Procedia Computer Science

Volume 51, 2015, Pages 374–383

ICCS 2015 International Conference On Computational Science

Using Genetic Algorithms for Maximizing Technical Efficiency in Data Envelopment Analysis

Martín González[1], Jose J. López-Espín[1], Juan Aparicio[1], Domingo Giménez[2], and Jesús T. Pastor[1]

[1] Centro de Investigación Operativa, Miguel Hernández University, Spain
martingonzes@gmail.com, jlopez@umh.es, j.aparicio@umh.es, jtpastor@umh.es
[2] Departamento de Informática y Sistemas, University of Murcia, Spain
domingo@um.es

Abstract
Data Envelopment Analysis (DEA) is a non-parametric technique for estimating the technical efficiency of a set of Decision Making Units (DMUs) from a database consisting of inputs and outputs. This paper studies DEA models based on maximizing technical efficiency, which aim to determine the least distance from the evaluated DMU to the production frontier. Usually, these models have been solved through unsatisfactory methods used for combinatorial NP-hard problems. Here, the problem is approached by metaheuristic techniques and the solutions are compared with those of the methodology based on the determination of all the facets of the frontier in DEA. The use of metaheuristics provides solutions close to the optimum with low execution time.

Keywords: Data Envelopment Analysis, Closest targets, Mathematical Programming, Efficiency Methodologies, Genetic Algorithms

1 Introduction

Over the past 50 years technologies have been estimated using many different approaches [9]. The two principal methods are stochastic frontiers, which use econometric techniques, and Data Envelopment Analysis (DEA), which is a non-parametric technique based on mathematical programming for the evaluation of technical efficiency of a set of decision making units (DMUs) that consume inputs to produce outputs [10]. Unlike other efficiency methodologies, DEA simultaneously provides both an efficiency score and benchmarking information through efficient targets. In DEA, the efficiency score is obtained from the distance between the assessed DMU and a point at the frontier of the technology, which serves as an efficient target for the assessed DMU.

An important stream of the recent literature in DEA is concerned with determining the least distance to the frontier from an assessed inefficient DMU or, equivalently, obtaining the

 Selection and peer-review under responsibility of the Scientific Programme Committee of ICCS 2015

doi:10.1016/j.procs.2015.05.257

efficient targets that maximize the final level of technical efficiency (see [2, 3, 5] to name but a few). This contrasts with the usual approaches followed from the origins of DEA, where only the furthest efficient targets are calculated for computational reasons. Indeed, maximizing technical efficiency is computationally difficult while minimizing technical efficiency is easier, since this is usually associated with the resolution of a standard linear program.

Regarding papers that have studied the computational aspects of DEA models associated with the determination of the least distance to the frontier and, therefore, related to the maximization of the technical efficiency, we cite Aparicio et al. [4] and Jahanshahloo et al. [11, 12, 13]. Some 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 Section 2, all these approaches have their strong and weak points and, consequently, there is currently no approach accepted as the best solution to the problem.

The approach in [4], based on Mixed Integer Linear Programming, is used and we use metaheuristics to try to solve the model these authors introduced. The complexity of the model makes it difficult to generate solutions satisfying all the restrictions. In [7, 14] heuristics were used to generate valid solutions for a subset of restrictions of the problem of maximizing technical efficiency in DEA. In this paper, all the constraints have been incorporated (in the previous papers only 9 of 14 constraints were considered), the heuristics are improved, and new ones are developed, so initial populations of solutions satisfying all the constraints are generated. Additionally, the solutions generated by our approach for a battery of simulated databases are compared with the solutions obtained through the determination of all the facets of the frontier in DEA (see [1]). In this case, the optimizer CPLEX was used to measure technical efficiency.

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, and the existing approaches for maximizing efficiency are outlined. Then, heuristic methods to generate initial populations with valid solutions are studied in Section 3. After that, a Genetic Algorithm to improve the solutions is discussed in Section 4. In Section 5, the results of some experiments are summarized. Section 6 concludes the paper and outlines some possible research directions.

2 Data Envelopment Analysis and the Problem to be Solved

DEA involves the use of Mathematical Programming to construct a non-parametric piece-wise surface over the data in the input-output space. Technical efficiency measures associated with the performance of each DMU are then calculated relative to this surface, as a distance from it.

Some notation is needed. Assume there are data on m inputs and s outputs for n DMUs (firms, universities, farms, etc). For the j-th DMU these are represented by $x_{ij} \geq 0$, $i = 1, \ldots, m$, and $y_{rj} \geq 0$, $r = 1, \ldots, s$, respectively.

The basic DEA models are the CCR [8] and the BCC [6]. 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. The "original" Enhanced Russell Graph measure [15] can be calculated for DMU k, $k = 1, \ldots, n$ as follows:

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

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 limitation, a weakness shared with other traditional measures in DEA. Specifically, the original Enhanced Russell Graph measure minimizes technical efficiency in equation 1. In order to maximize instead of minimize technical efficiency, it seems sufficient to change "min" for "max" in equation 1. However, this is not true. In this case, we could show that the solutions generated by the model would not be technically efficient, but inefficient (see [4]) and, therefore, could not serve as valid benchmark for the assessed DMU.

This problem was behind the introduction of different approaches to maximizing technical efficiency suitably in DEA. Some of these approaches propose using Mixed Integer Linear Programs to overcome the problem [4]. In the case of DMU k, the model to be solved would be:

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

One weakness of the approach in equation 2 is that it uses a "big M" to model the key constraints (c.7) and (c.8). Specifically, it allows us to link d_{jk} to α_{jk} by means of the binary variable b_{jk}. The value of M can be calculated if and only if all the facets that define the technology are previously determined. Unfortunately, the identification of all these facets is a combinatorial NP-hard problem. Something similar happens with the other approaches devoted to maximizing technical efficiency. All of them are related in some sense to a big M or the direct calculation of all the efficient facets (see Jahanshahloo et al. [11, 12, 13]).

In this paper, we apply a Genetic Algorithm to solve equation 2. In order to check the goodness of our approach, the results are compared with those obtained from the determination of all the facets of the frontier in DEA using a set of simulated numerical examples.

3 Heuristic Methods to Obtain Valid Solutions

Each solution of equation 2 is composed of β_k, α_{jk}, t_{ik}^-, $t_{rk}^+, \nu_{ik}, \mu_{rk}, d_{jk} \in \mathbb{R}^+$ and $b_{jk} \in \{0,1\}$, with $i = 1,\ldots,m, j = 1,\ldots,n, r = 1,\ldots,s$. In [7] some heuristics were presented to generate solutions of 13 of the 14 constraints in the equation. These heuristics are now improved so that the number of solutions satisfying the 14 constraints greatly increases. Two methods to generate the initial population of solutions are combined. If non valid solutions are obtained by using method 1, method 2 is used. The second method has a higher computational cost, and so it is used only when the first method fails.

3.1 Method 1

1. For a given k, the process starts generating b_{jk} $\forall j$ based on c.9., with the restrictions: the number of b_{jk} equal to 0 should be greater than s and lower than $s+m$ to calculate the values of α_{jk} and d_{jk} $\forall j$ by means of a system of equations in the next steps. α_{jk}, d_{jk} and b_{jk} are related through c.7. and c.8. The number of b_{jk} equal to 0 is s, with which the number of equations and of unknowns coincides. The positions of these zero values are generated randomly.

2. t_{rk}^+ $\forall r$ and β_k are generated using algorithm 1 in order to satisfy c.1. In this algorithm, the values of t_{rk}^+ are generated randomly between 0 and 1. Next, β_k is obtained using c.1. If β_k is lower than 0, then t_{rk}^+ $\forall r$ are decreased, and if β_k is greater than 1, then t_{rk}^+ $\forall r$ are increased. The process continues until $0 < \beta_k < 1$.

Require: $Y \in \mathbb{R}^{+^{h\times n}}$, DMU k
Ensure: $\forall r, t_{rk}^+ \in \mathbb{R}^+$, $0 < \beta_k < 1$
  Generate $\forall r, t_{rk}^+$ randomly between 0 and 1
  Obtain β_k using c.1.
  while $\beta_k \leq 0$ OR $\beta_k \geq 1$ **do**
    if $\beta_k < 0$ **then**
      Generate r randomly, and $t_{rk}^+ = t_{rk}^+/(2.0 + random(0,1,2))$
    else
      Generate r randomly, and $t_{rk}^+ = t_{rk}^+ * (2.0 + random(0,1,2))$
    end if
    Obtain β_k using c.1.
  end while

Algorithm 1: Generate t_{rk}^+ and β_k

3. Next, α_{jk} $\forall j$ are calculated using c.3. The number of α_{jk} different from 0 is equal to s through step 1. The values of α are calculated using c.3. by solving the system of equations.

4. t_{ik}^- are calculated using c.2. by solving the system of equations.

5. Finally, ν_{ik} $\forall i$, μ_{rk} $\forall r$ and d_{jk} $\forall j$ are calculated. The number of d_{jk} equal to 0 is the same as the number of α different from 0. Therefore, the values of ν_{ik} are generated randomly and those of μ_{rk} are obtained by solving system c.4.

3.2 Method 2

This method is used to recalculate the non valid solutions in method 1.

1. The first step coincides with that of the previous method: $b_{jk}\ \forall j$ are randomly generated based on c.9., with s values of b_{jk} equal to 0.

2. The values α are generated randomly, with $0 < \alpha_{jk} \leq 1$, for the same reason mentioned in step 3 of method 1.

3. Next, $\alpha_{jk}\ \forall j$ are modified using algorithms 2 and 3 in order to satisfy c.1, c.2., c.3., c.11., and c.12. In algorithm 2, since β_k is between 0 and 1, $\forall i$ the maximum value of $\beta_k x_{ik}$ in c.2. is equal to x_{ik}. Then, $\forall i$, $\sum_{j=1}^{n} \alpha_{jk} x_{ij}$ must be lower than x_{ik} in order to satisfy c.11. Therefore, the α with least effect in c.3. ($\alpha_{j_0 k}$) must be decreased. Otherwise, $\forall r$, $\sum_{j=1}^{n} \alpha_{jk} y_{rj}$ must be greater than y_{rk} in order to satisfy c.12. In the same way, the α with least effect in c.2. is increased. Algorithm 3 has been developed considering c.1. and c.3. In it the α with least effect in c.2. and c.3. ($\alpha_{j_0 k}$) is calculated in order to satisfy c.1.

Require: $Y \in \mathbb{R}^{+\,s \times n}$, $X \in \mathbb{R}^{+\,m \times n}$, DMU k
Ensure: $\forall r, t_{rk}^{+} \in \mathbb{R}$; $\forall i, t_{ik}^{-} \in \mathbb{R}$; $\forall j, \alpha_{jk} \in \mathbb{R}^{+}$, $0 < \beta_k < 1$
  for $i = 1, \ldots, m$ **do**
    if $x_{ik} < \sum_{j=1}^{n} \alpha_{jk} x_{ij}$ **then**
      Find j_0 / $\frac{1}{m}\sum_{i=1}^{m} x_{ij_0} - \frac{1}{s}\sum_{i=1}^{s} y_{ij_0} = max_{j=1,\ldots,n}\{\frac{1}{m}\sum_{i=1}^{m} x_{ij} - \frac{1}{s}\sum_{i=1}^{s} y_{ij}\}$
      $\alpha_{j_0 k} = \alpha_{j_0 k} * 0.95$
    end if
  end for
  for $r = 1, \ldots, s$ **do**
    if $y_{rk} > \sum_{j=1}^{n} \alpha_{jk} y_{rj}$ **then**
      Find j_0 / $\frac{1}{s}\sum_{i=1}^{s} y_{ij_0} - \frac{1}{m}\sum_{i=1}^{m} x_{ij_0} = max_{j=1,\ldots,n}\{\frac{1}{s}\sum_{i=1}^{s} y_{ij} - \frac{1}{m}\sum_{i=1}^{m} x_{ij}\}$
      $\alpha_{j_0 k} = \alpha_{j_0 k} * 1.05$
    end if
  end for
  $\forall j$ adjust α_{jk} with algorithm 3.
  Adjust β_k to satisfy c.11. and c.12. (step 4)
  Obtain $t_{rk}^{+}\ \forall r$ and $t_{ik}^{-}\ \forall i$ using c.2. and c.3. (step 4)

Algorithm 2: Adjust α_{jk} to satisfy c.2. and c.3.

4. Adjust β_k to satisfy c.2., c.3., c.11. and c.12. If c.11. is violated, then β_k is increased by a factor to satisfy c.2. and c.11. Otherwise, if c.12. is violated, then β_k is decreased by a factor to satisfy c.3. and c.12. This factor is decreased in each iteration for a finer adjustment. In each iteration $t_{rk}^{+}\ \forall r$ and $t_{ik}^{-}\ \forall i$ are obtained using c.2.and c.3.

5. Finally, $\nu_{ik}\ \forall i$, $\mu_{rk}\ \forall r$ and $d_{jk}\ \forall j$ are calculated as in step 5 of method 1.

4 Genetic Algorithm

A Genetic Algorithm is proposed here to improve the solutions obtained with the previous heuristic methods. A population of valid solutions of equation 2 is explored. As mentioned

Require: $Y \in \mathbb{R}^{+^{s\times n}}$, $X \in \mathbb{R}^{+^{m\times n}}$, DMU k, α_{jk} $\forall j$
Ensure: $\forall j, \alpha_{jk} \in \mathbb{R}^+$
$\forall j, 1 \leq j \leq n, p_j = \sum_{r=1}^{s} y_{rj}/y_{rk}$
repeat
Find j_0 / $\frac{1}{s}\sum_{i=1}^{s} y_{ij_0} + \frac{1}{m}\sum_{i=1}^{m} x_{ij_0} = min_{j=1,\ldots,n}\{\frac{1}{s}\sum_{i=1}^{s} y_{ij} + \frac{1}{m}\sum_{i=1}^{m} x_{ij}\}$
sum $= \sum_{j=1,\ldots,n, j\neq j_0} \alpha_{jk} p_j$
$\alpha_{j_0 k} = \frac{s - sum}{p_{j_0}}$
if $\alpha_{j_0 k} \leq 0$ **then**
$\forall j, \alpha_{jk} = \alpha_{jk} * 0.95$
end if
until $\alpha_{j_0 k} > 0$

Algorithm 3: Adjust α_{jk} to satisfy c.1.

in the previous section, a solution is composed by β_k, α_{jk}, t_{ik}^-, t_{rk}^+, ν_{ik}, μ_{rk}, $d_{jk} \in \mathbb{R}^+$ and $b_{jk} \in \{0, 1\}$, with $i = 1, \ldots, m$, $j = 1, \ldots, n$, $r = 1, \ldots, s$. Valid solutions are those satisfying the 14 constraints in equation 2, and the fitness is given by the objective function in this equation. The main characteristics of the algorithm are:

- *Initialization.* The initial population is generated with the above heuristics. Given the difficulty of obtaining valid solutions for the 14 constraints [14], some non-valid solutions can be included initially in the population. The individuals in the population are ordered by the fitness, with the non-valid solutions at the end of the population.

- *End Condition.* The Genetic Algorithm works by combining and mutating individuals in the population until some end condition is reached. Normally, a maximum number of iterations or a maximum number without improving the best solution is established. In our case, some values have been tested, and after a number of iterations there were no significant differences in the goodness of the solution found. For the experiments we have fixed the number of iterations to 1000.

- *Selection.* Only valid solutions are selected for combination, and the non-valid solutions (those not fulfilling all the restrictions) in the population will be substituted for new valid solutions generated.

  The best individuals in the population are selected to be combined, leaving at least 50% of the overall space to generate new solutions. That is, if the total capacity is 200 chromosomes, and there are 120 valid chromosomes, the first 100 (50% of the total) are selected for combination, and the remaining space is allocated for new solutions. In this case, the 20 worst valid solutions are discarded. If the valid solutions represents less than the 50% of the total size, all the valid solutions are selected to be crossed, and all the non-valid elements are discarded for the next generation.

- *Crossover*

  From those individuals selected for crossing, pairs of individuals are selected randomly and combined to generate two descendants. The process continues until there are sufficient new elements to substitute those not selected for combination. The new individuals are included in the ordered population.

  Crossover is fundamental to improve solutions, so three types of crossover have been implemented and evaluated. Because each individual has components of six types (β, t^+,

t^-, ν, μ and d), each combination will work with only one of these types. One crossover considers only one component and the others randomly select the type of component in each combination:

- *Crossover 1.* Only one feature of the chromosomes is considered, β. It appears in the objective function, and so its modification directly affects the fitness. The mean of β_1 and β_2 of the two ascendants is obtained and perturbed by adding and subtracting a value randomly generated between 0 and 1, γ: $\beta_1^{'} = \frac{\beta_1+\beta_2}{2} + \gamma$ and $\beta_2^{'} = \frac{\beta_1+\beta_2}{2} - \gamma$. If some β is outside the range $[0, 1]$, the descendants are generated with a new γ. The chromosomes so generated can be non-valid, so the values of t_{ik}^- and t_{rk}^+ are recalculated so that constraints c.1, c.2 and c.3 are fulfilled. The other values (ν_{ik}, μ_{rk} and b_{jk}) are inherited from the ascendants. The new chromosomes are evaluated to see if they are valid or not, and their fitness is calculated.

- *Crossover 2.* The values of t^+, t^-, ν, μ or d are crossed. In each combination only parameters of one type are combined, and the type is randomly selected. Each parameter has several elements, and a middle point crossover combination is used: if the ascendants are $a_1 = (a_{11}, a_{12})$ and $a_2 = (a_{21}, a_{22})$ the two descendants are $d_1 = (a_{11}, a_{22})$ and $d_2 = (a_{21}, a_{12})$. Moreover, as in crossover 1, the value of the other parameters are recalculated, and the new chromosomes are evaluated and inserted in the ordered population.

- *Crossover 3.* This is a combination of the previous crossovers. All the parameters are candidates for crossing, and one is randomly selected. The crossover works as explained, with recalculation of the generated elements and insertion in the population.

- *Mutation.* Each individual has a 10% probability of being selected for mutation. One of the parameters in the individual is selected randomly, and new values are randomly generated for this parameter. The values of the other parameters are not recalculated. If the new element is valid, its fitness is calculated and the individual is inserted in the ordered population.

5 Experimental Results

The generation of valid solutions is a difficult problem, and in previous works the problem of building valid solutions for 9 [14] or 13 [7] of the 14 constraints in equation 2 was tackled with heuristic methods. The simplified methods presented in Section 3 allow us to work with all the constraints. The execution time and the percentage of valid solutions generated with the previous heuristics and with those presented in this work are compared in Table 1, where the time is expressed in seconds and the values are the mean of ten executions for each problem size. The standard deviation is also shown. With the new heuristics more valid solutions are generated, for all the constraints and with a lower execution time. Thus, these heuristics can be used for the initial generation of individuals in the population of the Genetic Algorithm.

With an approximation method like the Genetic Algorithm we can not ensure the optimum solution is found. So, the solutions obtained with the Genetic Algorithm are compared with those obtained with CPLEX [1], which generates optimum solutions but at the expense of very large execution times and is impracticable for large problems. Table 2 compares the solutions

size			9 constraints [14]		13 constraints [7]		14 constraints	
m	n	s	time (sec)	% val.	time (sec)	% val.	time (sec)	% val.
2	15	1	$26.42_{51.44}$	$82_{35.58}$	$33.21_{10.82}$	$72_{18.12}$	$0.09_{0.02}$	$100_{0.00}$
3	25	2	$6.72_{16.03}$	$90_{30.46}$	$72.89_{15.56}$	$24_{20.97}$	$0.88_{0.68}$	$96_{2.85}$
4	30	2	$0.22_{0.16}$	$100_{0.00}$	$89.84_{18.63}$	$16_{21.13}$	$0.88_{1.74}$	$95_{1.49}$
5	40	3	$13.13_{20.64}$	$74_{43.40}$	$116.39_{12.86}$	$1.6_{2.49}$	$27.22_{42.38}$	$92_{9.07}$
6	60	4	$2.01_{1.13}$	$35_{44.07}$	$117.26_{14.15}$	$0.06_{0.10}$	$93.46_{70.08}$	$53_{35.57}$

Table 1: Execution cost and percentage of valid solutions with the previous heuristics and with the heuristics in Section 3, varying the problem size.

m	n	s	crossover 1	crossover 2	crossover 3
2	50	1	2.1e-07	2.1e-07	2.1e-07
3	10	2	**0.0344**	0.0373	0.0381
3	30	2	0.1122	0.1075	**0.1042**
3	60	2	0.0778	0.0714	**0.0689**
4	30	2	0.1431	0.1437	**0.1417**
4	30	4	0.1612	0.1029	**0.1054**

Table 2: Comparison of the solution obtained with the Genetic Algorithm using different crossover functions with the optimum solution (obtained with CPLEX).

obtained with the Genetic Algorithm with the three crossover considered with those obtained with CPLEX (the optimum ones), for different problem sizes and varying the number of DMUs, inputs and outputs. Each entry in the table represents the mean of ten experiments, and for each experiment the mean of the difference of the solution provided by CPLEX and that obtained with the Genetic Algorithm for all the DMUs. There are always small differences with respect to the optimum, so the Genetic Algorithm is a good alternative to an exact method when this can not be used due to huge computational cost. The differences between the three crossovers is not significant, but in general the third crossover gives better results, which can be expected because the space of solutions is searched by varying the parameters of all the available types.

In general, and as is normal in a metaheuristic, the initial iterations give important improvements in the fitness, and successive iterations give only small improvements if any. This happens for all the sizes experimented with, as is observed in Figure 1, which compares the fitness with the Genetic Algorithm with the three crossovers implemented with the optimum obtained with CPLEX, for one DMU in a problem with $m = 4$, $n = 30$ and $s = 3$. The best fitnesses obtained every 5 iterations up to 30 are represented. Similar results are obtained for other DMUs and other problem sizes.

Small problem sizes are used in Table 2 because the execution time of CPLEX increases exponentially with the problem size. This is seen in Figure 2, which compares (in logarithmic scale) the execution time with CPLEX and the Genetic Algorithm when varying the problem size.

6 Conclusions and Future Works

Maximizing technical efficiency or, equivalently, determining least distance measures are topics of relevance in recent DEA literature. However, it is well-known that from a computational point of view this has usually been solved by unsatisfactory approaches associated with a combinatorial NP-hard problem.

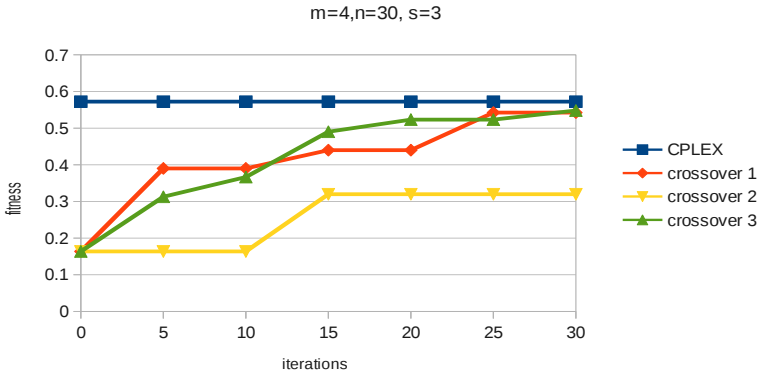

Figure 1: Comparison of the optimum fitness and those obtained with the Genetic Algorithm with the three crossovers considered at different iterations, for one DMU in a problem with $m = 4$, $n = 30$ and $s = 3$

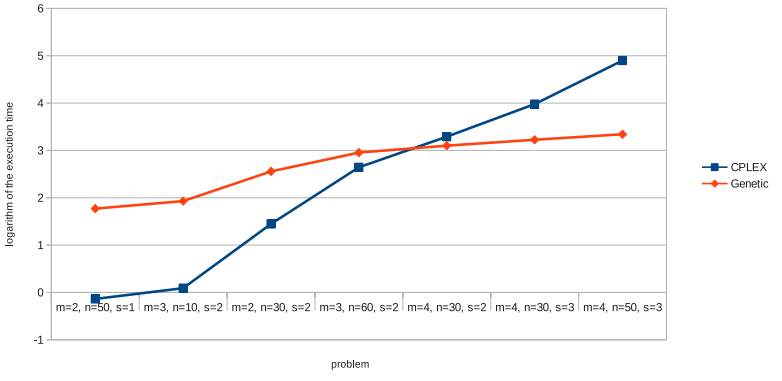

Figure 2: Comparison of the execution time (in seconds and logarithmic scale) of CPLEX and the Genetic Algorithm

This paper improves previous heuristics for the generation of valid solutions for an optimization problem for DEA. The new heuristic provides more valid solutions which satisfy all the constraints in the model and with a lower execution time. A Genetic Algorithm has been developed working with this initial population of valid and non-valid solutions to generate more valid solutions and to improve the best fitness obtained. The Genetic Algorithm gives solutions close to the optimum and is competitive with an exact method with high computational cost, which can not be used for large problems.

A deeper analysis should be made to tune the Genetic Algorithm to the problem to obtain better solutions with lower execution times.

We have studied the problem associated with the so-called Enhanced Russell Graph measure. Nevertheless, there are a lot of measures in DEA that can be used in the maximization of

technical efficiency. In this way, programming the approach based on metaheuristic algorithms to solve all of them can be seen as appropriate and interesting future work.

Acknowledgements

This work was supported by the Spanish MINECO, as well as European Commission FEDER funds, under grant TIN2012-38341-C04-03. Additionally, the authors are grateful to the Santander Chair of Efficiency and Productivity of the University Miguel Hernández of Elche for partially supporting this research.

References

[1] H. Amatatsu and T. Ueda. Measurement of simultaneous scale and mix changes in inputs and outputs using DEA facets and RTS. *European Journal of Operational Research*, 223(3):752–761, 2012.

[2] A. Amirteimoori and S. Kordrostami. A euclidean distance-based measure of efficiency in data envelopment analysis. *Optimization*, 59:985–996, 2010.

[3] J. Aparicio and J. T. Pastor. Closest targets and strong monotonicity on the strongly efficient frontier in DEA. *Omega*, 44:51–57, 2014.

[4] J. Aparicio, J. L. Ruiz, and I. Sirvent. Closest targets and minimum distance to the Pareto-efficient frontier in DEA. *Journal of Productivity Analysis*, 28:209–218, 2007.

[5] C. Baek and J. Lee. The relevance of DEA benchmarking information and the Least-Distance Measure. *Mathematical and Computer Modelling*, 49:265–275, 2009.

[6] R. D. Banker, A. Charnes, and W. W. Cooper. Some models for estimating technical and scale inefficiencies in data envelopment analysis. 30:1078–1092, 1984.

[7] C. Benavente, J. J. López-Espín, J. Aparicio, J. T. Pastor, and D. Giménez. Closets targets, benchmarking and data envelopment analysis: a heuristic algorithm to obtain valid solutions for the shortest projection problem. In *11th International Conference on Applied Computing*, 2014.

[8] A. Charnes, W. W. Cooper, and E. Rhodes. Measuring the efficiency of decision making units. *European Journal of Operational Research*, 2(429–444), 1978.

[9] T. Coelli, D. S. P. Rao, and G. E. Battese. *An Introduction to Efficiency and Productivity Analysis*. Kluwer Academic Publishers, 1998.

[10] 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 Publishers, 2000.

[11] 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*, 177:42–54, 2007.

[12] G. R. Jahanshahloo, F. Hosseinzadeh Lotfi, and M. Zohrehbandian. Finding the piecewise linear frontier production function in data envelopment analysis. *Applied Mathematics and Computation*, 163:483–488, 2005.

[13] G. R. Jahanshahloo, J. Vakili, and S. M. Mirdehghan. Using the minimun distance of DMUs from the frontier of the PPS for evaluating group performance of DMUs in DEA. *Asia-Pacific Journal of Operational Research*, 29(2):1250010, 2012.

[14] J. J. López-Espín, J. Aparicio, D. Giménez, and J. T. Pastor. Benchmarking and data envelopment analysis. An approach based on metaheuristics. In *Proceedings of the International Conference on Computational Science, ICCS 2014, Cairns, Queensland, Australia, 10-12 June, 2014*, pages 390–399, 2014.

[15] J. T. Pastor, J. L. Ruiz, and I. Sirvent. An enhanced DEA Russell graph efficiency measure. *European Journal of Operational Research*, 115:596–607, 1999.