A Parameterized Scheme of Metaheuristics with Exact Methods for determining the Principle of Least Action in Data Envelopment Analysis

Martín González
Center of Operations Research
University Miguel Hernández
Elche, Alicante, Spain

Jose J. López-Espín
Center of Operations Research
University Miguel Hernández
Elche, Alicante, Spain

Juan Aparicio
Center of Operations Research
University Miguel Hernández
Elche, Alicante, Spain

Domingo Giménez
Department of Computer Science and Systems
University of Murcia, Spain

El-Ghazali Talbi
University of Lille, France

***Abstract*—Data Envelopment Analysis (DEA) is a non-parametric methodology for estimating technical efficiency of a set of Decision Making Units (DMUs) from a dataset of inputs and outputs. This paper is devoted to computational aspects of DEA models under the application of the Principle of Least Action. This principle guarantees that the efficient closest targets are determined as benchmarks for each assessed unit. Usually, these models have been addressed in the literature by applying unsatisfactory techniques, based fundamentally on combinatorial NP-hard problems. Recently, some heuristics have been developed to partially solve these DEA models. This paper improves the heuristic methods used in previous works by applying a combination of metaheuristics and an exact method. Also, a parameterized scheme of metaheuristics is developed in order to implement metaheuristics and hybridations/combinations, adapting them to the particular problem proposed here. In this scheme, some parameters are used to study several types of metaheuristics, like Greedy Random Adaptative Search Procedure, Genetic Algorithms or Scatter Search. The exact method is included inside the metaheuristic to solve the particular model presented in this paper. A hyperheuristic is used on top of the parameterized scheme in order to search, in the space of metaheuristics, for metaheuristics that provide solutions close to the optimum. The method is competitive with exact methods, obtaining fitness close to the optimum with low computational time.**

I. Introduction

This paper is devoted to computational aspects of Data Envelopment Analysis (DEA) models under the application of the Principle of Least Action. DEA is a non-parametric technique (it is not necessary to set a specific functional form for the underlying production function) which is based on mathematical programming and generates a polyhedral technology. There are different DEA efficiency measures, depending on how the distance from the evaluated DMU to the frontier of the technology is operationalized. Each DEA efficiency measure is calculated by solving a mathematical programming model. DEA models provide both an efficiency score for each of the assessed DMUs and information on the targets that have been used in the efficiency assessment in the case of inefficient DMUs. The targets are the coordinates, in the input-output space, of the efficient projection point on the frontier and thus represent levels of operation of inputs and outputs that can make the corresponding inefficient DMU perform efficiently. Overall, targets indicate keys for the inefficient DMUs to improve their performance and, therefore, are managerially important.

One common criticism of targets yielded by traditional DEA efficiency models is that they are located far from the assessed DMU (see, for example, Aparicio et al. [1]), being too demanding and not easily achievable for firms. This has aroused increasing interest among researchers to develop DEA measures of technical efficiency capable of yielding achievable targets. Examples are the papers by Briec and Lesourd [2], Pastor and Aparicio [3], and Aparicio and Pastor [4], [5], [6]. The philosophy behind all these approaches is the application of the Principle of Least Action (PLA), which always seeks the efficient targets closest to the evaluated unit (Aparicio et al. [7]).

A point to highlight is that applying the PLA is computationally more difficult than obtaining the furthest efficient targets, since these are usually associated with the resolution of a standard linear program, something that does not happen with the determination of the least distance to the production frontier (see Briec [2]). Regarding papers that have studied the computational aspects of DEA models associated with the PLA, some of these approaches are based on Mixed Integer Linear Programming ([1]), others are derived from algorithms that facilitate the determination of all the facets of a polyhedron, while still others apply metaheuristics such as genetic algorithms. As we will argue in Section II, all these approaches have strong and weak points and there is currently no approach accepted as the best solution to the problem of determining closest efficient targets from a computational point of view.

The approach in [1] has been recently studied from a

978-1-5090-4601-0/17/$31.00

metaheuristic perspective (Benavente et al. [8], López-Espín et al. [9] and González et al. [10]). In [8], [9] heuristics were used to generate valid solutions for a subset of restrictions of the problem, while in [10] all the constraints are incorporated, the heuristics are improved, and new ones are developed, so initial populations of solutions satisfying all the constraints are generated.

Our paper takes up where González et al. [10] left off in the application of metaheuristics to the approach in [1]. The improvement of previous heuristics for the generation of valid solutions is a possible option, but greatly limits the search for valid solutions for large problem sizes, because when the number of variables grows, the number of valid solutions decreases.

Exact methods can also be used to solve these problems. The main drawback of these methods is the great amount of time needed to solve a NP-hard problem. When the problem grows, the number of possible combinations between variables increases exponentially. The studies on the execution time show that when heuristics are used, the execution time is lower than the exact methods for the same problems.

The contributions of this work include the development of an algorithm that belongs to the class of hybrid metaheuristics [11]. Metaheuristics and exact methods are complementary optimization strategies in terms of the quality of solutions and the search time used to find them. The hybrid metaheuristic that combines metaheuristics with exact method is called Matheuristic in this paper. In addition, a parameterized scheme of metaheuristics is introduced to use metaheuristics in combination with the same exact method and to compare them. Metaheuristic algorithms are very powerful in dealing with this kind of problem [12]. In this paper, some examples of Greedy Random Adaptative Search Procedure, Genetic Algorithms or Scatter Search are developed, and combinations of them are created using the parameterized scheme. The metaheuristics work with an initial population of valid and invalid solutions generated by the exact method, trying to obtain a greater number of valid solutions and better fitnesses, using techniques of improvement, combinations and mutations. A hyperheuristic is implemented to take control over the parameterized metaheuristic scheme. This method is able to create several metaheuristics and compare them, searching in the space of metaheuristics. Furthermore, the hyperheuristic is implemented as a metaheuristic with the same parameterized scheme. In this way, satisfactory metaheuristics (determined by the values of the parameters in the parameterized metaheuristic scheme) are obtained. Figure 1 shows the combination of all the mentioned methods.

The remainder of the paper is organized as follows. In Section II, a brief introduction of the main notions associated with Data Envelopment Analysis is presented, and the existing approaches for determining closest targets are outlined. The working problem is also presented in this section. The Matheuristic used to generate and improve valid solutions is studied in Section III. A parameterized scheme to improve the solutions is discussed in Section IV. In Section V, the results of some experiments are summarized. Section VI concludes the paper and outlines some possible research directions.

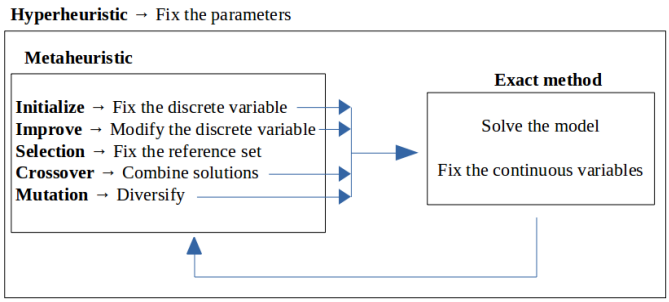

Fig. 1: Scheme of a hyperheuristic working on top of hybrid metaheuristic-exact methods

II. 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.

Before solving the Mathematical Programming model, we introduce some notations. Let us assume that data on m inputs and s outputs for n DMUs are observed. For the j-th DMU these are represented by $x_{ij} \geq 0$, $i = 1, \ldots, m$, and $y_{rj} \geq 0$, $r = 1, \ldots, s$.

$$\max \left\{ \beta_k - \frac{1}{m} \sum_{i=1}^{m} \frac{t_{ik}^-}{x_{ik}} \right\}$$

s.t.

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

Basic DEA models are based on radial projections to the production frontier. However, many other approaches give the projection freedom, so the final efficient targets do not conserve the mix of inputs and outputs. This is the case, for example, of the original Enhanced Russell Graph measure [13]. However, it presents a drawback shared with other traditional measures in DEA. In particular, the traditional Enhanced Russell Graph measure yields efficient targets that are far from DMU k. Therefore, in order to determine the closet efficient targets, [1] includes the following Mixed Integer

Linear Programming model that must be solved for each DMU k in the sample.

The definition and interpretation of the decision variables and constraints of model 1 can be found in [1].

One weakness of the approach in model 1 is that it uses a "big M" in (c.7) and (c.8). These constraints allow 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 DEA technology are previously determined. Unfortunately, the identification of all these facets is a combinatorial NP-hard problem. This weakness will be overcome in the new approach introduced here, since the new methodology does not need to resort to a big M to obtain the desired result.

III. MATHEURISTIC

The algorithm presented in this paper, called matheuristic, combines metaheuristics with exact methods, delivering the necessary work and improving performance. It has been developed to solve model 1. In this approach, a solution is encoded as a combination of values for all the variables in model 1. This solution is composed of the variables β_k, α_{jk}, t^-_{ik}, $t^+_{rk}, \nu_{ik}, \mu_{rk}, d_{jk} \in \mathbb{R}^+$ and $b_{jk} \in \{0,1\}$, with $i = 1,\ldots,m, j = 1,\ldots,n, r = 1,\ldots,s$. Only the solutions that meet all 14 constraints of the model 1 are considered valid. The others are classified as invalid solutions, and the algorithm will try to improve them and transform them into valid ones.

The matheuristic follows the same skeleton as a common metaheuristic, but includes an exact method to solve the mathematical problem. The configuration of this algorithm is described in the following steps: Initialize (generate the binary variable b_{jk} and solve the model 1 using an exact method), improvement, selection, combine and mutate. A previous step is included where the efficiency of each DMU is determined, separating the efficient DMUs that form the frontier to use them to solve the inefficient ones. Algorithm 1 shows the steps of this matheuristic.

The steps of the algorithm are explained in greater depth:

1) **Find the efficient DMUs**: The algorithm solves an additive model 2 (Charnes et al.[14]) to find which DMUs of the set are efficient. A set, E, is created with all the efficient DMUs. The number of efficient DMUs is denoted by w. Now, for each inefficient DMU, model 1 is solved only using the efficient DMUs from $E = \{j = 1,\ldots,n/A_j = 0\}$. So, the number of variables in model 1 decreases, and the variables α_{jk} and d_{jk} are the most affected in constraints (c.2), (c.3) and (c.4), because now only the DMUs in set E are used to solve these constraints.

$$
\begin{aligned}
& A_k = \max \left\{ \textstyle\sum_{i=1}^{m} \tau^-_{ik} + \sum_{r=1}^{s} \tau^+_{rk} \right\} \\
& \text{s.t.} \\
& \textstyle\sum_{j=1}^{n} \lambda_{jk} x_{ij} = x_{ik} - \tau^-_{ik} \quad \forall i \quad (c.1) \\
& \textstyle\sum_{j=1}^{n} \lambda_{jk} y_{rj} = y_{rk} + \tau^+_{rk} \quad \forall r \quad (c.2) \\
& \lambda_{jk} \geq 0 \quad (c.3) \\
& \tau^-_{ik} \geq 0 \quad (c.4) \\
& \tau^+_{rk} \geq 0 \quad (c.5)
\end{aligned}
\tag{2}
$$

Require: Inputs (x_{ij}), Outputs (y_{rj})
Ensure: E, Solution with best fitness.

```
Find the number of efficient DMUs (w) with the additive model and save them on the efficient set (E)
for k = 1,...,n do
  if k ∉ E then
    for i = 1,...,InitialPopulation do
      Generate the binary variable b_pk randomly, where p = 1,...,w.
      Solve model 1 (using set E of DMUs) to generate solutions with an exact method. Some solutions can be invalid.
    end for
    Improve valid and invalid solutions.
    while not End Condition do
      Select a number of valid and invalid solutions.
      Combine the selected solutions to create new ones.
      Improve the new solutions generated and mutate some solutions randomly.
    end while
  end if
  Save the best solution.
end for
```

Algorithm 1: Algorithm to create and improve solutions

2) **Initialize**: This step is used to generate the initial population. The algorithm includes two steps:
   a) **Generate the binary variable**: For the inefficient DMUs, a binary variable (b_{pk}) is created. Each DMU k has a variable b_{pk}, $p = 1,\ldots,w$. It is randomly created, with the same probability of 0 or 1.
   b) **Solve the model**: Once b_{pk} has been generated, model 1 is solved with an exact method, obtaining values for the rest of variables. The exact method tries to solve the model while fulfilling the 14 constraints. If a valid solution is found, the fitness value is saved and the solution is marked as valid. If not, it is marked as invalid. The exact method always finds a solution, which can be valid or invalid.
3) **Improvement**: A number of valid and invalid solutions, selected randomly from the reference set, are improved. The algorithm tries to improve the fitness value from the valid solutions, and tries to transform the invalid solutions into valid ones. In each solution, a single value of b_{pk} (randomly selected) is modified, and model 1 is solved again using the exact method. For valid solutions, if the fitness is improved, the new value of b_{pk} and the fitness obtained are saved. If the fitness does not improve, the new b_{pk} is discarded, and this step is repeated, changing another value of the initial b_{pk}. For invalid solutions, if the new b_{pk} generates a valid solution (solving model 1 with the exact method), the value

is saved and the improvement stops for this solution. A maximum number of evaluations is established, both for valid and invalid solutions.

4) **Selection**: The solutions are sorted, with the valid ones first, in order from the highest to the lowest fitness, followed by the invalid ones, randomly ordered. A percentage of solutions is selected to be combined and mutated. The best solutions (with the highest fitness) are selected from the valid set. From the invalid set, the solutions are selected randomly.
5) **Combine**: The algorithm includes a combination function. It combines pairs of solutions, randomly chosen from those previously selected. The pair of solutions must be from the same group, combining valid solutions with valid solutions and invalid solutions with invalid ones. These combinations generate new solutions that inherit some characteristics from their parents. For all the combinations, the algorithm only uses b_{pk}. The rest of the parameters are obtained solving model 1 with the exact method. To carry out these combinations, a multi-point crossover operator has been developed which generates an offspring by copying its genes from the parents, chosen according to a randomly constructed crossover mask. This mask is used to generate a new b_{pk}, and contains ones and zeros randomly generated with the same probability. The selected values from each of the two solutions are determined by the mask in each position, taking the value from the first solution if there is 1 in the mask, and from the second one if not. With this new variable, model 1 is solved using the exact method.
6) **Improvement and mutation**: The solutions generated by crossover are improved using the same function as in step 3. A mutation step is also included. The mutation is used to diversify the search. It modifies an invalid solution by randomly generating a new b_{pk}, and model 1 is solved with the exact method. Usually, the solutions obtained by mutation are not better than the previous ones, and in most cases they are invalid. To try to make them valid and useful for the algorithm, some mutated solutions (randomly selected) are improved in the same way.

Steps 4, 5 and 6 are repeated until a stopping condition is reached. The stopping condition states a certain number of repetitions for the improvements, combinations and mutations (`while` in Algorithm 1). It also states that when the best solution is not improved in a specified number of iterations, the algorithm ends.

IV. A Parameterized Scheme of Metaheuristcs

In a previous work [10], the solution of the problem was addressed with a genetic algorithm. Now, not only a genetic algorithm is used, and a modified version of the parameterized scheme of metaheuristics created in [15] is applied. This offers the possibility of analysing a large number of different metaheuristics with the objective of finding the best one, which allows us to obtain a satisfactory solution. The scheme proposed in this section includes parameters to modify the matheuristic used in Section III. The parameterized scheme comprises a skeleton (Algorithm 2) with six basic functions: Initialize, EndCondition, Selection, Combine, Improve and Include. The reference set with all the solutions is called S.

Require: S, ParamINI, ParamEND, ParamSEL, ParamCOM, ParamIMP, ParamINC.
1: Initialize (S,ParamINI)
2: **while** (not EndCondition(S,ParamEND)) **do**
3: SS = Select (S,ParamSEL)
4: SS1 = Combine (SS,ParamCOM)
5: SS2 = Improve (SS1,ParamIMP)
6: S = Include (SS2,ParamINC)
7: **end while**
Algorithm 2: Parameterized metaheuristic scheme

To analyze how the parameterized scheme can be applied to our problem, three basic metaheuristics are considered: Greedy Randomized Adaptive Search Procedure (GRASP), Scatter Search (SS) and Genetic Algorithms (GA) ([15]). They are implemented within the scheme, and the inclusion of the parameters allows us to experiment with the three basic metaheuristics and with different types of hybridations.

This scheme is applied to the algorithm shown in Section III. For that, some modifications are made to the basic Algorithm 2. What is shown below is the particular application of this parameterized scheme to Algorithm 1. The scheme is applied for each inefficient DMU (`for` loop in Algorithm 1, line 2).

- **Initialize**: This step generates the initial population of the algorithm in Section III. The initial population is composed of a total of INEIni valid and invalid solutions. A percentage of these solutions (PEIIni) is improved with an intensification (IIEIni) (line 8 in Algorithm 1). Finally, a number of solutions (FNEIni) constitutes the reference set, denoted by S in Algorithm 2.
- **EndCondition**: Two conditions are considered for the `while` in line 9: A maximum number of iterations (MNIEnd) or a maximum number of iterations without improving the best solution (NIREnd).
- Select: The solutions are clustered in two groups: valid and invalid solutions. The best NBESel solutions of the valid group are selected, and NWESel are randomly selected from the invalid ones. If the number of valid or invalid solutions is lower than the corresponding parameter, all the solutions are selected. So, the set SS is divided into two subsets: SS_v and SS_i.
- **Combine**: The solutions selected are combined by pairs. A number of crossovers between the best solutions (PBBCom) using the set SS_v and the worst ones (PWWCom) using the set SS_i is performed. Combinations between valid and invalid solutions are not carried out. Two sets are created to be improved: $SS1_v$ and $SS1_i$
- **Improve**: A percentage (PEIImp) of the solutions generated is improved with a specified intensification (IIEImp).

Two improvement functions are used: The first tries to improve the valid solutions ($SS1_v$) and the second tries to make valid solutions from the invalid ones ($SS1_i$). This improvement is applied in the solutions created in the initialization and in the solutions obtained after the combinations (lines 8 and 12 in Algorithm 1). To explore the solutions space, a diversification is included (equivalent to the mutation in a Genetic Algorithm). A percentage of solutions (PEDImp) is diversified using the mutation function explained in Section III, with a specific intensification (IIDImp) for improvement of the elements (with the function shown in step 3 in Section III) obtained by diversification.

- **Include**: The best FNEIni elements are selected for the next iteration. Valid and invalid solutions could be included, selecting the invalid ones randomly.

Table I summarizes the parameters used in the basic functions of the scheme. A parameterized scheme has already been used and tested in [15], where more parameters were considered. The number and meaning of the parameters would change if other basic metaheuristics are considered or if the basic functions are implemented in a different way, but the parameters and the basic metaheuristics considered here allow us to experiment with different heuristics and combinations/hybridations, and to improve previous results.

Table II shows three sets of values of the parameters, which correspond to three basic metaheuristics. The values of each parameter have been designed with the aim of representing the basic metaheuristics used in the experiments. These values have been decided to perform the tests efficiently. Some parameters are set to 0 to drive the metaheuristic to its most specific form. Each metaheuristic uses different tools to obtain good solutions. A Genetic Algorithm (GA) creates a number of initial solutions, combines only the best ones and mutates some elements, with a certain improvement. By contrast, a Greedy Randomized Adaptive Search Procedure (GR) creates more initial solutions, and improves them through high intensification. The Scatter Search (SS) is a population based algorithm, which works with a small set of solutions, with good and scattered elements, combining them and tries to improve each solution.

Apart from these basic metaheuristics, a huge number of combinations/hybridations can be considered simply by selecting different values for the parameters. The best metaheuristic from those obtained with the parameterized scheme could be obtained by generating all the possible combinations of the parameters and by applying them to some small training problems. In this way, the metaheuristic (given by the values of the parameters) which gives the best results for the training set can be considered as a satisfactory metaheuristic for the problem in hand. The number of possible combinations of the parameters in the parameterized metaheuristic scheme is huge, and the problem of obtaining the best metaheuristic for the training set is an optimization problem. So, it is a suitable problem for metaheuristics. Taking into account that model 1 is intended to maximize the objective function, a hyperheuristic is developed to find the parameter configuration that obtains the maximum value. For this, the hyperherustic uses the function value like a fitness and tries to obtains the optimal metaheuristic that provides the maximum value. A hyperheuristic can be developed as a metaheuristic searching for satisfactory metaheuristics, and can be developed on top of the parameterized metaheuristic scheme. The hyperheuristic uses the same scheme (Algorithm 2) as the metaheuristic. In this case, and for low execution times, only the Initialize and Combine functions are considered:

1) **Initialize:** A number HINEIni metaheuristis are generated.
2) **Combine:** A number of combinations between metaheuristics in the initial population is established (HPCom).

The combination uses the same method used for the metaheuristics, although, in this case, by combining the parameters of the metaheuristics selected to create new metaheuristics, which are improved by increasing a randomly selected parameter. The increment of the parameter is specified with an intensification parameter.

V. Experimental Results

Experiments were conducted to analyze the effectiveness of the parameterized scheme of metaheuristics and the hyperheuristic developed. The number of valid solutions obtained with the algorithm proposed in this paper is studied for different population sizes. The fitness values obtained with basic metaheuristics similar to GA, GR or SS are compared with hybrid metaheuristics generated with a hyperheuristic using the scheme. We call the metaheuristic generated by the hyperheuristic "general metaheuristic". Finally, the fitness and time obtained with a satisfactory metaheuristic are compared with those with an exact method. For all the experiments, the IBM ILOG CPLEX Optimization Studio (CPLEX) is used. The system used in the experiments is a NUMA node with 4 Intel hexa-core Nehalem-EX EC E7530, with 24 cores, at 1.87 GHz and 32 GB of RAM.

One of the most critical aspects of the algorithm presented in Section III, lies in the creation of valid solutions. To study this aspect, the Algorithm 1 has been executed for three initial population sizes (10, 100 and 1000), with 10 executions for each size. The execution is performed using only the first part of the algorithm, without improvements, crossovers or mutations. Execution time, fitness, and the percentage of valid solutions generated with the algorithm are compared in Tables III (small problems) and IV (big problems), which also show the standard deviation (as a subscript) in the percentage of valid solutions. For the other measures the standard deviation is not shown because the values do not suffer significant changes. Times are expressed in seconds.

When the problem size increases, the number of valid solutions usually decreases. For small problems, increasing the initial population implies a substantial improvement in the fitness obtained. When the problem size is larger, this

TABLE I: Parameters used in each basic function of the parameterized metaheuristic scheme

Function	Parameter	Description
Initialize	INEIni	Initial number of elements
	FNEIni	Final number of elements selected for the iterations
	PEIIni	Percentage of elements to improve in the initialization
	IIEIni	Intensification of the improvement of elements in the initialization
End Condition	MNIEnd	Maximum number of iterations
	NIREnd	Maximum number of iterations without improving the best solution
Selection	NBESel	Number of best valid elements selected
	NWESel	Number of invalid elements selected
Combination	PBBCom	Number of combinations between valid elements
	PWWCom	Number of combinations between invalid elements
Improve	PEIImp	Percentage of elements obtained by combination to be improved
	IIEImp	Intensification of the improvement
	PEDImp	Percentage of elements to diversify
	IIDImp	Intensification of the improvement of elements obtained by diversification

TABLE II: Values of the parameters for the three basic metaheuristics considered

Metaheuristic	IINEIni	FNEIni	PEIIni	IIEIni	NBESel	NWESel	PBBCom
GA	500	250	0	0	100	100	100
GR	1500	1	100	25	0	0	0
SS	1000	30	50	10	10	10	50

Metaheuristic	PWWCom	PEIImp	IIEImp	PEDImp	IDEImp	MNIEnd	NIREnd
GA	100	0	0	10	10	25	5
GR	0	0	0	0	0	25	5
SS	50	50	10	0	0	25	5

TABLE III: Execution time, percentage of valid solutions and fitness obtained for all the DMUs, obtained with Algorithm 1, for several small problems and three values of parameter INEIni

size			Hybrid method (10)			Hybrid method (100)			Hybrid method (1000)			Efficiency	
m	n	s	% val.	fitness	time(sec)	% val.	fitness	time(sec)	% val.	fitness	time(sec)	Efficient	Inefficient
2	30	1	$55.55_{7.00}$	0.137	0.01	$100.00_{0.00}$	0.413	6.23	$100.00_{0.00}$	0.413	59.50	27	3
3	30	2	$20.00_{6.53}$	0.059	0.01	$83.92_{5.90}$	0.355	4.12	$100.00_{0.00}$	0.566	46.81	20	10
4	30	2	$19.05_{10.32}$	0.041	1.32	$95.23_{1.47}$	0.419	4.05	$100.00_{0.00}$	0.48	49.35	21	9
4	30	3	$0.00_{0.00}$	0.00	1.42	$45.45_{12.72}$	0.146	2.37	$90.90_{1.10}$	0.414	27.29	11	19
5	30	3	$20.00_{7.07}$	0.247	0.02	$60.00_{8.36}$	0.282	3.62	$90.0_{9.08}$	0.398	30.03	10	20
6	30	4	$0.00_{0.00}$	0.00	0.01	$50.00_{27.38}$	0.073	1.15	$100.0_{0.00}$	0.208	13.71	2	28

footnotesize

TABLE IV: Execution time, percentage of valid solutions and fitness obtained for all the DMUs, obtained with Algorithm 1, for several large problems and three values of parameter INEIni

size			Hybrid method (10)			Hybrid method (100)			Hybrid method (1000)			Efficiency	
m	n	s	% val.	fitness	time(sec)	% val.	fitness	time(sec)	% val.	fitness	time(sec)	Efficient	Inefficient
2	100	1	$89.58_{3.11}$	0.359	2.01	$100.00_{0.00}$	0.459	22.12	$100.00_{0.00}$	0.462	226.50	96	4
3	100	2	$34.48_{2.35}$	0.312	2.58	$89.65_{5.28}$	0.40	20.25	$100.00_{0.00}$	0.505	201.52	87	13
4	100	2	$19.75_{8.14}$	0.261	2.33	$81.48_{2.29}$	0.360	22.50	$100.00_{0.00}$	0.484	223.05	81	19
4	100	3	$2.63_{1.12}$	0.006	2.73	$19.74_{5.22}$	0.051	21.04	$71.05_{12.33}$	0.273	209.91	66	34
5	100	3	$15.15_{7.28}$	0.261	2.32	$39.39_{6.12}$	0266	20.87	$75.75_{2.87}$	0.432	193.62	76	24
6	100	4	$0.00_{0.00}$	0.00	1.00	$2.38_{1.09}$	0.017	15.32	$30.95_{12.54}$	0.115	151.12	42	58

parameter has more influence and better solutions are generated. The modification of the initial population significantly improves both the amount of valid solution found and the fitness obtained, but at the expense of larger execution times (Table IV).

Another aspect to be considered is the amount of inefficient DMUs in each problem. As discussed in Section III, the algorithm initially estimates the number of efficient DMUs, and tries to evaluate the fitness of the inefficient ones. As shown in Tables III and IV, if the number of DMUs to be evaluated is fixed and the number of inputs and outputs increases, the number of efficient DMUs also increases (due to the way the input and output data are generated for testing). When the number of inefficient DMUs increases, it is necessary to use larger initial populations. More valid solutions could be generated with larger populations, but the execution time increases.

A hyperheuristic is used to generate hybrid metaheuristics. The method works by generating random metaheuristics, using as a seed the parameters of the three basic metaheuristics. The values for the parameters are randomly generated, in a range between the minimum and maximum values of the basic metaheuristics in Table II. A number of new metaheuristics are generated by a combination working as the combination for the algorithm in Section III. The hyperheuristic uses parameters for the number of metaheuristics to generate and of combina-

tions. 20 metaheuristics are generated in the experiment: 10 in the initial population (HINEIni) and 10 with combinations (HPCom).

In the experiments, the hyperheuristic is applied to each problem. Figure 2 shows a comparison of the fitness obtained using the three basic metaheuristics (GRASP, Genetic Algorithm and Scatter Search), whose parameters are shown in Table II, with the fitness obtained with the best metaheuristic obtained by the hyperheuristic. The fitness obtained by CPLEX is also shown. The results are shown for optimality problems and different sizes. However, the fitness obtained by CPLEX in the last problem (6-50-4) is not shown because the time to solve this kind of NP-hard problem is huge. Each value is the average of 10 executions with the same inputs and outputs.

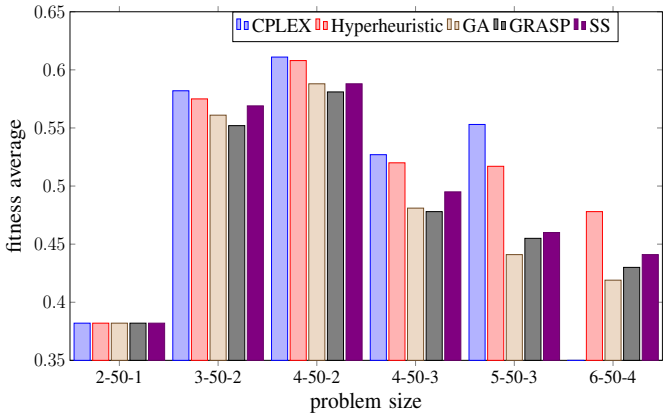

Fig. 2: Fitness obtained by the three basic metaheuristics, an exact method and a hyperheuristic, varying the problem size

The metaheuristics obtain satisfactory values of the fitness, with low execution times. So, they are competitive with exact methods for large problems when these are impracticable. The application of the hyperheuristic generates better results than the metaheuristics, especially for large problems. Table V shows the values of the parameters of the metaheuristic obtained by the hyperheuristic for each problem. The parameters obtained with the hyperheuristic are similar for different problems. A large number of elements are created, and more than 50% of the elements are improved. The number of selected elements does not have a great influence. By contrast, the number of combinations should be high. Improvement and end parameters vary widely. Some problems improve many elements while others include more mutations.

The hyperheuristic is trained with small problems to generate a general metaheuristic with satisfactory behavior for those problems, and which is expected to work satisfactorily for other problems The metaheuristics for each problem size can be combined in different ways. To analyze the behavior of a general metaheuristic, a value for each parameter is selected (from the 6 problems proposed), creating a metaheuristic that combines features of all the problems evaluated. For example, if the highest value is selected for each parameter, Table VI shows that the fitness obtained with this general metaheuristic improves the fitness generated by the hyperheuristic when it is directly applied to each problem.

The parameterized scheme of metaheuristics is used to generate a satisfactory metaheuristic by training a hyperheuristic with small problems. Better fitness is obtained, but the optimum solution is not always reached. Even so, metaheuristics are an alternative to exact methods for large problems, for which the execution time of exact methods is unaffordable. The execution time of an exact method is compared with that of the general hybrid metaheuristic in Figure 3. In the graph it can be observed how for small sizes the exact method without metaheuristics improves, in computational time, to the proposed algorithm. As the size of the problem increases, the hybrid algorithm improves this characteristic. At the problem with 4-50-2 size the times are similar, and when the size grows, the hybrid algorithm improves to the exact method alone. For the biggest problem considered (6-50-4 is not included due to huge execution time with CPLEX), the general metaheuristic is approximately 100 times faster than the exact method, and the difference increases with the problem size.

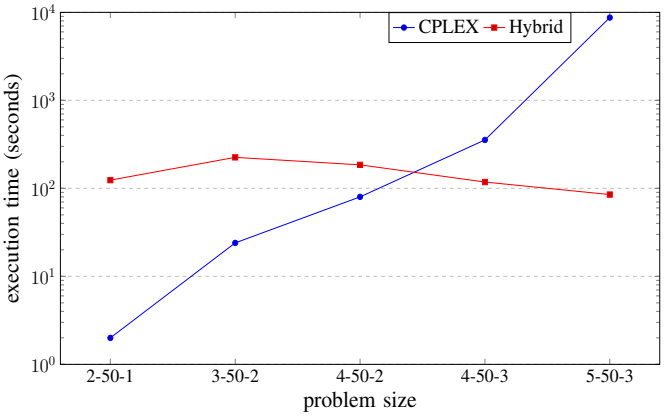

Fig. 3: Comparison of the execution time (in seconds and logarithmic scale) between an exact method and the hybrid metaheuristic developed

VI. Conclusions and Future Works

Determining closest efficient targets is a topic of relevance in recent DEA literature. However, it is well-known that from a computational point of view this has usually been tackled with unsatisfactory approaches, associated with combinatorial NP-hard problems.

This paper improves previous heuristics for the generation of valid solutions for an optimization problem in DEA - the Enhanced Russell Graph. The new algorithm provides more valid solutions which satisfy all the constraints in the model and with low execution times. A parameterized scheme has been developed by working with an initial population of valid and invalid solutions to generate more valid solutions and to improve the fitness. Future applications will incorporate parameters for the control of the exact method.

TABLE V: Metaheuristic parameters obtained by the hyperheuristic for different problem sizes

Problem size	IINEIni	FNEIni	PEIIni	IIEIni	NBESel	NWESel	PBBCom
$m = 2$, $n = 50$, $s = 1$	1050	63	75	11	46	66	15
$m = 3$, $n = 50$, $s = 2$	977	94	50	10	59	69	69
$m = 4$, $n = 50$, $s = 2$	1238	85	68	8	65	60	94
$m = 4$, $n = 50$, $s = 3$	1451	33	59	3	10	68	59
$m = 5$, $n = 50$, $s = 3$	1500	171	92	18	32	10	78
$m = 6$, $n = 50$, $s = 4$	1406	177	68	13	31	53	63

Problem size	PWWCom	PEIImp	IIEImp	PEDImp	IIDImp	MNIEnd	NIREnd
$m = 2$, $n = 50$, $s = 1$	83	35	10	8	3	5	3
$m = 3$, $n = 50$, $s = 2$	100	42	4	2	5	12	5
$m = 4$, $n = 50$, $s = 2$	89	13	4	9	2	10	3
$m = 4$, $n = 50$, $s = 3$	100	23	7	7	5	8	5
$m = 5$, $n = 50$, $s = 3$	29	33	8	10	8	7	4
$m = 6$, $n = 50$, $s = 4$	41	28	8	9	5	5	5

TABLE VI: Comparison between the fitness obtained with the direct application to different problems of the hyperheuristic and with the general metaheuristic obtained by the hyperheuristic

Schemes	Hyperheuristic	General Metaheuristic
$m = 2$, $n = 50$, $s = 1$	0.393197	0.393197
$m = 3$, $n = 50$, $s = 2$	0.576276	0.578523
$m = 4$, $n = 50$, $s = 2$	0.605534	0.609163
$m = 4$, $n = 50$, $s = 3$	0.518284	0.520264
$m = 5$, $n = 50$, $s = 3$	0.488011	0.502334
$m = 6$, $n = 50$, $s = 4$	0.454793	0.429670

A hyperheuristic is developed on top of the parameterized metaheuristic scheme. It is trained with some problems to generate a hybrid metaheuristic with which satisfactory results are obtained. The metaheuristic obtained is preferable to exact methods for large problems, for which exact methods are not applicable due to their huge execution time. The hyperheuristic presented is a basic one, and a deeper analysis is needed to generate better hybrid metaheuristics.

ACKNOWLEDGEMENTS

J. Aparicio and M. González thank the financial support from the Spanish 'Ministerio de Economa, Industria y Competitividad' (MINECO), the 'Agencia Estatal de Investigacion' and the 'Fondo Europeo de Desarrollo Regional' under grant MTM2016-79765-P (AEI/FEDER, UE). Additionally, D. Giméenez thanks the financial support from the Spanish MINECO, as well as by European Commission FEDER funds, under grant TIN2015-66972-C5-3-R.

REFERENCES

[1] 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.

[2] W. Briec and J. B. Lesourd. Metric Distance Function and Profit: Some Duality Results. *Journal of Optimization Theory and Applications*, 101(1):15-33, 1999.

[3] J. T. Pastor and J. Aparicio. The relevance of DEA benchmarking information and the Least-Distance Measure: Comment. *Mathematical and Computer Modelling*, 52:397-399, 2010.

[4] J. Aparicio and J. T. Pastor. A well-defined efficiency measure for dealing with closest targets in DEA. *Applied Mathematics and Computation*, 219:9142-9154, 2013.

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

[6] J. Aparicio and J. T. Pastor. On how to properly calculate the Euclidean distance-based measure in DEA. *Optimization*, 63(3):421-432, 2014.

[7] J. Aparicio, B. Mahlberg, J. T. Pastor and B. K. Sahoo. Decomposing technical inefficiency using the principle of least action. *European Journal of Operational Research*, 239:776-785, 2014.

[8] 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.

[9] 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*, 390-399, 2014.

[10] M. Gónzalez, J. J. López-Espín, J. Aparicio, D. Giménez and J. T. Pastor. Using Genetic Algorithms for Maximizing Technical Efficiency in Data Envelopment Analysis. In *Proceedings of the International Conference on Computational Science, ICCS 2015, Reykjavk, Iceland, 01-03 June, 2015*, 51:374-383, 2015.

[11] E.-G. Talbi. Hybrid metaheuristics. Studies in computational intelligence. *Springer Verlag, Berlin, Germany*, vol. 434, 2013.

[12] E.-G. Talbi. Metaheuristics: from design to implementation. *John Wiley and Sons*, 2009.

[13] 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.

[14] A. Charnes, W. W. Cooper, B. Golany, L. Seiford and J. Stutz. Foundations of data envelopment analysis for ParetoKoopmans efficient empirical production functions. *Journal of Econometrics*, 30:91-107, 1985.

[15] F. Almeida, D. Giménez, J. J. López-Espín and M. Pérez-Pérez. Parameterized schemes of metaheuristics: basic ideas and applications with Genetic algorithms, Scatter Search and GRASP. *IEEE Transactions on Systems, Man, and Cybernetics*, 43(3):570-586, 2013.