A Parallel Application of Matheuristics in Data Envelopment Analysis

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

[1] Center of Operations Research, Miguel Hernández University of Elche, Alicante, Spain

[2] Department of Computer Science and Systems, University of Murcia, Spain

Abstract. Data Envelopment Analysis (DEA) is a non-parametric methodology for estimating technical efficiency and benchmarking. In general, it is desirable that DEA generates the efficient closest targets as benchmarks for each assessed unit. This may be achieved through the application of the Principle of Least Action. However, the mathematical models associated with this principle are based fundamentally on combinatorial NP-hard problems, difficult to be solved. For this reason, this paper uses a parallel matheuristic algorithm, where metaheuristics and exact methods work together to find optimal solutions. Several parallel schemes are used in the algorithm, being possible for them to be configured at different stages of the algorithm. The main intention is to divide the number of problems to be evaluated in equal groups, so that they are resolved in different threads. The DEA problems to be evaluated in this paper are independent of each other, an indispensable requirement for this algorithm. In addition, taking into account that the main algorithm uses exact methods to solve the mathematical problems, different optimization software has been evaluated to compare their performance when executed in parallel. The method is competitive with exact methods, obtaining fitness close to the optimum with low computational time.

1 Introduction

Data Envelopment Analysis (DEA) is a mathematical programming, non-parametric technique commonly used to measure the relative performance of a set of homogeneous processing units, which use several inputs to produce several outputs. These operating units are usually called Decision Making Units (DMUs) in recognition of their autonomy in setting their input and output levels. Thanks to being a non-parametric technique, DEA does not need to suppose a particular functional form for the production function, technical efficiency may be easily evaluated with multiple inputs and outputs and it also produces relevant benchmarking information from a managerial point of view. In particular, DEA provides both input and output efficient targets, the coordinates of the projection point on the estimated efficient frontier, and represents levels of operation that can make the corresponding inefficient DMU perform efficiently.

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

Traditional DEA measures maximize the total technical effort associated with the evaluated unit in order to reach the efficient frontier. Instead, it seems more natural to assume that inefficient DMUs apply a Principle of Least Action, a well-known law in physics, with the aim of being technically efficient. Otherwise, inefficient units would need to make an extra effort, decreasing inputs and/or increasing outputs, to reach the frontier. The application of this 'natural' Principle of Least Action is linked to the determination of the closest targets on the efficient frontier of the corresponding DEA production possibility set. This drawback of traditional DEA measures has aroused increasing interest among researchers to develop new models capable of yielding achievable targets. Examples are the papers by Briec and Lesourd [2], Pastor and Aparicio [3], Aparicio and Pastor [1,4,5,6] and Aparicio et al. [7].

The application of the Principle of Least Action has been recently studied from a 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, thereby generating initial populations of solutions that satisfy all constraints.

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 contributions of this work include the development of a parallel algorithm that belongs to the class of hybrid metaheuristics [11]. New parallel features have been included in the matheuristic developed in [12]. The algorithm developed is focused on the need to solve multiple simultaneous models. This is due to the DEA problem that concerns us, in which numerous models must be analyzed for each DMU evaluated. The aim is to separate the number of DMUs to be evaluated in the most efficient way in the different available threads. For this, message-passing (MPI) and shared memory (OpenMP) programming have been considered.

The remainder of the paper is organized as follows. In Section 2, a brief introduction to the main notions associated with Data Envelopment Analysis is presented, and existing approaches for determining closest targets are outlined. The working problem is also presented in this section. The parallel algorithm used to generate and improve valid solutions is studied in Section 3. In Section 4, the results of some experiments are summarized. Section 5 concludes the paper and outlines some possible lines of research.

2 Data Envelopment Analysis and the Problem to be Solved

DEA involves the use of mathematical programming to construct a non-parametric piecewise 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 $y_{rj} \geq 0$, $r = 1, \ldots, s$.

One of the models that can be solved by applying the Principle of Least Action in DEA is that by [1]:

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

s.t.

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

The definition and interpretation of the decision variables and constraints of the 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.

3 Parallel Algorithm

In order to improve the performance of the algorithm developed in [12], it has been parallelized at different levels. Some of the objectives to parallelize the algorithm are related to the difficulty of solving problems such as the one proposed

in section 2, where a lot of computing time is taken to find satisfactory solutions. So, the proposed parallelization models are intended to reduce computation time to solve these models and improve the fitness of the solutions obtained. For this, both shared memory (OpenMP) and message-passing (MPI) schemes have been used, since it is possible to use them either separately or in combination. The metaheuristic parallelised in this scheme is based on generating initial solutions to the given model, and improving them with the intention of finding satisfactory solutions. For this, the following scheme is followed: Initialization, Improvement, Selection, Crossing and Diversification. Each parallelization models tries to improve the solution quality or computational time:

- OpenMP: Shared memory parallelization functions have been included within the algorithm's own functions, making them faster. Mainly, these improvements have been included in the initialization function, where it is possible to distribute the generation of solutions between the different threads, and in the improvement and crossing functions, where these tasks can be divided into smaller functions. This model of parallelism has been introduced with the intention of improving the internal loops of the main metaheuristic algorithm. Therefore, in all parts of the algorithm where many models must be evaluated (initialization, improvement, crossing), whenever the evaluation of the models are independent of each other, this type of parallelism is introduced.
- MPI: Message-passing functions have been included over and above metaheuristics. These functions optimize the computing time and the fitness of the final solutions. This level of parallelism is found in a higher level than the shared memoria scheme. In this way, the number of models to be evaluated are divided in equal parts in the different threads. Thus, the computation time is decreased as the number of cores increases.If it is not possible to divide all the DMUs into equal groups, the remaining units will be assigned randomly to the different cores.

The OpenMP scheme can also be used at the same level as the MPI scheme. A comparison between these two schemes is developed in the results of the experiment. This comparison is made to decide where implement each of the schemes. Figure 1 shows how the algorithm works and all the possible configurations.

In the literature, there are several mathematical methods that can solve both mixed integer linear programming (MILP) and linear programming (LP) problems. In the algorithm developed in this paper, a MILP-based decomposition is used to divide the main DEA problem, which is difficult to solve, into smaller LP-type problems that are easier to solve. In this regard, an exact method able of optimally solving numerous LP problems is needed. For this task, two exact methods that work optimally are evaluated to measure their performance in combination with our parallel algorithm. The software packages used are CPLEX and GUROBI.

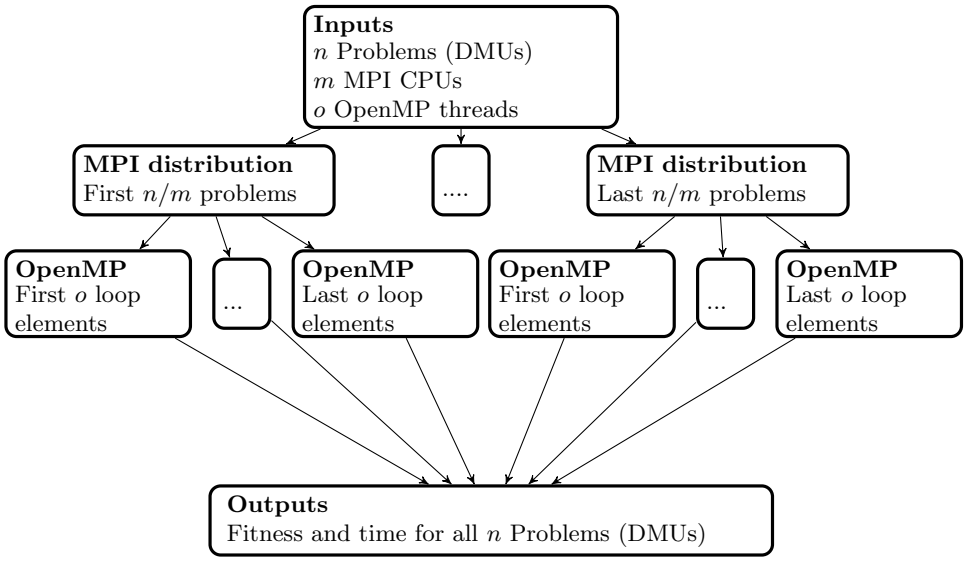

Fig. 1: Generation of metaheuristics and structure of the algorithm.

4 Experimental Results

Experiments were conducted to analyze the effectiveness of the parallel scheme developed. Two different exact methods are compared in terms of fitness and time, working in parallel. The performance of each exact method is also evaluated. For all the experiments, the IBM ILOG CPLEX Optimization Studio (CPLEX) and GUROBI are used. The system used in the experiments is a AMD Phenom II X6 1075T CPU (hexacore) at 3 GHz with 16 GBytes of RAM, private L1 and L2 caches of 64 KBytes and 512 KBytes respectively, and a L3 cache of 6 MBytes shared by all cores. For all the experiments, a standard problem has been taken with the following dimensions: 3 inputs, 2 outputs and 50 DMUs. Regarding the data, in our simulations the m inputs and s outputs of each of the n DMUs are generated at random but taking into account that the production function that governs the production situation is the well-known Cobb-Douglas function [13].

First, we analize the behavior of the different models of parallelism in the main algorithm will be studied. For this, the two available paradigms (MPI and OpenMP) will be used to perform the same function: divide the DMUs to be evaluated in equal groups, thus creating parallel executions of the algorithm, and therefore, of the metaheuristics and exact methods. In this evaluation, while comparing the parallel models, the different exact methods used in this paper are also compared: CPLEX and GUROBI. Therefore, there is going to be a comparison between how the MPI model works with CPLEX and GUROBI and how the OpenMP model also works with CPLEX and GUROBI.

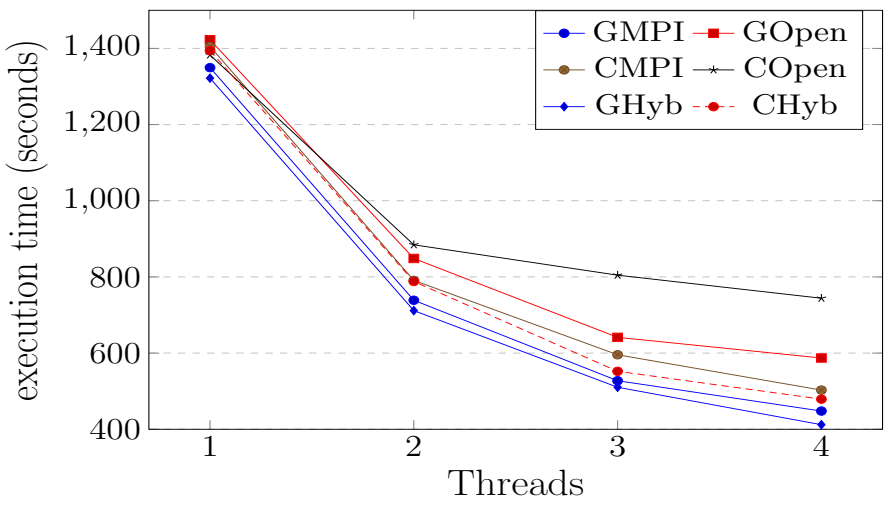

Fig. 2: Comparison of the execution time (in seconds) using different threads with the same problem, for each optimizer and different parallel algorithms: GUROBI with MPI (GMPI) and OpenMP (GOpen), and CPLEX with MPI (CMPI) and OpenMP (COpen).

Figure 2 shows how, as the number of CPUs increases, the computation time decreases. However, it can be seen how, when the number of CPUs grows, the improvement in the computational time decreases, becoming lower each time . This is due to the fact that, depending on the number of DMUs to be evaluated, from certain divisions, the resulting DMU groups have practically the same number of elements, therefore the computation time is smaller, but similar. In addition, this figure also shows that the exact methods chosen work well in parallel, being possible to execute multiple instances of them at the same time. It must be taken into account that the files in which the problems to be solved are written must have different names to avoid any confusion during the execution. This experiment shows that the MPI message-passing programming model works better than the OpenMP shared memory model in this first step of the algorithm. Therefore reaffirming what was stated in section 3.

Another important parameter to analyze, is the objective value achieved by each optimizer. For this, it is necessary to emphasize that the initial generation of solutions is done in a random way, so that the initial solutions generated by an optimizer are not the same as for the other, since the executions are independent of each other. To evaluate the results, 20 executions have been made for each optimizer, making 5 for each set of threads. The results shown in graph 3 show the average values of all the executions for each set of threads. Analyzing the obtained values, it can be affirmed that, after performing several experiments, the CPLEX optimizer obtains better fitness values in all the sets. As mentioned

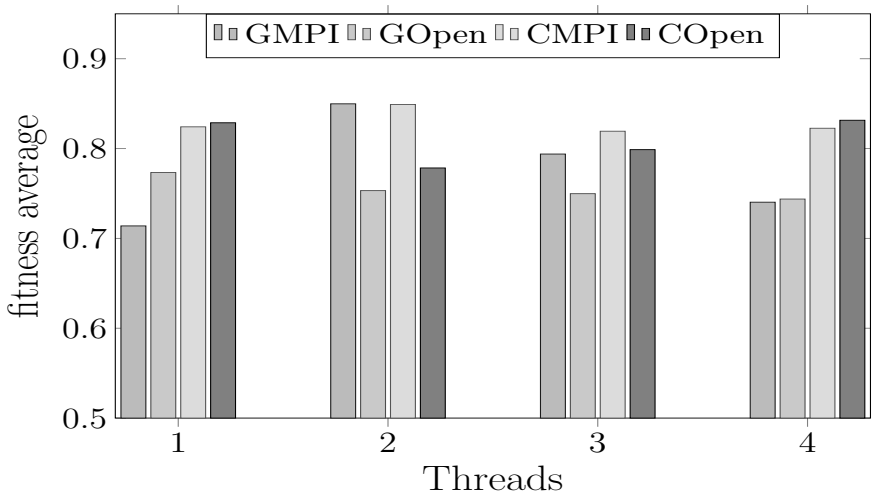

Fig. 3: Objective values obtained for each optimizer and different parallel algorithms: GUROBI with MPI (GMPI) and OpenMP (GOpen), and CPLEX with MPI (CMPI) and OpenMP (COpen). The values are the average of 5 executions.

before, the generation of initial solutions is randomly created and is independent for each optimizer. Even so, after a statistical study, CPLEX obtained better fitness values in this experiment.

5 Conclusions and Future Works

The application of the Principle of Least Action in DEA 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 inadequate approaches, associated with combinatorial NP-hard problems.

Parallel algorithms are good solutions for solving these kind of problems. This is because a high number of independent problems must be solved, and those problems can be divided and solved by different threads. Furthermore, several optimized software packages can be used, but not all of them work well in parallel. To improve performance, is necessary to include optimizers that can be executed in different instances. For that, CPLEX and GUROBI are tools for these tasks. The parallel algorithm proposed in these paper works in an optimal way with both of these optimizers and with different parallel paradigms.

For future work, we propose the use of other free optimizers, and to check their performance compared with the most powerful optimizers in the market. In addition, it is also desirable to incorporate improvements in parallelism, so that the internal functions of the metaheuristic algorithm can be executed more quickly and accurately.

Acknowledgements

J. Aparicio and M. González thank the financial support from the Spanish 'Ministerio de Economía, Industria y Competitividad' (MINECO), the 'Agencia Estatal de Investigacion' and the 'Fondo Europeo de Desarrollo Regional' under grant MTM2016-79765-P (AEI/FEDER, UE).

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. Closest 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, Reykjavík, 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. M.González, J.J. López Espín, J. Aparicio, D. Giménez and E. Talbi. A Parameterized Scheme of Metaheuristics with Exact Methods for determining the Principle of Least Action in Data Envelopment Analysis. *Program of the 2017 IEEE Congress on Evolutionary Computation.*
13. C.W. Cobb, P.H. Douglas A theory of production. *The American Economic Review*, 18(1):139-165, 1928.