# On the performance of the hybridisation between migrating birds optimisation variants and differential evolution for large scale continuous problems

Eduardo Segredo[a,b,*], Eduardo Lalla-Ruiz[c], Emma Hart[a], Stefan Voß[c,d]

[a]*School of Computing, Edinburgh Napier University, 10 Colinton Road, Edinburgh, EH10 5DT, Scotland, United Kingdom*
[b]*Departamento de Ingeniería Informática y de Sistemas, Universidad de La Laguna, San Cristóbal de La Laguna, Spain*
[c]*Institute of Information Systems, University of Hamburg, Hamburg, Germany*
[d]*Escuela de Ingeniería Industrial, Pontificia Universidad Católica de Valparaíso, Valparaíso, Chile*

**Abstract**

*Migrating Birds Optimisation* (MBO) is a nature-inspired approach which has been shown to be very effective when solving a variety of combinatorial optimisation problems. More recently, an adaptation of the algorithm has been proposed that enables it to deal with continuous search spaces. We extend this work in two ways. Firstly, a novel leader replacement strategy is proposed to counter the slow convergence of the existing MBO algorithms due to low selection pressure. Secondly, MBO is hybridised with *adaptive* neighbourhood operators borrowed from *Differential Evolution* (DE) that promote exploration and exploitation. The new variants are tested on two sets of continuous large scale optimisation problems. Results show that MBO variants using adaptive, *exploration*-based operators outperform DE on the CEC benchmark suite with 1000 variables. Further experiments on a second suite of 19 problems show that MBO variants outperform DE on 90% of these test-cases.

*Keywords:* migrating birds optimization; differential evolution; large scale continuous problem; global optimization; leader replacement strategy; continuous neighborhood search

## 1. Introduction

Due to their hardness and practical relevance, complex and challenging problems in fields such as engineering, business, economy, and biology require solutions to be produced in a reasonable computational time. Much scientific research is directed towards gaining knowledge and insights into methods and ways to solve hard problems. Many of those problems can only be modelled by continuous variables with non-linear objective functions. Traditional exact methods are not suitable when dealing with such cases, leading to the need of other types of approaches. This gives rise to the use of heuristic algorithms, and particularly, of meta-heuristics, specially in those cases where users, as stated by Storn & Price (1997), require approaches capable of handling non-differentiable, non-linear, and multi-modal functions.

A popular option for addressing optimisation problems are nature-inspired approaches. Those types of algorithms are inspired, for example, by natural events, animal collective behaviour, and swarm intelligence, where their components extracted from the nature are translated into optimisation and used for solving complex problems (Yang, 2008; Parpinelli & Lopes, 2011; Xing & Gao, 2014). In particular, natural systems exhibiting swarming behaviours have received wide attention from the research community. For example, in the case of birds, flock members of several species profit from sharing information or cooperating one with each other by travelling and living together (Conradt, 2012; Sumpter, 2006). Recently, Duman et al.

*Corresponding author
*Email addresses:* e.segredo@napier.ac.uk (Eduardo Segredo), eduardo.lalla-ruiz@uni-hamburg.de (Eduardo Lalla-Ruiz), e.hart@napier.ac.uk (Emma Hart), stefan.voss@uni-hamburg.de (Stefan Voß)

(2012) proposed a population-based algorithm inspired on the V-formation flight of migratory birds for combinatorial optimisation, which was termed as *Migrating Birds Optimisation* (MBO). The algorithm exploits the concepts of cooperative search by including the share of information among individuals that are connected via the previously mentioned V-formation. As discussed below, in the related literature, MBO has been successfully applied to a wide range of combinatorial problems. In contrast, less attention has been paid to understanding how best MBO can be adapted to continuous optimisation domains. Although a few approaches exist, they have weaknesses in that they have not provided competitive results taking into account the particular test suites they have been applied. Furthermore, those proposals have only been applied to continuous problems with low dimensionalities.

Another well-known solving methodology for addressing optimisation problems is *Evolutionary Computation* (EC) (Bäck et al., 2000). Its main goal is to study, develop, and analyse algorithms following the biological notion of evolution within the Darwinian principles. Its key features include population-based collective learning processes, self-adaptation, and robustness. Within EC, *Differential Evolution* (DE) (Storn & Price, 1997) is a popular global search algorithm that has rapidly grown in both the number of applications and studies devoted to the analysis of its performance. Its simplicity, flexibility, and easy implementation has led to DE being one of the most used and powerful tools when dealing with global continuous optimisation.

The structure of DE contains a narrow set of exploration moves (Neri & Tirronen, 2010), which gives rise to a plethora of modifications that include introducing additional components into DE and/or making modifications within the DE structure. A taxonomy was proposed by Neri & Tirronen (2010) that classifies hybrid DE schemes into two classes: DE *integrating an extra component* and *modified structures of* DE. Nevertheless, to the best of our knowledge, there is currently no research that analyses the influence of the over-arching schemes that govern the overall control of DE. Thus, we propose an extension of the existing taxonomy with a new class of modified versions of DE providing MBO approaches as exemplary elements withing this class:

- **External ruling structures over** DE. This includes those algorithms where a given structure or scheme determining the relationship and selection of the individuals of the population is defined. As discussed later, these types of algorithms *a priori* define how individuals share information among them.

This article extends MBO in order to adapt it to large scale continuous optimisation problems. It takes advantage of the wide range of *neighbourhood operators* that have been well studied in DE to develop hybridised MBO algorithms that either bias the search towards exploration or exploitation. Secondly, it introduces a novel *replacement* operator that addresses an existing weakness in the standard MBO algorithm that tends to lead to slow convergence due to low selection pressure. This is of particular importance in solving large scale problems due to the size of the search space. Specifically the contributions of the paper are as follows:

- Novel MBO variants that incorporate state-of-the art adaptive neighbour generation operators.
- A novel elitist leader replacement strategy within MBO that provides high selection-pressure, and is therefore more suited to large search spaces.
- Extensive empirical investigation using 34 functions with 1000 variables from two different test-suites to evaluate the relative performance of the MBO variants in comparison to state-of-the-art DE variants.
- A detailed statistical analysis of the performance that highlights the superiority of MBO variants that use *exploration* operators compared to DE on the majority of instances tested.

*Large Scale Global Optimisation* (LSGO) has been extensively studied in the recent literature, and represents a very active research line in continuous optimisation (LaTorre et al., 2015; Mahdavi et al., 2015). The term *large scale problems* refers to those problems with a large dimensionality, typically more than 100 decision variables (Kazimipour et al., 2013). We note that the best performing approaches specifically proposed to deal with large scale problems usually combine multiple algorithms to solve an instance. One of the best examples is the *Multiple Offspring Sampling* (MOS) (LaTorre et al., 2013) framework, which was

the winner approach in the most recent editions of the competition on LSGO organised at the *IEEE Congress on Evolutionary Computation*. This framework allows multiple types of algorithms to be applied during the search. The specific algorithm applied at the current moment of the optimisation process is dynamically selected by means of a quality measure, such as the average fitness increment of the newly created individuals. The particular version of MOS that won the competition consisted of a hybrid approach combining a GA, a direct search method and a local search procedure. The second best performing algorithm of the competition was an *Iterative Hybrid Differential Evolution with Local Search* (IHDELS) (Molina & Herrera, 2015). It iteratively applies a DE variant and a local search selected from among two different options.

In this paper, our goal is not to outperform those state-of-the-art approaches which rely heavily on algorithm portfolios: a direct comparison to these hybrid approaches would not be fair. Instead, the goal is to develop new solvers that benefit from the hybridisation of MBO and DE approaches, and can ultimately be used as additional components of those state-of-the-art portfolio-based schemes. At this point, it is worth mentioning that obtaining the source code of those state-of-the-art optimisers is somewhat difficult. For instance, there is one version of MOS available [1], but it is older than the particular version applied to win the competition. As a result, it is quite difficult to analyse whether the incorporation of new components to those state-of-the-art schemes would provide any benefit. Bearing the above in mind, the incorporation of our MBO-based algorithmic proposals into state-of-the-art approaches is out of the scope of the current work. This issue, however, will be addressed as a future line of work.

The remainder of this paper is structured as follows. Section 2 goes over those works related to the contribution of this paper. Afterwards, Section 3 describes our novel MBO proposals for continuous spaces and how they can be applied when tackling optimisation problems. Then, the experimental evaluation carried out in this work, as well as their discussion, are shown in Section 4. Finally, Section 5 presents the main conclusions extracted from the work and suggests several directions for further research.

## 2. Literature review

This section is devoted to revise the literature related to the application of MBO to both combinatorial and continuous optimisation (Section 2.1), and to the hybridisation of DE with other algorithmic approaches (Section 2.2).

### *2.1. Migrating birds optimisation for combinatorial and continuous problems*

MBO was first proposed by Duman et al. (2012) as a meta-heuristic inspired by the V-formation flight of migrating birds. Since its introduction, it has been applied to a wide range of applications, particularly in combinatorial optimisation. Its performance was assessed by means of the well-known *Quadratic Assignment Problem* (QAP) using several instances belonging to the QAPLIB (Burkard et al., 1997), as well as to novel problem instances belonging to the *Printed Circuit Board* (PCB) problem. Results showed that MBO behaved better than other population-based approaches such as *Particle Swarm Optimisation* (PSO), *Genetic Algorithm* (GA), and *Scatter Search* (SS), among others. Other applications to combinatorial problems followed, the most important of which are summarised below.

Duman & Elikucuk (2013) applied MBO to solve the *Credit Card Detection Problem* and investigated the impact of sharing information among the individuals of MBO instead of discarding that information, finding that differences between sharing information or discarding it were not significant.

Shen et al. (2015) addressed the *University Course Timetabling Problem* (UCTP) by means of MBO and a modified version called m-MBO. The variant m-MBO included some modifications such as the hybridisation with an *Iterated Local Search* (ILS), different leader replacement and information sharing mechanisms, and the removal of tour iterations. Results indicated that m-MBO was able to outpace the original MBO.

In other work, Soto et al. (2016) developed an algorithm based on MBO for the *Machine-Part Cell Formation Problem*. Their approach was able to provide the optimal solution in all the considered instances. Additionally, their specific implementation incorporated parallel procedures in order to enhance several

[1] The source code of that MOS version can be downloaded through the following URL: `http://sci2s.ugr.es/EAMHCO#Software`.

sorting processes carried out in the algorithm, thus allowing execution times to be reduced in comparison to the sequential sorting algorithm.

An improved MBO algorithm to minimise the total flow-time for a hybrid *Flow-shop Scheduling Problem* was proposed by Pan & Dong (2014). Together with MBO, they also introduced an enhanced version that combined a diversified initialisation method, a mixed neighbourhood in order to provide neighbour solutions, a leaping mechanism for escaping from suboptimal solutions, a problem-specific meta-heuristic, and a local search procedure. Both MBO schemes performed better than the other algorithms taken into consideration for comparison purposes.

The *Dynamic Berth Allocation Problem* (DBAP) and the *Quay Crane Scheduling Problem* (QCSP) were addressed by efficient approaches based on MBO by Lalla-Ruiz et al. (2015). The proposed algorithms were able to provide high-quality solutions by means of short computational times. In the same work, methods for improving the final solutions attained by MBO were also introduced. Those methods, however, did not provide a significant improvement of the quality of the solutions obtained by the MBO schemes.

Finally, Lalla-Ruiz et al. (2017) introduced a novel meta-heuristic termed as *Multi-Leader Migrating Birds Optimisation* (MMBO). This new population-based meta-heuristic is inspired by MBO and enforces the algorithmic translation of the nature migratory event at hand by taking into account specialised works studying birds' flight behaviour. To assess the performance of MMBO and its contribution with respect to MBO, the same instances of the QAP addressed in Duman et al. (2012) were considered. The experimental evaluation showed that MMBO was able to provide high-quality solutions and outperform MBO for all the instances taken into account, thus demonstrating the significant contribution of MMBO with respect to MBO.

Concerning continuous optimisation, only a few papers are found in the related literature, describing new MBO variants. Makas & Yumusak (2013) proposed two different versions of MBO. The first one was a hybridisation with an *Artificial Bee Colony* (ABC) cooperative search, while the second one used a linearly decreasing generation procedure. In both cases, the way in that neighbours were generated was not explained, and therefore, we cannot designate both approaches as continuous MBO variants.

Later, Alkaya et al. (2014) presented, as far as we know, the first MBO approach for dealing with continuous functions. In this case, a neighbour generation operator based on hyper-spheres was considered. However, detailed information about its implementation was not given. The algorithm was applied to a set of functions with low dimensionalities: 2, 10, and 30 decision variables. Although in the original experimental evaluation it was not compared to other algorithms, the algorithm was entered in the 2014 *International Conference on Swarm Intelligence* (ICSI) competition on single-objective optimisation, and consequently, a comparison was reported in Tan et al. (2015). MBO applied together with the neighbourhood operator based on hyper-spheres was not able to provide competitive solutions with respect to other approaches, such as DE and PSO, among others. Oz (2017) proposed an MBO algorithm to address the multi-objective version of the task allocation problem. The proposed solution approach was compared to exact and approximate algorithms. MBO was able to provide the optimal solution in most of the cases. Niroomand et al. (2015) developed a modified MBO algorithm for tackling the closed loop layout with exact distances in flexible manufacturing systems. The reported computational results indicated that MBO outperformed the state-of-the-art approaches. Alkaya & Algin (2015) proposed an MBO variant for the obstacle neutralisation providing competitive results.

Finally, in Lalla-Ruiz et al. (2016), the authors of the current work introduced hybrid approaches combining, on the one hand, MBO and MMBO, and on the other hand, a neighbourhood operator based on DE, with the aim of enabling the former to be applicable to continuous decision spaces. This was the first time that MMBO was proposed to deal with continuous problems, and more particularly with large scale ones, as well as the first time that MBO was applied to large scale continuous optimisation. The experimental evaluation showed that MBO was able to provide statistically similar results, and even better for some test cases, than those attained by DE executed separately. Furthermore, MBO showed a better performance than MMBO for the set of problems considered. Considering the promising results achieved by MBO combined with a neighbourhood operator based on DE, the contributions of the current work in this realm are:

- Novel neighbourhood operators based on two adaptive DE variants, which promote either exploration or exploitation in the whole approach.

- A novel elitist leader replacement mechanism to be integrated into MBO-based schemes aimed to increase their selection pressure.
- A wide and detailed experimental evaluation considering two different test suites.

### *2.2. Hybrid approaches based on differential evolution*

Since we propose a hybridisation between MBO and DE, it is worth exploring the literature to look for previous research related to hybrid approaches that make use of DE in a similar way to the one we propose, i.e., DE is used for generating neighbour solutions while MBO entails an external ruling structure over DE. Although in the following we describe those papers that are most related to our contributions, the reader is directed to detailed surveys with an in-depth perspective of the state-of- the-art concerning DE (Das et al., 2016; Lozano & García-Martínez, 2010; Mahdavi et al., 2015).

Das et al. (2008) proposed PSO-DV, a combination between PSO and DE, which incorporates the mutant vector generation strategy from DE into the PSO velocity update process. They tested PSO-DV on a suite of unconstrained benchmark functions with a maximum number of 30 decision variables. The proposed hybrid approach led to a better global search algorithm than the isolated versions of PSO and DE.

Similarly, Omran et al. (2009) proposed a hybridisation of a variant of PSO, called *Bare Bones* PSO, and DE. This hybrid scheme was termed as BBDE, and applied the mutant vector generation strategy of DE to the attractor associated to each particle. The computational experiments were carried out taking into account a set of unconstrained benchmark functions and image classification problems. Results showed that, while DE performed better than BBDE for uni-modal benchmark functions, in the multi-modal case, BBDE exhibited a better performance.

In another work, Pholdee & Bureerat (2013) proposed a hybrid algorithm that incorporated the trial vector generation scheme of DE in a gradient based *Real-Coded Population-Based Incremental Learning* (RCPBIL) algorithm. Their computational experience was performed over multi-objective unconstrained and constrained functions, as well as optimisation design problems. Although RCPBIL demonstrated to be inefficient for solving some of the test cases, after the incorporation of the trial vector generation strategy of DE, its performance significantly increased. An interesting observation in regard to this work is that the authors tested their proposal with improved parameter settings, raising the question regarding the use of mechanisms for adapting the parameters of a given approach.

Ghosh et al. (2012) proposed a hybridisation between a *Covariance Matrix Adaptation Evolutionary Strategy* (CMA-ES) and DE by incorporating the trial vector generation strategy and the selection operator of the latter into the structure of the former, which uses the adaptation of the covariance matrix with the aim of identifying the function landscape. The experimental evaluation was performed over functions with a maximum number of 50 decision variables, and reported that the proposed hybridisation behaved better than CMA-ES and DE executed independently.

Finally, Stanarevic (2012) proposed a hybridisation between an ABC scheme and the mutant vector generation strategy of DE. The experimental evaluation, which was performed over a set of functions with 10, 100, and 500 dimensions, showed that better results were attained by the hybrid proposal.

Despite the fact that there is existing research that incorporates some DE components into some algorithmic frameworks, we are not aware on any previous work enforcing a fixed relationship among individuals of the population, such as provided with MBO. Furthermore, it seems that hybrid schemes based on DE achieve a better performance in comparison to DE considered as an isolated algorithm. Based on this, the contributions of this paper with respect to the current literature are:

- Proposing and assessing the incorporation of a fixed relationship among individuals, as the one considered by MBO, into DE.
- Addressing the research question of whether a hybrid scheme combining MBO and DE is able to improve the performance of DE executed as an independent approach.

**Algorithm 1** Pseudocode of Migrating Birds Optimisation (MBO)

**Require:** $n$, $K$, $m$, $k$, and $x$
1: Generate $n$ initial individuals at random and arbitrarily place them on a logical V-formation
2: $g = n$
3: **while** $(g < K)$ **do**
4:  **for** $(j = 1 : m)$ **do**
5:  Try to improve the leader individual through the generation of $k$ neighbours starting from it
6:  $g = g + k$
7:  **for all** (follower individual $s$ in the population) **do**
8:  Try to improve $s$ through the generation of $k - x$ neighbours starting from it, and the best unused $x$ neighbours of its immediate predecessor attending to the current V-formation
9:  $g = g + (k - x)$
10:  **end for**
11:  **end for**
12:  Move the leader to the end of the V-formation and forward one of its immediate followers as the leader
13: **end while**
14: **return** the best individual in the population

## 3. Migrating birds optimisation algorithm for continuous search spaces

As previously mentioned, MBO is a nature-inspired algorithm based on the migration flow of birds. It considers a *flock* (population) of *birds* (individuals), that are aligned in a V-formation during the flight (search). Taking into account that formation, the first individual is denoted as the *leader*. Remaining individuals are denoted as *followers*. Individuals maintain a cooperative relationship among them by means of sharing information. This information is unidirectionally shared in the form of individuals transferred from the neighbourhood of a particular individual to the neighbourhoods of its immediate followers. Information sharing starts from the leader individual and moves toward its followers by considering the V-formation as a scheme determining which individuals share information with whom (see Figure 1). At this point, we should note that, essentially, the V-formation is an arbitrary spatial structure imposed on the population, and consequently, it is not related to the problem being solved. Once the population of solutions is generated, those solutions are connected, in terms of information sharing, by following the said V-formation. As a result, the V-formation remains as a characteristic of MBO for restricting and managing the way information is shared among individuals. The parameters of the original MBO scheme are denoted as follows:

- Number of individuals in the population ($n$)
- Maximum number of individuals generated or evaluated ($K$)
- Number of iterations carried out before the leader is updated ($m$)
- Neighbourhood size ($k$)
- Number of neighbours to be shared among a particular individual and its followers ($x$)

Algorithm 1 describes the operation of MBO (Duman et al., 2012). The first step consists of randomly generating $n$ individuals as the initial population and placing them arbitrarily on a logical V-formation (step 1). That V-formation is depicted in Figure 1. As it can be observed, the V-formation establishes that every individual has a unique predecessor, with the exception of the individual located at the front of the V-formation, which has no predecessor. For instance, by observing Figure 1 (left-hand side), the immediate predecessor of individuals $ind_2$ and $ind_3$ in the V-formation would be individual $ind_1$, while the immediate predecessor of individual $ind_7$ would be individual $ind_5$. At this point, we should note that positions of individuals in the V-formation change depending on the current moment of the search process by applying a particular leader replacement strategy periodically, as it will be described later.

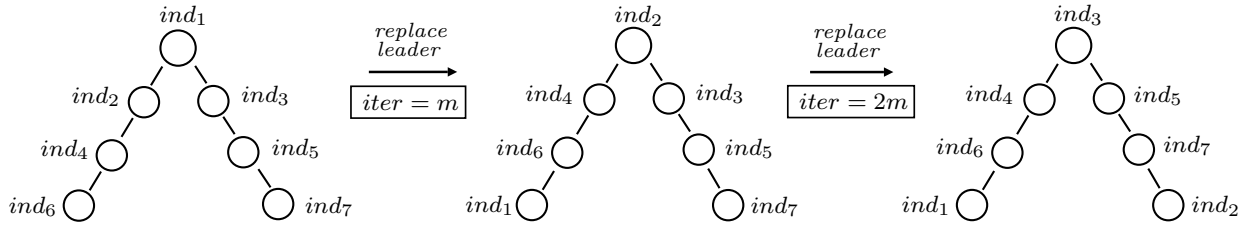


Figure 1: Operation of the leader replacement scheme in the original implementation of MBO

Once the population is initialised, the current number of individuals generated during the execution, i.e., $g$, is initially set to $n$ (step 2). During the search process, firstly, $k$ neighbours are generated starting from the leader individual by using a particular neighbour generation strategy (step 5). In case the leader individual is improved by its best neighbour, in terms of the objective function value, the latter replaces the former in the population. The way new neighbours are generated will be described in Section 3.2.

Then, for each follower individual $s$, the next steps are carried out by considering the order established by the V-formation (steps 7–10). In the first place, $k-x$ neighbours are generated starting from $s$. Afterwards, the neighbourhood of $s$ receives the fittest (i.e. best) valued unused $x$ neighbours of its immediate predecessor in the V-formation. Finally, if $s$ is improved by its fittest neighbour, then the former is replaced by the latter in the population. The unused neighbours of a particular individual $\gamma$ are those individuals belonging to its neighbourhood that are worse than $\gamma$ regarding the objective function value. Therefore, they are those individuals that have not been able to replace $\gamma$ in the population. Note that the leader is the unique individual that shares neighbours with its two immediate followers, located at left and right wings of the V-formation. Each of the remaining individuals only shares neighbours with its immediate follower, with the exception of the last two individuals located at the end of the left and right wings of the V-formation, which do not share neighbours with any other individual in the population. Figure 1 depicts the V-formation structure for a population of seven individuals. As it can be observed, in the first V-formation (left-hand side), individual $ind_1$ is connected to individuals $ind_2$ and $ind_3$, while individual $ind_2$ is connected to individual $ind_4$ and individual $ind_3$ to individual $ind_5$, and so on. Considering that V-formation, individual $ind_2$, for instance, generates neighbour solutions by means of a given neighbourhood structure and its fittest unused neighbours are shared with its immediate follower, which is individual $ind_4$ in this case.

The V-formation is maintained until a prefixed number of iterations $m$ is performed (step 4). Following this step, the current leader becomes the last individual in the V-formation and one of its immediate followers located at the left or right wing of the V-formation becomes the new leader (step 12). The above steps are executed until a maximum number of individuals $K$ is generated (step 3).

Since we propose a novel leader replacement strategy, the original leader replacement strategy incorporated into MBO (step 12 of Algorithm 1) is first explained in detail. Every time the leader has to be updated, one of its immediate followers located at the left or right wing of the V-formation becomes the new leader, and the previous leader is moved to the end of the left or right wing, respectively. The remaining individuals belonging to the left or right wing are therefore shifted towards the front of the V-formation. With each leader update, the algorithm alternates left and right wings, starting the run from one of both arbitrarily.

Figure 1 illustrates this procedure. Two subsequent leader replacements produced at iterations $m$ and $2 \cdot m$ are depicted. In the first replacement, the left immediate follower of the leader ($ind_2$) becomes the new leader. Afterwards, individuals $ind_4$ and $ind_6$ are shifted towards the front of the V-formation. Finally, the previous leader ($ind_1$) is moved to the end of the left wing. In the second replacement, the operation of the strategy is exactly the same, but considers the right wing of the V-formation instead of the left one. This replacement scheme is executed every $m$ iterations, by alternating left and right wings, until the stopping criterion of MBO is met.

### 3.1. Elitist leader replacement strategy with follower cloning

In this work, we additionally propose a novel leader replacement strategy to be incorporated into MBO-based schemes. Our hypothesis is that the original leader replacement scheme of MBO does not provide a high enough selection pressure, and therefore, the convergence speed to promising solutions is not fast enough when dealing with certain types of problems. In order to increase the selection pressure of MBO-based schemes, our novel leader replacement scheme incorporates both an elitist selection mechanism and cloning of some of the best individuals in the population. As a result of these modifications, it is expected that the whole optimisation scheme is able to move the balance towards exploitation, thus increasing the convergence speed to promising solutions. Hence, instead of the original leader replacement scheme of MBO shown in step 12 of Algorithm 1, we apply the novel leader replacement scheme described in Algorithm 2.

The new pseudocode is shown in Algorithm 2. As can be observed, the proposed replacement scheme starts by determining the fittest immediate successor of the current leader (steps 1–4). The method *getLeaderFollower*, provides the immediate follower of the current leader located at the left or right wing of the V-formation. It receives the selection of the particular wing as a parameter. Once the fittest immediate follower is determined, it is cloned through the method *cloneLeaderFollower*, which receives as a parameter the wing of the V-formation where the follower to be cloned is placed. Afterwards, the current leader is saved as the previous one (step 6), and the clone becomes the new leader of the formation (step 7).

The previous leader is then moved to the end of the wing where the fittest immediate follower was cloned, but only in the case that a previous leader is fitter than the individual situated at the end of that wing (steps 8–10). Otherwise, the previous leader is discarded. In order to achieve this, methods *getWingLastIndividual* and *setWingLastIndividual* are used. The former obtains the individual placed at the end of the left or right wing of the V-formation (step 8), while the latter allows a particular individual to be located at the end of the left or right wing (step 9). Both methods receive a parameter indicating the particular wing of the V-formation where they must work. Furthermore, the method *setWingLastIndividual* also receives the individual to be placed at the end of the corresponding wing as a second parameter.

The last step consists of shifting those individuals situated at the opposite wing to that from which the fittest immediate follower was cloned (step 11). The immediate follower of the leader that was not cloned is penalised by moving it to the end of its corresponding wing. Hence, the remaining individuals located at that wing are shifted towards the front of the V-formation. At this point, we note that cloning and shifting wings of our leader replacement scheme alternate during the whole run of MBO, as the fittest immediate follower of the current leader might be placed at the left or the right wings. Finally, it is worth pointing out that the computational complexity of a particular MBO-based approach incorporating our novel leader replacement mechanism does not increase in comparison to the application of the original replacement scheme.

In order to clarify the operation of the novel leader replacement scheme for MBO, the above procedure is described in Figure 2 with two different examples. The first example (top of Figure 2) shows the V-formation before (left-hand side) and after (right-hand side) our novel leader replacement procedure is applied. Considering a minimisation problem, the left immediate follower of the current leader ($ind_2$) is

**Algorithm 2** Pseudocode of the elitist leader replacement strategy with follower cloning

```
1: if (getLeaderFollower(left).isFitterThan(getLeaderFollower(right))) then
2:    cloningWing = left; shiftingWing = right
3: else
4:    cloningWing = right; shiftingWing = left
5: end if
6: prevLeader = currentLeader
7: currentLeader = cloneLeaderFollower(cloningWing)
8: if (prevLeader.isFitterThan(getWingLastIndividual(cloningWing))) then
9:    setWingLastIndividual(cloningWing, prevLeader)
10: end if
11: shiftFormation(shiftingWing)
```

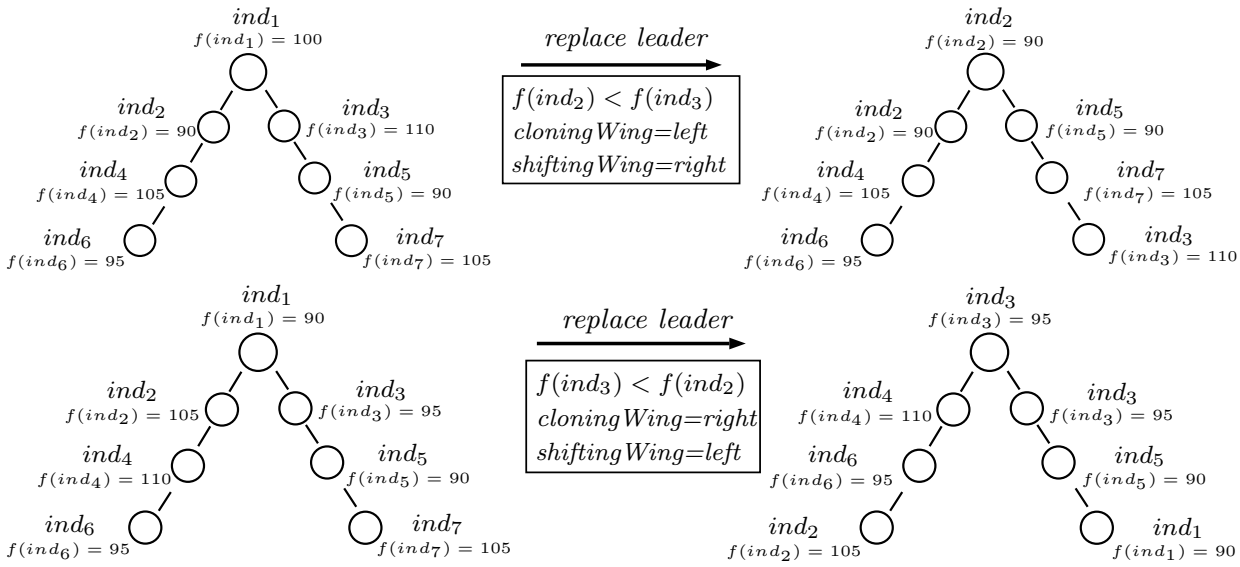


Figure 2: Operation of our novel elitist leader replacement strategy with follower cloning

fitter than the right immediate follower ($ind_3$). As a result, the cloning wing is the left one and the shifting wing is the right one. In the cloning wing, the clone of $ind_2$ becomes the new leader. Moreover, as the individual placed at the end of the left wing ($ind_6$) is fitter than the previous leader ($ind_1$), the latter is discarded. In the shifting wing, the immediate follower of the leader that was not cloned ($ind_3$) is penalised by moving it to the end of the V-formation. Individuals $ind_5$ and $ind_7$ are therefore shifted towards the front of the V-formation. In the second example (bottom of Figure 2) the operation is similar. In this case, the right immediate follower of the current leader ($ind_3$) is fitter than the other immediate follower ($ind_2$), and consequently, the former is the individual to be cloned. Furthermore, the cloning wing is the right one and the shifting wing is the left one. In the cloning wing, the clone of $ind_3$ becomes the new leader. The main difference in comparison to the first example is that the individual placed at the end of the right wing ($ind_7$) is worse than the previous leader ($ind_1$). The previous leader $ind_1$ thus replaces $ind_7$ in the population. In the shifting wing, the immediate follower of the leader that was not cloned ($ind_2$) is penalised by moving it to the end of the V-formation. Hence, individuals $ind_4$ and $ind_6$ are shifted towards the front.

### *3.2. Neighbour generation operators based on differential evolution*

This section is devoted to describing the different variants of DE used to define the neighbourhood operators for MBO, thus allowing the latter to be enabled for continuous search spaces. As already mentioned, DE is a well-known search method which was particularly proposed for global continuous optimisation (Storn & Price, 1997), and has been shown to obtain high performance when dealing with these types of problems (Das et al., 2016). Hence, our hypothesis is that a hybridisation combining the features of MBO for sharing information among individuals placed in a structured population and a neighbour generation operator based on DE, might provide better results when solving continuous problems than those obtained by DE considered as an independent scheme.

Algorithm 3 shows the operation of DE. A vector $\vec{X} = [x_1, x_2, \ldots, x_i, \ldots, x_D]$ with $D$ real-valued decision variables or dimensions $x_i$ is used in order to encode individuals. As we previously mentioned, in the related literature, the term *large scale problems* is used to refer to those optimisation problems with a large dimensionality, typically $D > 100$. In the case of box-constrained problems, the feasible region $\Omega$ is defined by $\Omega = \prod_{i=1}^{D}[a_i, b_i]$, where $a_i$ and $b_i$ represent the lower and upper bounds of variable $x_i$, respectively.

Regarding the most widely used nomenclature for DE (Storn & Price, 1997), i.e., DE/$x$/$y$/$z$, where $x$ is the individual to be mutated, $y$ defines the number of difference vectors used, and $z$ indicates the crossover strat-

**Algorithm 3** Pseudocode of Differential Evolution (DE)

**Require:** $n$, $F$, $CR$

1: Generate $n$ individuals or target vectors at random as the initial population
2: **while** (stopping criterion is not satisfied) **do**
3:  **for** $(j = 1 : n)$ **do**
4:  The individual $\vec{X}_j$ belonging to the current population is referred to as the target vector
5:  Obtain a mutant vector $\vec{V}_j$ through the mutant generation strategy
6:  Combine $\vec{X}_j$ and $\vec{V}_j$ through the crossover operator to get the trial vector $\vec{U}_j$
7:  Repair infeasible values of $\vec{U}_j$
8:  Select the fittest individual between $\vec{X}_j$ and $\vec{U}_j$ as the survivor for next generation
9:  **end for**
10: **end while**
11: **return** the best individual in the population

egy, our neighbourhood operators are based on the schemes DE/*rand/1/bin* and DE/*current-to-p-best/1/bin*, where the term *bin* refers to the *binomial crossover*. The operation of both aforementioned DE variants, as well as the reasons why we have selected them, are described in the following.

*3.2.1. An explorative neighbour generation operator based on* DE/*rand/1/bin*

The rationale behind the selection of this DE version is twofold. Firstly, a configuration of this DE variant, from among a set with more than 80 different parameterisations, was able to provide the best performance for a wide range of functions belonging to one of the test suites tackled in this work (Kazimipour et al., 2014). Secondly, in past research (Segura et al., 2015), it showed to be the best performing DE variant when dealing with a set of scalable continuous problems. The operation of this DE version, which is shown in Algorithm 3, and consequently of our novel neighbourhood operator, is explained as follows. At this point, it is important to remark that our novel neighbour generation operators for MBO only consist of steps 4–7 of Algorithm 3, i.e., the *trial vector generation strategy* of DE, which will be described in the following lines.

Starting from a particular individual $\vec{X}_{j=1...n}$ of the current population, denoted as the *target vector* in DE terminology (step 4), and $n$ being the population size, a new neighbour is generated by means of the next steps. First, the *mutant generation strategy* rand/1 is applied for obtaining a *mutant vector* $\vec{V}_j$ (step 5). This procedure is described in Equation 1. We should note that $r_1$, $r_2$, and $r_3$ are mutually exclusive integers chosen at random from the range $[1, n]$, with all of them being also different from the index $j$. The fact that only a random procedure is used to select the individuals considered by the mutant generation strategy is the main reason why this DE variant promotes exploration rather than exploitation. In addition, the *mutation scale factor*, denoted by $F$, also allows the exploration and exploitation abilities of DE to be balanced. Small $F$ values promote exploitation, while large $F$ values make the approach more explorative.

$$\vec{V}_j = \vec{X}_{r_3} + F \times (\vec{X}_{r_1} - \vec{X}_{r_2}) \tag{1}$$

Once the mutant vector is obtained, it is combined with the target vector through the application of a crossover operator so as to generate the *trial vector* $\vec{U}_j$ (step 6). The combination of the mutant vector generation strategy and the crossover operator is usually referred to as the *trial vector generation strategy*. The binomial crossover is controlled by means of the crossover rate $CR$, and uses Equation 2 for producing a trial vector. The decision variable $i$ belonging to individual $\vec{X}_j$ is referred to as $x_{j,i}$. A random number uniformly distributed in the interval $[0, 1]$ is given by $rand_{j,i}$, and $i_{rand} \in [1, 2, ..., D]$ is an index randomly chosen which ensures that at least one decision variable belonging to the mutant vector is inherited by the trial one. Hence, variables are inherited from the mutant vector with probability $CR$. In the remaining cases, variables are inherited from the target vector.

$$u_{j,i} = \begin{cases} v_{j,i} & if\ (rand_{j,i} \leq CR\ or\ i = i_{rand}) \\ x_{j,i} & otherwise \end{cases} \tag{2}$$

As it can be observed in Equations 1 and 2, the trial vector generation strategy may generate individuals outside the feasible region $\Omega$. In this situation, an infeasible value in a particular variable of a trial vector is randomly reinitialised in the corresponding feasible range of that variable. Once all entries of the the trial vector are feasible, it becomes the newly generated neighbour (step 7).

In the case of MBO, the reader should recall that the above trial vector generation strategy is applied in steps 5 and 8 of Algorithm 1 for producing the neighbourhood of a given individual. In cases where DE is applied as an independent approach, the trial vector generation strategy is applied starting from each target vector $\vec{X}_{j=1...n}$ in the population (step 3). In addition, once a trial vector is obtained, it is compared against its corresponding target vector in terms of the objective function value. The fittest individual from among both survives for the next generation (step 8). If both individuals have the same objective value, the trial vector survives. Finally, the initial population of the algorithm is generated at random (step 1) and the algorithm evolves the population through consecutive generations until a given stopping criterion is satisfied (step 2).

*3.2.2. An exploitative neighbour generation operator based on* DE*/current-to-p-best/1/bin*

With the aim of providing a neighbour generation operator which promotes intensification rather than exploration, and based on previous work carried out by the authors (Segura et al., 2015), in the current paper we also consider this particular DE variant. The operation of this DE version is exactly the same as that shown in Algorithm 3. The mutant generation strategy, however, is different.

In this variant, a mutant vector $\vec{V}_j$ is created starting from a target vector $\vec{X}_j$ as it is described in Equation 3. Indexes $r_1$ and $r_2$ are mutually exclusive integers randomly selected from the range $[1, n]$, and also different from the index $j$. Furthermore, the individual $\vec{X}_{r_3}$ is randomly selected from the fittest $p \times 100\%$ individuals. Some of the fittest individuals in the population are taken into account by the mutant generation scheme, and consequently, this DE version is more exploitative than the approach DE/rand/1/bin, which only uses randomness for selecting the individuals involved in the mutant generation scheme.

$$\vec{V}_j = \vec{X}_j + K \times (\vec{X}_{r_3} - \vec{X}_j) + F \times (\vec{X}_{r_1} - \vec{X}_{r_2}) \tag{3}$$

In addition to the mutation scale factor $F$, parameter $p$ can be used in order to set the balance between the exploration and exploitation capabilities of the algorithm. By considering large $p$ values, the scheme is more explorative, while it becomes more exploitative with small $p$ values. Finally, parameter $K$ is also introduced, but in order to make the configuration of the approach easier, $K = F$ is usually considered in the related literature (Segura et al., 2015; Zhang & Sanderson, 2009).

*3.2.3. Control of the mutation scale factor $F$ and the crossover rate $CR$ by means of JADE*

It is clear that in both aforementioned DE-based neighbourhood operators, values for parameters $F$ and $CR$ must be set. The advantages that *adapting* the parameters of a particular algorithm during its execution might provide instead of keeping them fixed across the whole run are widely known (Karafotias et al., 2015), which discusses the benefits of *parameter control* in comparison to *parameter tuning*. A significant amount of research on parameter adaptation has been carried out regarding DE (Das et al., 2016; Tvrdík et al., 2013).

One of the most promising schemes for adapting the mutation scale factor $F$ and the crossover rate $CR$ is that applied by JADE (Zhang & Sanderson, 2009). In the case of MBO, the control mechanisms provided by JADE are applied at the beginning of our proposed neighbourhood operators for generating values of parameters $F$ and $CR$. Hence, a new neighbour is obtained by using those newly produced values (steps 5 and 8 of Algorithm 1). In case of considering DE as an independent optimisation procedure, JADE generates values for $F$ and $CR$ at the beginning of the trial vector generation strategy, thus producing a new trial vector by using those newly generated values (steps 5 and 6 of Algorithm 3). In this way, every individual has associated its own values for parameters $F$ and $CR$.

In JADE, a particular value of parameter $F$ is generated at random by means of a *Cauchy* distribution with location factor $\mu_F$ and scale parameter 0.1. If the value obtained is lower than 0, then another value is randomly produced. If it is higher than 1, however, it is truncated to 1. The location factor $\mu_F$ is initialised to 0.5. In the case of MBO, it is updated at each iteration $m$ after step 10 of Algorithm 1, while in DE, it

is modified at each generation after step 8 of Algorithm 3. The updating mechanism considers the *Lehmer mean* ($mean_L$) of the successful values of $F$ ($S_F$), the previous value of $\mu_F$, and a parameter $c$ representing the adaptation speed of $\mu_F$. Considering MBO, $S_F$ contains those values of $F$ associated to neighbours that have been able to replace any individual in the population in order to survive for the next generation, i.e., at steps 5 and 8 of Algorithm 1. In the case of DE, $S_F$ consists of those values of $F$ associated to trial vectors that have been able to replace their corresponding target vectors in the population so as to survive for the next generation, i.e., at step 8 of Algorithm 3. The updating procedure of $\mu_F$ is illustrated in Equation 4.

$$\mu_F = (1 - c) \cdot \mu_F + c \cdot mean_L(S_F) \tag{4}$$

The control mechanism of $CR$ is akin to the control procedure of $F$. A value of $CR$ is randomly generated through a *Normal* distribution with mean $\mu_{CR}$ and standard deviation 0.1, and truncated to the range $[0, 1]$. The mean $\mu_{CR}$ is initialised to 0.5 and updated by taking into account the arithmetic mean ($mean_A$) of the successful values of $CR$ ($S_{CR}$), the previous value of $\mu_{CR}$, and a parameter $c$ being the adaptation speed of $\mu_{CR}$. Equation 5 shows the update mechanism for $\mu_{CR}$.

$$\mu_{CR} = (1 - c) \cdot \mu_{CR} + c \cdot mean_A(S_{CR}) \tag{5}$$

## 4. Experimental evaluation and discussion

In this section we describe the computational experiments to evaluate the newly proposed MBO variants. The approach making use of the novel leader replacement scheme, which was described in Section 3.1, will be denoted as E-MBO in the rest of the paper. Furthermore, the original implementation of MBO was also executed. Both E-MBO and MBO were combined with the different neighbour generation operators based on DE presented in Section 3.2, thus providing novel schemes for dealing with continuous problems. Finally, in order to assess the contribution of the hybridisation between both variants of MBO and DE, the different DE versions used to define the neighbourhood operators were also run as independent optimisation procedures.

***Experimental method***. Both E-MBO and MBO, as well as the different DE variants, were implemented through the *Meta-heuristic-based Extensible Tool for Cooperative Optimisation* (METCO) (León et al., 2009). Experiments were run on one Debian GNU/Linux computer with four AMD® Opteron™ processors (model number 6348 HE) at 2.8 GHz and 64 GB RAM. Since the experiments use stochastic methods, each run was repeated 30 times. Comparisons were performed using the following statistical procedure, which has been applied in previous work by the authors (Segura et al., 2016). First, a *Shapiro-Wilk test* was performed to check if the values of the results followed a normal (Gaussian) distribution. If so, the *Levene test* checked for the homogeneity of the variances. If the samples had equal variance, an ANOVA *test* was done. Otherwise, a *Welch test* was performed. For non-Gaussian distributions, the non-parametric *Kruskal-Wallis* test was used. For all tests, a significance level $\alpha = 0.05$, corrected using the Dunn-Šidák correction, was considered.

***Problem sets***. We adopt the set of continuous optimisation problems proposed for the competition on LSGO (Li et al., 2013) [2] organised during CEC'13. It is important to note that this test suite is the most recently proposed for large scale global optimisation in the field of the CEC, and therefore, it was also considered for the LSGO competition organised during CEC'15 [3]. The set consists of 15 different functions ($f_1$–$f_{15}$) to be minimised with different features: fully-separable functions ($f_1$–$f_3$), partially additively separable functions ($f_4$–$f_{11}$), overlapping functions ($f_{12}$–$f_{14}$), and a non-separable function ($f_{15}$). In the current work, we fixed the number of decision variables $D$ to 1000 for all functions, with the exception of $f_{13}$ and $f_{14}$, where 905 decision variables were considered because of overlapping subcomponents. These are the values suggested by the LSGO competition organisers.

[2] To get further information about the way each of the functions $f_1$–$f_{15}$ was designed, as well as information about their particular features, the reader is referred to `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.696.6494&rep=rep1&type=pdf`.

[3] Although at CEC'14, CEC'16 and CEC'17 there were special sessions on LSGO, no competitions were organised.

Table 1: Benchmark functions

| **CEC Functions** | | |
|---|---|---|
| **Name** | **Bounds** | **Global Optimum** |
| $f_1$: Shifted Elliptic Function | $[-100, 100]^D$ | 0 |
| $f_2$: Shifted Rastrigin's Function | $[-5, 5]^D$ | 0 |
| $f_3$: Shifted Ackley's Function | $[-32, 32]^D$ | 0 |
| $f_4$: 7-nonseparable, 1-separable Shifted and Rotated Elliptic Function | $[-100, 100]^D$ | 0 |
| $f_5$: 7-nonseparable, 1-separable Shifted and Rotated Rastrigin's Function | $[-5, 5]^D$ | 0 |
| $f_6$: 7-nonseparable, 1-separable Shifted and Rotated Ackley's Function | $[-32, 32]^D$ | 0 |
| $f_7$: 7-nonseparable, 1-separable Shifted Schwefel's Function | $[-100, 100]^D$ | 0 |
| $f_8$: 20-nonseparable Shifted and Rotated Elliptic Function | $[-100, 100]^D$ | 0 |
| $f_9$: 20-nonseparable Shifted and Rotated Rastrigin's Function | $[-5, 5]^D$ | 0 |
| $f_{10}$: 20-nonseparable Shifted and Rotated Ackley's Function | $[-32, 32]^D$ | 0 |
| $f_{11}$: 20-nonseparable Shifted Schwefel's Function | $[-100, 100]^D$ | 0 |
| $f_{12}$: Shifted Rosenbrock's Function | $[-100, 100]^D$ | 0 |
| $f_{13}$: Shifted Schwefel's Function with Conforming Overlapping Subcomponents | $[-100, 100]^D$ | 0 |
| $f_{14}$: Shifted Schwefel's Function with Conflicting Overlapping Subcomponents | $[-100, 100]^D$ | 0 |
| $f_{15}$: Shifted Schwefel's Function | $[-100, 100]^D$ | 0 |
| **SOCO Functions** | | |
| **Name** | **Bounds** | **Global Optimum** |
| $s_1$: Shifted Sphere Function | $[-100, 100]^D$ | -450 |
| $s_2$: Shifted Schwefel's Problem 2.21 | $[-100, 100]^D$ | -450 |
| $s_3$: Shifted Rosenbrock's Function | $[-100, 100]^D$ | 390 |
| $s_4$: Shifted Rastrigin's Function | $[-5, 5]^D$ | -330 |
| $s_5$: Shifted Griewank's Function | $[-600, 600]^D$ | -180 |
| $s_6$: Shifted Ackley's Function | $[-32, 32]^D$ | -140 |
| $s_7$: Shifted Schwefel's Problem 2.22 | $[-10, 10]^D$ | 0 |
| $s_8$: Shifted Schwefel's Problem 1.2 | $[-65.536, 65.536]^D$ | 0 |
| $s_9$: Shifted Extended $f_{10}$ | $[-100, 100]^D$ | 0 |
| $s_{10}$: Shifted Bohachevsky | $[-15, 15]^D$ | 0 |
| $s_{11}$: Shifted Schaffer | $[-100, 100]^D$ | 0 |
| $s_{12}$: Hybrid Composition Function | $[-100, 100]^D$ | 0 |
| $s_{13}$: Hybrid Composition Function | $[-100, 100]^D$ | 0 |
| $s_{14}$: Hybrid Composition Function | $[-5, 5]^D$ | 0 |
| $s_{15}$: Hybrid Composition Function | $[-10, 10]^D$ | 0 |
| $s_{16}$: Hybrid Composition Function | $[-100, 100]^D$ | 0 |
| $s_{17}$: Hybrid Composition Function | $[-100, 100]^D$ | 0 |
| $s_{18}$: Hybrid Composition Function | $[-5, 5]^D$ | 0 |
| $s_{19}$: Hybrid Composition Function | $[-10, 10]^D$ | 0 |

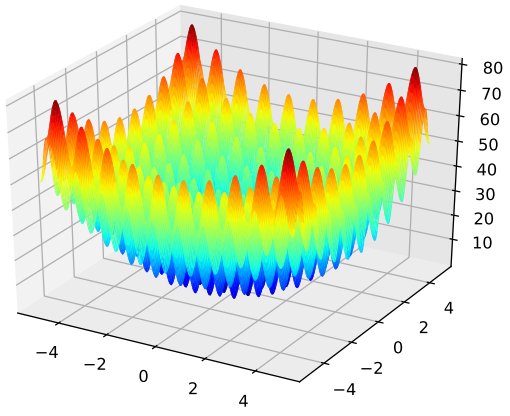


Figure 3: Landscape of the *Rastrigin's* function considering $D = 2$ decision variables or dimensions

With the aim of generalising the conclusions extracted from the analyses carried out with the above set of problems, we also performed experiments with a completely different test suite provided for a special issue on evolutionary algorithms and other meta-heuristics for large scale optimisation, belonging to the journal *Soft Computing* (Lozano et al., 2011). This set consists of 19 scalable continuous optimisation problems ($s_1$–$s_{19}$) to be minimised that combine different properties involving modality, separability, and the ease of optimisation dimension by dimension. Particularly, 7 shifted uni-modal functions ($s_1$–$s_2$ and $s_7$–$s_{11}$), 4 shifted multi-modal functions ($s_3$–$s_6$), and 8 hybrid composition functions ($s_{12}$–$s_{19}$) were provided [4]. As in the case of the first set of problems, $D = 1000$ decision variables were considered for this test suite.

Table 1 shows a summary of the functions tested in the current work, including information about the bounds of the decision variables and the value of the global optimum for each of them. As it can be observed, all the test cases are based on transformations and/or combinations of well-known base functions, such as the *Sphere* function and the *Rastrigin's* function, among others. For instance, Equation 6 shows the formal definition of the *Rastrigin's* function, where $\vec{x}$ is a vector with $D$ decision variables or dimensions. The goal would be to find the values of the $D$ decision variables belonging to vector $\vec{x}$ such that $f_{rastrigin}(\vec{x})$ is minimised. In the particular case of the *Rastrigin's* function, decision variables of vector $\vec{x}$ can take values from the range $[-5, 5]$. Figure 3 illustrates the landscape of that function by considering $D = 2$ dimensions.

$$f_{rastrigin}(\vec{x}) = \sum_{i=1}^{D} [x_i^2 - 10cos(2\pi x_i) + 10] \tag{6}$$

***Experiments' enumeration.*** Table 2 shows a summary of the different experiments performed to evaluate the proposals. Besides the description of the main goal of each experiment, it also reports a column devoted to outline the best overall performing method in each case. In the last two experiments the best overall variants of each algorithm are compared considering the two well-known aforementioned problem suites.

### *4.1. First experiment:* MBO *variants applying an explorative neighbourhood operator based on* DE*/rand/1/bin with fixed parameters*

As it was previously mentioned, one of the main aims of our experiments is to analyse the performance of our MBO-based approaches in comparison to DE executed as an independent optimiser. For this first experiment, we selected an explorative DE version with fixed parameters. Particularly, the variant DE/rand/1/bin was selected based on previous research (Kazimipour et al., 2014) that evaluated more than 80 different configurations for problems $f_1$–$f_{15}$. The best performing configuration had parameters $n = 150$, $F = 0.5$, and $CR = 0.9$, and is the one considered for comparison purposes in this experiment.

The neighbourhood operator based on DE/rand/1/bin used by E-MBO and MBO was applied with the same parameter values than those used by that DE version executed independently, i.e., $F = 0.5$ and $CR = 0.9$. In order to tune the parameters of E-MBO and MBO, several configurations of both schemes with different parameter values were analysed in a preliminary study. Due to space restrictions, that preliminary study is given as supplementary material. The best performing configuration of E-MBO, which we will refer to as E-MBO-1, was applied with parameter values $n = 150$, $k = 9$, $m = 5$, and $x = 2$, while the best performing configuration of MBO, which we will refer to as MBO-0, was run with parameter values $n = 150$, $k = 7$, $m = 10$, and $x = 1$. Finally, following the recommendations given by the LSGO contest, a stopping criterion equal to $3 \times 10^6$ function evaluations was fixed for E-MBO-1, MBO-0, and DE/rand/1/bin.

Table 3 shows, for each problem, the mean, the median, and the standard deviation (SD) of the objective function error achieved by the aforementioned approaches at the end of the executions. Additionally, for each problem, data in **boldface** show the approach that provided the lowest mean and/or median of the error at the end of the runs. E-MBO-1 was able to provide the lowest mean and median of the error in 6 out of 15 problems ($f_3$, $f_5$, $f_6$, $f_7$, $f_9$, and $f_{15}$), while MBO-0 and DE/rand/1/bin obtained the lowest mean and median of the error for 3 ($f_2$, $f_8$, and $f_{11}$) and 4 problems ($f_1$, $f_4$, $f_{12}$, and $f_{13}$), respectively. For

[4] To get further information about the formal definition and particular features of this test suite, the reader is referred to `http://sci2s.ugr.es/sites/default/files/files/TematicWebSites/EAMHCO/testfunctions-SOCO.pdf`.

Table 2: Overview of experiments. Considering a particular experiment, bullet points in the last column indicate the best-performing overall approaches from among those specified in the corresponding second column. In the case of the MBO-based schemes, the particular neighbourhood operator applied is denoted between brackets.

| Experiment | Methods | Problems | Goal | Best schemes | | |
|---|---|---|---|---|---|---|
| | | | | MBO | E-MBO | DE |
| **First** | MBO [DE/rand/1/bin (fixed parameters)]<br>E-MBO [DE/rand/1/bin (fixed parameters)]<br>DE/rand/1/bin (fixed parameters) | CEC | Analysing the performance of MBO approaches combined with a DE-based explorative neighbourhood operator with fixed parameters | | • | |
| **Second** | MBO [Adaptive DE/rand/1/bin (JADE)]<br>E-MBO [Adaptive DE/rand/1/bin (JADE)]<br>Adaptive DE/rand/1/bin (JADE) | CEC | Analysing the performance of MBO schemes making use of a DE-based adaptive (JADE) explorative neighbourhood operator | • | • | |
| **Third** | MBO [Adaptive DE/current-to-p-best/1/bin (JADE)]<br>E-MBO [Adaptive DE/current-to-p-best/1/bin (JADE)]<br>Adaptive DE/current-to-p-best/1/bin (JADE) | CEC | Analysing the performance of MBO schemes making use of a DE-based adaptive (JADE) exploitative neighbourhood operator | | | • |
| **Fourth** | MBO [Adaptive DE/rand/1/bin (JADE)]<br>E-MBO [Adaptive DE/rand/1/bin (JADE)]<br>Adaptive DE/current-to-p-best/1/bin (JADE) | CEC | Comparing the performance of the best overall MBO, E-MBO and DE approaches | • | • | • |
| **Fifth** | MBO [Adaptive DE/rand/1/bin (JADE)]<br>E-MBO [Adaptive DE/rand/1/bin (JADE)]<br>Adaptive DE/current-to-p-best/1/bin (JADE) | SOCO | Comparing the performance of the best overall MBO, E-MBO and DE approaches with a different test suite | • | | |

the remaining functions ($f_{10}$ and $f_{14}$), MBO-0 achieved the lowest mean of the error while DE/rand/1/bin provided the lowest median of the error, and vice-versa. We note that considering the results provided by E-MBO-1 and MBO-0 together, **the lowest mean and median of the error were provided by the MBO-based approaches in 9 out of 15 problems**, while DE/rand/1/bin provided the lowest mean and median of the error in only 4 out of 15 problems.

In order to complement the above observations, a pairwise statistical comparison among the different optimisation schemes, for each function, is shown in Table 4, which was carried out by following the statistical procedure described at the beginning of Section 4. Particularly, p-values and results of the statistical comparison between the first and second algorithms of each pair are presented. In cases where statistically significant differences arose, p-values are shown in **boldface** [5]. Moreover, it also shows if the first approach outperformed the second one ($\uparrow$), whether the first scheme was outperformed by the second one ($\downarrow$), and if statistically significant differences did not appear between both approaches ($\leftrightarrow$). For example, in the comparison between E-MBO-1 and MBO-0 considering function $f_3$, both schemes presented statistically significant

[5] The reader should recall that the significance level $\alpha = 0.05$ was corrected using the Dunn-Šidák correction. Hence, the corrected significance level is $\alpha_{SID} = 1.695e-2$ (considering $m = 3$ different pairwise statistical comparisons).

Table 3: Mean, median, and standard deviation (SD) of the error achieved by E-MBO-1 [DE/rand/1/bin (fixed parameters)], MBO-0 [DE/rand/1/bin (fixed parameters)] and DE/rand/1/bin (fixed parameters) at the end of 30 executions for problems $f_1$–$f_{15}$

| Alg. | E-MBO-1 | | | MBO-0 | | | DE/rand/1/bin (fixed params.) | | |
|---|---|---|---|---|---|---|---|---|---|
| Func. | Mean | Median | SD | Mean | Median | SD | Mean | Median | SD |
| $f_1$ | 5.189e+06 | 4.006e+06 | 4.993e+06 | 4.081e+06 | 3.487e+06 | 2.962e+06 | **2.063e+06** | **1.611e+06** | 1.301e+06 |
| $f_2$ | 7.981e+03 | 7.938e+03 | 4.273e+02 | **7.790e+03** | **7.820e+03** | 3.344e+02 | 8.464e+03 | 8.446e+03 | 3.057e+02 |
| $f_3$ | **2.131e+01** | **2.131e+01** | 2.666e-02 | 2.153e+01 | 2.153e+01 | 1.571e-02 | 2.153e+01 | 2.153e+01 | 1.218e-02 |
| $f_4$ | 1.712e+10 | 1.593e+10 | 6.070e+09 | 1.656e+10 | 1.497e+10 | 5.870e+09 | **1.521e+10** | **1.366e+10** | 6.269e+09 |
| $f_5$ | **5.915e+06** | **7.748e+06** | 2.957e+06 | 8.351e+06 | 8.387e+06 | 3.234e+05 | 8.193e+06 | 8.192e+06 | 3.573e+05 |
| $f_6$ | **1.060e+06** | **1.060e+06** | 1.413e+03 | 1.061e+06 | 1.061e+06 | 1.063e+03 | 1.060e+06 | 1.061e+06 | 1.264e+03 |
| $f_7$ | **1.729e+08** | **1.533e+08** | 8.242e+07 | 2.477e+08 | 2.090e+08 | 9.374e+07 | 2.265e+08 | 2.116e+08 | 7.566e+07 |
| $f_8$ | 2.299e+13 | 2.364e+13 | 1.558e+13 | **1.594e+13** | **1.543e+13** | 1.206e+13 | 2.746e+13 | 1.776e+13 | 2.229e+13 |
| $f_9$ | **8.110e+07** | **7.595e+07** | 1.781e+07 | 3.774e+08 | 5.334e+08 | 2.474e+08 | 2.582e+08 | 1.279e+08 | 2.293e+08 |
| $f_{10}$ | 9.413e+07 | 9.413e+07 | 2.495e+05 | 9.399e+07 | **9.401e+07** | 1.964e+05 | **9.399e+07** | 9.401e+07 | 2.299e+05 |
| $f_{11}$ | 5.588e+10 | 5.178e+10 | 2.389e+10 | **5.268e+10** | **4.862e+10** | 2.117e+10 | 6.050e+10 | 5.606e+10 | 2.460e+10 |
| $f_{12}$ | 3.213e+08 | 2.137e+08 | 3.999e+08 | 2.695e+08 | 2.059e+08 | 2.224e+08 | **1.589e+08** | **8.135e+07** | 2.090e+08 |
| $f_{13}$ | 5.993e+09 | 5.629e+09 | 1.558e+09 | 6.139e+09 | 5.950e+09 | 9.654e+08 | **5.767e+09** | **5.563e+09** | 1.724e+09 |
| $f_{14}$ | 9.162e+10 | 8.386e+10 | 2.592e+10 | **8.591e+10** | 8.314e+10 | 3.030e+10 | 8.771e+10 | **8.031e+10** | 2.977e+10 |
| $f_{15}$ | **5.292e+07** | **4.887e+07** | 1.548e+07 | 6.737e+07 | 6.046e+07 | 2.285e+07 | 6.830e+07 | 6.507e+07 | 2.365e+07 |

Table 4: Pairwise statistical comparison among E-MBO-1 [DE/rand/1/bin (fixed parameters)], MBO-0 [DE/rand/1/bin (fixed parameters)], and DE/rand/1/bin (fixed parameters) considering their results achieved at the end of 30 executions for problems $f_1$–$f_{15}$

| | E-MBO-1 vs. MBO-0 | | E-MBO-1 vs. DE/rand/1/bin (fixed params.) | | MBO-0 vs. DE/rand/1/bin (fixed params.) | |
|---|---|---|---|---|---|---|
| Func. | p-value | Stat. | p-value | Stat. | p-value | Stat. |
| $f_1$ | 2.036e-01 | $\leftrightarrow$ | **1.667e-06** | $\downarrow$ | **7.425e-05** | $\downarrow$ |
| $f_2$ | 5.915e-02 | $\leftrightarrow$ | **4.867e-06** | $\uparrow$ | **3.441e-11** | $\uparrow$ |
| $f_3$ | **2.872e-11** | $\uparrow$ | **5.494e-35** | $\uparrow$ | 1.983e-01 | $\leftrightarrow$ |
| $f_4$ | 7.193e-01 | $\leftrightarrow$ | 1.103e-01 | $\leftrightarrow$ | 2.550e-01 | $\leftrightarrow$ |
| $f_5$ | **2.290e-08** | $\uparrow$ | **7.475e-06** | $\uparrow$ | 7.764e-02 | $\leftrightarrow$ |
| $f_6$ | 1.984e-01 | $\leftrightarrow$ | 5.444e-01 | $\leftrightarrow$ | 5.543e-01 | $\leftrightarrow$ |
| $f_7$ | **3.666e-04** | $\uparrow$ | **1.480e-03** | $\uparrow$ | 5.250e-01 | $\leftrightarrow$ |
| $f_8$ | 2.866e-02 | $\leftrightarrow$ | 6.574e-01 | $\leftrightarrow$ | 2.559e-02 | $\leftrightarrow$ |
| $f_9$ | **5.308e-08** | $\uparrow$ | **3.131e-07** | $\uparrow$ | 1.738e-01 | $\leftrightarrow$ |
| $f_{10}$ | 1.902e-02 | $\leftrightarrow$ | 3.326e-02 | $\leftrightarrow$ | 8.592e-01 | $\leftrightarrow$ |
| $f_{11}$ | 6.152e-01 | $\leftrightarrow$ | 3.219e-01 | $\leftrightarrow$ | 1.925e-01 | $\leftrightarrow$ |
| $f_{12}$ | 6.898e-01 | $\leftrightarrow$ | **1.732e-04** | $\downarrow$ | **4.101e-04** | $\downarrow$ |
| $f_{13}$ | 2.488e-01 | $\leftrightarrow$ | 7.007e-01 | $\leftrightarrow$ | 1.882e-01 | $\leftrightarrow$ |
| $f_{14}$ | 4.355e-01 | $\leftrightarrow$ | 4.779e-01 | $\leftrightarrow$ | 8.360e-01 | $\leftrightarrow$ |
| $f_{15}$ | **6.522e-03** | $\uparrow$ | **2.002e-03** | $\uparrow$ | 6.152e-01 | $\leftrightarrow$ |

differences (p-value $= 2.872e-11$), with E-MBO-1 being the best approach ($\uparrow$) since it provided the lowest mean and median of the error. Bearing the above in mind, in the following we provide a list of observations that are further discussed:

- E-MBO-1 was able to statistically outperform MBO-0 in 5 out of 15 problems ($f_3$, $f_5$, $f_7$, $f_9$, and $f_{15}$), while the latter was not able to outperform the former in any test case. For the remaining problems, no statistically significant differences appeared between both schemes.
- With respect to DE/rand/1/bin, E-MBO-1 outperformed it in 6 out of 15 functions ($f_2$, $f_3$, $f_5$, $f_7$, $f_9$, and $f_{15}$), while the former was statistically better than the latter only in two problems ($f_1$, and $f_{12}$). Statistical differences did not arise for the remaining test cases.
- MBO-0 was statistically better than DE/rand/1/bin only in the case of problem $f_2$, while DE/rand/1/bin outperformed MBO-0 considering two test cases ($f_1$ and $f_{12}$). Both schemes did not present statistically significant differences for the remaining functions.

**The fact that E-MBO-1 outperformed MBO-0 in 5 out of 15 problems demonstrates the contribution of the novel elitist leader replacement strategy**. In this particular experiment, the neighbour generation operator based on DE/rand/1/bin promotes exploration instead of exploitation. In some test cases, by combining this operator with the novel elitist leader replacement scheme, the intensification ability of E-MBO increases with respect to MBO, and therefore, so does the convergence speed to better solutions.

Consequently, E-MBO-1 provided a better performance than MBO-0 considering 33.3% of the test cases. The above can be reinforced by the fact that E-MBO-1 was able to outperform DE/rand/1/bin in 6 out of 15 functions, while MBO-0 was able to beat DE/rand/1/bin taking into account a unique test case.

DE/rand/1/bin was outperformed by at least one of both MBO-based schemes considering 6 out of 15 functions ($f_2$, $f_3$, $f_5$, $f_7$, $f_9$, and $f_{15}$), while the former was statistically better than both MBO-based approaches in two out of 15 problems ($f_1$, and $f_{12}$). The above means that **the MBO-based schemes provided statistically better or similar solutions than DE/rand/1/bin in 13 out of 15 problems, which represents 86.6% of the test cases**. As a result, the contribution of combining a MBO-based approach with an explorative neighbourhood operator based on DE/rand/1/bin is also demonstrated, since the hybrid optimisation schemes were able to perform better than DE/rand/1/bin considered as an independent approach in a wide range of problems. Moreover, taking into account that in past research the said DE variant was able to provide the best results in the majority of the problems addressed herein, the contribution of E-MBO and MBO is even more noticeable.

### *4.2. Second experiment:* MBO *variants applying an explorative neighbourhood operator based on an adaptive* DE*/rand/1/bin (*JADE*)*

In this experiment, we analyse whether updating parameters $F$ and $CR$ of the explorative neighbourhood operator based on DE/rand/1/bin during the execution of E-MBO and MBO significantly changes the behaviour of both schemes. In addition we investigate if controlling those parameters provides any advantage in comparison to using fixed values as in the first experiment.

With respect to the neighbourhood operator based on the adaptive DE/rand/1/bin, the values for parameter $F$ were randomly generated by applying a *Cauchy* distribution with location factor $\mu_F = 0.5$ and scale parameter equal to 0.1. The main difference of this approach with respect to the control mechanism provided by JADE for parameter $F$, which was described in Section 3.2, is that the updating mechanism of the location factor $\mu_F$ (Equation 4) was not applied, i.e., the location factor $\mu_F$ remained fixed to the value 0.5 during the whole run. In the case of $CR$, the control mechanism provided by JADE for that parameter was used, and applied with adaptation speed $c = 0.1$. This particular DE variant was selected as Segura et al. (2015) demonstrated that the control mechanism provided by JADE for parameter $F$ decreases the performance of an explorative DE version, such as DE/rand/1/bin, in comparison to the usage of a random distribution, like *Cauchy*, for obtaining the values of $F$.

In order to look for the best parameterisation for E-MBO and MBO, the parameter tuning procedure carried out during the first experiment was also performed herein. In this case, the best performing configurations for both E-MBO and MBO, which we will refer to as E-MBO-0 and MBO-0, respectively, were applied with parameter values $n = 150$, $k = 7$, $m = 10$, and $x = 1$. In addition to E-MBO-0 and MBO-0, the adaptive DE/rand/1/bin variant was also executed as an independent scheme with $n = 150$ individuals. Finally, the stopping criterion for E-MBO-0, MBO-0, and the adaptive DE/rand/1/bin was fixed to $3 \times 10^6$ evaluations.

Table 5 shows, for each problem, the mean, the median, and the standard deviation (SD) of the objective function error achieved by the three schemes at the end of their runs. In this case, E-MBO-0 was able to provide the lowest mean and median of the error in 4 out of 15 problems ($f_4$, $f_5$, $f_8$, and $f_{13}$), while MBO-0 and the adaptive variant of DE/rand/1/bin obtained the lowest mean and median of the error in 3 ($f_9$, $f_{11}$, and $f_{14}$) and 2 problems ($f_2$, and $f_3$), respectively. In 5 functions ($f_1$, $f_6$, $f_7$, $f_{12}$, and $f_{15}$), E-MBO-0 achieved the lowest mean of the error while MBO-0 provided the lowest median of the error. Finally, MBO-0 provided the lowest median of the error while the adaptive version of DE/rand/1/bin achieved the lowest mean of the error considering function $f_{10}$. It is worth highlighting that based on the results attained by E-MBO-0 and MBO-0 together, **the MBO-based variants were able to provide the lowest mean and median of the error in 12 problems**, which represents 80% of the test cases. Only for problems $f_2$ and $f_3$ the adaptive DE/rand/1/bin attained the lowest mean and median of the error.

With the aim of supporting the aforementioned information, Table 6, shows the pairwise statistical comparison among E-MBO-0, MBO-0, and the adaptive variant of DE/rand/1/bin. We share the following observations regarding that statistical comparison:

Table 5: Mean, median, and standard deviation (SD) of the error achieved by E-MBO-0 [Adaptive DE/rand/1/bin (JADE)], MBO-0 [Adaptive DE/rand/1/bin (JADE)] and the adaptive version of DE/rand/1/bin (JADE) at the end of 30 executions for problems $f_1$–$f_{15}$

| Alg. | E-MBO-0 | | | MBO-0 | | | DE/rand/1/bin (JADE) | | |
|---|---|---|---|---|---|---|---|---|---|
| Func. | Mean | Median | SD | Mean | Median | SD | Mean | Median | SD |
| $f_1$ | **1.334e-02** | 8.177e-03 | 1.818e-02 | 2.513e-02 | **6.033e-03** | 5.523e-02 | 1.194e+03 | 1.190e+03 | 4.276e+01 |
| $f_2$ | 2.020e+03 | 2.016e+03 | 1.001e+02 | 1.917e+03 | 1.919e+03 | 1.137e+02 | **9.805e+02** | **9.823e+02** | 2.147e+01 |
| $f_3$ | 2.024e+01 | 2.024e+01 | 1.698e-02 | 2.022e+01 | 2.023e+01 | 1.604e-02 | **2.015e+01** | **2.015e+01** | 3.375e-03 |
| $f_4$ | **6.283e+10** | **5.370e+10** | 3.181e+10 | 6.824e+10 | 5.441e+10 | 4.060e+10 | 3.868e+11 | 3.928e+11 | 7.640e+10 |
| $f_5$ | **3.119e+06** | **2.992e+06** | 6.504e+05 | 3.298e+06 | 3.171e+06 | 5.957e+05 | 7.876e+06 | 7.877e+06 | 4.667e+05 |
| $f_6$ | **1.051e+06** | 1.055e+06 | 1.141e+04 | 1.054e+06 | **1.055e+06** | 8.332e+03 | 1.055e+06 | 1.058e+06 | 1.154e+04 |
| $f_7$ | **9.981e+08** | 9.145e+08 | 5.131e+08 | 1.026e+09 | **8.976e+08** | 5.107e+08 | 2.336e+09 | 2.338e+09 | 4.010e+08 |
| $f_8$ | **1.102e+15** | **9.825e+14** | 5.639e+14 | 1.149e+15 | 1.069e+15 | 5.638e+14 | 9.872e+15 | 9.675e+15 | 2.236e+15 |
| $f_9$ | 2.373e+08 | 2.426e+08 | 4.135e+07 | **2.304e+08** | **2.190e+08** | 4.502e+07 | 5.852e+08 | 5.848e+08 | 3.238e+07 |
| $f_{10}$ | 9.368e+07 | 9.364e+07 | 4.821e+05 | 9.354e+07 | **9.362e+07** | 7.530e+05 | **9.348e+07** | 9.369e+07 | 7.512e+05 |
| $f_{11}$ | 8.686e+10 | 7.076e+10 | 5.553e+10 | **7.693e+10** | **6.851e+10** | 3.645e+10 | 1.683e+11 | 1.658e+11 | 4.027e+10 |
| $f_{12}$ | **4.488e+03** | 4.428e+03 | 5.818e+02 | 4.525e+03 | **4.344e+03** | 6.388e+02 | 2.583e+04 | 2.590e+04 | 9.857e+02 |
| $f_{13}$ | **1.489e+10** | **1.448e+10** | 3.445e+09 | 1.569e+10 | 1.525e+10 | 4.021e+09 | 2.843e+10 | 2.876e+10 | 2.275e+09 |
| $f_{14}$ | 2.245e+11 | 2.108e+11 | 9.288e+10 | **2.054e+11** | **1.802e+11** | 9.870e+10 | 3.839e+11 | 3.855e+11 | 5.039e+10 |
| $f_{15}$ | **1.635e+07** | 1.616e+07 | 1.611e+06 | 1.666e+07 | **1.610e+07** | 2.112e+06 | 6.655e+07 | 6.729e+07 | 4.283e+06 |

Table 6: Pairwise statistical comparison among E-MBO-0 [Adaptive DE/rand/1/bin (JADE)], MBO-0 [Adaptive DE/rand/1/bin (JADE)] and the adaptive version of DE/rand/1/bin (JADE) considering their results achieved at the end of 30 executions for problems $f_1$–$f_{15}$

| | E-MBO-0 vs. MBO-0 | | E-MBO-0 vs. DE/rand/1/bin (JADE) | | MBO-0 vs. DE/rand/1/bin (JADE) | |
|---|---|---|---|---|---|---|
| Func. | p-value | Stat. | p-value | Stat. | p-value | Stat. |
| $f_1$ | 3.292e-01 | $\leftrightarrow$ | **2.872e-11** | $\uparrow$ | **2.872e-11** | $\uparrow$ |
| $f_2$ | **4.562e-04** | $\downarrow$ | **3.793e-33** | $\downarrow$ | **1.164e-29** | $\downarrow$ |
| $f_3$ | **9.357e-05** | $\downarrow$ | **1.459e-24** | $\downarrow$ | **1.184e-22** | $\downarrow$ |
| $f_4$ | 5.946e-01 | $\leftrightarrow$ | **2.872e-11** | $\uparrow$ | **2.872e-11** | $\uparrow$ |
| $f_5$ | 2.698e-01 | $\leftrightarrow$ | **5.928e-39** | $\uparrow$ | **2.220e-39** | $\uparrow$ |
| $f_6$ | 5.543e-01 | $\leftrightarrow$ | 2.760e-02 | $\leftrightarrow$ | 6.043e-02 | $\leftrightarrow$ |
| $f_7$ | 9.882e-01 | $\leftrightarrow$ | **5.317e-10** | $\uparrow$ | **1.229e-09** | $\uparrow$ |
| $f_8$ | 7.338e-01 | $\leftrightarrow$ | **2.872e-11** | $\uparrow$ | **2.315e-20** | $\uparrow$ |
| $f_9$ | 5.392e-01 | $\leftrightarrow$ | **1.450e-41** | $\uparrow$ | **9.967e-41** | $\uparrow$ |
| $f_{10}$ | 7.117e-01 | $\leftrightarrow$ | 7.562e-01 | $\leftrightarrow$ | 9.411e-01 | $\leftrightarrow$ |
| $f_{11}$ | 9.293e-01 | $\leftrightarrow$ | **2.675e-07** | $\uparrow$ | **8.487e-10** | $\uparrow$ |
| $f_{12}$ | 9.764e-01 | $\leftrightarrow$ | **2.872e-11** | $\uparrow$ | **2.872e-11** | $\uparrow$ |
| $f_{13}$ | 4.126e-01 | $\leftrightarrow$ | **1.583e-23** | $\uparrow$ | **2.003e-19** | $\uparrow$ |
| $f_{14}$ | 3.077e-01 | $\leftrightarrow$ | **3.496e-08** | $\uparrow$ | **2.493e-08** | $\uparrow$ |
| $f_{15}$ | 5.274e-01 | $\leftrightarrow$ | **1.575e-38** | $\uparrow$ | **1.039e-41** | $\uparrow$ |

- With respect to the adaptive DE/rand/1/bin, each of both MBO-based schemes outperformed it in 11 out of 15 functions ($f_1$, $f_4$, $f_5$, $f_7$–$f_9$, and $f_{11}$–$f_{15}$), while the former was statistically better than both MBO-based approaches only in two problems ($f_2$ and $f_3$). Statistical differences did not arise between each of both MBO-based algorithms and the adaptive DE/rand/1/bin in problems $f_6$ and $f_{10}$.
- E-MBO-0 and MBO-0 were not statistically outperformed by any other scheme in 13 out of 15 problems.
- The adaptive version of DE/rand/1/bin was not statistically outperformed by any other scheme in 4 out of 15 problems.

The fact that E-MBO-0 was able to statistically outperform the adaptive DE/rand/1/bin in 11 problems, while MBO-0 was statistically better considering the same 11 functions, shows their clear superiority. The adaptive DE/rand/1/bin outperformed both MBO-based approaches in only two functions. The above means that **the MBO-based schemes provided better or similar solutions than the adaptive DE/rand/1/bin in 13 out of 15 problems**, which represents more than 86% of the test cases. Bearing the above in mind, both MBO variants making use of the novel neighbour generation operator based on the adaptive DE/rand/1/bin, could be applied for solving problems that we do not have enough *a priori* information about, like black-box continuous problems, instead of using the adaptive variant of DE/rand/1/bin as an independent approach. Both MBO-based variants are likely to provide better results than those achieved by the adaptive DE/rand/1/bin. Furthermore, we can confirm conclusions deduced from the first experiment. **The hybridisation between a MBO-based algorithm and an explorative neighbour generation**

Table 7: Pairwise statistical comparison among E-MBO and MBO using the neighbourhood operator based on the adaptive DE/rand/1/bin (JADE) and both approaches applying the operator based on DE/rand/1/bin with fixed parameters (non-adaptive), considering their results achieved at the end of 30 executions for problems $f_1$–$f_{15}$

| | Adaptive E-MBO vs. Non-adaptive E-MBO | | Adaptive MBO vs. Non-adaptive MBO | |
|---|---|---|---|---|
| Func. | p-value | Stat. | p-value | Stat. |
| $f_1$ | **2.872e-11** | ↑ | **2.872e-11** | ↑ |
| $f_2$ | **1.391e-37** | ↑ | **8.664e-44** | ↑ |
| $f_3$ | **4.778e-82** | ↑ | **2.872e-11** | ↑ |
| $f_4$ | **1.148e-10** | ↓ | **9.445e-11** | ↓ |
| $f_5$ | **1.811e-03** | ↑ | **4.808e-37** | ↑ |
| $f_6$ | **1.548e-06** | ↑ | **1.067e-06** | ↑ |
| $f_7$ | **6.373e-11** | ↓ | **1.395e-10** | ↓ |
| $f_8$ | **2.872e-11** | ↓ | **2.872e-11** | ↓ |
| $f_9$ | **2.872e-11** | ↓ | 4.598e-01 | ↔ |
| $f_{10}$ | **3.742e-05** | ↑ | **1.205e-03** | ↑ |
| $f_{11}$ | 2.463e-02 | ↔ | **6.036e-04** | ↓ |
| $f_{12}$ | **2.872e-11** | ↑ | **2.872e-11** | ↑ |
| $f_{13}$ | **3.879e-11** | ↓ | **4.614e-14** | ↓ |
| $f_{14}$ | **2.260e-10** | ↓ | **8.864e-09** | ↓ |
| $f_{15}$ | **1.164e-13** | ↑ | **2.872e-11** | ↑ |

**operator based on DE/rand/1/bin is beneficial**, since both hybrid approaches were able to perform better than this DE variant used as an independent optimisation scheme, for a wide range of problems. For this particular case, where an adaptive version of DE/rand/1/bin is considered, the previous fact is even more noticeable than in the case of the first experiment with the non-adaptive DE/rand/1/bin.

Finally, in order to compare the results achieved by E-MBO and MBO using the neighbourhood operator based on the adaptive DE/rand/1/bin (configurations E-MBO-0 and MBO-0 of the second experiment) and the results attained by both schemes using the neighbour generation operator based on the non-adaptive DE/rand/1/bin (schemes E-MBO-1 and MBO-0 of the first experiment), Table 7 shows the p-values obtained from that comparison. It can be observed that the adaptive E-MBO statistically outperformed the non-adaptive E-MBO in 8 out of 15 problems ($f_1$–$f_3$, $f_5$, $f_6$, $f_{10}$, $f_{12}$, and $f_{15}$), while the latter was statistically better than the former in 6 functions. Only in the case of function $f_{11}$, the adaptive and non-adaptive E-MBO did not have statistically significant differences. In the case of MBO, the adaptive version also provided better results than the non-adaptive variant in 8 out of 15 test cases (exactly the same functions as in the case of E-MBO), while the latter statistically outperformed the former in 6 problems. Only for test case $f_9$, the adaptive MBO and the non-adaptive MBO did not present significant differences.

In some problems, such as $f_1$ and $f_{12}$, among others, adapting parameter values during the execution provided a clear advantage, while in other test cases, like $f_4$ and $f_{13}$, keeping parameter values fixed for the whole run allowed better results to be achieved. Nevertheless, **when using the adaptive version of the explorative neighbourhood operator based on DE/rand/1/bin, both MBO-based variants were able to attain better results for a larger number of problems** than those achieved by using the non-adaptive version of the explorative neighbour generation operator. The benefits of parameter control in comparison to parameter tuning are thus demonstrated. Despite of the above, it would be interesting to study whether a smart selection procedure, such as a hyper-heuristic based on selection similar to that applied by Segredo et al. (2016), would allow the best performing approach, from among the four MBO-based variants considered in this comparison, to be automatically selected depending on the particular features of the problem addressed. The above, however, is out of the scope of this paper and will be addressed as a future line of research.

### *4.3. Third experiment: MBO variants applying an exploitative neighbourhood operator based on an adaptive DE/current-to-p-best/1/bin (JADE)*

This experiment was devoted to studying the behaviour of E-MBO and MBO when combined with a neighbourhood operator based on an exploitative adaptive version of DE. We considered the variant DE/current-to-p-best/1/bin, which was described in Section 3.2. The reader should recall that in previous research (Segura et al., 2015), it was shown that the mechanism provided by JADE for controlling parameter $F$ decreases the performance of explorative DE variants with respect to update $F$ through a

Table 8: Mean, median, and standard deviation (SD) of the error achieved by E-MBO-8 [Adaptive DE/current-to-p-best/1/bin (JADE)], MBO-8 [Adaptive DE/current-to-p-best/1/bin (JADE)] and the adaptive version of DE/current-to-p-best/1/bin (JADE) at the end of 30 executions for problems $f_1$–$f_{15}$

| Alg. | E-MBO-8 | | | MBO-8 | | | DE/current-to-p-best/1/bin (JADE) | | |
|---|---|---|---|---|---|---|---|---|---|
| Func. | Mean | Median | SD | Mean | Median | SD | Mean | Median | SD |
| $f_1$ | 3.017e+08 | 2.791e+08 | 9.852e+07 | 2.858e+08 | 2.656e+08 | 7.070e+07 | **5.924e+02** | **1.722e+02** | 1.380e+03 |
| $f_2$ | 1.549e+04 | 1.564e+04 | 4.513e+02 | 1.554e+04 | 1.548e+04 | 4.547e+02 | **7.108e+03** | **7.088e+03** | 7.821e+02 |
| $f_3$ | 2.017e+01 | 2.017e+01 | 6.429e-03 | **2.016e+01** | **2.016e+01** | 6.641e-03 | 2.044e+01 | 2.044e+01 | 8.249e-03 |
| $f_4$ | 4.935e+10 | 4.819e+10 | 7.718e+09 | 5.059e+10 | 5.030e+10 | 1.008e+10 | **3.966e+09** | **4.032e+09** | 1.211e+09 |
| $f_5$ | **1.950e+06** | **1.950e+06** | 2.486e+05 | 2.020e+06 | 2.053e+06 | 2.682e+05 | 4.064e+06 | 4.088e+06 | 3.081e+05 |
| $f_6$ | **1.052e+06** | **1.051e+06** | 4.706e+03 | 1.054e+06 | 1.053e+06 | 4.241e+03 | 1.055e+06 | 1.058e+06 | 1.186e+04 |
| $f_7$ | 4.451e+08 | 4.197e+08 | 1.107e+08 | 4.379e+08 | 4.265e+08 | 1.021e+08 | **3.881e+06** | **3.542e+06** | 1.633e+06 |
| $f_8$ | 3.466e+14 | 3.501e+14 | 8.802e+13 | 3.554e+14 | 3.632e+14 | 1.240e+14 | **4.711e+12** | **3.906e+12** | 3.522e+12 |
| $f_9$ | **2.109e+08** | **2.077e+08** | 1.868e+07 | 2.164e+08 | 2.158e+08 | 1.961e+07 | 3.469e+08 | 3.494e+08 | 2.071e+07 |
| $f_{10}$ | **9.308e+07** | **9.298e+07** | 4.678e+05 | 9.312e+07 | 9.317e+07 | 4.608e+05 | 9.359e+07 | 9.383e+07 | 8.166e+05 |
| $f_{11}$ | 8.627e+10 | 8.375e+10 | 2.155e+10 | 8.850e+10 | 9.011e+10 | 2.209e+10 | **1.705e+08** | **1.601e+08** | 4.288e+07 |
| $f_{12}$ | 1.787e+10 | 1.696e+10 | 2.236e+09 | 1.764e+10 | 1.771e+10 | 1.810e+09 | **6.127e+03** | **5.859e+03** | 8.029e+02 |
| $f_{13}$ | 8.474e+09 | 8.538e+09 | 1.091e+09 | 8.518e+09 | 8.445e+09 | 1.015e+09 | **1.835e+08** | **1.884e+08** | 7.226e+07 |
| $f_{14}$ | 1.470e+11 | 1.415e+11 | 3.471e+10 | 1.507e+11 | 1.468e+11 | 2.918e+10 | **1.306e+08** | **8.998e+07** | 9.241e+07 |
| $f_{15}$ | 1.755e+08 | 1.287e+08 | 1.410e+08 | 1.854e+08 | 1.528e+08 | 1.263e+08 | **1.246e+06** | **1.247e+06** | 1.230e+05 |

Table 9: Pairwise statistical comparison among E-MBO-8 [Adaptive DE/current-to-p-best/1/bin (JADE)], MBO-8 [Adaptive DE/current-to-p-best/1/bin (JADE)] and the adaptive version of DE/current-to-p-best/1/bin (JADE) considering their results achieved at the end of 30 executions for problems $f_1$–$f_{15}$

| | E-MBO-8 vs. MBO-8 | | E-MBO-8 vs. DE/current-to-p-best/1/bin (JADE) | | MBO-8 vs. DE/current-to-p-best/1/bin (JADE) | |
|---|---|---|---|---|---|---|
| Func. | p-value | Stat. | p-value | Stat. | p-value | Stat. |
| $f_1$ | 6.898e-01 | $\leftrightarrow$ | **2.872e-11** | $\downarrow$ | **2.872e-11** | $\downarrow$ |
| $f_2$ | 5.946e-01 | $\leftrightarrow$ | **2.872e-11** | $\downarrow$ | **1.506e-42** | $\downarrow$ |
| $f_3$ | 2.818e-02 | $\leftrightarrow$ | **3.822e-75** | $\uparrow$ | **3.484e-75** | $\uparrow$ |
| $f_4$ | 5.969e-01 | $\leftrightarrow$ | **6.429e-25** | $\downarrow$ | **1.206e-21** | $\downarrow$ |
| $f_5$ | 2.996e-01 | $\leftrightarrow$ | **2.070e-36** | $\uparrow$ | **6.955e-35** | $\uparrow$ |
| $f_6$ | 7.515e-02 | $\leftrightarrow$ | **5.101e-05** | $\uparrow$ | **3.940e-03** | $\uparrow$ |
| $f_7$ | 7.967e-01 | $\leftrightarrow$ | **2.872e-11** | $\downarrow$ | **2.872e-11** | $\downarrow$ |
| $f_8$ | 9.058e-01 | $\leftrightarrow$ | **2.872e-11** | $\downarrow$ | **2.872e-11** | $\downarrow$ |
| $f_9$ | 2.691e-01 | $\leftrightarrow$ | **2.821e-34** | $\uparrow$ | **8.564e-33** | $\uparrow$ |
| $f_{10}$ | 7.627e-01 | $\leftrightarrow$ | **4.267e-06** | $\uparrow$ | **3.701e-06** | $\uparrow$ |
| $f_{11}$ | 6.933e-01 | $\leftrightarrow$ | **2.872e-11** | $\downarrow$ | **2.872e-11** | $\downarrow$ |
| $f_{12}$ | 9.882e-01 | $\leftrightarrow$ | **2.872e-11** | $\downarrow$ | **2.872e-11** | $\downarrow$ |
| $f_{13}$ | 8.712e-01 | $\leftrightarrow$ | **1.445e-27** | $\downarrow$ | **1.457e-28** | $\downarrow$ |
| $f_{14}$ | 6.528e-01 | $\leftrightarrow$ | **2.872e-11** | $\downarrow$ | **2.872e-11** | $\downarrow$ |
| $f_{15}$ | 5.444e-01 | $\leftrightarrow$ | **2.872e-11** | $\downarrow$ | **2.872e-11** | $\downarrow$ |

random distribution. However, in the same work, it was shown that the proposed control mechanism is suitable for more exploitative DE versions, and particularly, for DE/current-to-p-best/1/bin. The control mechanisms provided by JADE for adapting both parameters $F$ and $CR$, which were explained at the end of Section 3.2, were applied with an adaptation speed $c = 0.1$. Furthermore, parameter $p$ of the mutant generation strategy current-to-p-best/1 was fixed to a low value, i.e., $p = 0.05$, following the recommendations given by Zhang & Sanderson (2009).

To seek the best configuration of E-MBO and MBO, the same tuning approach performed at previous experiments was carried out herein. In this case, the best performing configurations of E-MBO and MBO, which we will refer to as E-MBO-8 and MBO-8, respectively, were applied with parameter values $n = 350$, $k = 11$, $m = 5$, and $x = 1$. In addition to the execution of E-MBO-8 and MBO-8, the adaptive DE/current-to-p-best/1/bin variant was also run as an independent algorithm. Since for this particular experiment a more exploitative version of DE was considered, its population size was set to $n = 350$ individuals with the aim of balancing the exploration and exploitation abilities of the algorithm. Finally, the stopping criterion for E-MBO-8, MBO-8, and the adaptive DE/current-to-p-best/1/bin was set to $3 \times 10^6$ function evaluations, as in the case of previous experiments.

Table 8 shows the mean, the median, and the standard deviation (SD) of the objective function error achieved by the three schemes at the end of their runs, for each of the considered problems. E-MBO-8 was able to provide the lowest mean and median of the error in 4 out of 15 problems ($f_5$, $f_6$, $f_9$, and $f_{10}$), while MBO-0 obtained the lowest mean and median of the error only for problem $f_3$. For the remaining 10 functions, DE/current-to-p-best/1/bin executed separately provided the lowest mean and median of the error.

Table 10: Pairwise statistical comparison between E-MBO-0 [Adaptive DE/rand/1/bin (JADE)] and MBO-0 [Adaptive DE/rand/1/bin (JADE)], and the adaptive version of DE/current-to-p-best/1/bin (JADE), considering their results achieved at the end of 30 executions for each problem $f_1$–$f_{15}$

| | E-MBO-0 vs. DE/current-to-p-best/1/bin (JADE) | | MBO-0 vs. DE/current-to-p-best/1/bin (JADE) | |
|---|---|---|---|---|
| Func. | p-value | Stat. | p-value | Stat. |
| $f_1$ | **2.872e-11** | ↑ | **2.872e-11** | ↑ |
| $f_2$ | **5.614e-26** | ↑ | **2.266e-26** | ↑ |
| $f_3$ | **3.446e-41** | ↑ | **1.140e-44** | ↑ |
| $f_4$ | **2.872e-11** | ↓ | **2.872e-11** | ↓ |
| $f_5$ | **8.431e-09** | ↑ | **1.509e-07** | ↑ |
| $f_6$ | 2.658e-02 | ↔ | 8.635e-02 | ↔ |
| $f_7$ | **2.872e-11** | ↓ | **2.872e-11** | ↓ |
| $f_8$ | **2.872e-11** | ↓ | **2.872e-11** | ↓ |
| $f_9$ | **2.035e-16** | ↑ | **5.958e-16** | ↑ |
| $f_{10}$ | 3.912e-01 | ↔ | 2.739e-01 | ↔ |
| $f_{11}$ | **2.872e-11** | ↓ | **2.872e-11** | ↓ |
| $f_{12}$ | **2.128e-09** | ↑ | **1.151e-08** | ↑ |
| $f_{13}$ | **2.239e-20** | ↓ | **3.688e-19** | ↓ |
| $f_{14}$ | **2.872e-11** | ↓ | **2.872e-11** | ↓ |
| $f_{15}$ | **2.996e-30** | ↓ | **4.999e-27** | ↓ |

Table 9, shows the pairwise statistical comparison among E-MBO-8, MBO-8, and the adaptive DE/current-to-p-best/1/bin, for each problem, from where the following observations can be extracted:

- Regarding the adaptive DE/current-to-p-best/1/bin, each of both MBO-based variants outperformed it in 5 out of 15 functions ($f_3$, $f_5$, $f_6$, $f_9$, and $f_{10}$), while the former was statistically better than both MBO-based schemes in the remaining 10 problems.
- E-MBO-8 and MBO-8 were not statistically outperformed by any other scheme in 4 and 5 problems, respectively.
- The adaptive version of DE/current-to-p-best/1/bin was not statistically outperformed by any other scheme in 10 out of 15 problems.

What we can highlight from this experiment is that **the hybridisation between each of both MBO-based variants and an adaptive DE version that promotes exploitation, such as DE/current-to-p-best/1/bin, seems to be not so useful** as the combination of those MBO-based schemes and a DE version that promotes exploration, like DE/rand/1/bin, as it was shown in previous experiments. In the particular case of this third experiment, executing DE/current-to-p-best/1/bin as a independent approach provides better overall performance.

### *4.4. Fourth experiment: comparison of best approaches in terms of overall performance*

Previous experiments have compared E-MBO and MBO, each of them applied with a neighbourhood operator based on a particular DE variant, in contrast to that DE version considered as a independent algorithm. In the first and second experiments, the hybridisation between each MBO variant and a neighbourhood operator based on the explorative DE/rand/1/bin, showed to be beneficial in comparison to that DE version run separately. Advantages were even more noticeable in the case of using the neighbourhood operator based on the adaptive DE/rand/1/bin. Only in the case of the third experiment, where the more exploitative version DE/current-to-p-best/1/bin was considered, the said DE variant run separately provided better overall performance than the MBO-based approaches. Bearing the above in mind, in the current experiment, we statistically compare the best MBO variants against the best DE version in terms of overall performance. Particularly, configurations E-MBO-0 and MBO-0 applied during the second experiment, i.e., those making use of the neighbourhood operator based on the adaptive DE/rand/1/bin, were chosen. In the case of DE, the adaptive DE/current-to-p-best/1/bin using the same parameterisation than that applied in the third experiment was selected. The results of that comparison are shown in Table 10.

As it can be observed, each of both MBO-based schemes was able to provide statistically better solutions than DE/current-to-p-best/1/bin in 6 out of 15 problems, while the latter attained statistically better results

Table 11: Mean, median, and standard deviation (SD) of the error achieved by E-MBO-0 [Adaptive DE/rand/1/bin (JADE)] and MBO-0 [Adaptive DE/rand/1/bin (JADE)], and the adaptive version of DE/current-to-p-best/1/bin (JADE), at the end of 30 executions for problems $s_1$–$s_{19}$

| Alg. | E-MBO-0 | | | MBO-0 | | | DE/current-to-p-best/1/bin (JADE) | | |
|---|---|---|---|---|---|---|---|---|---|
| Func. | Mean | Median | SD | Mean | Median | SD | Mean | Median | SD |
| $s_1$ | **1.648e-13** | 1.705e-13 | 4.047e-14 | 1.876e-13 | **1.421e-13** | 1.924e-13 | 1.070e-08 | 3.246e-09 | 2.830e-08 |
| $s_2$ | 1.161e+02 | 1.155e+02 | 4.179e+00 | 1.143e+02 | 1.154e+02 | 4.173e+00 | **8.991e+01** | **8.990e+01** | 8.730e-01 |
| $s_3$ | 2.738e+03 | 2.738e+03 | 1.766e+02 | **2.695e+03** | **2.657e+03** | 1.658e+02 | 3.281e+03 | 3.224e+03 | 2.067e+02 |
| $s_4$ | 1.687e+03 | 1.698e+03 | 8.136e+01 | **1.608e+03** | **1.593e+03** | 8.783e+01 | 2.230e+03 | 2.237e+03 | 3.331e+01 |
| $s_5$ | 1.019e-02 | 5.684e-14 | 4.855e-02 | **5.211e-14** | **5.684e-14** | 1.683e-14 | 1.077e-01 | 8.627e-03 | 2.188e-01 |
| $s_6$ | 1.437e+00 | 1.357e+00 | 2.216e-01 | **1.325e+00** | **1.286e+00** | 1.403e-01 | 7.191e+00 | 9.844e+00 | 4.331e+00 |
| $s_7$ | **8.492e-09** | 7.936e-10 | 2.744e-08 | 4.699e-05 | **7.139e-10** | 2.477e-04 | 1.941e-02 | 3.176e-03 | 8.106e-02 |
| $s_8$ | 1.192e+06 | 1.163e+06 | 1.417e+05 | 1.176e+06 | 1.190e+06 | 1.270e+05 | **2.372e+04** | **2.366e+04** | 1.574e+03 |
| $s_9$ | 1.663e+03 | 1.635e+03 | 1.647e+02 | **1.536e+03** | **1.508e+03** | 1.358e+02 | 2.663e+03 | 2.657e+03 | 7.284e+01 |
| $s_{10}$ | 1.285e+02 | 1.280e+02 | 1.013e+01 | **1.222e+02** | **1.218e+02** | 9.377e+00 | 6.571e+02 | 6.574e+02 | 1.160e+01 |
| $s_{11}$ | 1.677e+03 | 1.672e+03 | 1.487e+02 | **1.555e+03** | **1.564e+03** | 1.532e+02 | 2.712e+03 | 2.696e+03 | 6.369e+01 |
| $s_{12}$ | 1.796e+02 | 1.688e+02 | 4.023e+01 | **1.472e+02** | **1.377e+02** | 4.113e+01 | 9.984e+02 | 1.009e+03 | 8.123e+01 |
| $s_{13}$ | **2.324e+03** | 2.305e+03 | 2.950e+02 | 2.366e+03 | **2.265e+03** | 5.096e+02 | 3.521e+03 | 3.549e+03 | 2.526e+02 |
| $s_{14}$ | 1.304e+03 | 1.299e+03 | 7.954e+01 | **1.248e+03** | **1.238e+03** | 7.073e+01 | 1.697e+03 | 1.702e+03 | 3.042e+01 |
| $s_{15}$ | **1.251e+01** | **1.260e+01** | 3.268e+00 | 1.295e+01 | 1.283e+01 | 3.445e+00 | 1.384e+02 | 1.394e+02 | 5.898e+00 |
| $s_{16}$ | 5.254e+02 | 5.047e+02 | 8.597e+01 | **4.789e+02** | **4.922e+02** | 1.037e+02 | 2.389e+03 | 2.408e+03 | 7.012e+01 |
| $s_{17}$ | 2.166e+03 | 2.110e+03 | 1.981e+02 | **1.938e+03** | **1.944e+03** | 2.346e+02 | 4.390e+03 | 4.421e+03 | 1.230e+02 |
| $s_{18}$ | 7.471e+02 | 7.388e+02 | 4.423e+01 | **7.279e+02** | **7.315e+02** | 4.017e+01 | 8.384e+02 | 8.403e+02 | 1.291e+01 |
| $s_{19}$ | 7.527e+01 | 7.506e+01 | 9.193e+00 | **7.100e+01** | **7.057e+01** | 8.081e+00 | 4.833e+02 | 4.854e+02 | 1.427e+01 |

Table 12: Pairwise statistical comparison among E-MBO-0 [Adaptive DE/rand/1/bin (JADE)] and MBO-0 [Adaptive DE/rand/1/bin (JADE)], and the adaptive version of DE/current-to-p-best/1/bin (JADE), at the end of 30 executions for problems $s_1$–$s_{19}$

| | E-MBO-0 vs. MBO-0 | | E-MBO-0 vs. DE/current-to-p-best/1/bin (JADE) | | MBO-0 vs. DE/current-to-p-best/1/bin (JADE) | |
|---|---|---|---|---|---|---|
| Func. | p-value | Stat. | p-value | Stat. | p-value | Stat. |
| $s_1$ | 8.032e-02 | ↔ | **1.816e-11** | ↑ | **1.583e-11** | ↑ |
| $s_2$ | 1.984e-01 | ↔ | **3.105e-26** | ↓ | **2.872e-11** | ↓ |
| $s_3$ | 3.376e-01 | ↔ | **1.017e-15** | ↑ | **1.659e-17** | ↑ |
| $s_4$ | **5.999e-04** | ↓ | **3.184e-30** | ↑ | **1.260e-30** | ↑ |
| $s_5$ | 1.778e-02 | ↔ | **1.728e-08** | ↑ | **1.292e-11** | ↑ |
| $s_6$ | 5.276e-02 | ↔ | **2.961e-03** | ↑ | **2.103e-03** | ↑ |
| $s_7$ | 6.898e-01 | ↔ | **2.872e-11** | ↑ | **5.228e-11** | ↑ |
| $s_8$ | 6.514e-01 | ↔ | **1.954e-28** | ↓ | **1.270e-29** | ↓ |
| $s_9$ | **1.782e-03** | ↓ | **3.132e-29** | ↑ | **1.627e-36** | ↑ |
| $s_{10}$ | **1.471e-02** | ↓ | **1.700e-82** | ↑ | **1.350e-83** | ↑ |
| $s_{11}$ | **2.745e-03** | ↓ | **2.989e-31** | ↑ | **2.356e-32** | ↑ |
| $s_{12}$ | **3.194e-03** | ↓ | **3.789e-39** | ↑ | **3.891e-40** | ↑ |
| $s_{13}$ | 6.048e-01 | ↔ | **5.773e-11** | ↑ | **5.225e-09** | ↑ |
| $s_{14}$ | **5.731e-03** | ↓ | **4.556e-25** | ↑ | **9.359e-30** | ↑ |
| $s_{15}$ | 6.077e-01 | ↔ | **3.426e-55** | ↑ | **2.916e-56** | ↑ |
| $s_{16}$ | 6.320e-02 | ↔ | **1.485e-64** | ↑ | **3.396e-56** | ↑ |
| $s_{17}$ | **1.390e-04** | ↓ | **2.788e-44** | ↑ | **1.460e-40** | ↑ |
| $s_{18}$ | 8.385e-02 | ↔ | **1.398e-12** | ↑ | **3.260e-16** | ↑ |
| $s_{19}$ | 6.093e-02 | ↔ | **1.012e-64** | ↑ | **1.076e-61** | ↑ |

than the former in 7 functions. Only for test cases $f_6$ and $f_{10}$ no statistically significant differences arose. We can therefore conclude that **it is worth applying these particular MBO variants combined with an adaptive DE-based neighbourhood operator that promotes exploration, since both were able to perform significantly better than the best overall DE version considering 40% of the problems**.

For some problems, such as $f_3$ and $f_9$, among others, the MBO-based approaches provided better performance than DE, while for other test functions, like $f_4$ and $f_{13}$, the latter attained better performance than the former. As a result, it would be interesting to analyse if a smart selection procedure would allow the best performing approach to be automatically selected depending on the features of a given problem. In this particular case, the pool of candidate algorithms would not only consist of different MBO variants, but also of different DE versions. Nevertheless, as we previously mentioned at the end of Section 4.2, the above is out of the scope of this paper and will be addressed as future research.

### *4.5. Fifth experiment: comparison of best approaches in terms of overall performance with another test suite*

In order to strengthen the conclusions drawn from the experiments performed with the first test suite, the goal here is to study whether the MBO variants are able to provide competitive results in comparison to

DE for a completely different set of large scale continuous problems. In this case, the test suite consists of 19 scalable continuous functions ($s_1$–$s_{19}$), as mentioned at the beginning of Section 4. The number of decision variables was fixed to $D = 1000$, as in the case of previous experiments.

As in the case of the fourth experiment, E-MBO and MBO were applied together with the neighbourhood operator based on the adaptive DE/rand/1/bin. Particularly, configurations E-MBO-0 and MBO-0, i.e., those providing the best overall performance considering the second experiment, were executed. In the case of DE, the adaptive DE/current-to-p-best/1/bin was run by using the same parameterisation than that applied in the third experiment. Finally, and following the recommendations given by Lozano et al. (2011), a stopping criterion equal to $5000 \times D$ function evaluations, i.e., $5 \times 10^6$, was fixed for the three approaches.

Table 11 shows the mean, the median, and the standard deviation (SD) of the objective function error achieved by the three schemes at the end of their runs, for each of the problems $s_1$–$s_{19}$, while Table 12, shows the pairwise statistical comparison among them. As it can be observed, E-MBO-0 was able to provide the lowest mean and median of the error only for function $s_{15}$, while DE/current-to-p-best/1/bin attained the lowest mean and median of the error for problems $s_2$ and $s_8$. For problems $s_1$, $s_7$, and $s_{13}$, E-MBO-0 provided the lowest mean of the error, while MBO-0 achieved the lowest median. Finally, we note that MBO-0 obtained the lowest mean and median of the error for the remaining 13 problems.

Considering the results provided by E-MBO-0 and MBO-0 together, **the lowest mean and median of the error were provided by the MBO-based schemes in 17 out of 19 problems**, while DE/current-to-p-best/1/bin provided the lowest mean and median of the error in two test cases only. In fact, **each of both MBO variants statistically outperformed DE/current-to-p-best/1/bin considering those 17 functions**, which represents almost 90% of the tested problems. As in the case of the fourth experiment, we can therefore conclude that **it is worth applying MBO variants incorporating an adaptive DE-based neighbourhood operator that promotes exploration, since they were able to perform significantly better than the best overall DE version considering almost all problems of a completely different test suite**.

## 5. Conclusions and further research

We have described novel MBO variants to address continuous optimisation problems, and in particular, large scale ones. In addition to the original MBO algorithm, a new variation, which we termed as E-MBO, making use of a novel elitist leader replacement scheme has been proposed and studied. Furthermore, in order to enable the analysed MBO variants for dealing with continuous decision spaces, we have described methods by which they can be combined with novel neighbourhood operators based on different trial vector generation strategies of DE. A wide experimental evaluation has been performed through several well-known large scale continuous test suites, including comparisons of our hybrid proposals to those DE variants used for defining the neighbourhood operators considered as independent optimisation schemes.

We conclude that there is benefit to combining explorative DE variants with migrating-bird algorithms such as E-MBO/MBO. Results demonstrated that a particular combination of E-MBO/MBO with a neighbourhood operator based on an explorative DE variant, such as DE/rand/1/bin, was able to statistically outperform that DE version executed independently in a wide range of functions. In the case of the hybridisation between E-MBO/MBO and a neighbourhood operator based on the adaptive version of DE/rand/1/bin (JADE) advantages are even more noticeable. Nevertheless, the combination of E-MBO/MBO with a more exploitative DE flavour, like DE/current-to-p-best/1/bin, did not provide significant advantages with respect to that particular DE variant executed as an independent approach.

Considering the quality of the solutions achieved at the end of the executions by the best-performing MBO-based variants and DE, the former were able to obtain the best results in 40% of the test cases. Furthermore, experiments with a completely different set of problems showed that the best overall configurations of the MBO variants were able to statistically outperform the best overall configuration of DE in almost 90% of the test cases, thus strengthening the above conclusions. Finally, we remark that the contribution of the MBO-based schemes is even more noteworthy taking into account that DE has shown to be one of the best global continuous search strategies since its inception.

As the particular MBO-based approach or DE version providing the best performance changes depending on the features of the problem or test suite addressed, it would be worth exploring the possibility of applying smart mechanisms that automatically select the best-performing approach to be applied during the optimisation procedure. This way, the study of our approaches with regard to other meta-heuristics, as well as its usage in hyper-heuristics based on selection, might be a possibility to address the above issue. In addition, to expand the use of the proposed algorithms, novel replacement strategies, as well as new neighbourhood operators, not necessarily based on DE, could be investigated. For instance, since several variants of DE have been proposed in the related literature to deal with both continuous and discrete decision variables simultaneously, those DE versions have been applied to solve *Mixed Integer Linear Programming* (MILP) models (Zhang & Chen, 2017; Mohammadi et al., 2017). As a result, if the MBO variants presented in the current work were hybridised with those DE versions specifically proposed for solving MILP models, MBO may be applied to deal with such types of problems as well. Another interesting line of future work might be the application of our proposals to optimisation problems with lower dimensionalities, with the aim of analysing whether their behaviour is altered or not. Finally, we aim to adapt and apply our proposed MBO approaches to real-world applications involving the resolution of continuous optimisation problems, such as power engineering problems, the design of gas circuit breakers, and chamber design problems, among others.

## References


Alkaya, A. F., & Algin, R. (2015). Metaheuristic based solution approaches for the obstacle neutralization problem. *Expert Systems with Applications*, *42*, 1094–1105.

Alkaya, A. F., Algin, R., Sahin, Y., Agaoglu, M., & Aksakalli, V. (2014). Performance of migrating birds optimization algorithm on continuous functions. In Y. Tan, Y. Shi, & C. A. Coello Coello (Eds.), *Advances in Swarm Intelligence: 5th International Conference, ICSI 2014, Hefei, China, October 17-20, 2014, Proceedings, Part II* (pp. 452–459). Cham: Springer International Publishing. doi:10.1007/978-3-319-11897-0_51.

Bäck, T., Fogel, D., & Michalewicz, Z. (2000). *Evolutionary computation 1: basic algorithms and operators*. CRC Press.

Burkard, R. E., Karisch, S. E., & Rendl, F. (1997). QAPLIB – A quadratic assignment problem library. *Journal of Global Optimization*, *10*, 391–403. doi:10.1023/A:1008293323270.

Conradt, L. (2012). Models in animal collective decision-making: information uncertainty and conflicting preferences. *Interface Focus*, *2*, 226–240. doi:10.1098/rsfs.2011.0090.

Das, S., Abraham, A., & Konar, A. (2008). Particle swarm optimization and differential evolution algorithms: technical analysis, applications and hybridization perspectives. In Y. Liu, A. Sun, H. T. Loh, W. F. Lu, & E.-P. Lim (Eds.), *Advances of Computational Intelligence in Industrial Systems* (pp. 1–38). Berlin, Heidelberg: Springer. doi:10.1007/978-3-540-78297-1_1.

Das, S., Mullick, S. S., & Suganthan, P. (2016). Recent advances in differential evolution – An updated survey. *Swarm and Evolutionary Computation*, *27*, 1 – 30. doi:10.1016/j.swevo.2016.01.004.

Duman, E., & Elikucuk, I. (2013). Solving credit card fraud detection problem by the new metaheuristics migrating birds optimization. In I. Rojas, G. Joya, & J. Cabestany (Eds.), *Advances in Computational Intelligence: 12th International Work-Conference on Artificial Neural Networks, IWANN 2013, Puerto de la Cruz, Tenerife, Spain, June 12-14, 2013, Proceedings, Part II* (pp. 62–71). Berlin, Heidelberg: Springer. doi:10.1007/978-3-642-38682-4_8.

Duman, E., Uysal, M., & Alkaya, A. F. (2012). Migrating birds optimization: a new metaheuristic approach and its performance on quadratic assignment problem. *Information Sciences*, *217*, 65 – 77. doi:10.1016/j.ins.2012.06.032.

Ghosh, S., Das, S., Roy, S., Islam, S. M., & Suganthan, P. (2012). A differential covariance matrix adaptation evolutionary algorithm for real parameter optimization. *Information Sciences*, *182*, 199 – 219. doi:10.1016/j.ins.2011.08.014.

Karafotias, G., Hoogendoorn, M., & Eiben, A. E. (2015). Parameter control in evolutionary algorithms: trends and challenges. *IEEE Transactions on Evolutionary Computation*, *19*, 167–187. doi:10.1109/TEVC.2014.2308294.

Kazimipour, B., Li, X., & Qin, A. (2014). Effects of population initialization on differential evolution for large scale optimization. In *2014 IEEE Congress on Evolutionary Computation (CEC)* (pp. 2404–2411). doi:10.1109/CEC.2014.6900624.

Kazimipour, B., Li, X., & Qin, A. K. (2013). Initialization methods for large scale global optimization. In *2013 IEEE Congress on Evolutionary Computation (CEC)* (pp. 2750–2757).

Lalla-Ruiz, E., de Armas, J., Expósito-Izquierdo, C., Melián-Batista, B., & Moreno-Vega, J. M. (2017). Multi-leader migrating birds optimization: a novel nature-inspired metaheuristic for combinatorial problems. *International Journal of Bio-Inspired Computation*, *10*, 89–98.

Lalla-Ruiz, E., Expósito-Izquierdo, C., de Armas, J., Melián-Batista, B., & Moreno-Vega, J. M. (2015). Migrating birds optimization for the seaside problems at container terminals. *Applied Mathematics*, *2015*, 1–12. doi:10.1155/2015/781907.

Lalla-Ruiz, E., Segredo, E., Voß, S., Hart, E., & Paechter, B. (2016). Analysing the performance of migrating birds optimisation approaches for large scale continuous problems. In J. Handl, E. Hart, R. P. Lewis, M. López-Ibáñez, G. Ochoa, & B. Paechter (Eds.), *Parallel Problem Solving from Nature – PPSN XIV: 14th International Conference, Edinburgh, UK, September 17-21, 2016, Proceedings* (pp. 134–144). Cham: Springer International Publishing. doi:10.1007/978-3-319-45823-6_13.

LaTorre, A., Muelas, S., & Peña, J. M. (2013). Large scale global optimization: experimental results with MOS-based hybrid algorithms. In *2013 IEEE Congress on Evolutionary Computation* (pp. 2742–2749). doi:10.1109/CEC.2013.6557901.

LaTorre, A., Muelas, S., & Peña, J. M. (2015). A comprehensive comparison of large scale global optimizers. *Information Sciences*, *316*, 517 – 549. doi:10.1016/j.ins.2014.09.031.

León, C., Miranda, G., & Segura, C. (2009). METCO: a parallel plugin-based framework for multi-objective optimization. *International Journal on Artificial Intelligence Tools*, *18*, 569–588. doi:10.1142/S0218213009000275.

Li, X., Tang, K., Omidvar, M., Yang, Z., & Qin, K. (2013). *Benchmark Functions for the CEC'2013 Special Session and Competition on Large Scale Global Optimization*. Technical Report Evolutionary Computation and Machine Learning Group, RMIT University Australia.

Lozano, M., & García-Martínez, C. (2010). Hybrid metaheuristics with evolutionary algorithms specializing in intensification and diversification: overview and progress report. *Computers & Operations Research*, *37*, 481 – 497. doi:10.1016/j.cor.2009.02.010.

Lozano, M., Molina, D., & Herrera, F. (2011). Editorial scalability of evolutionary algorithms and other metaheuristics for large-scale continuous optimization problems. *Soft Computing*, *15*, 2085–2087. doi:10.1007/s00500-010-0639-2.

Mahdavi, S., Shiri, M. E., & Rahnamayan, S. (2015). Metaheuristics in large-scale global continues optimization: a survey. *Information Sciences*, *295*, 407 – 428. doi:10.1016/j.ins.2014.10.042.

Makas, H., & Yumusak, N. (2013). New cooperative and modified variants of the migrating birds optimization algorithm. In *2013 International Conference on Electronics, Computer and Computation (ICECCO)* (pp. 176–179). doi:10.1109/ICECCO.2013.6718257.

Mohammadi, M., Dantan, J.-Y., Siadat, A., & Tavakkoli-Moghaddam, R. (2017). A bi-objective robust inspection planning model in a multi-stage serial production system. *International Journal of Production Research*, *0*, 1–26. doi:10.1080/00207543.2017.1363425.

Molina, D., & Herrera, F. (2015). Iterative hybridization of DE with local search for the CEC'2015 special session on large scale global optimization. In *2015 IEEE Congress on Evolutionary Computation (CEC)* (pp. 1974–1978). doi:10.1109/CEC.2015.7257127.

Neri, F., & Tirronen, V. (2010). Recent advances in differential evolution: a survey and experimental analysis. *Artificial Intelligence Review*, *33*, 61–106. doi:10.1007/s10462-009-9137-2.

Niroomand, S., Hadi-Vencheh, A., Şahin, R., & Vizvári, B. (2015). Modified migrating birds optimization algorithm for closed loop layout with exact distances in flexible manufacturing systems. *Expert Systems with Applications*, *42*, 6586–6597.

Omran, M. G., Engelbrecht, A. P., & Salman, A. (2009). Bare bones differential evolution. *European Journal of Operational Research*, *196*, 128 – 139. doi:10.1016/j.ejor.2008.02.035.

Oz, D. (2017). An improvement on the migrating birds optimization with a problem-specific neighboring function for the multi-objective task allocation problem. *Expert Systems with Applications*, *67*, 304–311.

Pan, Q.-K., & Dong, Y. (2014). An improved migrating birds optimisation for a hybrid flowshop scheduling with total flowtime minimisation. *Information Sciences*, *277*, 643 – 655. doi:10.1016/j.ins.2014.02.152.

Parpinelli, R., & Lopes, H. (2011). New inspirations in swarm intelligence: a survey. *International Journal of Bio-Inspired Computation*, *3*, 1–16. doi:10.1504/IJBIC.2011.038700.

Pholdee, N., & Bureerat, S. (2013). Hybridisation of real-code population-based incremental learning and differential evolution for multiobjective design of trusses. *Information Sciences*, *223*, 136 – 152. doi:10.1016/j.ins.2012.10.008.

Segredo, E., Lalla-Ruiz, E., Hart, E., Paechter, B., & Voß, S. (2016). Hybridisation of evolutionary algorithms through hyper-heuristics for global continuous optimisation. In P. Festa, M. Sellmann, & J. Vanschoren (Eds.), *Learning and Intelligent Optimization: 10th International Conference, LION 10, Ischia, Italy, May 29 – June 1, 2016, Revised Selected Papers* (pp. 296–305). Cham: Springer International Publishing.

Segura, C., Coello, C. A. C., Segredo, E., & Aguirre, A. H. (2016). A novel diversity-based replacement strategy for evolutionary algorithms. *IEEE Transactions on Cybernetics*, *46*, 3233–3246. doi:10.1109/TCYB.2015.2501726.

Segura, C., Coello, C. A. C., Segredo, E., & León, C. (2015). On the adaptation of the mutation scale factor in differential evolution. *Optimization Letters*, *9*, 189–198. doi:10.1007/s11590-014-0723-0.

Shen, L., Asmuni, H., & Weng, F. (2015). A modified migrating bird optimization for university course timetabling problem. *Jurnal Teknologi*, *72*, 89–96. doi:10.11113/jt.v72.2949.

Soto, R., Crawford, B., Almonacid, B., & Paredes, F. (2016). Efficient parallel sorting for migrating birds optimization when solving machine-part cell formation problems. *Scientific Programming*, *2016*, 1–39. doi:10.1155/2016/9402503.

Stanarevic, N. (2012). Hybridizing artificial bee colony (ABC) algorithm with differential evolution for large scale optimization problems. *International Journal of Mathematics and Computers in Simulation*, *6*, 194–202.

Storn, R., & Price, K. (1997). Differential evolution – A simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, *11*, 341–359. doi:10.1023/A:1008202821328.

Sumpter, D. (2006). The principles of collective animal behaviour. *Philosophical Transactions of the Royal Society of London B: Biological Sciences*, *361*, 5–22. doi:10.1098/rstb.2005.1733.

Tan, Y., Li, J., & Zheng, Z. (2015). ICSI 2014 competition on single objective optimization (ICSI-2014-BS). *arXiv preprint arXiv:1501.02128*, .

Tvrdík, J., Poláková, R., Veselský, J., & Bujok, P. (2013). Adaptive variants of differential evolution: towards control-parameter-free optimizers. In I. Zelinka, V. Snášel, & A. Abraham (Eds.), *Handbook of Optimization: From Classical to Modern Approach* (pp. 423–449). Berlin, Heidelberg: Springer. doi:10.1007/978-3-642-30504-7_17.

Xing, B., & Gao, W.-J. (2014). Emerging biology-based CI algorithms. In *Innovative Computational Intelligence: A Rough Guide to 134 Clever Algorithms* (pp. 217–317). Cham: Springer International Publishing. doi:10.1007/978-3-319-03404-1_17.

Yang, X.-S. (2008). *Nature-inspired metaheuristic algorithms*. Luniver Press.

Zhang, J., & Sanderson, A. C. (2009). JADE: adaptive differential evolution with optional external archive. *IEEE Transactions*

*on Evolutionary Computation*, *13*, 945–958. doi:10.1109/TEVC.2009.2014613.

Zhang, X. Y., & Chen, L. (2017). A re-entrant hybrid flow shop scheduling problem with machine eligibility constraints. *International Journal of Production Research*, *0*, 1–13. doi:10.1080/00207543.2017.1408971.