Memetic Comp. (2013) 5:295–309
DOI 10.1007/s12293-013-0120-8

REGULAR RESEARCH PAPER

Evaluating the Multiple Offspring Sampling framework on complex continuous optimization functions

Antonio LaTorre · Santiago Muelas · José-María Peña

Received: 26 June 2012 / Accepted: 29 May 2013 / Published online: 7 June 2013

Abstract In this contribution we present a study on the combination of Differential Evolution and the IPOP-CMA-ES algorithms. The hybrid algorithm has been constructed by using the Multiple Offspring Sampling framework, which allows the seamless combination of multiple metaheuristics in a dynamic algorithm capable of adjusting the participation of each of the composing algorithms according to their current performance. In this study we analyze the existing synergies, if any, emerging from the combination of the two algorithms. For this purpose, the COCO suite used in BBOB 2009 and 2010 Workshops has been used. The experimental results on the noiseless testbed show a robust behavior of the algorithm and a good scalability as the dimensionality increases. In the noisy testbed, the algorithm shows a good performance on functions with moderate to severe noise.

Keywords Hybridization · Multiple Offspring Sampling · Evolution Strategies · IPOP-CMA-ES · Differential Evolution · Benchmarking · Continuous optimization

A. LaTorre (✉) · S. Muelas · J.-M. Peña
Department of Computer Systems Architecture and Technology, Facultad de Informática, Universidad Politécnica de Madrid, Boadilla del Monte, 28660 Madrid, Spain
e-mail: atorre@fi.upm.es

S. Muelas
e-mail: smuelas@fi.upm.es

J.-M. Peña
e-mail: jmpena@fi.upm.es

A. LaTorre
Instituto Cajal, Consejo Superior de Investigaciones Científicas (CSIC), Madrid, Spain

1 Introduction

Continuous optimization is a field of research which is getting more and more attention in the last years. Many real-world problems from very different domains (biology, engineering, data mining, etc.) can be formulated as the optimization of a continuous function. These problems have been tackled using Evolutionary Algorithms (EA) [17] or similar metaheuristics [30].

Selecting an appropriate algorithm to solve a continuous optimization problem is not a trivial task. Although a particular algorithm can be configured to perform properly in a given scale of problems (considering the number of variables as their dimensionality), the behavior of the algorithm can degrade as this dimensionality increases, even if the nature of the problem remains the same.

In this contribution, the Multiple Offspring Sampling (MOS) [15] framework has been used to combine an Increasing Population Covariance Matrix Adaptation Evolution Strategy (IPOP-CMA-ES) and a Differential Evolution (DE) algorithm. This framework allows the combination of different metaheuristics following an HRH (High-level Relay Hybrid) approach (this nomenclature will be reviewed in Sect. 2) in which the number of evaluations that each algorithm can carry out is dynamically adjusted.

The purpose of this study is two-fold. On the first hand, we want to study if a successful hybridization of Differential Evolution and the IPOP-CMA-ES algorithms is possible. Our hypothesis is that it should be possible to combine both algorithms to exploit the strength of each algorithm in several groups of functions. In this sense, several functions from the benchmark will be studied in detail to see if some synergies emerge from this combination (Sect. 5.1) that help one algorithm to overcome its limitations on certain groups of functions and vice-versa. On the second hand, we will

present the results on the benchmark proposed in the COCO suite [11,13] used at the BBOB 2009 and 2010 Workshops to offer a general view of the performance of the hybrid algorithm proposed in this contribution. In particular, a detailed description of the experimentation conducted for this work (including a thorough parameter tuning phase) is provided, as well as the full results obtained on both the noiseless and the noisy testbeds and an analysis presenting the main conclusions extracted from this study.

The rest of the paper is organized as follows: in Sect. 2, relevant related work is briefly reviewed. Section 3 details the proposed algorithm. In Sect. 4 the experimental scenario and the parameter tuning carried out are described. Section 5 presents and comments on the results obtained and lists the most relevant facts from this analysis. Finally, Sect. 6 contains the concluding remarks obtained from this work.

2 Related work

The HRH terminology was introduced in [28], one of the first attempts to define a complete taxonomy of hybrid metaheuristics. This taxonomy is a combination of a hierarchical and a flat classification structured into two levels. The first level defines a hierarchical classification in order to reduce the total number of classes, whereas the second level proposes a flat classification, in which the classes that define an algorithm may be chosen in an arbitrary order. From this taxonomy, the following four basic hybridization strategies can be derived, according to how the combination of the different algorithms is implemented: (a) LRH (Low-level relay hybrid): one metaheuristic is embedded into a single-solution metaheuristic. (b) HRH (High-level relay hybrid): two metaheuristics are executed in sequence. (c) LTH (Low-level teamwork hybrid): one metaheuristic is embedded into a population-based metaheuristic. (d) HTH (High-level teamwork hybrid): two metaheuristics are executed in parallel. For this work, we review some relevant work on the HRH group, the one the algorithm proposed in this paper belongs to. In the last years there has been an intense research in HRH algorithms, combining different types of metaheuristics. In the following subsections we will review some relevant approaches combining DE with other techniques, CMA-ES with other techniques and, finally, some similar approaches to the MOS framework described in this paper, stressing the main differences between these algorithms and our contribution.

2.1 HRH algorithms combining DE with other techniques

The DE algorithm is one of the Evolutionary Algorithms that has been recently hybridized using this kind of strategy. In the following paragraphs some of the most recent and representative approaches will be reviewed.

Gao and Wang [7] proposed CSDE1, a memetic DE algorithm, to optimize thirteen 30-dimensional continuous problems. CSDE1 uses simplex (Nelder–Mead method) to carry out the Local Search (LS) using also chaotic systems to create the initial population. CSDE1 applies the local search only to the best individual in the population at each generation.

Tirronen et al. [29] designed a hybrid DE algorithm that combined the Hooke–Jeeves Algorithm (HJA) and the Stochastic Local Search (SLS), coordinated by an adaptive rule that estimates fitness diversity using the ratio between the standard deviation and the average fitness of the population. This algorithm was compared against a regular DE and an Evolution Strategy (ES) on the problem of weighting coefficients to detect defects in paper production.

The FAMA (Fast Adaptive Memetic Algorithm) proposed in [2], is a memetic algorithm with a dynamic parameter setting and two local searchers adaptively launched, either one by one or simultaneously, according to the needs of the evolution. The employed local search methods are: the Hooke–Jeeves and the Nelder–Mead methods. The Hooke–Jeeves method is executed only on the elite individual, whereas the Nelder–Mead simplex is carried out on 11 randomly selected individuals. FAMA includes a self-adaptive criterion based on a fitness diversity measure and the iteration number. Mutation probability and other search parameters depend also on the diversity measure. The FAMA algorithm was compared against Tirronen's algorithm and SFMDE obtaining better results for the problem of permanent magnet synchronous motors [3].

2.2 HRH algorithms combining CMA-ES with other techniques

Contrary to DE, there has not been the same effort in combining CMA-ES with other algorithms, probably due to the difficulty of sharing and adjusting the inner information of this algorithm.

In [9], the authors proposed a VNS algorithm in which the intensification component of the algorithm was a CMA-ES algorithm. A similar approach was followed by Molina et al. [19], who proposed a memetic algorithm with local search chains in which the IPOP-CMA-ES algorithm is used as an intensification algorithm.

In [21] a slightly different approach is followed. In this paper, the authors propose a hybrid of Particle Swarm Optimization (PSO) and CMA-ES, in which several instances of the CMA-ES algorithm run in parallel as particles of the swarm. The rationale behind this combination is to exploit the global exploration capabilities of PSO and the local exploration power of CMA-ES.

Kämpf and Robinson [14] combine Differential Evolution and CMA-ES in an HRH scheme, in which one algorithm is applied after the other for a prefixed number of generations. Experimental results show emergent synergies between both algorithms: the hybrid algorithm has similar convergence speed to that of CMA-ES and similar robustness to that of Differential Evolution.

Furthermore, both algorithms have obtained remarkable results in previous sessions and competitions on continuous optimization [1,23,24]. For this reason, and to explore new approaches for combining the IPOP-CMA-ES algorithm with other metaheuristics, we have proposed a hybrid algorithm combining both of them, which is benchmarked on the BBOB 2010 testbed.

2.3 Similar approaches to the MOS framework

In this section we review two relevant similar approaches to the MOS framework, comparing their characteristics and highlighting the main differences among them.

In [18], an Evolutionary Algorithm with competing heuristics is presented. In this algorithm, several heuristics are considered to create new candidate solutions. The probability to select one of the available heuristics is proportional to a measure of how successful one of the heuristics has been in creating new better solutions. These probabilities can be reset through the evolutionary process if they reach some prefixed critical values. Compared to MOS, there are some important differences. First, before creating a new solution, this algorithm must select the heuristic to be used according to its probability, which may not be suitable to combine any type of heuristic that may require a budget of evaluations to be useful. Second, the success measure is cumulative, until a reset is carried out, which means that the adjustment of the probability of selecting one heuristic can be very slow in some situations in which different heuristics result to be more effective at different stages of the search process.

In [22], a Population-based Algorithm Portfolios is presented and evaluated in numerical optimization problems. This algorithm is based on the theory of investment portfolios developed in the field of economics and tries to minimize the "risk" (probability of not reaching the optimum) by combining several population-based algorithms. Compared to the algorithm proposed in this paper, there are several differences. On the first hand, MOS works with a shared overall population, whereas the algorithm proposed by Peng et al. assigns an independent population to each algorithm. On the other hand, MOS does not impose a distributed architecture: it can be used within an island model, but this is not embedded in its design. From this point of view, MOS could be seen as a more general algorithm, as it allows both centralized and distributed implementations of hybrid algorithms.

3 Algorithm presentation

Multiple Offspring Sampling (MOS) is a framework for the development of Dynamic Hybrid Evolutionary Algorithms [15]. MOS provides the functional formalization necessary to design this type of algorithms, as well as the tools to identify and select the best performing configuration for the problem under study. In this context, the hybridization of several algorithms can lead to the following two situations:

- A collaborative synergy emerges among the different algorithms that improves the performance of the best one when it is used individually.
- A competitive selection of the best one takes place, in which the final performance is equivalent to the best competing algorithms, with a minimum overhead.

If we were given to choose, the first scenario would be obviously selected. Ideally, we would like the hybrid algorithm to be able to combine several complementary heuristics and obtain better results than each algorithm separately. However, this is not always possible and, in those cases, hybridization might still be useful if it alleviates the task of selecting the best algorithm for a particular problem. As we will show in Sect. 5.1, the hybrid algorithm proposed in this contribution belongs to this second category and allows the automatic selection of the best performing algorithm (between DE and IPOP-CMA-ES) for each function of the considered benchmark.

In the following sections we will review the Multiple Offspring Sampling framework (Sect. 3.1) and the hybrid algorithm proposed for this work based on the MOS framework (Sect. 3.2).

3.1 Multiple Offspring Sampling

In MOS, a key term is the concept of *technique*, which is a mechanism, decoupled from the main algorithm, to generate new candidate solutions. This means that, within a MOS-based algorithm, several techniques can be used simultaneously, and it is the main algorithm which selects among the available optimization techniques the most appropriate ones for the particular problem and search phase. A more concrete definition for these reproductive mechanisms follows:

Definition 1 A MOS reproductive technique is a mechanism to create new individuals in which: (a) a particular evolutionary algorithm model, (b) an appropriate solution encoding, (c) specific operators (if required), and (d) necessary parameters have been defined.

Furthermore, the use of multiple reproductive mechanisms simultaneously has to be controlled in some way. First,

we should define periodical checkpoints in which the participation of each technique in the overall process is adjusted according to its current performance. The periods between these checkpoints are the steps of our algorithm. Then, we should define the number of individuals that each technique j can generate at each step i ($\Pi_i^{(j)}$), which is called its participation ratio. This ratio is uniformly distributed at the beginning of the search process, and it is periodically updated according to a given policy. In the canonical version of MOS, this adjustment is carried out by what is known as a *Participation Function*. These functions can carry out simple static assignments or, more interestingly, dynamic adjustments according to a *Quality Measure* that evaluates how good the offspring of each technique j is during the step i from the point of view of that measure ($Q_i^{(j)}$).

At this point, different measures can be proposed, depending on the concept of quality that we consider. One option would be to consider the Fitness Average of the subset of the best individuals of the offspring generated by each technique as our measure for quality. Other alternatives could be used (Negative Slope Coefficient by Vanneschi [31], age, diversity, etc.). If the Fitness Average measure is selected, the quality value computed from the offspring population produced by a technique j during the step i ($O_i^{(j)}$) is obtained as defined in Eq. 1.

$$Q_i^{(j)} = \sum_{o \in O_i^{(j)}} \frac{fit(o)}{|O_i^{(j)}|} \tag{1}$$

At the end of each step, the quality of each of the available techniques is recomputed, according to the Quality Function (QF) under consideration. These quality values are then used by a Dynamic Participation Function (PF) to adjust the number of Fitness Evaluations allocated for each technique at each step (Eq. 2). This PF computes, at each step, a trade-off factor for each technique, $\Delta_i^{(j)}$, that represents the decrease in participation for the $j\text{-}th$ technique at the $i\text{-}th$ step, for every technique except the best performing ones. These techniques will increase their participation by the sum of all those $\Delta_i^{(j)}$ divided by the number of techniques with the best quality values.

$$\Pi_i^{(j)} = \begin{cases} \Pi_{i-1}^{(j)} + \eta & \text{if } j \in \Omega, \\ \Pi_{i-1}^{(j)} - \Delta_i^{(j)} & \text{otherwise} \end{cases} \tag{2}$$

$$\eta = \frac{\sum_{k \notin \Omega} \Delta_i^{(k)}}{|\Omega|}$$

$$\Omega = \left\{ l / Q_i^{(l)} \geq Q_i^{(m)} \, \forall l, m \in [1, n] \right\}$$

$$Q_i^{max} = max \left\{ Q_i^{(j)} \forall j \in [1, n] \right\}$$

The above-mentioned $\Delta_i^{(j)}$ values are computed as shown in Eq. 3. These $\Delta_i^{(j)}$ factors are computed from the relative difference between the quality of the *best* and the $j\text{-}th$ techniques, n being the number of available techniques. In this equation, ξ represents a reduction factor, i.e., the ratio that is transferred from one technique to the other(s). Finally, a minimum participation ratio can be established to guarantee that all the techniques are represented through all the search. This is done to avoid, if possible, premature convergence to undesired solutions caused by a technique that obtains all the participation in the early steps of the search and quickly converges to poor regions of the solution space, preventing the other techniques to collaborate at later stages of the process, in which they could be more beneficial.

$$\Delta_i^{(j)} = \xi \cdot \frac{Q_i^{max} - Q_i^{(j)}}{Q_i^{max}} \cdot \Pi_{i-1}^{(j)} \quad \forall j \in [1, n] / j \notin \Omega \tag{3}$$

Finally, the Multiple Offspring Sampling framework allows the development of both HTH (High-level Teamwork Hybrid) and HRH (High-level Relay Hybrid) algorithms (according to Talbi's nomenclature [28]). In the case of the HTH algorithms, two metaheuristics are executed in parallel, working at the same time on the resolution of the problem. On the other hand, in the case of the HRH algorithms, two metaheuristics are executed in sequence, one after the other, and changes of the executing algorithm are carried out according to a given policy. As the proposed algorithm is of the HRH type, more attention will be paid to this type of algorithms.

Algorithm 1 HRH MOS Algorithm

```
Create initial overall population of candidate solutions P_0
Initialize step length: FEs_0 = |P_0| · φ
Uniformly distribute participation among the n used techniques (T_j):
    ∀j Π_0^(j) = |P_0| / n
    Each technique produces a subset of individuals according to its participation:
    FEs_0^(j) = Π_0^(j) · FEs_0
Evaluate initial population P_0
while number of evaluations not exceeded do
    Start new step i
    Update Quality of T_j computed as the average quality of all the individuals
        created by technique T_j in step i − 1
    Update participation ratios from Quality values computed in Step 7:
        ∀j Π_i^(j) = PF(Q_i^(j))
    Update FEs allocated for each technique at current step:
        ∀j FEs_i^(j) = Π_i^(j) · FEs_i
    for every available technique T_j do
        while FEs_i^(j) not exceeded do
            Evolve
        end while
    end for
    if restart needed then
        Increase population size: |P_{i+1}| = 2 · |P_i|
        Restart techniques information and population
    end if
    Adjust step length to its new value: FEs_{i+1} = |P_{i+1}| · φ
end while
```

In terms of the MOS framework, the available techniques in a MOS-based HRH hybrid algorithm are used in sequence, one after the other, each of them reusing the output population of the previous technique. This approach fits better when there are non-population-based techniques, such as local searches, or algorithms with very different offspring sampling mechanisms as techniques are not constrained to produce a % of the common population. If different population sizes are used by different techniques, it is the responsibility of the technique to make grow/shrink the population in order to adjust it to its needs and to return a population of an appropriate size to the next technique. For example, if a population-based algorithm is combined with a local search, the latter could select one or more individuals from the output population of the population-based algorithm, modify them as needed and then include them in the original population by means of a predefined elitism procedure.

In this type of algorithms, the search process can be divided into a fixed number of *steps*, which is established at the beginning of the execution (as it was shown in [16]), or a dynamic number of steps, which depends on the adjustment of the population size conducted after restarting the algorithm (Sect. 3.2 describes this restarting in detail). In the first case, each step is assigned a fixed amount of Fitness Evaluations (FEs_i in Algorithm 1) that will not change through the execution of the algorithm. In the second case, which is the alternative used in this study, the step length depends on the population size at each phase of the search process. Its length is computed by multiplying the current population size by a step factor (ϕ in Algorithm 1). This way we can make the steps more homogeneous in time or, on the other hand, adapt their length to adjust how quick the participation adjustment should be done. In both cases, the evaluations assigned to each step, regardless they have been assigned statically or dynamically, are distributed by the Participation Function (PF).

Each technique can manage its number of allocated FEs at each step of the algorithm ($FEs_i^{(j)}$) in its own particular way. For example, a population-based technique, such as Differential Evolution, could execute several iterations of the algorithm, whereas a Local Search could decide to spend all its assigned evaluations in improving just one individual. The quality of the new individuals of each technique will be averaged at the end of the whole set of evaluations, as the division of the search into generations depends on each of the techniques.

The pseudocode given in Algorithm 1 summarizes the way an HRH MOS algorithm works. The reader is referred to [15] for more information on the MOS framework.

3.2 Proposed algorithm

In this contribution, an HRH Dynamic algorithm is proposed. This algorithm combines the explorative/exploitative strength of two heuristic search methods, that separately have proven to obtain very competitive results in either low or high dimensional problems. These algorithms are: the IPOP-CMA-ES algorithm [1], the best algorithm of the "*Special Session on Real-Parameter Optimization*" held at the CEC 2005 Congress, and the DE algorithm [27] which has demonstrated to obtain competitive results when executed independently and when combined with other algorithms [7,20].

For the adjustment of the participation of each technique in the overall search process, the Participation Function described in Eqs. 2 and 3 has been used. To measure the quality of the different techniques, the Quality Function in Eq. 4 has been proposed. This QF takes into account two desirable characteristics in a search algorithm: the Average Fitness Increment of the newly created individuals after a set of allocated Fitness Evaluations and the number of times that these improvements take place (Eq. 4). To compute the Average Fitness Increment we first need to obtain the differences between the fitness of each new individual and the average fitness of its parents per technique and step. In the case of Differential Evolution, those parents are the three differential vectors and the donor vector, whereas in the IPOP-CMA-ES algorithm the parents are the whole previous population. Then, the Average Fitness Increment of each technique is computed as the mean of all these differences.

This Quality Function uses the Average Fitness Increment as the effective QF only if there is consensus among both measures. If this is not the case, the raw number of fitness improvements is used. The logic behind this function is that, in some functions, the use of the Average Fitness Increment QF could impose too much selection pressure reducing also population diversity. As a result of this, the MOS combination strategy could be biased towards suboptimal solutions. In some particular situations, a technique which is not carrying out an effective search could introduce, for some reason, a large increment in the average fitness value of the new individuals. This could be due, for example, to a recombination of poor solutions. In such a case, it is easy for a technique to improve previous solutions. However, it could be more adequate to carry out small changes to good individuals in order to find the right "path" to the global optimum rather than carrying out substantial modifications to poor solutions. For this reason, a consensus of both measures is required in order to apply the more elitist Average Fitness Increment QF. If this is not the case, the number of fitness improvements is

used to guarantee a softer adjustment of participation.

$$Q_i^{(j)} = \begin{cases} \Sigma_{i-1}^{(j)} & if\ \Sigma_{i-1}^{(j)} > \Sigma_{i-1}^{(k)} \wedge \\ & \Gamma_{i-1}^{(j)} > \Gamma_{i-1}^{(k)} \qquad \forall j,k \in [1,n] \\ \Gamma_{i-1}^{(j)} & otherwise \end{cases} \tag{4}$$

$Q_i^{(j)} \equiv$ Quality of technique T_j in step i

$\Sigma_i^{(j)} \equiv$ Average Fitness Increment of T_j in step i

$\Gamma_i^{(j)} \equiv$ Number of Fitness improvements of T_j in step i

To summarize, the presented algorithm works as follows. All the available techniques are allocated the same number of FEs at the beginning of the execution. At the end of each step, the quality of the new solutions created by each technique is evaluated and, based on this quality, its participation ratio is adjusted accordingly. This participation ratio is used to compute the number of FEs that each technique will be allowed to use in the next step of the search. If a minimum participation ratio has been established, then the number of FEs can not go below this threshold.

If a restart is needed by any of the techniques, then the population and the step length are readjusted to their new values. In this case, a restart mechanism, similar to the one used by the IPOP-CMA-ES algorithm, was also used within the proposed algorithm. With this strategy, the algorithm is halted whenever a restart stopping criteria is met, reinitializing the population and increasing its size by a factor of two until a maximum population size is reached. As this restart mechanism depends on some specific conditions of the IPOP-CMA-ES technique, the restart can only take place when this technique is being executed. However, the effect of the restart affects to all the available techniques, as it is the overall population which is restarted. Moreover, the framework easily allows the use of additional restart mechanisms associated to the remaining techniques or overall restart mechanisms independent of these techniques.

4 Parameter tuning

The results reported for this work have been obtained from 50 independent executions of each of the functions in both the noiseless and the noisy testbeds described in [11,12] executed on the computer configuration displayed in Table 1.

Table 1 Computer configuration

PC	Intel Xeon 8 cores 1.86 Ghz CPU
Operating system	Ubuntu Linux 8.04
Prog. Language	C++
Compiler	GNU C++ 4.3.2

In this work, we have conducted a thorough parameter tuning study to analyze the influence of the selection of different values for some of the parameters of the algorithm on the performance of the hybrid algorithm. In particular, we have focused our attention on the key parameters of the hybrid algorithm: initial population size, maximum populations size (after restarts) and the step factor used to determine the length of one step of the algorithm.

Table 2 contains the parameters that have been kept fixed throughout the study. In particular, we have used recommended parameter values for both composing algorithms (Differential Evolution and IPOP-CMA-ES) as suggested in [1,10,17]. Regarding the minimum participation ratio and the reduction factor ξ, it has been fixed as recommended in previous studies [15].

On the other hand, Tables 4 and 5 contain the detailed information about the parameter tuning conducted for this study. They show the parameters of the hybrid algorithm that have been subject to adjustment. These parameters have been considered because they could play a key role in the convergence capability of the algorithm. As it can be seen, for the (initial) population size six different values have been tested. In the case of the maximum population size (after restarts), three different values have been considered. Finally, the same number of different values have been tested for the step factor parameter. All the possible combinations for these values of those three parameters have been tested, which becomes a total number of 54 configurations. The same set of values have been considered for both testbeds, although they have been adjusted independently, in order to be able to compare how differently the algorithm has to be set up to work properly on different sets of functions. In both cases, to avoid overfitting to any of the testbeds, the parameter tuning has been conducted on a subset of functions, randomly selecting one or two representatives, depending on the size, from each group of functions (i.e., separable functions, functions

Table 2 Common parameters

Parameter	Value
DE CR	0.5
DE F	0.5
DE crossover operator	Exponential
DE selection operator	Tournament 2
DE model	Classic
CMA-ES λ	Same as hybrid population size
CMA-ES μ	$\frac{\lambda}{2}$
Minimum participation ratio	5 %
Reduction factor ξ	0.05
Max. FEs.	$10^5 D$

Table 3 Functions randomly selected for the parameter tuning

Noiseless testbed	f_1, f_4, f_7, f_8, f_{11}, f_{14}, f_{18}, f_{19}, f_{21}, f_{22}
Noisy testbed	f_{102}, f_{106}, f_{108}, f_{110}, f_{113}, f_{117}, f_{121}, f_{123}, f_{127}, f_{128}

Table 4 Parameters of the algorithm for the noiseless testbed

Parameter	Value
Initial population size	25 50 75 **100** 150 200
Maximum pop. size (after restarts)	6,400 15,000 **25,000**
Step factor	5 **10** 20

Table 5 Parameters of the algorithm for the noisy testbed

Parameter	Value
Initial population size	25 50 75 100 150 **200**
Maximum pop. size (after restarts)	6,400 15,000 **25,000**
Step factor	5 10 **20**

with low or moderate conditioning, etc.). See Table 3 for the detailed list of functions finally selected.

The values that have been finally selected appear in bold in Tables 4 and 5 for both the noiseless and the noisy testbeds, respectively. These were the best configurations according to the statistical tests described later on in this section. It can be advanced that, for the noiseless testbed, the values for the initial population size and the step factor are intermediate values of those tested in the parameter tuning study. On the other hand, we can observe how, in the case of the noisy testbed, the selected values have moved to the limits of the considered values in the case of both the initial population size and the step factor. This gives an initial idea of how differently an algorithm can perform as the testbed is changed and the importance of a proper parameter tuning study.

Once the best configuration has been chosen for each testbed all the parameters of the algorithm remain the same for all the functions and, thus, the Crafting Effort (CrE) value is zero [13].

Table 6 presents a summary of the results of the statistical tests carried out to decide which configuration of parameters should be used for the noiseless testbed. This table shows the values for the parameters for both the best and the worst configurations on this subset of functions, the average ranking of each algorithm according to the Friedman test and the nWins value according to the nWins Procedure [15]. In that table we can observe how the best configurations have in common that they are using step factors of 10, whereas the worst configurations use all a step factor of 5. To validate statistically these results, the Wilcoxon signed-rank test, the Holm Procedure and the nWins Procedure were used. Both Wilcoxon and Holm agree that using a step factor of 10 is better than using a step factor of 20, whereas they gave statistical support to state that using a step factor of 10 is better than using a step factor of 5. The nWins value showed in Table 6 gives a quantitative view of how better are some configurations compared to others, from a statistical point of view. The nWins procedure basically counts in how many pairwise Wilcoxon tests one configuration is statistically better than another. In this case we can see that the best configuration is statistically better than 41 out of 54 other configurations.

Once the best configuration has been selected, the same procedure is repeated but now with the whole set of functions. The results obtained are very similar to those presented in Table 6, with the same best and worst configurations. However, now the best configuration is statistically better than 48 out of 54 other configurations.

Analogously, Table 7 presents a summary of the results of the statistical tests for the noisy testbed. In that table we can observe how the best configurations have in common that they are using step factors of 20 and large initial populations (150 individuals or more), whereas the worst configurations use all a step factor of 5 and the smallest considered population (25 individuals). The same statistical validation has been conducted on these data. In this case, both Wilcoxon and Holm tests gave statistical support to state that using a step factor of 20 is better than using a step factor of 5 and that using the smallest population of 25 (and also 50) is worse than using larger populations (mainly 200 and 150 individuals, but this also holds for some configurations using 100 individuals). Regarding the number of nWins, we can see that the best configurations are statistically better that most of the other ones (46–41 out of 54 other configurations, respectively). However, among them the differences are very small, both in the number of nWins and the average ranking according to the Friedman test. This means that there is not one single configuration that can bring good results, but several configurations with similar performance due to the superior importance of some of the considered parameters on the final behavior of the algorithm. In this case, those parameters are the initial population and the step factor, whereas the maximum population does not seem to be of great importance in this study.

As with the noiseless testbed, the same procedure is repeated with the whole set of functions. The results obtained are again very similar to those presented in Table 7, with the same best and worst configurations. In this case, the best configuration is statistically better than 48 out of 54 other configurations.

Table 6 Statistical tests of the benchmarked configurations for the subset of functions obtained by random sampling from the noiseless testbed

	Configuration	Average ranking	nWins
Best	pop: 100, max pop: 25,000, factor: 10	23.82	41
2nd	pop: 200, max pop: 6,400, factor: 10	24.17	28
3rd	pop: 150, max pop: 6,400, factor: 10	24.47	27
	...		
52nd	pop: 25, max pop: 6,400, factor: 5	34.32	−50
53rd	pop: 25, max pop: 15,000, factor: 5	36.48	−50
Worst	pop: 25, max pop: 25,000, factor: 5	37.19	−51

Table 7 Statistical tests of the benchmarked configurations for the subset of functions obtained by random sampling from the noisy testbed

	Configuration	Average ranking	nWins
Best	pop: 200, max pop: 25,000, factor: 20	18.62	46
2nd	pop: 100, max pop: 6,400, factor: 20	20.23	41
3rd	pop: 150, max pop: 15,000, factor: 20	20.76	41
	...		
52nd	pop: 25, max pop: 25,000, factor: 5	34.82	−38
53rd	pop: 25, max pop: 15,000, factor: 5	35.21	−39
Worst	pop: 25, max pop: 6,400, factor: 5	36.91	−44

Figures 1 and 2 depict two parallel coordinates plots of the parameters considered in the parameter tuning phase. Parallel coordinates is a useful technique which has been successfully used to represent high dimensional data as polylines in two dimensions. More recently, it has been used to capture the underlying interactions between the parameters of a PSO algorithm [6]. In this case, the first three axes represent the parameters of the configurations, whereas the fourth one represents the Friedman average ranking mentioned before (smaller values for the ranking means better configurations). These plots will help us to graphically confirm the conclusions obtained from the statistical analysis. To ease the visualization of the different polylines, a small jitter has been added to the three parameters (not to the ranking) in order to avoid the overlap of those lines and to be able to distinguish which lines join which values of the parameters.

In Fig. 1 we can observe how the most important parameter for the noiseless testbed is the step factor. Configurations using values of 10 or 20 are depicted with green/blue dashed lines, which represent better configurations (according to their average ranking), whereas configurations using a value of 5 are shown with red solid lines. For the other two parameters, configurations with good performance are distributed more uniformly, which means that these are not as critical as the step factor to determine the performance of the algorithm.

In Fig. 2 we can observe how, for the noisy testbed, the most important parameter is the initial population size. Configurations using large values for this parameter are depicted in blue/green dashed lines, which represent better configurations (according to their average ranking), whereas configurations using small initial populations are shown with red solid lines. Regarding the step factor, it is not clear which value obtains the overall best performance. Configurations using values of 20 seem to be slightly better in this case, as shown in Table 7, but there is not a clear distinction on this. For the other parameter, the maximum population size, configurations with good performance are distributed more uniformly, which means that this parameter is not as critical as the initial population size or the step factor to determine the performance of the algorithm.

5 Experimental results

In this section, we first analyze the behavior of the hybrid algorithm and show the ability of the MOS algorithm to adjust the participation of the composing algorithms and select the most appropriate for each problem. Then, the results obtained on both testbeds are presented. We will discuss first the results on the noiseless testbed and, afterwards, those on the noisy testbed.

5.1 Hybridization analysis

In Sect. 3 we presented and described the MOS algorithm. One of the key issues discussed in this section was the type of interaction emerging from the combination of two different

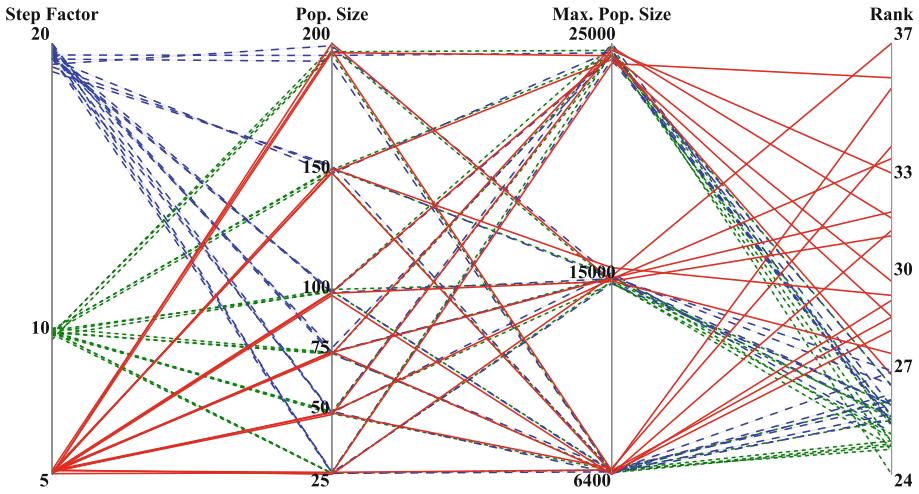


Fig. 1 Parallel coordinates representation of the parameter tuning for the noiseless testbed

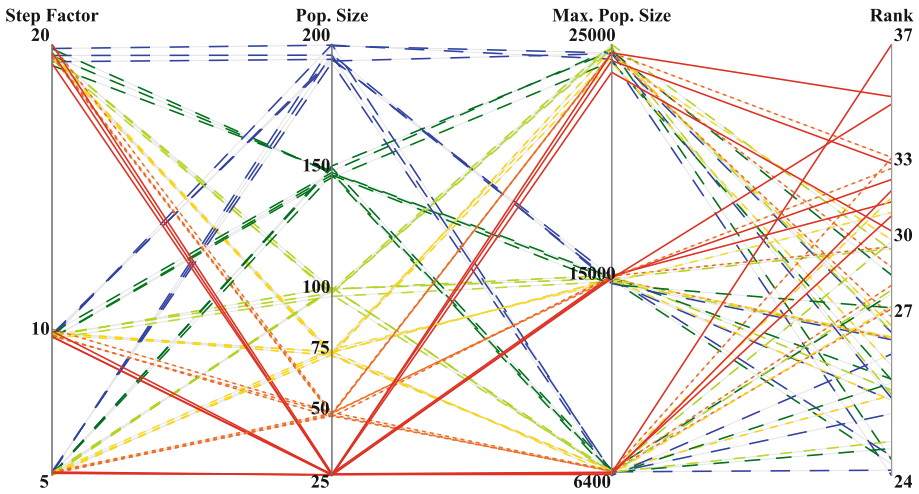


Fig. 2 Parallel coordinates representation of the parameter tuning for the noisy testbed

Evolutionary Algorithms. As we said, such a combination could lead to two different situations: a collaborative synergy of both algorithms or a competitive selection of the best performing one.

Ideally, the first one of these two behaviors is desired. This would be the case of problems in which several algorithms perform differently at different stages of the search. However, this is not usually the case and, most of the times, the second type of interaction (competitive selection) arises. When this happens, the hybrid algorithm should be able to evaluate the performance of each algorithm and adjust their participation accordingly. In the BBOB benchmark considered for this contribution, IPOP-CMA-ES has shown to be very competitive in terms of number of functions solved [25,26]. However, there are some functions in the benchmark that are especially difficult to solve by IPOP-CMA-ES. For example, as it can be seen in the results reported in [25], Rastrigin (f_3) and Büche-Rastrigin (f_4) functions were two of those functions. At this point, our hypothesis is that it should be possible to combine IPOP-CMA-ES with another algorithm (Differential Evolution in this case) for which those functions are easier to solve in such a way that the hybrid algorithm can decide at each point which algorithm should be used. Furthermore, the hybrid algorithm should take accurate decisions and maintain the good performance that the IPOP-CMA-ES has shown on other functions.

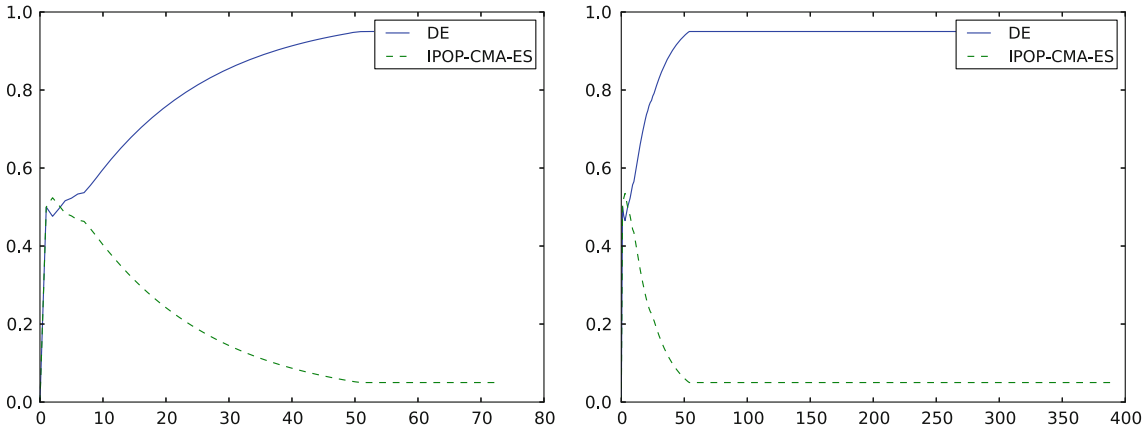
Table 8 shows the average errors for each algorithm (IPOP-CMA-ES, DE and MOS) on functions f_3 and f_4 in 20-D. As it can be observed, the MOS hybrid algorithm has a performance similar to that of DE, as both algorithms are able to solve both functions to the requested precision. However, IPOP-CMA-ES is not able to solve any of the two functions.

If we look at Fig. 3a and b we can see what is exactly happening in these two functions. In both cases, the hybrid algorithm quickly finds that DE is able to find better solutions for these functions and thus increases the participation ratio of this technique. The difference in the slope of the curves for both functions represents the relative difference in the performance of each algorithm on each function: DE is able to find better solutions than IPOP-CMA-ES in both functions, but the difference in the performance is more pronounced in f_4 than in f_3.

On the other hand, IPOP-CMA-ES is better than DE in many other functions. We can consider, for example, func-

Table 8 Average error for Rastrigin (f_3) and Büche-Rastrigin (f_4) functions in 20-D

	IPOP-CMA-ES	DE	MOS
f_3	5.62e+00 ± 1.69e+00	0.00e+00 ± 0.00e+00	0.00e+00 ± 0.00e+00
f_4	1.36e+01 ± 1.24e+00	0.00e+00 ± 0.00e+00	0.00e+00 ± 0.00e+00



(a) f3 in 20-D **(b)** f4 in 20-D

Fig. 3 Participation adjustment of the hybrid algorithm: DE dominance

Table 9 Average error for Ellipsoidal (f_{10}) and Weierstrass (f_{16}) functions in 20-D

	IPOP-CMA-ES	DE	MOS
f_{10}	0.00e+00 ± 0.00e+00	1.12e+04 ± 4.22e+03	0.00e+00 ± 0.00e+00
f_{16}	1.20e−05 ± 4.98e−05	4.04e+00 ± 9.51e−01	3.74e−06 ± 1.34e−06

tions f_{10} and f_{16}, for which the average error of the three algorithms is given in Table 9.

As in the previous case, the hybrid algorithm is able to detect which of the available techniques is finding better solutions and adjusts the participation ratios accordingly. However, now it is IPOP-CMA-ES the algorithm that sees its participation increased, as it is shown in Fig. 4a and b.

With these two examples we have shown how the hybrid algorithm is able to select, from multiple techniques, the one which is getting better results, increasing its participation. This type of hybridization is effective to create robust hybrid algorithms in which the limitations of one algorithm are complemented by other algorithm and vice-versa. With this in mind, we will proceed to the overall analysis of the hybrid algorithm in both the noiseless and the noisy testbeds.

5.2 Results on the noiseless testbed

Results from experiments according to Hansen et al. [13] on the benchmark functions of the noiseless testbed given in [4,11] are presented in Figs. 5 and 6. The runs have been conducted with the configuration depicted in Table 2 for the common parameters and the values obtained after the parameter tuning for the hybridization parameters highlighted in Table 6.

The overall results in the noiseless testbed are quite satisfactory in terms of achieved precision and scalability. The hybrid algorithm here presented is able to solve 24, 24, 24, 24, 22 and 21 functions out of 24 in 2, 3, 5, 10, 20 and 40 dimensions, respectively.[1]

Compared to the individual use of its composing algorithms, the hybrid algorithm obtains more stable results than any of them. Furthermore, functions $f3$ and $f4$ which are extremely hard to solve by the IPOP-CMA-ES algorithm, are now solved thanks to the hybridization with the DE algorithm. On the other hand, the most difficult function for our approach is f_{24}, for which convergence is never reached for dimension 20 or above. Nevertheless, this is somehow reasonable, as this function has been designed to be deceptive for Evolution Strategies (and the DE is also unable to deal with it).

Furthermore, it can also be observed that the proposed algorithm achieves one of the best results in terms of ECDF values, compared with the algorithms presented in

[1] The complete 120 results in the noiseless testbed can be accessed in the following URL: http://laurel.datsi.fi.upm.es/research.

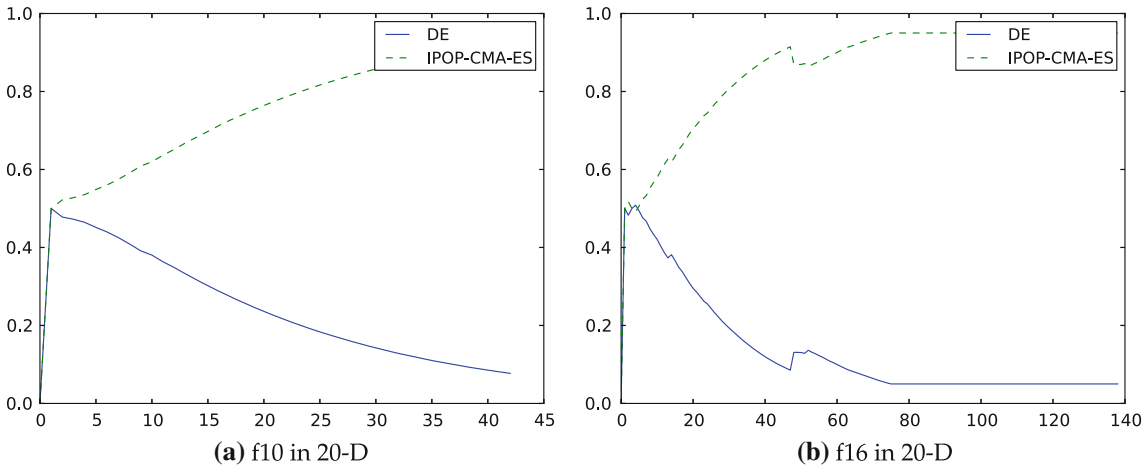


Fig. 4 Participation adjustment of the hybrid algorithm: IPOP-CMA-ES dominance

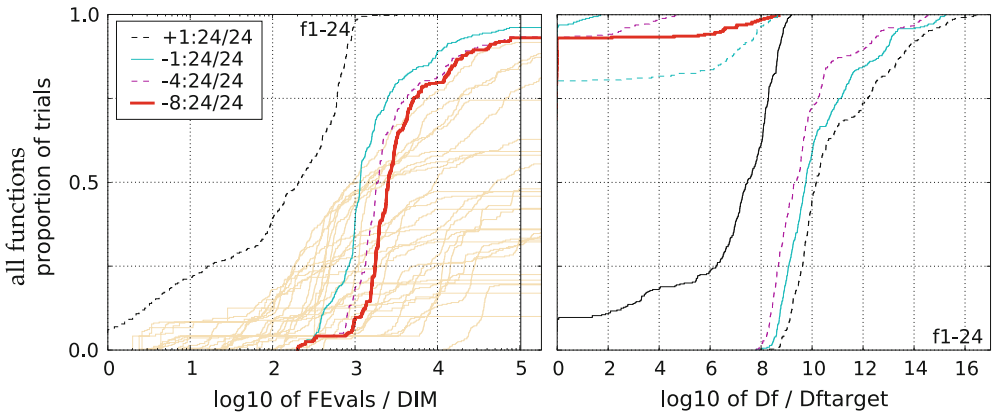


Fig. 5 Noiseless functions 5-D. *Left* subplots: Empirical Cumulative Distribution Function (ECDF) of the running time (number of function evaluations), divided by search space dimension D, to fall below $f_{\text{opt}} + \Delta f$ with $\Delta f = 10^k$, where k is the first value in the *legend*. The *vertical black line* indicates the maximum number of function evaluations. The *legends* indicate the number of functions that were solved in at least one trial. *Light brown lines* in the background show ECDFs for target value 10^{-8} of all algorithms benchmarked during BBOB 2009. *Right* subplots: ECDF of the best achieved Δf divided by 10^k (*upper left lines* in continuation of the *left* subplot), and best achieved Δf divided by 10^{-8} for running times of D, $10\,D$, $100\,D\ldots$ function evaluations (from *right* to *left* cycling *black-cyan-magenta*). $D = \mathsf{DIM}$ denotes search space dimension (5 in this case) and $\Delta f = \mathsf{Df}$ denotes the difference to the optimal function value

the BBOB-2009 workshop, for all the groups of functions, as it can be seen in Figs. 5, 6.

Additionally, we can also observe in those figures how the proposed algorithm is slower to converge to good solutions than many of the algorithms that appear in the comparison. This behavior is more pronounced at the beginning of the search (while log10 of FEvals/DIM is <3). From this point, the convergence speed of the algorithm increases and it is able to solve more functions than most of the reference algorithm from the BBOB 2009 workshop. To some degree, this behavior is normal, as the regulatory mechanisms implemented by the MOS framework need some time to take a decision and adjust the participation of each technique accordingly. Furthermore, in some cases this adjustment is not as fast as it would be desirable, especially when the number of FEs allocated for each step is relatively large. For this reason, it is important to tune the parameters in order to optimize the desired behavior of the hybrid algorithm: speed of convergence or precision. In this case, the latter has been considered (although it has not always to be the case).

Finally, we have conducted a statistical validation following the recommendations provided in [8]. For this comparison, we have considered the average error for all the functions and dimensions. First of all, we applied Friedman's test in order to see whether there are global differences in the results. Given that the p-value obtained ($4.82e{-}09$) is lower than the considered level of significance $\alpha = 0.05$, there are significant differences in the results for the noiseless testbed. According to this result, we conduct a post-hoc statistical analysis to detect particular differences among the

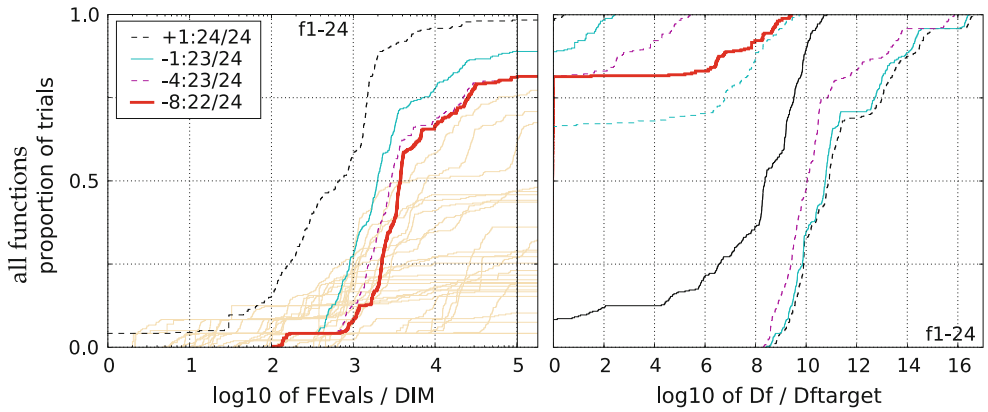


Fig. 6 Noiseless functions 20-D. See caption of Fig. 5

Table 10 Statistical validation (MOS is the control algorithm)

$\surd$ means that there are statistical differences with significance level $\alpha = 0.05$

MOS vs.	z-value	p-value	Holm p-value	Hochberg p-value
IPOP-CMA-ES	2.09E+00	3.65E−02	$3.65E-02^{\surd}$	$3.65E-02^{\surd}$
DE	4.45E+00	8.63E−06	$1.73E-05^{\surd}$	$1.73E-05^{\surd}$

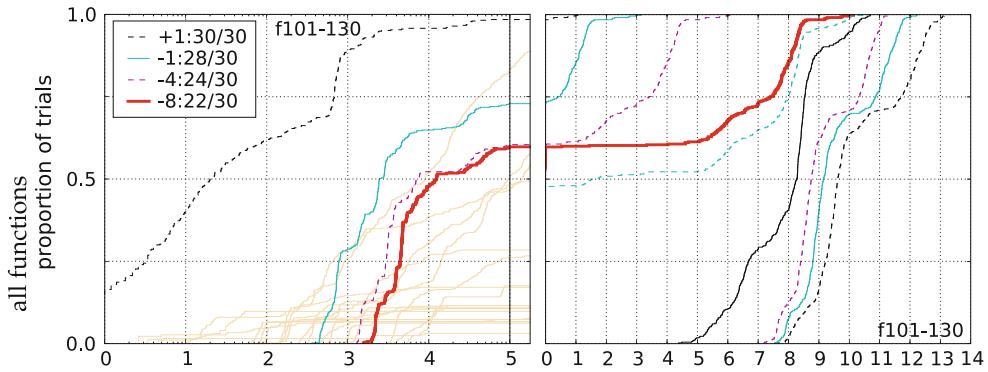


Fig. 7 Noisy functions 5-D. *Left* subplots: Empirical Cumulative Distribution Function (ECDF) of the running time (number of function evaluations), divided by search space dimension D, to fall below $f_{\text{opt}} + \Delta f$ with $\Delta f = 10^k$, where k is the first value in the *legend*. The *vertical black line* indicates the maximum number of function evaluations. The *legends* indicate the number of functions that were solved in at least one trial. *Light brown lines* in the background show ECDFs for target value 10^{-8} of all algorithms benchmarked during BBOB 2009. *Right* subplots: ECDF of the best achieved Δf divided by 10^k (*upper left lines* in continuation of the *left* subplot), and best achieved Δf divided by 10^{-8} for running times of D, $10\,D$, $100\,D\ldots$ function evaluations (from *right* to *left* cycling *black-cyan-magenta*). $D = \mathsf{DIM}$ denotes search space dimension (5 in this case) and $\Delta f = \mathsf{Df}$ denotes the difference to the optimal function value

algorithms. Table 10 presents the obtained results according to Holm and Hochberg tests. In both the cases, the tests were able to find significant differences, being MOS the control algorithm.

5.3 Results on the noisy testbed

The results for the benchmark functions of the noisy testbed given in [5,12] are presented in Figs. 7 and 8. As for the noiseless testbed, all the runs have been conducted with the configuration depicted in Table 2 for the common parameters and the values obtained after the parameter tuning for the hybridization parameters highlighted in Table 7.

The overall results in the noisy testbed are comparatively good in terms of achieved precision, but could be still improved, especially on functions with severe noise on larger dimensions. The hybrid algorithm here presented is able to solve 30, 28, 22, 19, 18 and 13 functions out of 30 in 2, 3, 5, 10, 20 and 40 dimensions, respectively. It seems that the noise added to the functions makes the performance of the algorithm deteriorate as the number of dimensions increases.

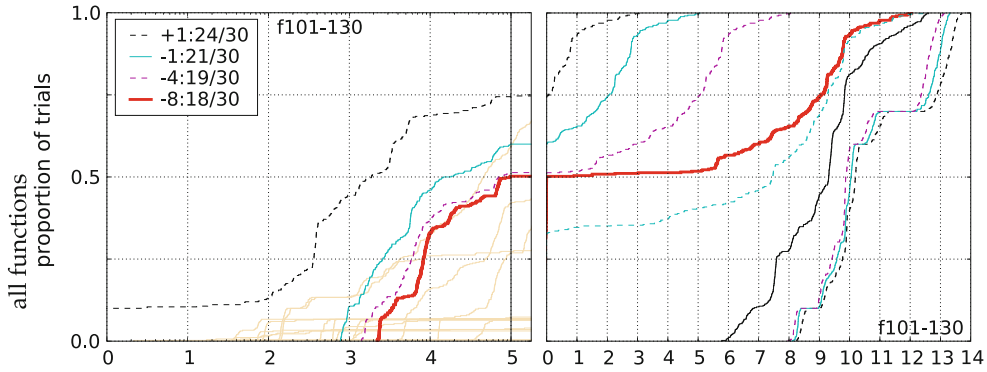


Fig. 8 Noisy functions 20-D. See caption of Fig. 7

Table 11 Statistical validation (MOS is the control algorithm)

MOS vs.	z-value	p-value	Holm p-value	Hochberg p-value
IPOP-CMA-ES	8.70E−01	3.85E−01	$3.85E-01$	$3.85E-01$
DE	7.12E+00	1.12E−12	$2.24E-12^{\surd}$	$2.24E-12^{\surd}$

$\surd$ means that there are statistical differences with significance level $\alpha = 0.05$

This effect is more pronounced in the case of those functions with severe noise than in those with a moderate noise.[2]

As it happened with the noiseless testbed, it can also be observed that the proposed algorithm achieves one of the best results in terms of ECDF values, compared with the algorithms presented in the BBOB-2009 workshop, for all the groups of functions, as it can be seen in Figs. 7 and 8. Actually, it obtains the best results in functions with moderate noise, whereas it is only improved by one of the algorithms of the workshop in the other two groups of functions.

Regarding the convergence speed, we can observe that the differences with other algorithms have been reduced, compared with those of the noiseless testbed. The additional difficulty introduced by the noise makes it difficult for all the algorithms to converge to good solutions quickly, which attenuates the overhead effect of the hybridization mechanisms of our algorithm. In particular, the hybrid algorithm is still slower in functions with moderate noise (although it finally converges to better results). For the other two groups of functions, the convergence speed is similar to that exhibited by reference algorithms. As in the case of the noiseless testbed, this behavior is more pronounced at the beginning of the search (roughly, while log10 of FEvals/DIM is <2, 3 or 4, depending on the case). From this point, the convergence speed of the algorithm increases.

Finally, we conducted the same statistical validation as for the noiseless testbed. First, Friedman's test gave a p-value of $4.81e-18$, which is clearly below the considered level of significance $\alpha = 0.05$. Thus, we used the same post-hoc tests in order to find the particular differences among the algorithms. Table 11 presents these results. It can be observed that, in this case, the DE algorithm is much worse than IPOP-CMA-ES and thus there are significant differences between MOS and DE, but not between MOS and IPOP-CMA-ES, as this is the selected algorithm for most of the functions of this testbed.

6 Conclusions

In this paper, a hybrid algorithm combining Differential Evolution and IPOP-CMA-ES has been presented and benchmarked on both the BBOB-2010 noiseless and noisy testbeds. A thorough parameter tuning has been conducted on several of the key parameters of the algorithm to analyze their influence in its performance. The results of this analysis have been graphically depicted for better comprehension. The study shows important differences between the values for the parameters of the algorithm needed to obtain the best results in the two different testbeds. Regarding the noiseless testbed, the experimental results show a good performance on all the groups of functions and a good scalability. In particular, the proposed algorithm has been able to solve 24, 24, 24, 24, 22 and 21 functions out of 24 in 2, 3, 5, 10, 20 and 40 dimensions, respectively. Furthermore, it obtains better results than its composing algorithms when used individually according to the conducted statistical validation. Additionally, a comparative analysis with the algorithms presented at the BBOB-2009 workshop reveals that our approach obtains one of the best results in terms of convergence. In the case of

[2] The complete 150 results in the noisy testbed can be accessed in the following URL: http://laurel.datsi.fi.upm.es/research.

the noisy testbed, the hybrid algorithm is able to solve 30, 28, 22, 19, 18 and 13 functions out of 30 in 2, 3, 5, 10, 20 and 40 dimensions, respectively. The results are very competitive in terms of convergence (the best results in one of the groups of functions) and not significantly slower than the reference algorithms, mainly because the extra difficulty introduced by the noise on these functions makes all the algorithms to converge more slowly. However, on multimodal functions with severe noise the results can still be improved, which opens new research directions in order to find better restart mechanisms and combinations of algorithms. The statistical analysis reveals better performance than the DE algorithm but not statistically better that IPOP-CMA-ES. This is due to the fact that the DE algorithm performs very poorly on this testbed and the hybrid algorithm selects IPOP-CMA-ES as the main algorithm most of the time.

Further research will investigate with new techniques to complement the two used algorithms in those functions in which the hybrid algorithms obtains worse results. A more thorough study on the control mechanisms, especially those related to the detection of the stagnation and the restart of the search process, could be also useful to increase the stability in those functions in which the convergence to the global optimum is not always obtained. Finally, a deeper study on the participation adjustment mechanisms would also be of great interest, especially to develop new criteria which will allow a faster detection and selection of one technique when it is significantly better than the others in order to reduce the gap in terms of convergence speed with other continuous optimizers.

Acknowledgments This work was financed by the Spanish Ministry of Science (TIN2010-21289-C02-02) and supported by the Cajal Blue Brain Project. The authors thankfully acknowledge the computer resources, technical expertise and assistance provided by the Centro de Supercomputación y Visualización de Madrid (CeSViMa) and the Spanish Supercomputing Network. A. LaTorre gratefully acknowledges the support of the Spanish Ministry of Science and Innovation (MICINN) for its funding throughout the Juan de la Cierva program.

References

1. Auger A, Hansen N (2005) A restart CMA evolution strategy with increasing population sizes. In: Proceedings of the 7th IEEE Congress on Evolutionary Computation, CEC 2005, IEEE Press, pp 1769–1776
2. Caponio A, Cascella GL, Neri F, Salvatore N, Summer M (2007) A fast adaptive memetic algorithm for on-line and off-line control design of PMSM drives. IEEE Trans Syst Man Cybern Part B 37:28–41
3. Caponio A, Neri F, Cascella GL, Salvatore N (2008) Application of memetic differential evolution frameworks to PMSM drive design. In: Proceedings of the 2008 IEEE Congress on Evolutionary Computation, CEC 2008 (IEEE World Congress on Computational Intelligence), pp 2113–2120
4. Finck S, Hansen N, Ros R, Auger A (2009) Real-parameter Black-Box optimization benchmarking 2009: presentation of the noiseless functions. Tech Rep 2009/20
5. Finck S, Hansen N, Ros R, Auger A (2010) Real-parameter black-box optimization benchmarking 2010: presentation of the noisy functions. Tech Rep
6. Franken N (2009) Visual exploration of algorithm parameter space. In: Proceedings of the Eleventh conference on Congress on Evolutionary Computation, pp 389–398
7. Gao Y, Wang YJ (2007) A memetic differential evolutionary algorithm for high dimensional functions' optimization. In: Proceedings of the Third International Conference on Natural Computation (ICNC 2007), pp 188–192
8. García S, Molina D, Lozano M, Herrera F (2009) A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: a case study on the CEC'2005 special session on real parameter optimization. J Heuristics 15(6):617–644
9. García-Martínez C, Lozano M, (2009) A continuous variable neighbourhood search based on specialised EAs: application to the noiseless BBO-benchmark. In: 11th Genetic and Evolutionary Computation Conference, GECCO 2009 (Companion). ACM, New York, pp 2287–2294
10. Hansen N, Kern S (2004) Evaluating the CMA evolution strategy on multimodal test functions. In: Proceedings of the 8th International Conference on Parallel Problem Solving from Nature—PPSN VIII. Springer-Verlag GmbH, Berlin, pp 282–291
11. Hansen N, Finck S, Ros R, Auger A (2009) Real-parameter black-box optimization benchmarking 2009: noiseless functions definitions. Tech Rep RR-6829, INRIA
12. Hansen N, Finck S, Ros R, Auger A (2009) Real-parameter black-box optimization benchmarking 2009: noisy functions definitions. Tech Rep
13. Hansen N, Auger A, Finck S, Ros R (2010) Real-parameter black-box optimization benchmarking 2010: experimental setup. Tech Rep RR-7215
14. Kämpf JH, Robinson D (2009) A hybrid CMA-ES and HDE optimisation algorithm with application to solar energy potential. Appl Soft Comput 9(2):738–745
15. LaTorre A (2009) A framework for hybrid dynamic evolutionary algorithms: multiple offspring sampling (MOS). PhD thesis, Universidad Politécnica de Madrid
16. LaTorre A, Muelas S, Peña JM (2010) Benchmarking a MOS-based algorithm on the BBOB-2010 noiseless function testbed. In: 12th Genetic and Evolutionary Computation Conference, GECCO 2010 (Companion), ACM, pp 1649–1656
17. LaTorre A, Muelas S, Peña JM (2011) A MOS-based dynamic memetic differential evolution algorithm for continuous optimization: a scalability test. Soft Comput—Fusion Found 15(11):2187–2199
18. Misik L, Krivy I (2002) Competing heuristics in evolutionary algorithms. In: Sincak P, Vascak J, Kvasnicka V (eds) New Trends in Intelligent Technologies. IOS Press, Intelligent Technologies—Theory and Applications, pp 159–165
19. Molina D, Lozano M, García-Martínez C, Herrera F (2010) Memetic algorithms for continuous optimisation based on local search chains. Evol Comput 18(1):27–63
20. Muelas S, LaTorre A, Peña JM (2009) A memetic differential evolution algorithm for continuous optimization. In: Proceedings of the 9th International Conference on Intelligent Systems Design and Applications, ISDA 2009, IEEE Press, pp 1080–1084
21. Muller CL, Baumgartner B, Sbalzarini IF (2009) Particle swarm CMA evolution strategy for the optimization of multi-funnel landscapes. IEEE Congress on Evolutionary Computation, 2009. CEC 2009:2685–2692

22. Peng F, Tang K, Chen G, Yao X (2010) Population-based algorithm portfolios for numerical optimization. IEEE Trans Evolut Comput 14(5):782–800
23. Qin AK, Suganthan PN (2005) Self-adaptive differential evolution algorithm for numerical optimization. In: Proceedings of the IEEE Congress on Evolutionary Computation. CEC 2005:1785–1791
24. Rönkkönen J, Kukkonen S, Price KV (2005) Real-parameter optimization with differential evolution. In: Proceedings of the 7th IEEE Congress on Evolutionary Computation, CEC 2005, IEEE Press, pp 506–513
25. Ros R (2010a) Black-box optimization benchmarking the IPOP-CMA-ES on the noiseless testbed: comparison to the BIPOP-CMA-ES. In: 12th Genetic and Evolutionary Computation Conference, GECCO 2010, ACM Request Permissions, pp 1503–1510
26. Ros R (2010b) Black-box optimization benchmarking the IPOP-CMA-ES on the noisy testbed: comparison to the BIPOP-CMA-ES. In: 12th Genetic and Evolutionary Computation Conference, GECCO 2010, ACM Request Permissions, pp 1511–1517
27. Storn R, Price KV (1995) Differential evolution—a simple and efficient adaptive scheme for global optimization over continuous spaces. Tech Rep
28. Talbi EG (2002) A taxonomy of hybrid metaheuristics. J Heuristics 8(5):541–564
29. Tirronen V, Neri F, Karkkainen T, Majava K, Rossi T (2007) A memetic differential evolution in filter design for defect detection in paper production. In: Proceedings of EvoWorkshops 2007, pp 330–339
30. Tseng LY, Chen C (2008) Multiple trajectory search for large scale global optimization. In: Proceedings of the 10th IEEE Congress on Evolutionary Computation, CEC 2008 (IEEE World Congress on Computational Intelligence), IEEE Press, pp 3052–3059
31. Vanneschi L, Tomassini M, Collard P, Vérel S (2006) Negative slope coefficient: a measure to characterize genetic programming fitness landscapes. In: Collet P (ed) Proceedings of the 9th European conference on Genetic Programming, EuroGP 2006. Springer-Verlag GmbH, Berlin, pp 179–189