

Super-fit Multicriteria Adaptive Differential Evolution

Fabio Caraffini ^{*} [†], Ferrante Neri ^{*} [†], Jixiang Cheng [‡], Gexiang Zhang [‡],
Lorenzo Picinali^{*}, Giovanni Iacca[§], Ernesto Mininno[†]

^{*} Centre for Computational Intelligence, School of Computer Science and Informatics, De Montfort University,
The Gateway, Leicester LE1 9BH, United Kingdom
Email: fabio.caraffini@email.dmu.ac.uk, fneri@dmu.ac.uk, and lpicinali@dmu.ac.uk

[†] Department of Mathematical Information Technology,
P.O. Box 35 (Agora), 40014 University of Jyväskylä, Finland
Email: fabio.caraffini@jyu.fi, ferrante.neri@jyu.fi, and ernesto.mininno@jyu.fi

[§]School of Electrical Engineering, Southwest Jiaotong University, Chengdu, P.R. China 610031
Email: chengjixiang0106@126.com and zhgx Dylan@126.com

[‡]INCAS³ - Innovation Centre for Advanced Sensors and Sensor Systems
P.O. Box 797, 9400 AT Assen, The Netherlands
Email: giovanniiacca@incas3.eu

Abstract—This paper proposes an algorithm to solve the CEC2013 benchmark. The algorithm, namely Super-fit Multicriteria Adaptive Differential Evolution (SMADE), is a Memetic Computing approach based on the hybridization of two algorithmic schemes according to a super-fit memetic logic. More specifically, the Covariance Matrix Adaptive Evolution Strategy (CMAES), run at the beginning of the optimization process, is used to generate a solution with a high quality. This solution is then injected into the population of a modified Differential Evolution, namely Multicriteria Adaptive Differential Evolution (MADE). The improved solution is super-fit as it supposedly exhibits a performance a way higher than the other population individuals. The super-fit individual then leads the search of the MADE scheme towards the optimum. Unimodal or mildly multimodal problems, even when non-separable and ill-conditioned, tend to be solved during the early stages of the optimization by the CMAES. Highly multi-modal optimization problems are efficiently tackled by SMADE since the MADE algorithm (as well as other Differential Evolution schemes) appears to work very well when the search is led by a super-fit individual.

I. INTRODUCTION

Differential Evolution (DE) is a simple, efficient, and versatile algorithm that has been successfully applied to many problems in engineering and applied sciences, see [1]. For example, in [2] a DE is applied to a sensor fusion problem. In [3] and [4] a modified DE is used to address a filter design in the context of signal processing. In [5] a DE based hybrid algorithm with compact structure has successfully applied in the field of industrial robotics. Other applications, in mobile and space robotics, of DE schemes with compact structures are reported in [6] and [7].

As reported in [8], although DE is a very good algorithmic framework, it is characterised by a wide margin of

improvement. During the latest years, computer scientists have proposed many enhanced versions of the original DE schemes. Paper [8] highlights that DE schemes suffer from a limited amount of search moves and an excessive determinism in the search logic. These facts can lead to a stagnation condition. On the other hand, DE schemes can be easily improved by integrating new search moves in the offspring generations. The pool of search moves can be easily broadened by including a certain degree of randomization in the generation of new solutions, see [9] and [10]. The amount of search moves in DE schemes has been also increased by explicitly using multiple search logics. For example, in [11] the use of multiple mutation strategies is proposed. In [12] and [13] a local search is integrated within the DE framework to support the offspring generation. In [4] multiple local search assist the offspring generation of DE.

A special way to perform the hybridisation between a DE framework and an explicit extra search logic, is by means of the super-fit memetic logic, see [14]. It has been observed that DE, unlike Evolutionary Algorithms EAs, performs well when one individual is characterised by a higher fitness with respect to the other population individuals. This effect is due to the fact that, in DE frameworks, the relationship between diversity and algorithmic performance is different with respect to other EAs, see [15], [16], [17], [18], and [18]. If we consider for example the Evolution Strategies (ES), a proper algorithmic functioning occurs when to the exploration necessity corresponds a high diversity condition and to the exploitation necessity corresponds a low diversity condition. In high diversity conditions, the algorithm is capable of exploring the entire decision space while in low diversity conditions the algorithm finalizes the search. DE schemes, unlike an ES, tend not to lose the diversity even when the algorithm is no longer producing enhancements.

When one individual has a much better performance than the others, the pivot individual (or super-fit individual) guides the search and is then outperformed by the other individuals of the population. Thus, in super-fit schemes an exploitative search is applied to a solution that is then inserted into a DE population. In order to generate this desirable condition, specifically in [14] an exploitative search at the beginning of the DE search is performed by applying the Particle Swarm Optimization to one individual of the population that is then injected in the DE population. The success of the super-fit logic within DE schemes has been confirmed in [19], where a super-fit individual generated by means of the Rosenbrock Algorithm [20] is applied to a DE having compact structure.

The present paper, in order to obtain a competitive algorithm for solving the testbed introduced in [21], proposes a super-fit hybridization of a DE algorithm. The initial improvement is achieved by means of the Covariance Matrix Adaptation Evolution Strategy (CMAES), see [22] and [23]. The DE scheme employed in this case is the recently developed Multicriteria Adaptive Differential Evolution (MADE), [24]. The resulting algorithm is here termed Super-fit Multicriteria Adaptive Differential Evolution (SMADE).

The remainder of this paper is organized in the following way. Section II describes the operators composing the proposed algorithm and their combination in a memetic fashion. Section III displays the numerical results obtained in this study on the testbed defined in [21] and, finally, Section IV presents the conclusion of this work.

II. SUPER-FIT MULTICRITERIA ADAPTIVE DIFFERENTIAL EVOLUTION

Without a loss of generality, in order to clarify the notation in this paper, we refer to the minimization problem of an objective function $f(\mathbf{x})$, where the candidate solution $\mathbf{x}$ is a vector of n design variables (or genes) in a decision space D .

A. Covariance Matrix Adaptation Evolution Strategy

The CMAES algorithm is a popular and efficient optimizer for continuous optimization problems. The main powerful advantage of CMAES is that it adapts the search by sampling solutions in a region of the decision space that takes the shape of the basin of attraction. This feature makes CMAES very efficient for non-separable and ill-conditioned problems, especially when the fitness landscape is not highly multimodal. More specifically, CMAES samples a set of candidate solutions from a multi-variate normal distribution. The equivalent of the variance in a multi-variate distribution is the covariance matrix whose dimensions are $n \times n$ and that gives the name to this algorithm. The fitness values of the sampled candidate solutions are calculated and the solution are then sorted. The performance of the sampled solutions are then used to update the state variables including the covariance matrix, i.e. modifying the shape of the multi-variate distribution for the new set of samples. Implementations details of CMAES are in [23] and [25]. It is worthwhile mentioning that several CMAES variants have been proposed in the literature, see e.g. [26], [27].

In the CMAES framework, at each step, λ new solutions are sampled from a multivariate normal distribution whose

mean and covariance matrix are adaptively updated from a weighted sum of the best μ solutions in the population. The loop goes on until a stop condition is met. It is interesting to note that, compared to classic evolution strategies employing the (μ, λ) or $(\mu + \lambda)$ survivor selection schemes, CMAES is said to employ a $(\mu/\mu_w, \lambda)$ strategy, where μ_w represent the weights on the best μ individuals used to update the distribution model. For the sake of completeness, a simplified pseudo-code illustrating the main steps of CMAES, see Algorithm 1.

Algorithm 1 Pseudo-code of CMAES

```

initialize covariance matrix  $\mathbf{C} = \mathbf{I}$ , and step-size  $\sigma$ 
initialize the mean vector  $\mathbf{m}$  with a random sample in  $D$ 
while CMAES stop criterion is not met do
    (activate Covariance Matrix Repairing method, if needed)
    sample  $\lambda$  new individuals from distribution  $\mathcal{N}(\mathbf{m}, \sigma^2 \mathbf{C})$ 
    evaluate individuals and sort them based on their fitness
    update  $\mathbf{m}$  based on a weighted sum of the best  $\mu$  individuals
    update covariance matrix  $\mathbf{C}$  and step-size  $\sigma$ 
end while
return best individual  $\mathbf{x}_e$ 

```

B. Multicriteria Adaptive Differential Evolution

A population of NP individuals is sampled within the decision space D . As in DE schemes, each element is taken into consideration. Similar to several successful DE versions proposed in the literature, such as [11], [28], and [29], MADE makes use of multiple mutation/crossover strategies. More specifically, for each individual of the population, in order to generate the offspring, the four mutation/crossover strategies in Fig. 1 are considered. In accordance with the analysis reported in [8], the employment of multiple search strategies is here proposed to increase the pool of search moves and thus enhance the DE performance.

Each strategy is associated to a probability p_k for $k = \{1, 2, 3, 4\}$. These probability values dynamically vary during the evolution. At the beginning of the optimization process, the probability values are set to be the same, i.e. $p_k = 0.25$, $\forall k$. The probability values are then updated on the basis of a feedback on the evolution. More specifically, at the generation g the Normalized Relative Fitness Improvement (NRFI) and Normalized Distance to the Best Individual (NDBI), measuring respectively the fitness impact (η_i^g) and the diversity impact (τ_i^g) related to the individual i at the generation g , are calculated. These two metrics are formulated as follows:

- 1) Normalized relative fitness improvement (NRFI):

$$\eta_i^g = \frac{\tilde{\eta}_i^g}{\max\{\tilde{\eta}_i^g | i = 1, 2, \dots, NP\}} \quad (1)$$

where

$$\tilde{\eta}_i^g = \frac{f(\mathbf{x}_i^g) - f(\mathbf{x}_i^{g+1})}{f(\mathbf{x}_i^g) - f(\mathbf{x}_{best}^g)} \quad (2)$$

- 2) Normalized distance to the best individual (NDBI):

$$\tau_i^g = \frac{\tilde{\tau}_i^g}{\max\{\tilde{\tau}_i^g | i = 1, 2, \dots, NP\}} \quad (3)$$

1) rand/1/bin:

$$\mathbf{u}_{i,j}^g = \begin{cases} \mathbf{x}_{r_1,j}^g + F(i) \cdot (\mathbf{x}_{r_2,j}^g - \mathbf{x}_{r_3,j}^g) & \text{if } \text{rand}_j(0,1) \leq Cr(i) \text{ or } j = j_r \\ \mathbf{x}_{i,j}^g & \text{otherwise} \end{cases}$$

2) rand/2/bin:

$$\mathbf{u}_{i,j}^g = \begin{cases} \mathbf{x}_{r_1,j}^g + F(i) \cdot (\mathbf{x}_{r_2,j}^g - \mathbf{x}_{r_3,j}^g) + F(i) \cdot (\mathbf{x}_{r_4,j}^g - \mathbf{x}_{r_5,j}^g) & \text{if } \text{rand}_j(0,1) \leq Cr(i) \text{ or } j = j_r \\ \mathbf{x}_{i,j}^g & \text{otherwise} \end{cases}$$

3) rand-to-best/2/bin:

$$\mathbf{u}_{i,j}^g = \begin{cases} \mathbf{x}_{r_1,j}^g + F(i) \cdot (\mathbf{x}_{best,j}^g - \mathbf{x}_{i,j}^g) + F(i) \cdot (\mathbf{x}_{r_1,j}^g - \mathbf{x}_{r_2,j}^g) + F(i) \cdot (\mathbf{x}_{r_3,j}^g - \mathbf{x}_{r_4,j}^g) & \text{if } \text{rand}_j(0,1) \leq Cr(i) \text{ or } j = j_r \\ \mathbf{x}_{i,j}^g & \text{otherwise} \end{cases}$$

4) current-to-rand/1:

$$\mathbf{u}_i^g = \mathbf{x}_i^g + \text{rand}(0,1) \cdot (\mathbf{x}_{r_1}^g - \mathbf{x}_i^g) + F(i) \cdot (\mathbf{x}_{r_2}^g - \mathbf{x}_{r_3}^g)$$

Fig. 1. Recombination strategies used by MADE

where

$$\tilde{\tau}_i^g = \sqrt{\sum_{j=1}^D (\mathbf{x}_{i,j}^g - \mathbf{x}_{best,j}^g)^2} \quad (4)$$

Here, NRFI measures the degree of fitness improvement and NDBI weights the potential of escaping from a local optimum. The two indexes are then combined to generate the impact γ_i^g related to each individual i at the generation g by means of the so called *number-of-dominating-impacts* method, consisting of the following:

$$\gamma_i^g = |\{(\eta_j^g, \tau_j^g) | (\eta_i^g, \tau_i^g) \succeq (\eta_j^g, \tau_j^g), j \in h_i^g\}| \quad (5)$$

where $|\cdot|$ denotes the cardinality of a set; $\mathbf{a} \succeq \mathbf{b}$ means that vector $\mathbf{a} = (a_1, \dots, a_k)$ dominates vector $\mathbf{b} = (b_1, \dots, b_k)$, i.e., $\forall i \in \{1, \dots, k\}, a_i \geq b_i \wedge \exists i \in \{1, \dots, k\}, a_i > b_i$; $h_i^g = \{\bigcup_{k=1}^K s_k^g\} \setminus s_{n_i}^g$ denotes the set of indexes of the target vectors which use the operator differing from the operator n_i used by $\mathbf{x}_i^g$.

On the basis of the calculated impact of the operator, the credit of each operator in the current generation is calculated by

$$r_k^g = \frac{\sum_{n_i \in s_k^g} \gamma_i^g}{|s_k^g|} \quad (6)$$

where $k = 1, \dots, K$ (in our case $K = 4$ as we have four recombination strategies). Then the operator quality is updated by an additive relaxation mechanism, i.e.,

$$q_k^g = (1 - \beta) \cdot q_k^g + \beta \cdot r_k^g \quad (7)$$

where β ($\beta \in [0, 1]$) is the adaptation rate; q_k^0 is initialized with 1. Finally, the probability of each operator to be selected at the next generation is updated by

$$p_k^{g+1} = p_{min} + (1 - K \cdot p_{min}) \frac{q_k^g}{\sum_{k=1}^K q_k^g} \quad (8)$$

Apart from the trial vector generation strategy, the control parameters F and Cr are also adaptively adjusted by introducing a degree of randomization and rewarding the successful strategies. Let sF_k and sCr_k denote the sets containing the LP (stands for Learning Period) most recent values of F

and Cr associated with the k th strategy which generates trial vectors entering the next generation. At each generation g , for each target vector $\mathbf{x}_i^g$ that selects the k th trial vector generation strategy, the scale factor $F(i)$ is sampled according to a Cauchy distribution with a local parameter μF_k and a scale parameter 0.1 by:

$$F(i) = \text{randc}(\mu F_k, 0.1) \quad (9)$$

and re-sampled if $F(i) > 1$ or $F(i) \leq 0$; the crossover rate $Cr(i)$ is generated following a normal distribution with mean μCr_k and standard deviation 0.1 by

$$Cr(i) = \text{randn}(\mu Cr_k, 0.1). \quad (10)$$

and regenerated if $Cr(i) > 1$ or $Cr(i) < 0$.

In (9) and (10), the local parameter μF_k and the mean μCr_k are initialized as 0.5. At the end of each generation, for LP generations if $|sF_k| < LP$, the values of μF_k and μCr_k remain unchanged, otherwise old values of sF_k and sCr_k are first removed to guarantee $|sF_k| = |sCr_k| = LP$. Then they are updated by

$$\mu F_k = \frac{\sum_{F \in sF_k} F^2}{\sum_{F \in sF_k} F} \quad (11)$$

and

$$\mu Cr_k = \frac{\sum_{Cr \in sCr_k} Cr}{LP}. \quad (12)$$

By means of these procedure an offspring individual is eventually generated. As for all the DE based algorithms, if the offspring solution outperforms the parent that generated it, then the parent solution is replaced by its offspring otherwise no replacement occurs. For the sake of clarity, the pseudo-code describing the working principles of MADE is reported in Algorithm 2.

C. Super-fit Memetic Hybridization and parameter setting

At the beginning of the optimization process, a population of NP individuals is sampled within the decision space D . By following the principles explained in [14] and [19], in the proposed SMADE, CMAES is run for 30% of the total budget. At the end of this procedure, the best solution detected by CMAES is injected into the MADE initial population and

Algorithm 2 Pseudo-code of MADE algorithm

Require: NP, LP, β, p_{min}

```
1:  $g = 0, K = 4$ 
2: Generate initial population  $P^0 = \{\mathbf{x}_1^0, \mathbf{x}_2^0, \dots, \mathbf{x}_{NP}^0\}$  and
   evaluate  $P^0$ 
3: Set  $q_k^0 = 1, p_k^0 = 0.25, \mu F_k = \mu Cr_k = 0.5, sF_k =$ 
    $sCr_k = \emptyset, k = 1, \dots, K$ 
4: while termination criterion is not satisfied do
5:   for  $i = 1$  to  $NP$  do
6:     Select a strategy  $k$  via a roulette-wheel selection
       scheme for  $\mathbf{x}_i^g$ 
7:     Generate  $F(i)$  by (9) and regenerate it if  $F(i) > 1$ 
       or  $F(i) \leq 0$ 
8:     Generate  $Cr(i)$  by (10) and regenerate it if  $Cr(i) > 1$ 
       or  $Cr(i) < 0$ 
9:     Generate  $\mathbf{u}_i^g$  using  $k$ th strategy and parameter values
        $F(i)$  and  $Cr(i)$ 
10:    Evaluate  $\mathbf{u}_i^g$  as  $f(\mathbf{u}_i^g)$ 
11:    if  $f(\mathbf{u}_i^g) \leq f(\mathbf{x}_i^g)$  then
12:       $\mathbf{x}_i^{g+1} = \mathbf{u}_i^g; sF_k = sF_k \cup \{F(i)\}; sCr_k = sCr_k \cup$ 
         $\{Cr(i)\}$ 
13:    else
14:       $\mathbf{x}_i^{g+1} = \mathbf{x}_i^g$ 
15:    end if
16:  end for
17:  for  $k = 1$  to  $NP$  do
18:    Calculate fitness impact  $\eta_i^g$  by (1)
19:    Calculate diversity impact  $\tau_i^g$  by (3)
20:    Calculate aggregated impact  $\gamma_i^g$  by (5)
21:  end for
22:  for  $k = 1$  to  $K$  do
23:    Calculate reward  $r_k^g$  by (6)
24:    Update quality  $q_k^g$  by (7)
25:    Update probability  $p_k^g$  by (8)
26:    if  $|sF_k| \geq LP$  then
27:      Remove old values from  $sF_k$  and  $sCr_k$  so that
         $|sF_k| = |sCr_k| = LP$ 
28:      Update  $\mu F_k$  and  $\mu Cr_k$  by (11) and (12)
29:    end if
30:  end for
31:   $g = g + 1$ 
32: end while
Ensure: the best solution
```

replaces the worst solution of the population. The MADE algorithm continues the optimization until the budget reaches its limit. As for the CMAES part of the algorithm, we used the default parameter setting of the original Java implementation [25]. In particular, the evolution strategy was configured with the standard values $\lambda = \lfloor 4 + 3 \ln(D) \rfloor$, $\mu = \lfloor \lambda/2 \rfloor$, and initial step-size $\sigma = 0.2$, which are broadly accepted in the literature. The parameters of MADE, with reference to Fig. 2, are chosen in the following way: $NP = 50$, $\beta = 0.7$, $p_{min} = 0.02$. As mentioned in the algorithmic description, we set as initial values $\mu F_k = \mu Cr_k = 0.5$ and $q_k = 1$. While the setting $p_{min} = 0.02$ appears to be good for all the problems, we suggest that NP and β are selected from the intervals $[20, 60]$ and $[0.4, 0.8]$. The initial values of μF_k and μCr_k are selected at the centre of their interval and let evolve according to formulas (11) and (12). Thus the parameters to be selected

are essentially β and NP .

By means of this hybridization, we expect that uni-modal or mildly multi-modal problems are solved (or nearly solved) during the first 30% of the budget. Due to the CMAES features, non-separable and ill-conditioned problems, stated that they are not highly multi-modal, should quickly be solved. For highly multi-modal landscapes, although CMAES is not likely to detect solutions with an extraordinary high performance, its application can be beneficial to achieve some improvements. The improved solution will then act as a super-fit individual to guide the search of MADE towards the optimum. The choice of 30% has been carried out on the basis of a tuning and general considerations. Given the total budget for this competition, we observed that $1500 \times n$ fitness evaluations are enough, for CMAES, to achieve a reasonably good result and stop improving upon the solution. It has been observed that a shorter budget can be not enough to reach high quality solutions and a too long budget would not lead to any further improvement. Usually, uni-modal problems are very easily/quickly solved by CMAES while highly multi-modal problems are likely to never been solved by it. We noticed that if CMAES has not solved yet the problem within $1500 \times n$ fitness evaluation, the problem will unlikely be solved within a longer budget. On the other hand, MADE appears to efficiently explore the decision space and make use of the super-fit individual. Although sometimes MADE improvements can be slow, the action of this algorithmic component appears to be efficient when a long budget is assigned. On the basis of these consideration, we recommend that most of the budget is assigned to MADE and only a moderate portion of the budget is initially given to CMAES.

III. NUMERICAL RESULTS

The proposed SMADE has been run over the 28 test problems contained in the testbed CEC2013 described in [21]. The problems of the testbed have been considered in 10, 30, and 50 dimensions. The SMADE algorithm, for each test problem and dimensionality value, has been run 51 times. Each run has been continued for $10000 \times n$ fitness evaluations. Tables I, II, and III report best, worst, median, and mean fitness values (in terms of fitness error at the end of the budget) as well the corresponding standard deviation in 10, 30, and 50 dimensions.

The proposed SMADE succeeded at solving, at least once, eight problems in 10D, seven problems in 30D, and five problems in 50D.

Figs 2, 3, and 4 show the average performance trends for some of the test problems listed in Tables I, II, and III. More specifically, in each figure, one example where CMAES solve the problems in the early stages of the optimization is depicted. This is the case of unimodal and moderately multi-modal optimization problems with or without ill-conditioning, such as **f1**, **f2**, **f3**, and **f4**. In addition, in each figure, the average trends of some cases where the initial improvement and thus the super-fit individual assists the DE based evolution is also shown. This would be the case of highly multi-modal and complex problems, such as **f17**, **f20**, and **f25**.

TABLE I. RESULTS OF SMADE IN 10D.

Func.	Best	Worst	Median	Mean	Std
1	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00
2	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00
3	0.00e+00	6.32e+00	0.00e+00	2.48e-01	1.23e+00
4	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00
5	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00
6	0.00e+00	9.81e+00	9.81e+00	5.41e+00	4.76e+00
7	7.03e-03	2.32e+01	3.05e-01	2.27e+00	4.45e+00
8	2.00e+01	2.05e+01	2.04e+01	2.03e+01	1.03e-01
9	8.33e-01	3.84e+00	2.36e+00	2.29e+00	7.19e-01
10	0.00e+00	3.95e-02	1.23e-02	1.42e-02	9.58e-03
11	0.00e+00	9.95e-01	0.00e+00	9.75e-02	2.96e-01
12	1.99e+00	1.79e+01	6.96e+00	7.80e+00	4.10e+00
13	1.99e+00	2.73e+01	1.18e+01	1.21e+01	6.40e+00
14	1.87e-01	1.87e+01	3.48e+00	3.64e+00	4.39e+00
15	1.60e+02	1.14e+03	7.93e+02	7.36e+02	2.60e+02
16	3.00e-02	1.45e+00	2.90e-01	4.04e-01	3.14e-01
17	1.01e+01	1.08e+01	1.02e+01	1.03e+01	1.55e-01
18	1.42e+01	3.57e+01	2.50e+01	2.46e+01	4.68e+00
19	1.64e-01	6.79e-01	3.84e-01	3.95e-01	1.25e-01
20	1.60e+00	3.48e+00	2.67e+00	2.65e+00	4.48e-01
21	2.00e+02	4.00e+02	4.00e+02	3.83e+02	5.50e+01
22	8.29e+00	2.12e+02	2.60e+01	4.93e+01	5.33e+01
23	1.07e+01	1.38e+03	5.17e+02	5.78e+02	3.16e+02
24	1.17e+02	2.18e+02	2.05e+02	2.02e+02	1.76e+01
25	2.00e+02	2.08e+02	2.01e+02	2.02e+02	1.91e+00
26	1.02e+02	2.00e+02	1.08e+02	1.26e+02	3.69e+01
27	1.80e+02	4.00e+02	3.03e+02	3.37e+02	5.23e+01
28	3.00e+02	6.00e+02	3.00e+02	3.17e+02	6.87e+01

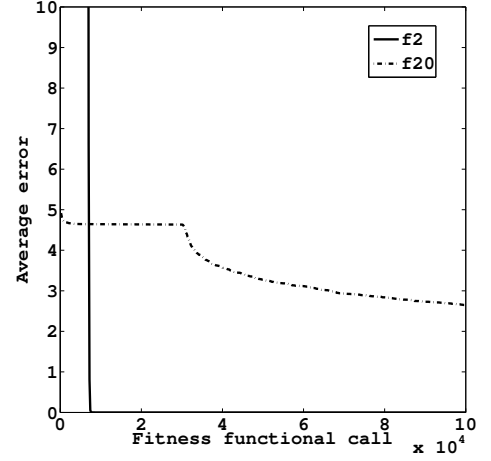


Fig. 2. Average fitness (error) trend of SMADE for f2 and f20 in 10D

TABLE II. RESULTS OF SMADE IN 30D.

Func.	Best	Worst	Median	Mean	Std
1	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00
2	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00
3	0.00e+00	3.52e+05	2.25e+00	9.82e+03	4.94e+04
4	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00
5	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00
6	0.00e+00	2.64e+01	0.00e+00	2.67e+00	7.85e+00
7	3.66e+00	6.53e+01	3.38e+01	3.25e+01	1.61e+01
8	2.08e+01	2.10e+01	2.10e+01	2.10e+01	4.80e-02
9	1.24e+01	3.15e+01	2.21e+01	2.23e+01	3.57e+00
10	0.00e+00	5.42e-02	1.23e-02	1.84e-02	1.34e-02
11	3.98e+00	1.99e+01	1.09e+01	1.09e+01	4.18e+00
12	2.89e+01	1.10e+02	5.47e+01	5.72e+01	1.70e+01
13	4.95e+01	2.24e+02	1.31e+02	1.28e+02	3.50e+01
14	1.31e+01	5.47e+02	7.29e+01	1.33e+02	1.27e+02
15	2.10e+03	6.12e+03	4.07e+03	4.10e+03	8.47e+02
16	3.46e-02	3.72e-01	1.06e-01	1.31e-01	7.57e-02
17	3.14e+01	4.08e+01	3.47e+01	3.48e+01	1.52e+00
18	5.28e+01	1.89e+02	8.12e+01	8.33e+01	2.06e+01
19	1.24e+00	3.90e+00	2.52e+00	2.55e+00	5.18e-01
20	8.54e+00	1.23e+01	1.05e+01	1.05e+01	8.07e-01
21	2.00e+02	4.44e+02	3.00e+02	3.27e+02	8.65e+01
22	1.27e+02	2.95e+02	1.66e+02	1.79e+02	4.50e+01
23	2.72e+03	6.41e+03	4.18e+03	4.22e+03	8.74e+02
24	1.40e+02	2.60e+02	2.40e+02	2.32e+02	2.57e+01
25	2.43e+02	2.94e+02	2.79e+02	2.78e+02	9.90e+00
26	1.37e+02	3.48e+02	2.00e+02	2.15e+02	5.25e+01
27	4.00e+02	8.56e+02	6.73e+02	6.47e+02	1.37e+02
28	1.00e+02	1.38e+03	3.00e+02	3.88e+02	3.23e+02

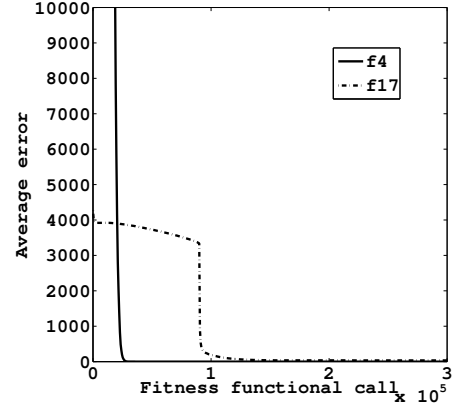


Fig. 3. Average fitness (error) trend of SMADE for f4 and f17 in 30D

TABLE III. RESULTS OF SMADE IN 50D.

Func.	Best	Worst	Median	Mean	Std
1	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00
2	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00
3	5.52e-06	9.30e+06	1.38e+04	3.81e+05	1.35e+06
4	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00
5	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00
6	0.00e+00	4.91e+01	4.34e+01	4.30e+01	6.28e+00
7	1.89e+01	1.02e+02	4.22e+01	4.32e+01	1.66e+01
8	2.10e+01	2.12e+01	2.11e+01	2.11e+01	3.85e-02
9	3.64e+01	5.25e+01	4.26e+01	4.36e+01	4.06e+00
10	0.00e+00	7.14e-02	2.46e-02	2.47e-02	1.48e-02
11	2.39e+01	8.76e+01	4.68e+01	4.81e+01	1.49e+01
12	7.06e+01	2.77e+02	1.55e+02	1.57e+02	4.52e+01
13	2.07e+02	4.57e+02	3.40e+02	3.35e+02	5.63e+01
14	5.41e+01	7.90e+02	3.14e+02	3.41e+02	2.05e+02
15	6.27e+03	1.05e+04	8.74e+03	8.54e+03	9.77e+02
16	1.95e-02	2.38e-01	7.91e-02	8.96e-02	4.24e-02
17	5.46e+01	8.45e+01	6.46e+01	6.57e+01	5.27e+00
18	1.19e+02	2.67e+02	1.91e+02	1.93e+02	3.46e+01
19	3.11e+00	7.63e+00	5.57e+00	5.43e+00	1.07e+00
20	1.72e+01	2.12e+01	1.92e+01	1.92e+01	8.86e-01
21	2.00e+02	1.12e+03	8.36e+02	8.46e+02	3.43e+02
22	8.69e+01	1.18e+03	2.64e+02	3.39e+02	2.24e+02
23	6.34e+03	1.30e+04	1.06e+04	9.89e+03	1.90e+03
24	2.72e+02	3.26e+02	3.02e+02	3.00e+02	1.20e+01
25	3.29e+02	3.92e+02	3.69e+02	3.68e+02	1.36e+01
26	1.82e+02	4.15e+02	2.01e+02	2.91e+02	9.70e+01
27	8.10e+02	1.57e+03	1.17e+03	1.18e+03	1.67e+02
28	4.00e+02	4.26e+03	4.00e+02	1.07e+03	1.27e+03

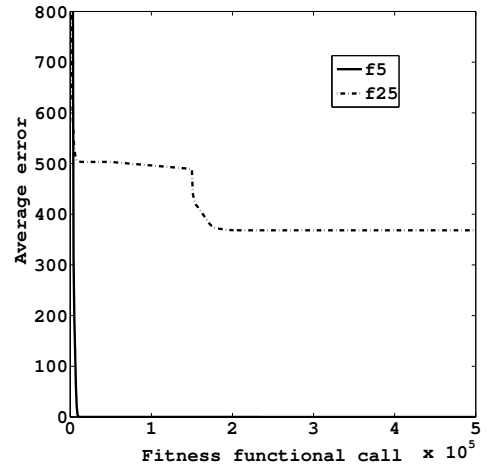


Fig. 4. Average fitness (error) trend of SMADE for f5 and f25 in 50D

TABLE IV. COMPUTATIONAL COMPLEXITY.

	T0	T1	$\widehat{T2}$	$(\widehat{T2} - T1)/T0$
D = 10	21.0	824.0	1847.0	48.7
D = 30		2005.0	5764.2	179
D = 50		3417.0	13280.8	469.7

A. Computational Complexity

Table IV reports four measures of the computational complexity of SMADE by following the procedure reported in [21]. More specifically, **T0** is the execution time of a benchmark program performing some simple mathematical operations in double precision, **T1** is the time needed to compute 200000 evaluations of the benchmark function **f14**, and **$\widehat{T2}$** it the average execution time (over 5 repetitions) of a run of SMADE while performing 200000 evaluations on the benchmark function **f3**. Parameters **T1**, **T1** and **$\widehat{T2}$** are here expressed in ms. A fourth, adimensional metric, is also defined as $(\widehat{T2} - T1)/T0$. In order to show how the algorithmic complexity varies with the problem dimension, **T1**, **$\widehat{T2}$** , $(\widehat{T2} - T1)/T0$ are reported for the three dimensionality values considered in this study (10, 30 and 50).

As in our previous experiments, the algorithm, as well as the program to compute its complexity, was coded in Java, wrapping the C/C++ implementation of the benchmark functions through JNI. The program was executed, in single thread, on a Linux Ubuntu system, version 12.04.2, with 2.66 GHz Intel Core 2 Quad processor, and 4 GB RAM. The timestamps were measured through the Java system call `System.currentTimeMillis()`.

The proposed SMADE contains two fairly complex components. The use of a covariance matrix has an important impact on the computational times as it introduces a quadratic complexity in the early stages of the optimization. On the other hand, the presence of CMAES at the beginning of the optimization is of great help to solve some problems in the benchmark and assists the DE framework to achieve solutions with a high quality. Although the MADE framework requires some non-trivial calculations to select the mutation strategy, its complexity is much lower than that of CMAES. In other words, the impact of MADE on complexity after the first 30% of the budget partially mitigates that of CMAES usage to generate the super-fit individuals. The computational complexity of the entire SMADE is thus super-linear but still sub-quadratic. This property makes the SMADE algorithm as an attractive option for optimization problem with a medium-relatively high number of dimensions.

B. Comparison against state-of-the-art algorithms

In order to provide a clear understanding of the SMADE performance, we compared the results shown in the previous section - on the same CEC 2013 benchmark and under the same conditions of budget and repetitions - with three different state-of-the-art optimization algorithms, namely the original CMAES [23], the Self-Adaptive Differential Evolution with pBX crossover (MDE-pBX) [29], and the Cooperatively Coevolving Particle Swarms Optimizer (CCPSO2) [30]. As suggested in their original papers, MDE-pBX was run with population size equal to 100 individuals and group size q equal

TABLE V. HOLM TEST ON THE FITNESS, REFERENCE ALGORITHM = SMADE (RANK = 3.18E+00)

j	Optimizer	Rank	z_j	p_j	δ/j	Hypothesis
1	MDE-pBX	2.49e+00	-4.47e+00	3.82e-06	5.00e-02	Rejected
2	CCPSO2	2.23e+00	-6.17e+00	3.37e-10	2.50e-02	Rejected
3	CMAES	1.87e+00	-8.49e+00	1.06e-17	1.67e-02	Rejected

to 15% of the population size, while CCPSO2 was executed with population size equal to 30 individuals, Cauchy/Gaussian sampling selection probability $p = 0.5$ and set of potential group sizes $S = \{2, 5\}$, $S = \{2, 5, 10\}$, $S = \{2, 5, 10, 25\}$, for experiments in 10, 30 and 50 dimensions, respectively. To provide a fair comparison, CMAES and SMADE were configured with the same values of λ , μ , σ .

Tables VI, VII and VIII, show the average fitness error (with respect to the target fitness) and its standard deviation, over 51 repetitions, obtained by the four algorithms under comparison after $T = 10000 \times D$ fitness evaluations. We report also the result of the Wilcoxon test [31] applied, with confidence level 0.95, to each pair-wise comparison on the final values obtained by SMADE against CMAES, MDE-pBX and CCPSO2 (“+”, “-” and “=” indicate, respectively, a better, worse or equivalent performance of SMADE against the other algorithms). It can be seen that SMADE is extremely competitive: indeed, it outperforms MDE-pBX in 12 cases in 10D, 16 cases in 30D, and 18 cases in 50D, thus showing an excellent scalability. Most impressively, SMADE systematically outperforms (with few exceptions) both CMAES and CCPSO2 in all the three dimensionality values. This result is confirmed by the Holm-Bonferroni procedure [32], which we performed as described in [33], with confidence level 0.05. As shown in Table V, SMADE is ranked first among the four algorithms, with the null-hypothesis (that SMADE and one of the other algorithms have indistinguishable performances) rejected in all the pair-wise comparisons.

IV. CONCLUSION

This paper proposes a novel algorithm for tackling the twenty-eight optimization problems listed in the CEC2013 benchmark in 10D, 30D, and 50D. The proposed algorithm applies the super-fit memetic logic to enhance upon a modified DE scheme. The super-fit logic consists of enhancing upon the initial performance of one solution by means of an exploitative component and injecting the improved solution into a starting population of DE. The solution has been here improved by means of the popular CMAES, thus allowing that unimodal and mildly multi-modal fitness landscapes are nearly solved in the initial stages of the optimization. The DE scheme here used is modified by the presence of multiple mutation strategies and an adaptive mechanism that tends to reward the most successful ones. This structure should then be able to handle high multi-modalities and thus continue the search performed by CMAES towards the optimum.

Numerical results show that the proposed SMADE is capable to solve several problems in the benchmark and can consistently achieve good results.

ACKNOWLEDGMENT

INCAS³ is co-funded by the Province of Drenthe, the Municipality of Assen, the European Fund for Regional De-

TABLE VI. AVERAGE ERROR $\pm$ STANDARD DEVIATION AND WILCOXON RANK-SUM TEST (REFERENCE =SMADE) FOR SMADE AGAINST MDE-PBX, CMAES, AND CCPSO2, ON CEC2013[21] IN 10 DIMENSIONS.

	SMADE	MDE-pBX		CMAES		CCPSO2	
f_1	0.00e+00 $\pm$ 0.00e+00	0.00e+00 $\pm$ 0.00e+00	boh	0.00e+00 $\pm$ 0.00e+00	=	1.92e-04 $\pm$ 1.16e-03	+
f_2	0.00e+00 $\pm$ 0.00e+00	4.15e+02 $\pm$ 9.60e+02	+	0.00e+00 $\pm$ 0.00e+00	=	9.93e+05 $\pm$ 7.58e+05	+
f_3	2.48e-01 $\pm$ 1.23e+00	4.96e+03 $\pm$ 4.66e+04	+	5.69e-01 $\pm$ 1.81e+00	=	2.13e+07 $\pm$ 3.13e+07	+
f_4	0.00e+00 $\pm$ 0.00e+00	6.50e-02 $\pm$ 6.38e-01	+	0.00e+00 $\pm$ 0.00e+00	=	8.80e+03 $\pm$ 2.50e+03	+
f_5	0.00e+00 $\pm$ 0.00e+00	0.00e+00 $\pm$ 7.54e-14	=	0.00e+00 $\pm$ 0.00e+00	=	2.94e-03 $\pm$ 8.06e-03	+
f_6	5.41e+00 $\pm$ 4.76e+00	6.18e+00 $\pm$ 4.73e+00	+	6.74e+00 $\pm$ 6.74e+00	=	1.52e+00 $\pm$ 2.91e+00	=
f_7	2.27e+00 $\pm$ 4.45e+00	5.63e+00 $\pm$ 7.94e+00	+	5.45e+08 $\pm$ 5.38e+09	+	3.27e+01 $\pm$ 8.31e+00	+
f_8	2.03e+01 $\pm$ 1.03e-01	2.05e+01 $\pm$ 1.06e-01	+	2.03e+01 $\pm$ 1.32e-01	=	2.04e+01 $\pm$ 7.66e-02	+
f_9	2.29e+00 $\pm$ 7.19e-01	2.37e+00 $\pm$ 1.41e+00	=	1.50e+01 $\pm$ 3.65e+00	+	4.98e+00 $\pm$ 9.13e-01	+
f_{10}	1.42e-02 $\pm$ 9.58e-03	1.25e-01 $\pm$ 9.20e-02	+	1.33e-02 $\pm$ 1.39e-02	=	1.47e+00 $\pm$ 6.44e-01	+
f_{11}	9.75e-02 $\pm$ 2.96e-01	2.48e+00 $\pm$ 1.61e+00	+	2.31e+02 $\pm$ 2.67e+02	+	1.97e+00 $\pm$ 1.18e+00	+
f_{12}	7.80e+00 $\pm$ 4.10e+00	1.06e+01 $\pm$ 4.76e+00	+	3.49e+02 $\pm$ 3.81e+02	+	2.64e+01 $\pm$ 7.93e+00	+
f_{13}	1.21e+01 $\pm$ 6.40e+00	2.01e+01 $\pm$ 7.94e+00	+	2.94e+02 $\pm$ 3.99e+02	+	3.56e+01 $\pm$ 8.30e+00	+
f_{14}	3.64e+00 $\pm$ 4.39e+00	1.25e+02 $\pm$ 1.12e+02	+	1.88e+03 $\pm$ 4.25e+02	+	5.27e+01 $\pm$ 3.86e+01	+
f_{15}	7.36e+02 $\pm$ 2.60e+02	7.26e+02 $\pm$ 2.61e+02	=	1.80e+03 $\pm$ 3.92e+02	+	8.92e+02 $\pm$ 2.18e+02	+
f_{16}	4.04e-01 $\pm$ 3.14e-01	5.43e-01 $\pm$ 4.49e-01	=	4.51e-01 $\pm$ 5.06e-01	=	1.18e+00 $\pm$ 2.19e-01	+
f_{17}	1.03e+01 $\pm$ 1.55e-01	1.32e+01 $\pm$ 1.85e+00	+	9.55e+02 $\pm$ 3.42e+02	+	1.60e+01 $\pm$ 2.07e+00	+
f_{18}	2.46e+01 $\pm$ 4.68e+00	1.93e+01 $\pm$ 4.67e+00	-	9.01e+02 $\pm$ 3.10e+02	+	5.41e+01 $\pm$ 6.33e+00	+
f_{19}	3.95e-01 $\pm$ 1.25e-01	6.44e-01 $\pm$ 2.09e-01	+	1.19e+00 $\pm$ 5.00e-01	+	7.65e-01 $\pm$ 2.52e-01	+
f_{20}	2.65e+00 $\pm$ 4.48e-01	2.87e+00 $\pm$ 5.25e-01	+	4.68e+00 $\pm$ 3.79e-01	+	3.50e+00 $\pm$ 1.96e-01	+
f_{21}	3.83e+02 $\pm$ 5.50e+01	3.98e+02 $\pm$ 1.99e+01	+	3.68e+02 $\pm$ 8.71e+01	=	3.73e+02 $\pm$ 6.75e+01	-
f_{22}	4.93e+01 $\pm$ 5.33e+01	1.33e+02 $\pm$ 1.04e+02	+	2.32e+03 $\pm$ 4.25e+02	+	7.27e+01 $\pm$ 4.98e+01	+
f_{23}	5.78e+02 $\pm$ 3.16e+02	8.82e+02 $\pm$ 3.08e+02	+	2.25e+03 $\pm$ 4.48e+02	+	1.15e+03 $\pm$ 2.58e+02	+
f_{24}	2.02e+02 $\pm$ 1.76e+01	2.02e+02 $\pm$ 1.56e+01	=	3.83e+02 $\pm$ 1.57e+02	+	2.01e+02 $\pm$ 2.54e+01	-
f_{25}	2.02e+02 $\pm$ 1.91e+00	2.00e+02 $\pm$ 1.23e+01	-	2.62e+02 $\pm$ 4.92e+01	+	2.12e+02 $\pm$ 1.09e+01	+
f_{26}	1.26e+02 $\pm$ 3.69e+01	1.47e+02 $\pm$ 4.36e+01	+	2.57e+02 $\pm$ 1.13e+02	+	1.58e+02 $\pm$ 2.40e+01	+
f_{27}	3.37e+02 $\pm$ 5.23e+01	3.06e+02 $\pm$ 2.76e+01	-	4.23e+02 $\pm$ 1.35e+02	+	4.27e+02 $\pm$ 4.91e+01	+
f_{28}	3.17e+02 $\pm$ 6.87e+01	3.07e+02 $\pm$ 5.78e+01	=	1.21e+03 $\pm$ 1.21e+03	+	3.01e+02 $\pm$ 1.28e+02	-

TABLE VII. AVERAGE ERROR $\pm$ STANDARD DEVIATION AND WILCOXON RANK-SUM TEST (REFERENCE =SMADE) FOR SMADE AGAINST MDE-PBX, CMAES, AND CCPSO2, ON CEC2013[21] IN 30 DIMENSIONS.

	SMADE	MDE-pBX		CMAES		CCPSO2	
f_1	0.00e+00 $\pm$ 1.27e-13	0.00e+00 $\pm$ 4.09e-13	=	0.00e+00 $\pm$ 5.57e-14	=	2.27e-13 $\pm$ 1.21e-12	+
f_2	0.00e+00 $\pm$ 1.74e-13	9.56e+04 $\pm$ 6.16e+04	+	0.00e+00 $\pm$ 7.88e-14	=	9.95e+05 $\pm$ 5.24e+05	+
f_3	9.82e+03 $\pm$ 4.94e+04	1.80e+07 $\pm$ 3.12e+07	+	1.53e+01 $\pm$ 1.23e+02	-	5.59e+08 $\pm$ 5.57e+08	+
f_4	0.00e+00 $\pm$ 1.86e-13	1.32e+01 $\pm$ 5.46e+01	+	0.00e+00 $\pm$ 5.08e-14	=	5.57e+04 $\pm$ 2.06e+04	+
f_5	1.14e-13 $\pm$ 2.75e-13	1.14e-13 $\pm$ 8.35e-13	=	1.14e-13 $\pm$ 1.14e-14	=	2.62e-08 $\pm$ 6.05e-08	+
f_6	2.67e+00 $\pm$ 7.85e+00	1.99e+01 $\pm$ 2.22e+01	+	3.05e+00 $\pm$ 8.33e+00	+	2.19e+01 $\pm$ 2.27e+01	+
f_7	3.25e+01 $\pm$ 1.61e+01	5.70e+01 $\pm$ 1.77e+01	+	9.83e+03 $\pm$ 6.44e+04	=	1.15e+02 $\pm$ 3.11e+01	+
f_8	2.10e+01 $\pm$ 4.80e-02	2.11e+01 $\pm$ 5.94e-02	+	2.09e+01 $\pm$ 6.66e-02	=	2.10e+01 $\pm$ 4.60e-02	+
f_9	2.23e+01 $\pm$ 3.57e+00	2.22e+01 $\pm$ 4.80e+00	=	4.45e+01 $\pm$ 6.99e+00	+	2.84e+01 $\pm$ 2.08e+00	+
f_{10}	1.84e-02 $\pm$ 1.34e-02	1.64e-01 $\pm$ 1.20e-01	+	1.73e-02 $\pm$ 1.25e-02	=	1.48e-01 $\pm$ 6.90e-02	+
f_{11}	1.09e+01 $\pm$ 4.18e+00	4.62e+01 $\pm$ 1.44e+01	+	9.47e+01 $\pm$ 2.36e+02	+	1.19e-01 $\pm$ 2.90e-01	-
f_{12}	5.72e+01 $\pm$ 1.70e+01	6.94e+01 $\pm$ 2.00e+01	+	7.11e+02 $\pm$ 9.66e+02	+	2.12e+02 $\pm$ 5.24e+01	+
f_{13}	1.28e+02 $\pm$ 3.50e+01	1.49e+02 $\pm$ 3.66e+01	+	1.64e+03 $\pm$ 1.66e+03	+	2.44e+02 $\pm$ 3.44e+01	+
f_{14}	1.33e+02 $\pm$ 1.27e+02	1.17e+03 $\pm$ 3.95e+02	+	5.21e+03 $\pm$ 7.37e+02	+	4.48e+00 $\pm$ 2.87e+00	-
f_{15}	4.10e+03 $\pm$ 8.47e+02	3.95e+03 $\pm$ 6.57e+02	=	5.37e+03 $\pm$ 6.73e+02	+	3.85e+03 $\pm$ 4.52e+02	=
f_{16}	1.31e-01 $\pm$ 7.57e-02	1.25e+00 $\pm$ 6.19e-01	+	1.26e-01 $\pm$ 8.87e-02	=	2.16e+00 $\pm$ 3.76e-01	+
f_{17}	3.48e+01 $\pm$ 1.52e+00	7.05e+01 $\pm$ 1.24e+01	+	3.95e+03 $\pm$ 7.89e+02	+	3.07e+01 $\pm$ 3.03e+00	-
f_{18}	8.33e+01 $\pm$ 2.06e+01	8.26e+01 $\pm$ 1.89e+01	=	4.23e+03 $\pm$ 8.31e+02	+	2.31e+02 $\pm$ 5.43e+01	+
f_{19}	2.55e+00 $\pm$ 5.18e-01	9.54e+00 $\pm$ 5.54e+00	+	3.66e+00 $\pm$ 9.52e-01	+	7.77e-01 $\pm$ 1.58e-01	-
f_{20}	1.05e+01 $\pm$ 8.07e-01	1.07e+01 $\pm$ 7.75e-01	+	1.50e+01 $\pm$ 6.45e-02	+	1.35e+01 $\pm$ 5.50e-01	+
f_{21}	3.27e+02 $\pm$ 8.65e+01	3.40e+02 $\pm$ 7.62e+01	+	3.05e+02 $\pm$ 9.01e+01	=	2.37e+02 $\pm$ 6.71e+01	-
f_{22}	1.79e+02 $\pm$ 4.50e+01	1.17e+03 $\pm$ 4.92e+02	+	6.97e+03 $\pm$ 1.06e+03	+	9.87e+01 $\pm$ 6.70e+01	-
f_{23}	4.22e+03 $\pm$ 8.74e+02	4.70e+03 $\pm$ 7.70e+02	+	6.76e+03 $\pm$ 6.82e+02	+	4.99e+03 $\pm$ 6.31e+02	+
f_{24}	2.32e+02 $\pm$ 2.57e+01	2.31e+02 $\pm$ 8.60e+00	-	8.19e+02 $\pm$ 6.15e+02	+	2.80e+02 $\pm$ 6.34e+00	+
f_{25}	2.78e+02 $\pm$ 9.90e+00	2.79e+02 $\pm$ 1.38e+01	=	3.46e+02 $\pm$ 1.45e+02	+	2.98e+02 $\pm$ 6.94e+00	+
f_{26}	2.15e+02 $\pm$ 5.25e+01	2.26e+02 $\pm$ 5.15e+01	=	5.51e+02 $\pm$ 5.14e+02	+	2.00e+02 $\pm$ 6.76e-01	-
f_{27}	6.47e+02 $\pm$ 1.37e+02	6.50e+02 $\pm$ 1.04e+02	=	8.53e+02 $\pm$ 2.42e+02	+	1.04e+03 $\pm$ 8.09e+01	+
f_{28}	3.88e+02 $\pm$ 3.23e+02	3.09e+02 $\pm$ 1.50e+02	-	1.96e+03 $\pm$ 3.40e+03	+	4.35e+02 $\pm$ 5.10e+02	+

TABLE VIII. AVERAGE ERROR $\pm$ STANDARD DEVIATION AND WILCOXON RANK-SUM TEST (REFERENCE =SMADE) FOR SMADE AGAINST MDE-PBX, CMAES, AND CCPSO2, ON CEC2013[21] IN 50 DIMENSIONS.

	SMADE	MDE-pBX		CMAES		CCPSO2	
f_1	2.27e-13 ± 0.00e+00	1.11e-11 ± 7.68e-11	+	2.27e-13 ± 0.00e+00	=	6.82e-13 ± 1.31e-12	+
f_2	2.27e-13 ± 0.00e+00	4.37e+05 ± 1.64e+05	+	2.27e-13 ± 0.00e+00	=	1.85e+06 ± 9.33e+05	+
f_3	3.81e+05 ± 1.35e+06	8.45e+07 ± 1.46e+08	+	9.64e+02 ± 4.49e+03	-	1.98e+09 ± 2.09e+09	+
f_4	2.27e-13 ± 0.00e+00	3.05e+01 ± 6.62e+01	+	2.27e-13 ± 0.00e+00	=	1.00e+05 ± 3.57e+04	+
f_5	6.82e-13 ± 2.38e-12	1.93e-11 ± 9.76e-11	+	6.82e-13 ± 1.57e-12	=	2.70e-10 ± 8.03e-10	+
f_6	4.30e+01 ± 6.28e+00	5.48e+01 ± 2.12e+01	+	4.32e+01 ± 7.10e+00	+	4.35e+01 ± 1.36e+01	+
f_7	4.32e+01 ± 1.66e+01	6.59e+01 ± 1.06e+01	+	4.19e+01 ± 1.70e+01	=	1.37e+02 ± 2.31e+01	+
f_8	2.11e+01 ± 3.85e-02	2.12e+01 ± 4.44e-02	+	2.11e+01 ± 9.75e-02	-	2.11e+01 ± 4.49e-02	=
f_9	4.36e+01 ± 4.06e+00	4.32e+01 ± 7.71e+00	=	7.66e+01 ± 7.77e+00	+	5.79e+01 ± 4.39e+00	+
f_{10}	2.47e-02 ± 1.48e-02	1.34e-01 ± 1.23e-01	+	2.24e-02 ± 1.49e-02	=	1.24e-01 ± 4.62e-02	+
f_{11}	4.81e+01 ± 1.49e+01	1.24e+02 ± 2.87e+01	+	2.19e+02 ± 4.56e+02	+	4.31e-01 ± 5.74e-01	-
f_{12}	1.57e+02 ± 4.52e+01	1.58e+02 ± 3.25e+01	+	2.25e+03 ± 1.37e+03	+	4.46e+02 ± 7.92e+01	+
f_{13}	3.35e+02 ± 2.63e+01	3.24e+02 ± 4.74e+01	=	3.36e+03 ± 1.09e+03	+	5.49e+02 ± 6.67e+01	+
f_{14}	3.41e+02 ± 2.05e+02	2.65e+03 ± 8.86e+02	+	8.82e+03 ± 1.04e+03	+	6.45e+00 ± 3.20e+00	-
f_{15}	8.54e+03 ± 9.77e+02	7.46e+03 ± 7.95e+02	=	9.09e+03 ± 9.43e+02	+	7.95e+03 ± 7.11e+02	+
f_{16}	8.96e-02 ± 4.24e-02	1.75e+00 ± 7.40e-01	+	8.01e-02 ± 4.72e-02	=	2.39e+00 ± 5.90e-01	+
f_{17}	6.57e+01 ± 5.27e+00	1.75e+02 ± 3.72e+01	+	6.97e+03 ± 1.07e+03	+	5.14e+01 ± 2.84e-01	-
f_{18}	1.93e+02 ± 3.46e+01	1.85e+02 ± 3.40e+01	=	7.08e+03 ± 9.14e+02	+	4.94e+02 ± 1.08e+02	+
f_{19}	5.43e+00 ± 1.07e+00	4.25e+01 ± 2.66e+01	+	6.32e+00 ± 1.18e+00	+	1.40e+00 ± 2.19e-01	-
f_{20}	1.92e+01 ± 8.86e-01	2.00e+01 ± 9.04e-01	+	2.50e+01 ± 9.73e-02	+	2.28e+01 ± 7.85e-01	+
f_{21}	8.46e+02 ± 3.43e+02	9.22e+02 ± 3.06e+02	+	8.12e+02 ± 3.73e+02	+	3.27e+02 ± 2.64e+02	-
f_{22}	3.39e+02 ± 2.24e+02	3.09e+03 ± 9.98e+02	+	1.19e+04 ± 1.26e+03	+	7.58e+01 ± 8.58e+01	-
f_{23}	9.89e+03 ± 1.90e+03	8.88e+03 ± 1.20e+03	=	1.18e+04 ± 8.52e+02	+	1.05e+04 ± 1.11e+03	+
f_{24}	3.00e+02 ± 1.20e+01	2.87e+02 ± 1.47e+01	+	1.64e+03 ± 1.05e+03	+	3.56e+02 ± 9.89e+00	+
f_{25}	3.68e+02 ± 1.36e+01	3.69e+02 ± 1.78e+01	+	4.94e+02 ± 1.88e+02	+	3.96e+02 ± 1.19e+01	+
f_{26}	2.91e+02 ± 9.70e+01	3.50e+02 ± 7.93e+01	+	6.04e+02 ± 7.11e+02	+	2.09e+02 ± 3.92e+01	-
f_{27}	1.18e+03 ± 1.67e+02	1.24e+03 ± 1.56e+02	+	1.28e+03 ± 2.51e+02	+	1.79e+03 ± 8.78e+01	+
f_{28}	1.07e+03 ± 1.27e+03	4.33e+02 ± 3.26e+02	=	3.27e+03 ± 5.60e+03	+	6.33e+02 ± 8.95e+02	+

velopment and the Ministry of Economic Affairs, Peaks in the Delta. The work of FN is supported by the Academy of Finland, Akatemiututkija 130600, "Algorithmic design issues in Memetic Computing" and Tutkijatohtori 140487, "Algorithmic Design and Software Implementation: a Novel Optimization Platform". The numerical experiments have been carried out on the computer network of the De Montfort University by means of the software for distributed optimization Kimeme [34]. The work of GZ and JC is supported by the National Natural Science Foundation of China (61170016), the Program for New Century Excellent Talents in University (NCET-11-0715) and SWJTU supported project (SWJTU12CX008)

REFERENCES

- [1] K. Price and R. Storn, "Differential evolution: A simple evolution strategy for fast optimization," *Dr. Dobbs's J. Software Tools*, vol. 22, no. 4, pp. 18–24, 1997.
- [2] R. Joshi and A. C. Sanderson, "Minimal representation multisensor fusion using differential evolution," *IEEE Transactions on Systems, Man and Cybernetics, Part A*, vol. 29, no. 1, pp. 63–76, 1999.
- [3] N. Karaboga, "Digital IIR Filter Design Using Differential Evolution Algorithm," *EURASIP Journal on Applied Signal Processing*, vol. 8, pp. 1269–1276, 2005.
- [4] V. Tirronen, F. Neri, T. Kärkkäinen, K. Majava, and T. Rossi, "An enhanced memetic differential evolution in filter design for defect detection in paper production," *Evolutionary Computation*, vol. 16, pp. 529–555, 2008.
- [5] F. Neri and E. Mininno, "Memetic Compact Differential Evolution for Cartesian Robot Control," *IEEE Computational Intelligence Magazine*, vol. 5, no. 2, pp. 54–65, 2010.
- [6] G. Iacca, F. Caraffini, and F. Neri, "Compact differential evolution light," *Journal of Computer Science and Technology*, vol. 27, no. 5, pp. 1056–1076, 2012.
- [7] G. Iacca, F. Caraffini, E. Mininno, and F. Neri, "Robot base disturbance optimization with compact differential evolution light," in *EvoApplications*, ser. Lecture Notes in Computer Science. Springer, 2012, pp. 285–294.
- [8] F. Neri and V. Tirronen, "Recent Advances in Differential Evolution: A Review and Experimental Analysis," *Artificial Intelligence Review*, vol. 33, no. 1–2, pp. 61–106, 2010.
- [9] J. Brest, S. Greiner, B. Bošković, M. Mernik, and V. Žumer, "Self-Adapting Control Parameters in Differential Evolution: A Comparative Study on Numerical Benchmark Problems," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 6, pp. 646–657, 2006.
- [10] E. Mininno, F. Neri, F. Cupertino, and D. Naso, "Compact Differential Evolution," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 1, pp. 32–54, 2011.
- [11] A. K. Qin, V. L. Huang, and P. N. Suganthan, "Differential Evolution Algorithm With Strategy Adaptation for Global Numerical Optimization," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 2, pp. 398–417, 2009.
- [12] N. Noman and H. Iba, "Accelerating Differential Evolution Using an Adaptive Local Search," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 1, pp. 107–125, 2008.
- [13] F. Neri and V. Tirronen, "Scale Factor Local Search in Differential Evolution," *Memetic Computing Journal*, vol. 1, no. 2, pp. 153–171, 2009.
- [14] A. Caponio, F. Neri, and V. Tirronen, "Super-fit control adaptation in memetic differential evolution frameworks," *Soft Computing - A Fusion of Foundations, Methodologies and Applications*, vol. 13, pp. 811–831, 2009.
- [15] A. Caponio, G. L. Cascella, F. Neri, N. Salvatore, and M. Sumner, "A fast adaptive memetic algorithm for on-line and off-line control design of PMSM drives," *IEEE Transactions on System Man and Cybernetics-part B*, vol. 37, no. 1, pp. 28–41, 2007.
- [16] F. Neri, J. I. Toivanen, G. L. Cascella, and Y. S. Ong, "An Adaptive Multimeme Algorithm for Designing HIV Multidrug Therapies," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 4, no. 2, pp. 264–278, 2007.
- [17] F. Neri, G. L. Cascella, N. Salvatore, A. V. Kononova, and G. Acciani, "Prudent-Daring vs Tolerant Survivor Selection Schemes in Control Design of Electric Drives," in *Applications of Evolutionary Computing*, ser. Lecture Notes in Computer Science, Rothlauf, F. et al., Ed., vol. LNCS 3907. Springer, 2006, pp. 805–809.
- [18] V. Tirronen and F. Neri, "Differential Evolution with Fitness Diversity Self-Adaptation," in *Nature-Inspired Algorithms for Optimisation, Studies in Computational Intelligence*, ser. Studies in Computational Intelligence, R. Chiong, Ed. Springer, 2009, vol. 193, pp. 199–234.
- [19] G. Iacca, R. Mallipeddi, E. Mininno, F. Neri, and P. N. Suganthan, "Super-fit and Population Size Reduction Mechanisms in Compact Differential Evolution," in *Proceedings of IEEE Symposium on Memetic Computing*, 2011, pp. 21–28.
- [20] H. H. Rosenbrock, "An automatic Method for finding the greatest or least Value of a Function," *The Computer Journal*, vol. 3, no. 3, pp. 175–184, 1960.
- [21] J. J. Liang, B. Y. Qu, P. N. Suganthan, and A. G. Hernandez-Daz, "Problem Definitions and Evaluation Criteria for the CEC 2013 Special Session on Real-Parameter Optimization," Zhengzhou University and Nanyang Technological University, Zhengzhou China and Singapore, Tech. Rep. 201212, 2013.
- [22] N. Hansen and A. Ostermeier, "Completely derandomized self-adaptation in evolution strategies," *Evolutionary Computation*, vol. 9, no. 2, pp. 159–195, 2001.
- [23] N. Hansen, S. D. Müller, and P. Koumoutsakos, "Reducing the Time Complexity of the Derandomized Evolution Strategy with Covariance Matrix Adaptation (CMA-ES)," *Evolutionary Computation*, vol. 11, no. 1, pp. 1–18, 2003.
- [24] J. Cheng, G. Zhang, F. Caraffini, and F. Neri, "Multicriteria adaptive differential evolution for global numerical optimization," *Applied Soft Computing*, submitted.
- [25] N. Hansen, "The CMA Evolution Strategy," 2012, <http://www.lri.fr/~hansen/cmaesintro.html>.
- [26] A. Auger and N. Hansen, "A Restart CMA Evolution Strategy With Increasing Population Size," in *Proceedings of the IEEE Congress on Evolutionary Computation*, 2005, pp. 1769–1776.
- [27] C. Igel, T. Suttorp, and N. Hansen, "A Computational Efficient Covariance Matrix Update and a (1+1)-CMA for Evolution Strategies," in *Proceedings of the Genetic and Evolutionary Computation Conference*. ACM Press, 2006, pp. 453–460.
- [28] J. Zhang and A. C. Sanderson, "JADE: Adaptive Differential Evolution with Optional External Archive," vol. 13, no. 5, 2009, pp. 945–958.
- [29] S. Islam, S. Das, S. Ghosh, S. Roy, and P. Suganthan, "An Adaptive Differential Evolution Algorithm With Novel Mutation and Crossover Strategies for Global Numerical Optimization," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 42, no. 2, pp. 482–500, april 2012.
- [30] X. Li and X. Yao, "Cooperatively Coevolving Particle Swarms for Large Scale Optimization," *Evolutionary Computation, IEEE Transactions on*, vol. 16, no. 2, pp. 210–224, april 2012.
- [31] F. Wilcoxon, "Individual comparisons by ranking methods," *Biometrics Bulletin*, vol. 1, no. 6, pp. 80–83, 1945.
- [32] S. Holm, "A simple sequentially rejective multiple test procedure," *Scandinavian Journal of Statistics*, vol. 6, no. 2, pp. 65–70, 1979.
- [33] S. Garcia, A. Fernandez, J. Luengo, and F. Herrera, "A study of statistical techniques and performance measures for genetics-based machine learning: accuracy and interpretability," *Soft Computing*, vol. 13, no. 10, pp. 959–977, 2008.
- [34] Cyber Dyne Srl Home Page, "Kimeme," 2012, <http://cyberdynesoft.it/>.