Micro-Differential Evolution with Extra Moves Along the Axes

Conference Paper · April 2013

DOI: 10.1109/SDE.2013.6601441

3 authors:

Fabio Caraffini
De Montfort University
29 PUBLICATIONS **150** CITATIONS
SEE PROFILE

Ferrante Neri
De Montfort University
123 PUBLICATIONS **2,408** CITATIONS
SEE PROFILE

Ilpo Poikolainen
University of Jyväskylä
9 PUBLICATIONS **30** CITATIONS
SEE PROFILE

Micro-Differential Evolution with Extra Moves Along the Axes

Fabio Caraffini and Ferrante Neri
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 and fneri@dmu.ac.uk
Department of Mathematical Information Technology,
P.O. Box 35 (Agora), 40014
University of Jyväskylä, Finland
Email: fabio.caraffini@jyu.fi and ferrante.neri@jyu.fi

Ilpo Poikolainen
Department of Mathematical Information Technology,
P.O. Box 35 (Agora), 40014
University of Jyväskylä, Finland
Email: ilpo.poikolainen@jyu.fi

***Abstract*—This paper proposes a novel implementation of micro-Differential Evolution (μDE) that incorporates within the DE scheme an extra search move that attempts to improve the best solution by perturbing it along the axes. These extra moves complement the DE search logic and allows the exploration of the decision space from an alternative perspective. In addition, these extra moves at subsequent activations tend to explore a progressively narrowing area. This mechanism increases the exploitation of the original scheme thus helping μDE to prevent from stagnation. An experimental set-up including various test problems and dimensionality values has been considered. Numerical results show that the proposed algorithm enhances upon the original μDE performance and, despite its simplicity, is competitive with modern complex DE based algorithms.**

I. Introduction

Population-size a crucially important, if not the most important, parameter in algorithms that process multiple solutions, such as Evolutionary and Swarm Intelligence Algorithms (EAs and SIAs, respectively), see e.g. [1]. The tuning of this parameter is hard since, the success of a given problem can heavily depend on it. Looking at this issue from a complementary perspective, a robust algorithmic design might have a variable population size. This variation can be deterministic as in [2] or self-adaptive as in [3], [4], and [5]. The topic whether a large, a small, or unitary population is preferable is a topic under discussion in computational intelligence community.

In this regard, an extensive study on population-based algorithms and their advantages over single solution algorithms has been reported in [6]. Five distinct mechanisms that would justify the superiority of population-based algorithms over schemes that perturb a single solution have been identified and studied. The first mechanism is that a population offers a diversified pool of building blocks whose combination might generate new promising solutions. The second mechanism is the result of focusing of the search caused by recombination operators. Since most recombination operators have the property that, if both parents share the same value of a variable, then the offspring also has the same value in correspondence of that variable, see [7], recombination has the power of exploring the part of the search space where individuals disagree. In contrast, mutation explores the entire search space. According to this analysis this mechanism of focusing of the search by crossover can dramatically enhance the speed of the algorithm to detect a good solution. The third mechanism is the capability of a population to act as a low-pass filter of the landscape, ignoring short-length scale features in the landscape (e.g. shallow basins of attractions). The fourth mechanism is the possibility to search different areas of the decision space. This mechanism can be seen also in a different way: since population-based algorithms naturally perform an initial multiple sampling, the chance that an unlucky initial sample jeopardizes the entire algorithmic functioning is significantly mitigated. The fifth mechanism is the opportunity of using the population to learn about good parameters of the algorithm, i.e. to find a proper balance between exploration and exploitation.

For the above-listed reasons, the employment of a population-based algorithm would, in principle, be preferable when possible. However, in counter-tendency with the analysis in [6], some algorithms recently proposed in literature, although based on a single solution, still display an excellent performance, even when compared with that of modern complex population-based algorithms, see e.g. [8].

Contradictory results in literature are not only about the advisability of using or not a population within an optimization framework, but also about the proper sizing of the population. Some studies clearly suggest the usage of large populations in order to ensure the success of the algorithm, see [9]. On the other hand, in [10] and [11], it is shown that, if properly designed, a population-based algorithm with a very small population size can efficiently solve large scale problems, see also [12] and [13].

The latter kind of algorithms, i.e. population-based algorithm that use a small population, are indicated as micro algorithms and indicated by the prefix μ. An early implementation

of micro algorithm is the micro Genetic Algorithm (μGA), see e.g. [14] and [15]. Over the latest years, micro algorithms have been employed in various engineering applications as they are proven to be lighter in terms of hardware requirements and thus are prone to their use in embedded systems, see [16]. In addition, algorithms that make use of small populations are more exploitative than the large ones and thus quickly achieve improvements during the early stages of the optimization process. This feature makes micro-algorithms especially useful in real-time applications when a quick answer is needed, see e.g. [17]. The effect of small populations is obviously different when applied to various search strategies, see [18].

Amongst the various micro algorithms proposed in literature, micro-Differential Evolution (μDE) is a successfully applied scheme. For example, in [19] a μDE employing opposition-based mechanism has been proposed for image thresholding problems. In [20] a μDE apprach is proposed for evolving an indirect representation of the Bin Packing Problem.

This paper proposes a novel implementation of μDE, namely micro-Differential Evolution with Axis-moves (μDEA). The proposed μDEA is a DE/rand/1/exp scheme, see [21], that employs a very small population. In addition, μDEA makes use of an extra refinement operator that perturbs the solution of the micro-population characterized by the highest performance. This refinement operator attempts to improve upon the solution by means of an exploratory move in the direction of each variable.

The remainder of this paper is organized in the following way. Section II describes the working principles of the proposed μDEA. Section III displays the experimental results of this study. Section IV gives the conclusions of this work.

II. Micro-Differential Evolution with Axis Moves

Without a loss of generality, in order to clarify the notation in this paper, we refer to the minimization problem of an objective function $f(x)$, where the candidate solution x is a vector of n design variables (or genes) in a decision space D. The i^{th} design variable of the vector x is indicated as $x[i]$. The proposed μDEA algorithm consists of a DE framework and the extra moves along the axes. Section II-A and II-B describe framework and extra moves, respectively. Section II-C analyzes the μDE behavior and gives a justification to the proposed algorithmic structure.

A. Micro-Differential Evolution framework

At the beginning of the optimization process, a sampling of S_{pop} individuals is performed randomly with a uniform distribution function within the decision space D. In our implementation, the μDE population size S_{pop} has been set equal to 5.

At each generation, for each individual x_j of the S_{pop}, three individuals x_r, x_s and x_t are randomly extracted from the population. According to the DE logic, a provisional offspring x'_{off} is generated by mutation:

$$x'_{off} = x_t + F(x_r - x_s) \quad (1)$$

$x_{off} = x_j$
generate $i = round\,(n \cdot rand\,(0,1))$
$x_{off}[i] = x'_{off}[i]$
$k = 1$
while $rand\,(0,1) \leq Cr$ AND $k < n$ **do**
  $x_{off}[i] = x'_{off}[i]$
  $i = i + 1$
  if $i == n$ **then**
    $i = 1$
  end if
  $k = k + 1$
end while

Fig. 1. Pseudo code of the exponential crossover

where $F \in [0, 1+\epsilon[$ is a scale factor which controls the length of the exploration vector $(x_r - x_s)$ and thus determines how far from point x_j the offspring should be generated. With $F \in [0, 1+\epsilon[$, it is meant here that the scale factor should be a positive value which cannot be much greater than 1 (i.e. ϵ is a small positive value), see [22]. While there is no theoretical upper limit for F, effective values are rarely greater than 1.0. The mutation scheme given in Equation (1) is also known as DE/rand/1. In literature many other mutation variants have been proposed, see [21] and [23].

When the provisional offspring has been generated by mutation, a popular crossover, namely exponential crossover is applied to the parent solution x_j and the provisional offspring x'_{off}, see [22]. In this crossover scheme, the number of variables x_j that are exchanged during one crossover follows a geometric distribution. A geometric distribution is the discrete counterpart of the exponential distribution (that gives the name to this operator).

In the exponential crossover, a design variable of the provisional offspring $x'_{off}\,(j)$ is randomly selected and copied into the j^{th} design variable of the solution x_i. This guarantees that parent and offspring have different genotypes. Subsequently, a set of random numbers between 0 and 1 are generated. As long as $\mathrm{rand}\,(0,1) \leq CR$, where the crossover rate CR is a predetermined parameter, the design variables from the provisional offspring (mutant) are copied into the corresponding positions of the parent x_i. The first time that $\mathrm{rand}\,(0,1) > CR$ the copy process is interrupted. Thus, all the remaining design variables of the offspring are copied from the parent. When this crossover is combined with the DE/rand/1 mutation, the algorithm is referred to as DE/rand/1/exp (in our case μDE/rand/1/exp). For the sake of clarity the pseudo-code of the exponential crossover is shown in Fig. 1.

As shown in [24], it can easily be observed that for a given value of Cr, the meaning of the exponential crossover would change with the dimensionality of the problem. For low dimensionality problems the trial solution would inherit most of the genes from the elite while for high dimensionality problems, only a small portion of x_e would be copied into x_t. In order to avoid this problem and make the crossover action independent on the dimensionality of the problem, the

following quantity is fixed:

$$\alpha_e \approx \frac{n_e}{n} \tag{2}$$

where n_e is the number of genes we expect to copy from parent to offspring in addition to that gene deterministically copied. The probability that n_e genes are copied is $Cr^{n_e} = Cr^{n\alpha_e}$. In order to control the approximate amount of copied genes and to achieve that about n_e genes are copied into the offspring with probability 0.5, we imposed that

$$Cr^{n\alpha_e} = 0.5. \tag{3}$$

It can easily be seen that, for a chosen α_e, the crossover rate can be set on the basis of the dimensionality as follows:

$$Cr = \frac{1}{\sqrt[n\alpha_e]{2}}. \tag{4}$$

By means of formula (4), the expected quantity of information to be transmitted from parent to offspring is controlled.

B. Extra moves along the axes

Let us indicate with x_p the *pivot* individual, i.e. the individual of the micro-population that displays the best performance. With a given probability η, the pivot individual undergoes the following operator that perturbs a single solution along its n axes, i.e. separately perturbs each design variable. This operator can be seen as a modification of a classical hill-descend algorithm and employs the perturbation logic proposed in [25].

The implementation of this operator requires an additional solution, which will here be referred to as x_s. The pivot individual x_p is perturbed by computing, for each variable i:

$$x_s[i] = x_p[i] - \rho, \tag{5}$$

where ρ is the exploratory radius. Subsequently, if x_s outperforms x_p, its values (the values of the vector elements) are saved and the pivot solution is updated), otherwise a half step in the opposite direction is taken:

$$x_s[i] = x_p[i] + \frac{\rho}{2}. \tag{6}$$

Again, x_s replaces x_p if it outperforms it. If there is no update, i.e. the exploration was unsuccessful, the radius ρ is halved. This operation is repeated a limited prefixed amount of times $Iter$, thus working as a shallow local search. The current value of ρ is saved and used as the initial radius for the subsequent activation of this operator.

The complete pseudo-code of the proposed μDEA algorithm is shown in Fig. 2.

C. Algorithmic functioning

The DE algorithm is a very versatile and efficient optimizer for continuous optimization problem. However, the original scheme has a wide margin of improvement. For this reason, part of the computer science community put an energetic effort in order to propose DE variants that can outperform the original DE scheme over various optimization problems. Some of these variants turned out to be very successful.

```
generate randomly S_pop individuals of the initial population
and compute their fitness values
while the computational budget is smaller than the prefixed
amount do
  for j = 1 : S_pop do
    select three individuals x_r, x_s, and x_t
    compute mutant individual x_off = x_t + F(x_r − x_s)
    compute exponential crossover in Fig. 1
  end for
  for j = 1 : S_pop do
    compute f(x_j)
  end for
  if rand(0,1) < η then
    extract the pivot individual x_p from the micro-
    population
    x_s = x_p
    for k = 1 : Iter do
      for i = 1 : n do
        compute x_s[i] = x_p[i] − ρ
        if f(x_s) ≤ f(x_p) then
          x_p = x_s
        else
          compute x_s[i] = x_p[i] + ρ/2
          if f(x_s) ≤ f(x_p) then
            x_p = x_s
          end if
        end if
      end for
    end for
  end if
end while
```

Fig. 2. Pseudo code of the μDEA algorithm

For example, the so called jDE [26] proposed a controlled randomization of the DE parameters. Another popular DE variant based on controlled randomization of the parameters has been proposed in [27]. The Self-Adaptive Differential Evolution (SADE) proposed in [28] employs multiple mutation strategies and a randomized coordination scheme based on an initial learning. Another efficient coordination strategy for a multiple mutation structure has been proposed in [29]. A modified selection strategy, based on the location within the population, for the individuals undergoing mutation has been proposed in [30]. Another example of efficient DE variant has been proposed in [31] where a novel DE mutation is combined with a randomized fitness based selection of the individuals undergoing mutation.

As highlighted in [23] and [32], the reasons behind the wide margin of improvements for the original DE scheme are mainly two. The first reason is that DE scheme has a limited amount of search moves. Thus, DE variants that include extra moves into the original framework, usually, lead to improved versions. The extra moves can be explicitly implemented within the DE framework, see e.g. [33] and [34], or can be implicitly contained in other perturbation mechanism. The randomization, as shown in [32] and [35], plays a very important role as it allows the generation of candidate solutions that would not be generated by standard mutation and crossover operations. The second reason is that

DE can be excessively exploratory. As shown in [36], a typical challenge in DE functioning is that the solutions in a population can be diverse and still unable to outperform the individual with the best performance (i.e. DE can easily suffers from stagnation). In order to prevent from this condition, the employment of exploitative component or the implementation of exploitative actions can be beneficial to DE performance.

The μDE schemes, due to the fact that use a small population-size, are intrinsically more exploitative than standard DE schemes and this would, in principle, make them less prone to stagnation issues. On the other hand, small populations could potentially lead to an excessively quick diversity loss and thus to a premature convergence. The undesired premature convergence effect would actually have a major impact on the performance on a micro-Evolutionary Algorithm (μEA), such as a μGA. Unlike μEAs, the DE search logic does not appear to lead too often to a diversity loss. In low dimensions and for simple fitness landscapes, a DE with a small population would obviously lose the diversity and converge to a solution. On the other hand, in complex multi-modal and multi-dimensional problems (already in 30 dimensions), even though only a few solutions (e.g. 5) compose the population of a DE scheme, the μDE population tends to keep the diversity high and its solutions could still be distant within the decision space D. Since the distance among solutions in a μDE scheme is correlated to the position of the potential offspring, after initial improvements, a μDE can be too exploratory and generate new points far away from the interesting areas. On the contrary, in order to continue achieving fitness improvements, the algorithm may require to enhance the exploitation and focus the search in the areas of interest.

The proposed μDEA aims at compensating this effect by including within the search moves an alternative exploration rule for the neighbourhood of the best solution. The extra moves along the axes are supposed to offer, in a simplistic way, a support to the μDE framework. These moves offer an alternative search logic with respect to the normal DE mutation and crossover and, most importantly, performs thorough exploration of the most interesting areas so far detected, i.e. the areas surrounding the solution characterized by the best performance. As a result, μDEA explicitly incorporates extra moves within a μDE framework and increases the exploitation of the original scheme. Finally, the fact that the exploratory radius of the moves along the axes is not re-initialized (but used for the subsequent activation) results in a natural increase in the exploitation action of this operator. In this way, the moves along the axes explore a progressively narrowing area around the pivot solution x_p.

III. Numerical Results

All the test problems included in the following four testbeds have been considered in this study.

- The CEC2005 benchmark described in [37] in 30 dimensions (25 test problems)
- The BBOB2010 benchmark described in [38] in 100 dimensions (24 test problems)
- The CEC2008 benchmark described in [39] in 1000 dimensions (7 test problems)
- The CEC2010 benchmark described in [40] in 1000 dimensions (20 test problems)

Thus, 76 test problems have been considered in this study. For each algorithm in this paper (see following subsections) 100 runs have been performed. Each run has been continued for $5000 \times n$ fitness evaluations, where n is the dimensionality of the problem. For each test problem and each algorithm, the average final fitness value $\pm$ standard deviation over the 100 available runs has been computed. In order to strengthen the statistical significance of the results, for each test problem the Wilcoxon Rank-Sum test [41] has been also applied, with a confidence level of 0.95.

The proposed μDEA has been run with $S_{pop} = 5$, $F = 0.7$, $\alpha_e = 0.5$, see eq. (2), $Iter = 20$, $\eta = 0.25$, and $\rho = 0.4$ of the width of the decision space D.

The following algorithms with respective parameter setting have been considered for comparison against μDEA.

- A μDE with the same parameter setting of μDEA
- Self-Adaptive Differential Evolution (SADE) proposed in [28] with population size equal to 50 individuals.
- Adaptive Differential Evolution (JADE) proposed in [27] with population size equal to 60 individuals, group size factor $p = 0.05$ and parameters adaptation rate factor $c = 0.1$.
- Modified Differential Evolution with p-Best Crossover (MDE-pBX) proposed in [31] with population size equal to 100 individuals and group size q equal to 15% of the population size.

Tables I, II, III, and IV show the comparison against μDE for the four benchmarks under consideration. Tables V, VI, VII, and VIII, show the comparison against SADE, JADE, and MDE-pBX. The tables in this study display the average final fitness value over the 100 available runs and the corresponding standard deviation value. The results of the Wilcoxon test are also reported in terms of pair-wise comparisons. The symbols"=" and "+" ("-") indicate, respectively, a statistically equivalent performance and a better (worse) performance of RIS compared with the algorithm in the column label.

The numerical comparison between μDEA and μDE shows that the extra moves along the axes tend to have a positive effect on the algorithmic performance in the majority of the considered cases. This fact confirms the validity of the analysis reported in [23] about the lack of moves in DE frameworks and that extra moves appear to be beneficial for DE. In addition, as shown Tables III and IV, the success of μDEA with respect to μDE in high dimensions demonstrates that an increase in the exploitation is beneficial also in DE schemes that employ a micro-population. In our opinion, this fact can be interpreted by considering that even in the case micro-populations, DE solutions tend to be scattered in the decision space, thus using a large exploration step whilst a neighbourhood search would

TABLE I
AVERAGE FITNESS ± STANDARD DEVIATION AND WILCOXON RANK-SUM TEST (REFERENCE =μDEA) FOR μDEA AGAINST μDE ON CEC2005[37] IN 30 DIMENSIONS.

	μDEA	μDE	
f_1	$\mathbf{-4.50e+02 \pm 2.18e-13}$	$-3.98e+02 \pm 5.14e+02$	+
f_2	$\mathbf{-4.50e+02 \pm 2.43e-12}$	$-4.29e+02 \pm 1.28e+02$	+
f_3	$\mathbf{1.83e+05 \pm 1.05e+05}$	$1.66e+07 \pm 5.61e+06$	+
f_4	$6.74e+04 \pm 1.60e+04$	$\mathbf{7.59e+02 \pm 1.22e+03}$	-
f_5	$\mathbf{7.20e+03 \pm 2.22e+03}$	$9.49e+03 \pm 2.07e+03$	+
f_6	$\mathbf{8.52e+02 \pm 1.03e+03}$	$2.79e+07 \pm 1.94e+08$	=
f_7	$\mathbf{-1.80e+02 \pm 1.35e-02}$	$2.99e+11 \pm 1.20e+12$	+
f_8	$\mathbf{-1.20e+02 \pm 4.75e-03}$	$-1.19e+02 \pm 5.98e-02$	+
f_9	$\mathbf{-1.17e+02 \pm 1.16e+00}$	$-1.16e+02 \pm 1.78e+00$	=
f_{10}	$2.62e+02 \pm 2.05e+01$	$\mathbf{2.56e+02 \pm 1.89e+01}$	-
f_{11}	$\mathbf{1.18e+02 \pm 3.55e+00}$	$1.21e+02 \pm 2.50e+00$	+
f_{12}	$\mathbf{1.49e+03 \pm 2.90e+03}$	$1.55e+04 \pm 6.55e+03$	+
f_{13}	$-1.22e+02 \pm 1.45e+00$	$\mathbf{-1.27e+02 \pm 1.20e+00}$	-
f_{14}	$-2.86e+02 \pm 2.78e-01$	$\mathbf{-2.87e+02 \pm 2.60e-01}$	-
f_{15}	$1.45e+03 \pm 2.89e+00$	$\mathbf{1.45e+03 \pm 4.25e+00}$	-
f_{16}	$1.59e+03 \pm 1.56e+01$	$\mathbf{1.58e+03 \pm 1.20e+01}$	=
f_{17}	$1.74e+03 \pm 1.81e+01$	$\mathbf{1.61e+03 \pm 1.13e+01}$	-
f_{18}	$\mathbf{9.10e+02 \pm 5.26e-12}$	$9.10e+02 \pm 5.41e-02$	+
f_{19}	$\mathbf{9.10e+02 \pm 5.82e-12}$	$9.10e+02 \pm 1.64e-01$	+
f_{20}	$\mathbf{9.10e+02 \pm 5.61e-12}$	$9.10e+02 \pm 4.18e-01$	+
f_{21}	$\mathbf{1.72e+03 \pm 1.09e+01}$	$1.72e+03 \pm 8.91e+00$	+
f_{22}	$2.60e+03 \pm 5.80e+01$	$\mathbf{2.54e+03 \pm 4.93e+01}$	-
f_{23}	$1.73e+03 \pm 9.39e+00$	$\mathbf{1.72e+03 \pm 8.43e+00}$	-
f_{24}	$1.71e+03 \pm 1.44e+01$	$\mathbf{1.71e+03 \pm 9.66e+00}$	=
f_{25}	$\mathbf{1.88e+03 \pm 3.40e+02}$	$1.91e+03 \pm 1.37e+02$	=

TABLE II
AVERAGE FITNESS ± STANDARD DEVIATION AND WILCOXON RANK-SUM TEST (REFERENCE = μDEA) FOR μDEA AGAINST μDE ON BBOB2010[38] IN 100 DIMENSIONS.

	μDEA	μDE	
f_1	$\mathbf{7.95e+01 \pm 3.23e-03}$	$7.95e+01 \pm 2.08e-01$	+
f_2	$\mathbf{-1.50e+02 \pm 1.73e+02}$	$1.67e+03 \pm 1.13e+04$	+
f_3	$-2.20e+02 \pm 1.05e+02$	$\mathbf{-4.09e+02 \pm 2.79e+01}$	-
f_4	$-1.16e+02 \pm 1.12e+02$	$\mathbf{-3.81e+02 \pm 3.26e+01}$	-
f_5	$\mathbf{-8.26e+00 \pm 2.06e+00}$	$-6.49e+00 \pm 4.89e+00$	+
f_6	$\mathbf{8.73e+01 \pm 1.34e+02}$	$4.32e+02 \pm 1.03e+02$	+
f_7	$\mathbf{4.42e+02 \pm 1.71e+02}$	$6.35e+02 \pm 8.28e+01$	+
f_8	$\mathbf{2.74e+02 \pm 9.87e+01}$	$2.99e+02 \pm 9.18e+01$	+
f_9	$\mathbf{1.92e+02 \pm 5.80e+01}$	$2.26e+02 \pm 2.78e+01$	+
f_{10}	$\mathbf{5.65e+04 \pm 9.88e+04}$	$2.07e+05 \pm 2.87e+04$	+
f_{11}	$8.30e+02 \pm 1.30e+02$	$\mathbf{6.43e+02 \pm 6.68e+01}$	-
f_{12}	$\mathbf{6.52e+02 \pm 3.39e+03}$	$7.29e+04 \pm 3.11e+05$	+
f_{13}	$\mathbf{4.02e+01 \pm 1.04e+01}$	$6.89e+01 \pm 9.00e+01$	=
f_{14}	$\mathbf{-5.23e+01 \pm 1.73e-02}$	$-5.23e+01 \pm 2.41e-01$	+
f_{15}	$\mathbf{2.36e+03 \pm 3.43e+02}$	$2.87e+03 \pm 2.09e+02$	+
f_{16}	$\mathbf{9.04e+01 \pm 6.19e+00}$	$9.80e+01 \pm 3.31e+00$	+
f_{17}	$\mathbf{-7.56e+00 \pm 2.73e+00}$	$-3.41e+00 \pm 2.08e+00$	+
f_{18}	$\mathbf{1.75e+01 \pm 8.38e+00}$	$3.62e+01 \pm 7.70e+00$	+
f_{19}	$\mathbf{-9.29e+01 \pm 2.50e+00}$	$-9.08e+01 \pm 8.39e-01$	+
f_{20}	$-5.45e+02 \pm 2.07e-01$	$\mathbf{-5.46e+02 \pm 6.08e-02}$	-
f_{21}	$4.98e+01 \pm 6.24e+00$	$\mathbf{4.32e+01 \pm 2.40e+00}$	-
f_{22}	$-9.87e+02 \pm 9.58e+00$	$\mathbf{-9.95e+02 \pm 6.34e+00}$	-
f_{23}	$\mathbf{8.60e+00 \pm 8.04e-01}$	$9.85e+00 \pm 3.78e-01$	+
f_{24}	$\mathbf{1.71e+03 \pm 3.62e+02}$	$2.10e+03 \pm 1.88e+02$	+

be more beneficial. The extra moves along the axes, explore progressively narrowing neighbourhood and support the basic DE search moves to detect solutions characterized by a high quality.

Numerical results in 30 dimensions show that the proposed μDEA is, in general, slightly less promising that some of

TABLE III
AVERAGE FITNESS ± STANDARD DEVIATION AND WILCOXON RANK-SUM TEST (REFERENCE = μDEA) FOR μDEA AGAINST μDE ON CEC2008[39] IN 1000 DIMENSIONS.

	μDEA	μDE	
f_1	$\mathbf{-4.50e+02 \pm 1.42e-09}$	$5.49e+02 \pm 2.25e+03$	+
f_2	$\mathbf{-4.50e+02 \pm 2.53e-02}$	$-3.59e+02 \pm 1.33e+01$	+
f_3	$\mathbf{1.52e+03 \pm 8.92e+01}$	$2.83e+08 \pm 1.52e+09$	+
f_4	$5.77e+03 \pm 4.56e+02$	$\mathbf{2.16e+02 \pm 5.30e+01}$	-
f_5	$\mathbf{-1.80e+02 \pm 1.89e-03}$	$-1.70e+02 \pm 2.25e+01$	+
f_6	$\mathbf{-1.40e+02 \pm 5.01e-07}$	$-1.37e+02 \pm 7.81e-01$	+
f_7	$-1.35e+04 \pm 6.61e+01$	$\mathbf{-1.44e+04 \pm 2.61e+01}$	-

TABLE IV
AVERAGE FITNESS ± STANDARD DEVIATION AND WILCOXON RANK-SUM TEST (REFERENCE = μDEA) FOR μDEA AGAINST μDE ON CEC2010[40] IN 1000 DIMENSIONS.

	μDEA	μDE	
f_1	$\mathbf{4.44e-18 \pm 7.20e-19}$	$4.45e+07 \pm 1.87e+08$	+
f_2	$5.71e+03 \pm 3.66e+02$	$\mathbf{5.26e+02 \pm 4.45e+01}$	-
f_3	$\mathbf{2.47e-02 \pm 9.79e-02}$	$2.88e+00 \pm 8.15e-01$	+
f_4	$\mathbf{2.07e+13 \pm 3.99e+12}$	$3.12e+13 \pm 7.40e+12$	+
f_5	$\mathbf{4.49e+08 \pm 1.16e+08}$	$6.32e+08 \pm 8.94e+07$	+
f_6	$\mathbf{1.90e+07 \pm 3.01e+06}$	$2.04e+07 \pm 2.15e+05$	+
f_7	$1.92e+10 \pm 4.99e+09$	$\mathbf{1.83e+10 \pm 3.99e+09}$	=
f_8	$\mathbf{2.36e+10 \pm 1.22e+10}$	$2.70e+11 \pm 1.71e+12$	+
f_9	$\mathbf{1.71e+08 \pm 7.24e+06}$	$4.55e+08 \pm 2.53e+08$	+
f_{10}	$7.23e+03 \pm 2.88e+02$	$\mathbf{6.95e+03 \pm 2.85e+02}$	-
f_{11}	$\mathbf{1.48e+02 \pm 4.56e+01}$	$2.08e+02 \pm 2.18e+00$	+
f_{12}	$\mathbf{8.39e+04 \pm 1.48e+05}$	$3.79e+05 \pm 2.10e+04$	+
f_{13}	$\mathbf{2.26e+05 \pm 9.13e+04}$	$9.57e+07 \pm 4.98e+08$	=
f_{14}	$\mathbf{1.33e+08 \pm 2.53e+08}$	$9.38e+08 \pm 4.51e+07$	+
f_{15}	$\mathbf{7.31e+03 \pm 3.08e+02}$	$1.37e+04 \pm 3.75e+02$	+
f_{16}	$\mathbf{2.23e+02 \pm 1.08e+02}$	$4.11e+02 \pm 2.25e+00$	+
f_{17}	$\mathbf{1.14e+05 \pm 2.39e+05}$	$8.07e+05 \pm 2.43e+04$	+
f_{18}	$\mathbf{3.57e+04 \pm 1.21e+04}$	$3.71e+08 \pm 2.08e+09$	+
f_{19}	$5.47e+05 \pm 3.43e+04$	$\mathbf{2.82e+05 \pm 2.68e+04}$	-
f_{20}	$\mathbf{1.49e+04 \pm 1.10e+03}$	$1.99e+08 \pm 7.66e+08$	+

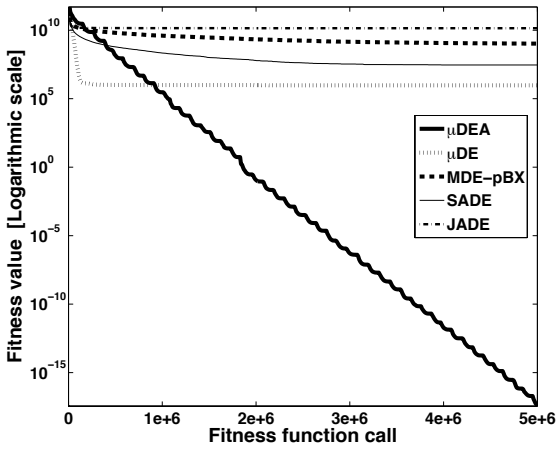

Fig. 3. Performance trend for f_1 of CEC2010 [40] in 1000 dimensions.

the other three modern DE versions but still is capable to display a respectable performance. More specifically, μDE out performs each of the other three algorithms in slightly less than half of the cases. In 100 dimensions, μDE is still slightly outperformed by SADE and MDE-pBX while is definitely competitive with JADE. The most interesting results of this study are reported in the large scale cases. In 1000 dimensions, μDE displays a surprisingly good performance with respect to the other modern DE based algorithms considered in this study. In high dimensions, μDEA displays the best performance (see Table VIII) by slightly outperforming SADE and clearly outperforming JADE and MDE-pBX. This result is especially interesting if we take into account that μDEA is a very simple and light (in terms of memory requirement and computational overhead) algorithm. It is important to remark that this study shows that small DE populations are more adequate than large ones to tackle large scale problems. Fig 3 shows the average performance in a case of successful application of the μDEA scheme.

TABLE V
AVERAGE FITNESS ± STANDARD DEVIATION AND WILCOXON RANK-SUM TEST (REFERENCE =μDEA) FOR μDEA AGAINST SADE, JADE AND MDE-PBX ON CEC2005[37] IN 30 DIMENSIONS.

	μDEA	MDE-pBX		SADE		JADE	
f_1	$-4.50e+02 \pm 2.18e-13$	$-4.50e+02 \pm 1.62e-13$	=	$\mathbf{-4.50e+02 \pm 2.90e-14}$	=	$-3.51e+02 \pm 1.99e+02$	+
f_2	$\mathbf{-4.50e+02 \pm 2.43e-12}$	$-4.50e+02 \pm 2.54e-03$	+	$-4.39e+02 \pm 2.31e+01$	+	$3.08e+02 \pm 7.32e+02$	+
f_3	$\mathbf{1.83e+05 \pm 1.05e+05}$	$2.81e+05 \pm 1.99e+05$	+	$8.97e+05 \pm 4.27e+05$	+	$3.71e+06 \pm 1.77e+06$	+
f_4	$6.74e+04 \pm 1.60e+04$	$\mathbf{-1.29e+02 \pm 9.67e+02}$	-	$-8.60e+01 \pm 5.87e+02$	-	$2.27e+03 \pm 1.68e+03$	-
f_5	$7.20e+03 \pm 2.22e+03$	$\mathbf{2.74e+03 \pm 6.34e+02}$	-	$2.86e+03 \pm 5.36e+02$	-	$3.91e+03 \pm 9.18e+02$	-
f_6	$8.52e+02 \pm 1.03e+03$	$4.33e+02 \pm 4.81e+01$	-	$\mathbf{4.16e+02 \pm 3.62e+01}$	-	$5.07e+06 \pm 1.53e+07$	+
f_7	$\mathbf{-1.80e+02 \pm 1.35e-02}$	$2.03e+06 \pm 1.88e+07$	+	$4.39e+02 \pm 6.16e+03$	+	$1.69e+13 \pm 1.78e+13$	+
f_8	$\mathbf{-1.20e+02 \pm 4.75e-03}$	$-1.19e+02 \pm 4.23e-01$	+	$-1.19e+02 \pm 4.24e-01$	+	$-1.19e+02 \pm 5.75e-02$	+
f_9	$-1.17e+02 \pm 1.16e+00$	$-1.17e+02 \pm 1.14e+00$	-	$\mathbf{-1.17e+02 \pm 4.60e-01}$	-	$-1.17e+02 \pm 1.22e+00$	=
f_{10}	$2.62e+02 \pm 2.05e+01$	$2.23e+02 \pm 2.44e+01$	-	$2.27e+02 \pm 2.25e+01$	-	$\mathbf{2.02e+02 \pm 2.20e+01}$	-
f_{11}	$1.18e+02 \pm 3.55e+00$	$\mathbf{1.11e+02 \pm 4.59e+00}$	-	$1.16e+02 \pm 3.54e+00$	-	$1.16e+02 \pm 4.48e+00$	-
f_{12}	$\mathbf{1.49e+03 \pm 2.90e+03}$	$3.77e+03 \pm 3.87e+03$	+	$4.90e+03 \pm 5.23e+03$	+	$1.72e+04 \pm 1.54e+04$	+
f_{13}	$-1.22e+02 \pm 1.45e+00$	$-1.19e+02 \pm 2.28e+00$	+	$-1.24e+02 \pm 9.45e-01$	-	$\mathbf{-1.26e+02 \pm 9.64e-01}$	-
f_{14}	$-2.86e+02 \pm 2.78e-01$	$\mathbf{-2.87e+02 \pm 4.50e-01}$	-	$-2.87e+02 \pm 4.22e-01$	-	$-2.87e+02 \pm 2.02e-01$	-
f_{15}	$1.45e+03 \pm 2.89e+00$	$1.46e+03 \pm 6.43e+00$	+	$\mathbf{1.44e+03 \pm 1.67e+00}$	-	$1.45e+03 \pm 3.45e+00$	=
f_{16}	$1.59e+03 \pm 1.56e+01$	$1.58e+03 \pm 1.11e+01$	-	$1.56e+03 \pm 8.59e+00$	-	$\mathbf{1.56e+03 \pm 6.50e+00}$	-
f_{17}	$1.74e+03 \pm 1.81e+01$	$1.62e+03 \pm 9.04e+00$	-	$1.62e+03 \pm 9.84e+00$	-	$\mathbf{1.59e+03 \pm 7.53e+00}$	-
f_{18}	$\mathbf{9.10e+02 \pm 5.26e-12}$	$9.10e+02 \pm 8.31e-11$	+	$9.10e+02 \pm 4.66e-09$	+	$9.10e+02 \pm 2.62e-01$	+
f_{19}	$\mathbf{9.10e+02 \pm 5.82e-12}$	$9.10e+02 \pm 2.42e-10$	+	$9.10e+02 \pm 5.64e-09$	+	$9.10e+02 \pm 1.31e-01$	+
f_{20}	$\mathbf{9.10e+02 \pm 5.61e-12}$	$9.10e+02 \pm 3.41e-11$	+	$9.10e+02 \pm 1.36e-10$	+	$9.10e+02 \pm 1.40e-01$	+
f_{21}	$1.72e+03 \pm 1.09e+01$	$1.70e+03 \pm 5.51e+00$	-	$1.70e+03 \pm 7.05e+00$	-	$\mathbf{1.69e+03 \pm 4.32e+00}$	-
f_{22}	$2.60e+03 \pm 5.80e+01$	$2.41e+03 \pm 4.97e+01$	-	$2.34e+03 \pm 3.99e+01$	-	$\mathbf{2.29e+03 \pm 3.44e+01}$	-
f_{23}	$1.73e+03 \pm 9.39e+00$	$1.70e+03 \pm 5.28e+00$	-	$1.71e+03 \pm 6.04e+00$	-	$\mathbf{1.70e+03 \pm 4.48e+00}$	-
f_{24}	$1.71e+03 \pm 1.44e+01$	$1.67e+03 \pm 1.55e+01$	-	$1.67e+03 \pm 1.21e+01$	-	$\mathbf{1.66e+03 \pm 1.40e+01}$	-
f_{25}	$1.88e+03 \pm 3.40e+02$	$1.83e+03 \pm 1.55e+02$	=	$\mathbf{1.78e+03 \pm 2.05e+02}$	-	$1.86e+03 \pm 4.65e+01$	=

TABLE VI
AVERAGE FITNESS ± STANDARD DEVIATION AND WILCOXON RANK-SUM TEST (REFERENCE =μDEA) FOR μDEA AGAINST SADE, JADE AND MDE-PBX ON BBOB2010[38] IN 100 DIMENSIONS.

	μDEA	MDE-pBX		SADE		JADE	
f_1	$7.95e+01 \pm 3.23e-03$	$7.95e+01 \pm 7.60e-05$	=	$\mathbf{7.95e+01 \pm 9.67e-13}$	=	$8.77e+01 \pm 7.64e+00$	+
f_2	$-1.50e+02 \pm 1.73e+02$	$-2.10e+02 \pm 6.06e-03$	-	$\mathbf{-2.10e+02 \pm 9.83e-13}$	-	$5.73e+04 \pm 8.69e+04$	+
f_3	$-2.20e+02 \pm 1.05e+02$	$3.29e+01 \pm 7.94e+01$	+	$-2.94e+02 \pm 4.40e+01$	-	$\mathbf{-3.11e+02 \pm 4.95e+01}$	-
f_4	$-1.16e+02 \pm 1.12e+02$	$4.03e+02 \pm 1.31e+02$	+	$\mathbf{-1.61e+02 \pm 1.19e+02}$	-	$-1.43e+02 \pm 9.83e+01$	-
f_5	$-8.26e+00 \pm 2.06e+00$	$-3.09e-02 \pm 1.27e+01$	+	$\mathbf{-9.16e+00 \pm 4.59e-01}$	-	$1.24e+02 \pm 5.08e+01$	+
f_6	$8.73e+01 \pm 1.34e+02$	$\mathbf{8.03e+01 \pm 3.32e+01}$	-	$1.11e+02 \pm 3.87e+01$	+	$3.92e+02 \pm 1.18e+02$	+
f_7	$4.42e+02 \pm 1.71e+02$	$3.70e+02 \pm 7.43e+01$	-	$3.39e+02 \pm 6.69e+01$	-	$\mathbf{3.08e+02 \pm 6.50e+01}$	-
f_8	$\mathbf{2.74e+02 \pm 9.87e+01}$	$3.40e+02 \pm 6.77e+01$	+	$2.82e+02 \pm 6.22e+01$	+	$7.24e+03 \pm 6.15e+03$	+
f_9	$\mathbf{1.92e+02 \pm 5.80e+01}$	$2.52e+02 \pm 3.75e+01$	+	$2.28e+02 \pm 2.45e+01$	+	$1.78e+03 \pm 1.33e+03$	+
f_{10}	$5.65e+04 \pm 9.88e+04$	$\mathbf{1.64e+04 \pm 7.99e+03}$	-	$5.22e+04 \pm 2.07e+04$	-	$1.94e+05 \pm 8.87e+04$	+
f_{11}	$8.30e+02 \pm 1.30e+02$	$\mathbf{9.16e+01 \pm 7.45e+00}$	-	$1.83e+02 \pm 2.72e+01$	-	$2.11e+02 \pm 2.76e+01$	-
f_{12}	$6.52e+02 \pm 3.39e+03$	$-5.99e+02 \pm 7.07e+01$	-	$\mathbf{-6.14e+02 \pm 7.07e+00}$	=	$2.22e+07 \pm 2.13e+07$	+
f_{13}	$4.02e+01 \pm 1.04e+01$	$3.47e+01 \pm 6.70e+00$	-	$\mathbf{3.20e+01 \pm 2.81e+00}$	-	$7.69e+02 \pm 2.46e+02$	+
f_{14}	$-5.23e+01 \pm 1.73e-02$	$-5.23e+01 \pm 2.55e-03$	=	$\mathbf{-5.23e+01 \pm 1.86e-03}$	=	$-4.65e+01 \pm 3.89e+00$	+
f_{15}	$2.36e+03 \pm 3.43e+02$	$1.66e+03 \pm 1.10e+02$	-	$\mathbf{1.35e+03 \pm 6.05e+01}$	-	$1.59e+03 \pm 8.67e+01$	-
f_{16}	$9.04e+01 \pm 6.19e+00$	$\mathbf{8.85e+01 \pm 4.46e+00}$	=	$9.72e+01 \pm 4.30e+00$	+	$1.01e+02 \pm 3.46e+00$	+
f_{17}	$-7.56e+00 \pm 2.73e+00$	$-1.35e+01 \pm 4.83e-01$	-	$-1.38e+01 \pm 5.21e-01$	-	$\mathbf{-1.45e+01 \pm 5.96e-01}$	-
f_{18}	$1.75e+01 \pm 8.38e+00$	$-4.84e+00 \pm 1.68e+00$	-	$-5.07e+00 \pm 1.98e+00$	-	$\mathbf{-8.79e+00 \pm 2.11e+00}$	-
f_{19}	$-9.29e+01 \pm 2.50e+00$	$\mathbf{-1.00e+02 \pm 7.13e-01}$	-	$-9.98e+01 \pm 6.72e-01$	-	$-9.50e+01 \pm 2.26e-01$	-
f_{20}	$\mathbf{-5.45e+02 \pm 2.07e-01}$	$-5.44e+02 \pm 1.14e-01$	+	$-5.45e+02 \pm 1.61e-01$	+	$-5.12e+02 \pm 9.92e+01$	+
f_{21}	$4.98e+01 \pm 6.24e+00$	$\mathbf{4.49e+01 \pm 5.88e+00}$	-	$4.63e+01 \pm 5.76e+00$	-	$4.93e+01 \pm 6.25e+00$	=
f_{22}	$-9.87e+02 \pm 9.58e+00$	$-9.92e+02 \pm 9.11e+00$	-	$-9.93e+02 \pm 7.97e+00$	-	$\mathbf{-9.94e+02 \pm 6.66e+00}$	-
f_{23}	$\mathbf{8.60e+00 \pm 8.04e-01}$	$9.34e+00 \pm 7.99e-01$	+	$9.17e+00 \pm 6.78e-01$	+	$1.07e+01 \pm 3.78e-01$	+
f_{24}	$1.71e+03 \pm 3.62e+02$	$4.75e+02 \pm 4.72e+01$	-	$\mathbf{3.73e+02 \pm 3.17e+01}$	-	$1.03e+03 \pm 4.44e+01$	-

In addition to the results presented above, the ranking among all the algorithms considered in this article has been performed by means of the Holm-Bonferroni procedure, see [42] and [43], for the 5 algorithms under study and the 76 problems under consideration. The Holm-Bonferroni procedure consists of the following. Considering the results in the tables above, the 5 algorithms under analysis have been ranked on the basis of their average performance calculated over the 76 test problems. More specifically, a score R_i for $i = 1, \ldots, N_A$ (where N_A is the number of algorithms under analysis, $N_A = 5$ in our case) has been assigned. The score has been assigned in the following way: for each problem, a score of 5 is assigned to the algorithm displaying the best performance, 4 is assigned to the second best, 3 to the third and so on. The algorithm displaying the worst performance scores 1. For each algorithm, the scores obtained on each problem are summed up averaged over the amount of test problems (76 in our case). On the basis of these scores the algorithms are sorted (ranked). With the calculated R_i values, RIS has been taken as a reference algorithm. Indicating with R_0 the rank of RIS, and with R_j for $j = 1, \ldots, N_A - 1$ the rank of one of the remaining eleven algorithms, the values z_j have been calculated as

$$z_j = \frac{R_j - R_0}{\sqrt{\frac{N_A(N_A+1)}{6N_{TP}}}} \tag{7}$$

where N_{TP} is the number of test problems in consideration ($N_{TP} = 76$ in our case). By means of the z_j values, the corresponding cumulative normal distribution values p_j have been calculated. These p_j values have then been compared with the corresponding δ/j where δ is the level of confidence, set to 0.05 in our case. Table IX displays the ranks, z_j values, p_j values, and corresponding δ/j obtained in this way. The rank of μDEA is shown in parenthesis. Moreover, it is indicated whether the null-hypothesis (that the two algorithms have indistinguishable performances) is "Rejected", i.e. μDEA statistically outperforms the algorithm under consideration, or

TABLE VII
AVERAGE FITNESS ± STANDARD DEVIATION AND WILCOXON RANK-SUM TEST (REFERENCE =μDEA) FOR μDEA AGAINST SADE, JADE AND MDE-PBX ON CEC2008[39] IN 1000 DIMENSIONS.

	muDEA	MDEpBX		SADE		JADE	
f_1	$\mathbf{-4.50e+02 \pm 1.42e-09}$	$1.20e+05 \pm 4.41e+04$	+	$5.06e+03 \pm 6.00e+03$	+	$1.02e+06 \pm 3.49e+05$	+
f_2	$\mathbf{-4.50e+02 \pm 2.53e-02}$	$-3.33e+02 \pm 4.09e+00$	+	$-3.19e+02 \pm 5.11e+00$	+	$-3.20e+02 \pm 7.98e+00$	+
f_3	$\mathbf{1.52e+03 \pm 8.92e+01}$	$3.13e+10 \pm 1.65e+10$	+	$9.97e+08 \pm 1.66e+09$	+	$4.40e+11 \pm 2.18e+11$	+
f_4	$5.77e+03 \pm 4.56e+02$	$7.60e+03 \pm 2.55e+02$	+	$5.88e+03 \pm 3.95e+02$	=	$\mathbf{4.43e+03 \pm 8.64e+02}$	-
f_5	$\mathbf{-1.80e+02 \pm 1.89e-03}$	$1.08e+03 \pm 4.60e+02$	+	$-1.20e+02 \pm 6.46e+01$	+	$8.70e+03 \pm 2.95e+03$	+
f_6	$\mathbf{-1.40e+02 \pm 5.01e-07}$	$-1.21e+02 \pm 5.10e-02$	+	$-1.21e+02 \pm 1.77e-01$	+	$-1.22e+02 \pm 5.79e-01$	+
f_7	$\mathbf{-1.35e+04 \pm 6.61e+01}$	$-1.11e+04 \pm 1.63e+02$	+	$-1.11e+04 \pm 1.28e+02$	+	$-1.19e+04 \pm 4.24e+02$	+

TABLE VIII
AVERAGE FITNESS ± STANDARD DEVIATION AND WILCOXON RANK-SUM TEST (REFERENCE =μDEA) FOR μDEA AGAINST SADE, JADE AND MDE-PBX ON CEC2010[40] IN 1000 DIMENSIONS.

	μDEA	MDE-pBX		SADE		JADE	
f_1	$\mathbf{4.44e-18 \pm 7.20e-19}$	$1.05e+09 \pm 6.58e+08$	+	$2.89e+07 \pm 1.02e+08$	+	$1.40e+10 \pm 6.91e+09$	+
f_2	$5.71e+03 \pm 3.66e+02$	$7.02e+03 \pm 2.38e+02$	+	$5.55e+03 \pm 2.99e+02$	-	$\mathbf{4.56e+03 \pm 1.04e+03}$	-
f_3	$\mathbf{2.47e-02 \pm 9.79e-02}$	$1.93e+01 \pm 4.76e-02$	+	$1.89e+01 \pm 2.83e-01$	+	$1.76e+01 \pm 6.75e-01$	+
f_4	$2.07e+13 \pm 3.99e+12$	$3.21e+12 \pm 9.76e+11$	-	$\mathbf{1.95e+12 \pm 8.82e+11}$	-	$2.62e+12 \pm 1.03e+12$	-
f_5	$4.49e+08 \pm 1.16e+08$	$1.54e+08 \pm 2.77e+07$	-	$1.03e+08 \pm 1.83e+07$	-	$\mathbf{8.58e+07 \pm 1.77e+07}$	-
f_6	$1.90e+07 \pm 3.01e+06$	$3.65e+06 \pm 1.75e+06$	-	$\mathbf{9.16e+05 \pm 1.21e+06}$	-	$3.48e+06 \pm 1.40e+06$	-
f_7	$1.92e+10 \pm 4.99e+09$	$\mathbf{6.79e+06 \pm 1.01e+07}$	-	$1.01e+08 \pm 2.36e+08$	-	$3.37e+09 \pm 3.66e+09$	-
f_8	$2.36e+10 \pm 1.22e+10$	$2.03e+08 \pm 1.63e+08$	-	$\mathbf{7.08e+07 \pm 3.71e+07}$	-	$6.31e+13 \pm 1.80e+14$	+
f_9	$\mathbf{1.71e+08 \pm 7.24e+06}$	$1.68e+09 \pm 1.00e+09$	+	$2.11e+08 \pm 2.93e+08$	+	$1.67e+10 \pm 5.87e+09$	+
f_{10}	$7.23e+03 \pm 2.88e+02$	$7.33e+03 \pm 2.55e+02$	+	$\mathbf{6.22e+03 \pm 3.15e+02}$	-	$7.50e+03 \pm 1.07e+03$	+
f_{11}	$\mathbf{1.48e+02 \pm 4.56e+01}$	$2.06e+02 \pm 2.40e+00$	+	$2.05e+02 \pm 4.34e+00$	+	$1.94e+02 \pm 7.49e+00$	+
f_{12}	$\mathbf{8.39e+04 \pm 1.48e+05}$	$2.92e+05 \pm 6.60e+04$	+	$3.15e+05 \pm 1.36e+05$	+	$2.32e+06 \pm 4.55e+05$	+
f_{13}	$\mathbf{2.26e+05 \pm 9.13e+04}$	$2.88e+09 \pm 3.17e+09$	+	$5.67e+07 \pm 2.48e+08$	=	$8.02e+10 \pm 4.76e+10$	+
f_{14}	$\mathbf{1.33e+08 \pm 2.53e+08}$	$1.04e+09 \pm 1.97e+08$	+	$3.77e+08 \pm 1.13e+08$	+	$1.31e+10 \pm 4.64e+09$	+
f_{15}	$7.31e+03 \pm 3.08e+02$	$7.44e+03 \pm 2.80e+02$	+	$\mathbf{6.49e+03 \pm 2.38e+02}$	-	$8.51e+03 \pm 1.03e+03$	+
f_{16}	$\mathbf{2.23e+02 \pm 1.08e+02}$	$3.84e+02 \pm 1.22e+00$	+	$3.82e+02 \pm 2.00e+00$	+	$3.83e+02 \pm 1.19e+01$	+
f_{17}	$\mathbf{1.14e+05 \pm 2.39e+05}$	$4.35e+05 \pm 8.33e+04$	+	$6.37e+05 \pm 2.00e+05$	+	$2.63e+06 \pm 7.56e+05$	+
f_{18}	$\mathbf{3.57e+04 \pm 1.21e+04}$	$3.73e+10 \pm 1.95e+10$	+	$7.60e+08 \pm 1.14e+09$	+	$4.42e+11 \pm 1.91e+11$	+
f_{19}	$\mathbf{5.47e+05 \pm 3.43e+04}$	$9.22e+05 \pm 1.06e+05$	+	$2.11e+06 \pm 1.61e+05$	+	$3.59e+06 \pm 7.17e+05$	+
f_{20}	$\mathbf{1.49e+04 \pm 1.10e+03}$	$4.18e+10 \pm 2.02e+10$	+	$2.26e+09 \pm 3.42e+09$	+	$5.48e+11 \pm 2.10e+11$	+

TABLE IX
HOLM TEST ON THE FITNESS, REFERENCE ALGRITHM = μDEA (RANK = 3.24E+00)

j	Optimizer	$Rank$	z_j	p_j	δ/j	Hypothesis
1	SADE	3.68e+00	2.14e+00	9.84e-01	5.00e-02	Accepted
2	MDE-pBX	3.08e+00	-7.54e-01	2.25e-01	2.50e-02	Accepted
3	μDE	2.53e+00	-3.39e+00	3.46e-04	1.67e-02	Rejected
4	JADE	2.46e+00	-3.71e+00	1.05e-04	1.25e-02	Rejected

"Accepted" if the distribution of values can be considered the same (there is no out-performance).

As shown in Table IX, the proposed μDEA is ranked second after SADE over all the 76 problems included in this study, thus confirming that μDEA is a valuable algorithm that makes use of a micro-population.

IV. CONCLUSION

This paper proposes a micro-Differential Evolution scheme that includes, in a memetic fashion, a shallow local search that performs, a limited amount of times exploitation in the directions of each variable of the candidate solution displaying the best performance. The extra moves according to the axes complement the search carried out by the DE logic and support the external workshop to detect solutions with a high performance. More specifically, DE schemes, even when characterized by a small population, tend to keep the candidate solutions far from each other. This may result into an excessive exploration, especially in high dimensions, thus resulting into an undesired stagnation condition. The extra moves increase the exploitation of the algorithm and allow an overall better performance. The comparison with modern DE based algorithms show that the proposed algorithm, notwithstanding its simplicity, is nearly as good as them for low dimensional problem, thus displaying a respectable performance. The comparison in large scale domains show that the proposed algorithm outperforms all the other algorithms contained in this study. From this finding, we can conclude that small populations in DE schemes can lead to performance higher than that of DE schemes that use large populations.

The proposed micro-Differential Evolution implementation appears a good and robust alternative that can be promisingly applied in those application characterized by a limted hardware, such as embedded systems, and in those problems that impose a modest computational overhead, such as real-time optimization problems. Future work will consider randomized operators and mechanisms that impose a narrowing of the search in the late stage of the optimization.

ACKNOWLEDGMENT

This research is supported by the Academy of Finland, Akatemiatutkija 130600, "Algorithmic design issues in Memetic Computing". 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 [44]. We thank Dr. Lorenzo Picinali, Dr. Nathan Jeffery and David Tunnicliffe for the technical support.

REFERENCES

[1] A. E. Eiben and S. K. Smit, "Parameter tuning for configuring and analyzing evolutionary algorithms," *Swarm and Evolutionary Computation*, vol. 1, no. 1, pp. 19–31, 2011.

[2] J. Brest and M. S. Maučec, "Population size reduction for the differential evolution algorithm," *Applied Intelligence*, vol. 29, no. 3, pp. 228–247, 2008.

[3] 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.

[4] 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.

[5] N. S. Teng, J. Teo, and M. H. A. Hijazi, "Self-adaptive population sizing for a tune-free differential evolution," *Soft Computing – A Fusion of Foundations, Methodologies and Applications*, vol. 13, no. 7, pp. 709–724, 2009.

[6] A. Prügel-Bennett, "Benefits of a population: Five mechanisms that advantage population-based algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 14, no. 4, pp. 500–517, 2010.

[7] N. J. Radcliffe, "Forma analysis and random respectful recombination," in *Proceedings of the 4th Int. Conf. Genet. Algorithms*. Morgan Kaufmann, 1991, pp. 222–229.

[8] G. Iacca, F. Neri, E. Mininno, Y. S. Ong, and M. H. Lim, "Ockham's Razor in Memetic Computing: Three Stage Optimal Memetic Exploration," *Information Sciences*, vol. 188, pp. 17–43, 2012.

[9] T. Chen, K. Tang, G. Chen, and X. Yao, "A large population size can be unhelpful in evolutionary algorithms," *Theoretical Computer Science*, vol. 436, pp. 54–70, 2012.

[10] K. E. Parsopoulos, "Cooperative micro-differential evolution for high-dimensional problems," in *Proceedings of the conference on Genetic and evolutionary computation*, 2009, pp. 531–538.

[11] ——, "Parallel cooperative micro-particle swarm optimization: A master-slave model," *Applied Soft Computing*, vol. 12, no. 11, pp. 3552–3579, Nov. 2012.

[12] S. Dasgupta, S. Das, A. Biswas, and A. Abraham, "On stability and convergence of the population-dynamics in differential evolution," *AI Communications - The European Journal on Artificial Intelligence*, vol. 22, no. 1, pp. 1–20, 2009.

[13] A. Rajasekhar, S. Das, and S. Das, "Abc: a micro artificial bee colony algorithm for large scale global optimization," in *GECCO (Companion)*, 2012, pp. 1399–1400.

[14] K. Krishnakumar, "Micro-genetic algorithms for stationary and non-stationary function optimization."

[15] C. A. Coello Coello and G. Toscano Pulido, "A micro-genetic algorithm for multiobjective optimization," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO2001*. Springer-Verlag, 2001, pp. 126–140.

[16] A. Rajasekhar, S. Das, and P. N. Suganthan, "Design of fractional order controller for a servohydraulic positioning system with micro artificial bee colony algorithm," in *IEEE Congress on Evolutionary Computation*, 2012, pp. 1–8.

[17] V. Tam, K.-Y. Cheng, and K.-S. Lui, "Using micro-genetic algorithms to improve localization in wireless sensor networks," *Journal of Communications*, vol. 1, no. 4, pp. 137–141, 2006.

[18] F. Viveros-Jimènez, E. Mezura-Montes, and A. Gelbukh, "Empirical analysis of a micro-evolutionary algorithm for numerical optimization," *International Journal of Physical Sciences*, vol. 7, no. 8, pp. 1235–1258.

[19] S. Rahnamayan and H. R. Tizhoosh, "Image thresholding using micro opposition-based differential evolution (micro-ode)," in *IEEE Congress on Evolutionary Computation*, 2008, pp. 1409–1416.

[20] M. A. Sotelo-Figueroa, H. J. P. Soberanes, J. M. Carpio, H. J. F. Huacuja, L. C. Reyes, and J. A. S. Alcaraz, "Evolving bin packing heuristic using micro-differential evolution with indirect representation," in *Recent Advances on Hybrid Intelligent Systems*, ser. Studies in Computational Intelligence, 2013, vol. 451, pp. 349–359.

[21] S. Das and P. Suganthan, "Differential evolution: A survey of the state-of-the-art," *Evolutionary Computation, IEEE Transactions on*, vol. 15, no. 1, pp. 4–31, feb. 2011.

[22] K. V. Price, R. Storn, and J. Lampinen, *Differential Evolution: A Practical Approach to Global Optimization*. Springer, 2005.

[23] 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.

[24] F. Neri, G. Iacca, and E. Mininno, "Disturbed Exploitation compact Differential Evolution for Limited Memory Optimization Problems," *Information Sciences*, vol. 181, no. 12, pp. 2469–2487, 2011.

[25] L.-Y. Tseng and C. Chen, "Multiple trajectory search for Large Scale Global Optimization," in *Proceedings of the IEEE Congress on Evolutionary Computation*, 2008, pp. 3052–3059.

[26] 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.

[27] J. Zhang and A. C. Sanderson, "JADE: Adaptive Differential Evolution with Optional External Archive," vol. 13, no. 5, 2009, pp. 945–958.

[28] 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.

[29] R. Mallipeddi, P. N. Suganthan, Q. K. Pan, and M. F. Tasgetiren, "Differential evolution algorithm with ensemble of parameters and mutation strategies," *Applied Soft Computing*, vol. 11, no. 2, pp. 1679–1696, 2011, the Impact of Soft Computing for the Progress of Artificial Intelligence.

[30] S. Das, A. Abraham, U. K. Chakraborty, and A. Konar, "Differential Evolution with a Neighborhood-based Mutation Operator," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 3, pp. 526–553, 2009.

[31] 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.

[32] 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.

[33] 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.

[34] F. Neri and V. Tirronen, "Scale Factor Local Search in Differential Evolution," *Memetic Computing Journal*, vol. 1, no. 2, pp. 153–171, 2009.

[35] M. Weber, V. Tirronen, and F. Neri, "Scale Factor Inheritance Mechanism in Distributed Differential Evolution," *Soft Computing - A Fusion of Foundations, Methodologies and Applications*, vol. 14, no. 11, pp. 1187–1207, 2010.

[36] J. Lampinen and I. Zelinka, "On Stagnation of the Differential Evolution Algorithm," in *Proceedings of 6th International Mendel Conference on Soft Computing*, P. Oŝmera, Ed., 2000, pp. 76–83.

[37] P. N. Suganthan, N. Hansen, J. J. Liang, K. Deb, Y.-P. Chen, A. Auger, and S. Tiwari, "Problem Definitions and Evaluation Criteria for the CEC 2005 Special Session on Real-Parameter Optimization," Nanyang Technological University and KanGAL, Singapore and IIT Kanpur, India, Tech. Rep. 2005005, 2005.

[38] N. Hansen, A. Auger, S. Finck, R. Ros *et al.*, "Real-Parameter Black-Box Optimization Benchmarking 2010: Noiseless Functions Definitions," INRIA, Tech. Rep. RR-6829, 2010.

[39] K. Tang, X. Yao, P. N. Suganthan, C. MacNish, Y. P. Chen, C. M. Chen, and Z. Yang, "Benchmark Functions for the CEC 2008 Special Session and Competition on Large Scale Global Optimization," Nature Inspired Computation and Applications Laboratory, USTC, China, Tech. Rep., 2007.

[40] K. Tang, X. Li, P. N. Suganthan, Z. Yang, and T. Weise, "Benchmark Functions for the CEC'2010 Special Session and Competition on Large-Scale Global Optimization," University of Science and Technology of China (USTC), School of Computer Science and Technology, Nature Inspired Computation and Applications Laboratory (NICAL): Hefei, Anhui, China, Tech. Rep., 2010.

[41] F. Wilcoxon, "Individual comparisons by ranking methods," *Biometrics Bulletin*, vol. 1, no. 6, pp. 80–83, 1945.

[42] S. Holm, "A simple sequentially rejective multiple test procedure," *Scandinavian Journal of Statistics*, vol. 6, no. 2, pp. 65–70, 1979.

[43] 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.

[44] Cyber Dyne Srl Home Page, "Kimeme," 2012, http://cyberdynesoft.it/.