

# Global Supervision for Compact Differential Evolution

Giovanni Iacca\*, Rammohan Mallipeddi†, Ernesto Mininno\*, Ferrante Neri\*, and Pannuthurai Nagaratnam Suganthan†

\*Department of Mathematical Information Technology, P.O. Box 35 (Agora), University of Jyväskylä, Finland,  
E-mail: giovanni.iacca@jyu.fi, ernesto.mininno@jyu.fi, and ferrante.neri@jyu.fi.

†School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore 639798,  
E-mail: mall0003@ntu.edu.sg and epnsugan@ntu.edu.sg.

**Abstract**—This paper proposes a novel Differential Evolution based algorithmic structure for solving continuous global optimization problems. The proposed structure makes use of the recently introduced concept of compact Differential Evolution as a search unit. Several compact units evolve simultaneously and interact in order to solve the optimization problem. In other words, the compact units are supposed to explore the decision space from diverse perspectives. The search work performed by the compact units is coordinated by a global supervision unit which processes by means of a global search the achievements obtained by the various compact units. More specifically, each compact unit performs a step of compact Differential Evolution and then feeds the achieved results to a global optimizer which recombines during one generation the candidate solutions and returns the improved genotypes to the corresponding compact units. In this implementation we selected as a global supervision unit a Differential Evolution algorithm with self-adaptive control parameters previously proposed in literature. The concept of global supervision, here introduced, appears to be very promising as it allows the improvement and development of the results locally obtained by each compact unit, thus preventing premature convergence of each unit and promoting a successful continuation of the search. Numerical results show that the resulting algorithm considered in this study displays a promising performance for a set of challenging test problems and is competitive with the state-of-the-art Differential Evolution based algorithms.

## I. INTRODUCTION

As observed in [1], albeit versatile and efficient, Differential Evolution (DE) contains some limitations and thus presents a wide margin of improvement. According to the analysis reported in [1], classical DE is characterized by a limited amount of search moves. This fact makes classical DE often stagnate and be incapable to reach a promising area of the decision space. Thus, in order to enhance the DE performance, alternative search moves should support the original scheme and promote a successful continuation of the optimization process. For example it is worthwhile mentioning the employment of extra or multiple mutation schemes, as for example the trigonometric mutation [2], the self-adaptation with multiple mutation schemes [3], and the generation of candidate solutions by means of an alternative rule (opposition-based), see [4]. The offspring generation, by means of the composition of two contributions, the first one resulting from the entire population, the second from a subset of it, is proposed in [5]. The progressive population

size reduction is employed in [6], [7] for handling high dimensional problems. In other works, local search algorithms support the DE search, see [8], [9], [10], [11] and [12]. The employment of structured population also appears to be an efficient way to improve DE, see [13], [14], and [15]. In modern DE-based algorithms randomization seems to play a very important role on the algorithmic performance, see [16]. It is a well-known fact that a scale factor randomization tends to improve upon the algorithmic performance as in the case of jitter and dither, see [1] and references therein. In a similar way, in [17] a controlled randomization of scale factor and crossover rate is proposed. In [18], the concept of parameter randomization is encoded within a sophisticated adaptive rule which is based on truncated Gaussian distributions. In [3] a randomized adaptation of the parameters is combined with multiple mutation schemes.

In this spirit and in order to address optimization under the lack of a full power computational device compact Differential Evolution (cDE) has been recently proposed in [19]. cDE, like other compact evolutionary algorithms (e.g. [20]), does not process a population of solutions but its statistic description. In addition, cDE employs the mutation and crossover typical of DE thus reproducing its search logic. An important property of cDE is that it requires the memory equivalent to the storage of four solutions regardless the (virtual) population size and the dimensionality of the problem. This result is attained by sampling, from an evolving distribution (representing the population), the candidate solutions undergoing recombination. Due to this mechanism, cDE can be seen as a DE which introduces a randomization within the solution generation and therefore introduces extra search moves which assist the DE structure and attempt to improve upon its performance. This fact has been experimentally proved in [19] and [21]. The main drawback of cDE (as well as for other compact algorithms), is that it is subject to premature convergence, especially when the dimensionality grows. Since the exploration of cDE is related to the shrinkage of the probability function modelling the population, the search tends to quickly narrow around the most promising candidate solutions. In this light, the algorithmic behavior of a cDE is assimilable to a local search rather than the global search characterizing classical DE.

Although cDE performs significantly better in higher di-

mensions with respect to other compact algorithms, see [19], since it requires much less resources in terms of memory and hardware with respect to modern population-based algorithms, see e.g. [3], it cannot be competitive with sophisticated population-based meta-heuristics for relatively high dimensional problems. This fact leads to the obvious conclusion that when memory requirements forbid the employment of population-based algorithms, compact algorithms can be a cheap and efficient solution. On the other hand, when there is no hardware limitation, population-based algorithms are often preferable.

This paper proposes, in the fashion of the concept of compact algorithms and especially cDE a novel algorithmic structure. This structure, namely Supervised compact Differential Evolution (ScDE), can be seen as a compromise between the necessity of having a respectable performance and the memory limitations. In details, multiple cDE units perform simultaneously the search of the decision space from various perspectives. These units are coordinated by means of global search algorithm which takes the role of supervising the multiple searches performed by the cDE units. This global search algorithm processes and updates the elite of each cDE promoting the recombination of the results attained separately by the various units and thus supporting the compact search.

The remainder of this paper is organized in the following way. A brief description of cDE, working principles of the supervised mechanism when simultaneous multiple cDE units are employed, and the proposed ScDE is given in Section II. Numerical results and conclusion are displayed in Section III and IV, respectively.

## II. SUPERVISED COMPACT DIFFERENTIAL EVOLUTION

In order to clarify the notation used throughout this article we refer to the minimization problem of an objective function  $f(x)$ , where  $x$  is a vector of  $n$  design variables in a decision space  $D$ . Without loss of generality, let us assume that design variables are normalized so that each search interval is  $[-1, 1]$ .

At the beginning of the optimization process,  $N_c$  ( $n \times 2$ )-matrices are generated. Parameter  $N_c$  stands for number of compact units. Each matrix  $PV_m$  ( $m = 1, 2, \dots, N_c$ ) is called Probability Vector, see [20]. More specifically,  $PV_m$  is a  $n \times 2$  matrix:

$$PV_m^t = [\mu^t, \sigma^t] \quad (1)$$

where  $\mu$  and  $\sigma$  are, respectively, vectors containing, for each design variable, mean and standard deviation values of a Gaussian Probability Distribution Function (PDF) truncated within the interval  $[-1, 1]$ . The height of the PDF has been normalized in order to keep its area equal to 1. The apex  $t$  indicates the generation (number of performed comparison).

In the initialization phase, for each design variable  $i$ ,  $\mu^1[i] = 0$  and  $\sigma^1[i] = \lambda$  where  $\lambda$  is a positive constant  $\gg 1$  (e.g.  $\lambda = 10$ ). This initialization of  $\sigma$  values is done in order to simulate a uniform distribution. Then, one individual is sampled from each  $PV_m$ . These individuals are indicated as elite  $x_e^m$  and are the elite individual related to the  $m^{th}$

probability vector  $PV_m$ . In other words,  $N_c$  compact units for cDE, see [19], are generated. For each unit  $m$ , a scale factor  $F^m$  is set. For simplicity of notation, let us refer to the generic  $m^{th}$  compact unit as  $PV$  and  $x_e$ , since the same operations are repeated within each unit. The search mechanism, within each unit and at each generation, is organized in the following way. Three individuals  $x_r$ ,  $x_s$  and  $x_t$  are pseudo-randomly sampled from the  $PV$ . More specifically, the sampling mechanism of a design variable  $x_r[i]$  associated to a generic candidate solution  $x_r$  from  $PV$  consists of the following steps. As mentioned above, for each design variable indexed by  $i$ , a truncated Gaussian PDF characterized by a mean value  $\mu[i]$  and a standard deviation  $\sigma[i]$  is associated. The formula of the PDF is:

$$PDF(\mu[i], \sigma[i]) = \frac{e^{-\frac{(x-\mu[i])^2}{2\sigma[i]^2}}}{\sigma[i] \left( \operatorname{erf}\left(\frac{\mu[i]+1}{\sqrt{2}\sigma[i]}\right) - \operatorname{erf}\left(\frac{\mu[i]-1}{\sqrt{2}\sigma[i]}\right) \right)} \sqrt{\frac{2}{\pi}} \quad (2)$$

where  $\operatorname{erf}$  is the error function, see [22].

From the PDF, the corresponding Cumulative Distribution Function (CDF) is constructed by means of Chebyshev polynomials according to the procedure described in [23]. It must be observed that the codomain of CDF is  $[0, 1]$ . In order to sample the design variable  $x_r[i]$  from  $PV$ , a random number  $\operatorname{rand}(0, 1)$  is sampled from a uniform distribution. The inverse function of CDF, in correspondence of  $\operatorname{rand}(0, 1)$ , is then calculated. This latter value is  $x_r[i]$ .

A provisional offspring  $x'_{off}$  is then generated by mutation, according to a DE logic, as:

$$x'_{off} = x_t + F(x_r - x_s) \quad (3)$$

where  $F \in [0, 2]$  is a scale factor which controls the length of the exploration vector  $(x_r - x_s)$  and thus determines how far from point  $x_t$  the offspring should be generated. This scale factor is the value  $F^m$  (corresponding to the  $m^{th}$  unit) mentioned above. The mutation scheme shown in formula (3) is also known as DE/rand/1. It is important to remark that other variants of the mutation rule have been proposed in literature for standard DE, see [1].

When the provisional offspring has been generated by mutation, the exponential crossover is applied. A design variable of the provisional offspring  $x'_{off}(j)$  is randomly selected and copied into the  $j^{th}$  design variable of the elite solution  $x_e$  (its copy). This guarantees that parent and offspring have different genotypes. Subsequently, a set of random numbers between 0 and 1 are generated. As long as  $\operatorname{rand}(0, 1) \leq Cr$ , 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  $\operatorname{rand}(0, 1) > Cr$  the copy process is interrupted. Thus, all the remaining design variables of the offspring are copied from the parent. For the sake of clarity the pseudo-code of the exponential crossover is shown in Fig. 1

In other words, this crossover operator generates offspring composed of the elite  $x_e$  and contains, within it, a section of

```

 $x_{off} = x_e$ 
generate  $j = 1 + \text{round}(n \times \text{rand}(0, 1))$ 
 $x_{off}(j) = x'_{off}(j)$ 
 $p \leftarrow 0$ 
while  $\text{rand}(0, 1) \leq Cr$  AND  $p < n - 1$  do
   $x_{off}(1 + (j + p) \bmod n) = x'_{off}(1 + (j + p) \bmod n)$ 
   $p = p + 1$ 
  generate  $\text{rand}(0, 1)$ 
end while

```

Fig. 1. Exponential crossover pseudo-code

the chromosome of the mutant vector  $x'_{off}$ .

When the offspring is generated, its fitness value is computed and compared with that of the elite individual. The comparison allows the definition of winner and loser solutions. The winner solution biases the virtual population by affecting the  $PV$  values. The update rule for  $\mu$  values is given by:

$$\mu^{t+1} = \mu^t + \frac{1}{N_p} (\text{winner} - \text{loser}), \quad (4)$$

where  $N_p$  is virtual population size. The update rule for  $\sigma$  values is given by:

$$(\sigma^{t+1})^2 = (\sigma^t)^2 + (\mu^t)^2 - (\mu^{t+1})^2 + \frac{1}{N_p} (\text{winner}^2 - \text{loser}^2) \quad (5)$$

where  $N_p$  is a parameter, namely virtual population size. Details for constructing formulas (4) and (5) are given in [20].

The formulas displayed in eq. (4) and (5) rule the convergence of the virtual population. More specifically, the mean value of the PDF representing the population is moved towards the winner solution while the standard deviation tends to progressively narrow around the most promising solution, thus resulting in a  $\sigma$  value tending toward zero. The latter condition is here indicated as convergence, see [19].

In this paper extra rules for modifying the  $PV$  values are employed. More specifically, with a probability  $M_p$  the  $PV$  is perturbed. The mean value  $\mu$  is perturbed according to the following formula:

$$\mu^{t+1} = \mu^{t+1} + 2\tau \cdot \text{rand}(0, 1) - \tau \quad (6)$$

where  $\tau$  is a weight representing the maximum amplitude of perturbation. Similar to typical DE schemes, a toroidal mechanism (see [24]) ensures that  $\mu$  is bounded by  $-1$  and  $1$  (for example  $1 + 0.1 = -0.9$ ). The perturbation rule for the sigma is given by:

$$(\sigma^{t+1})^2 = (\sigma^{t+1})^2 + \tau \cdot \text{rand}(0, 1). \quad (7)$$

It is worthwhile commenting the perturbation mechanism reported in formulas (6) and (7). The proposed compact DE scheme employs a fairly exploitative structure due to its nature and the employment of the exponential crossover. Thus, each compact unit is likely to prematurely converge towards suboptimal solution. In order to mitigate this effect the perturbation mechanism then inhibits the algorithmic convergence and forces the algorithm to search elsewhere in the decision space, possibly detecting new promising solutions. It can be

observed that the perturbation is equivalent to the replacement of part of the population in a population-based DE.

Each unit performs one offspring generation and possible elite replacement. When all the compact units performed one step, all the elites are inserted into an auxiliary population. Within this auxiliary population, the candidate solutions (the elites) are processed by means of one generation of global optimizer. After one generation, a new population of elite solutions is produced. The elite solutions are then injected into the corresponding compact units and replace the old elite solutions. It is obvious that a global optimizer employing a non-sorting selection mechanism (e.g. DE based algorithms) is preferable as it allows a natural reinsertion of the solutions into the corresponding units. We define the latter mechanism as ‘‘Supervision’’ unit and the novel algorithmic structure as Supervised compact Differential Evolution (ScDE). Supervision unit has the role of recombining the local achievements carried out by each compact unit and promote their exploitation for performing an efficient global search. The newly improved elite solutions after the application of the supervision unit should locally promote the search of unexplored areas of the decision space by affecting the values characterizing the virtual populations.

After a preliminary testing, we selected for this study the jDE, see [17], scheme as the supervision unit. For the sake of clarity, jDE is a DE/rand/1/bin structure (see for definitions [24]) employing a randomization of the control parameters on the basis of a probabilistic criterion. More specifically, each individual is composed of its genotype and its control parameters:

$$x_i = \langle x_{i,1}, x_{i,2}, \dots, x_{i,j}, \dots, x_{i,n}, F_i, CR_i \rangle.$$

In accordance with a self-adaptive logic, see e.g., [25] and [26], the variation operations are preceded by the parameter update. More specifically when, at each generation, the  $i^{th}$  individual  $x_i$  is taken into account and three other individuals are extracted pseudo-randomly, its parameters  $F_i$  and  $CR_i$  are updated according to the following scheme:

$$F_i = \begin{cases} F_l + F_u \text{rand}_1, & \text{if } \text{rand}_2 < \tau_1 \\ F_i, & \text{otherwise} \end{cases} \quad (8)$$

$$CR_i = \begin{cases} \text{rand}_3, & \text{if } \text{rand}_4 < \tau_2 \\ CR_i, & \text{otherwise} \end{cases} \quad (9)$$

where  $\text{rand}_j$ ,  $j \in \{1, 2, 3, 4\}$ , are uniform pseudo-random values between 0 and 1;  $\tau_1$  and  $\tau_2$  are constant values which represent the probabilities that parameters are updated,  $F_l$  and  $F_u$  are constant values which represent the minimum value that  $F$  could take and the maximum variable contribution to  $F$ , respectively. The newly calculated values of  $F_i$  and  $CR_i$  are then used for generating the offspring by applying DE/rand/1 mutation displayed in eq. (3) and binomial crossover as suggested in the original jDE in [17]. In this crossover strategy, each gene of the individual  $x'_{off}$  is exchanged with the corresponding gene of  $x_i$  with a uniform probability and the

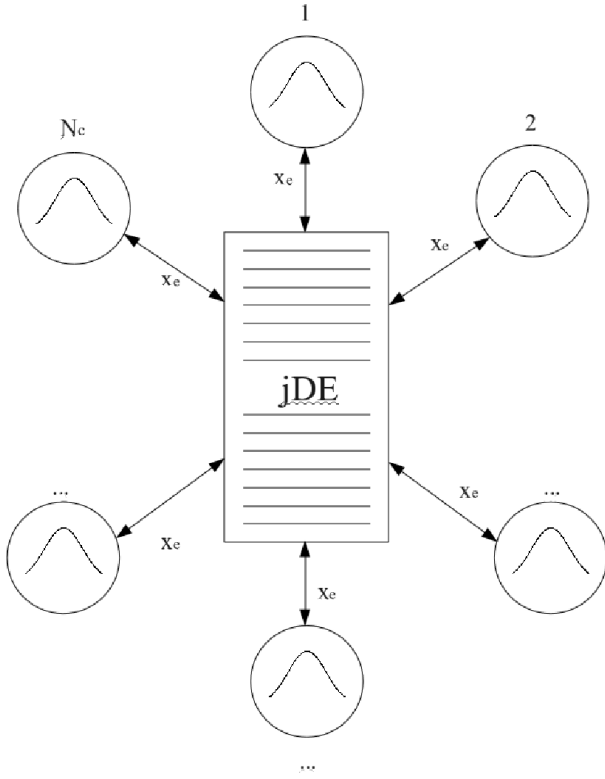


Fig. 2. Graphical representation of Supervised compact Differential Evolution

final offspring  $x_{off}$  is generated:

$$x_{off,j} = \begin{cases} x_{i,j} & \text{if } rand(0,1) < CR \\ x'_{off,j} & \text{otherwise} \end{cases} \quad (10)$$

where  $rand(0,1)$  is a random number between 0 and 1;  $j$  is the index of the gene under examination.

For the sake of clarity a graphical representation of ScDE is given in Fig 2.

Fig. 3 and 4 summarize the working principles (one generation) at the generic compact unit and supervision unit, respectively.

### III. NUMERICAL RESULTS

All the base problems defined in the CEC 2005 testbed, see [27], are considered in this study. More specifically, the following test functions have been included:

- $f_1$  Shifted sphere function.
- $f_2$  Shifted Schwefel's Problem 1.2
- $f_3$  Shifted Rotated High Conditioned Elliptic Function
- $f_4$  Shifted Schwefel's Problem 1.2 with Noise
- $f_5$  Schwefels Problem 2.6 with Global Optimum on Bounds
- $f_6$  Shifted Rosenbrocks Function
- $f_7$  Shifted rotated Griewank's Function with Bounds
- $f_8$  Shifted rotated Ackley's Function with Global Optimum on the Bounds
- $f_9$  Shifted Rastrigin's Function:
- $f_{10}$  Shifted rotated Rastrigin's Function
- $f_{11}$  Shifted Rotated Weierstrass Function

```

receive  $x_e$  from the supervision unit
{** Mutation **}
generate 3 individuals  $x_r, x_s$ , and  $x_t$  by means of PV
compute  $x'_{off} = x_t + F(x_r - x_s)$ 
{** Crossover **}
apply exponential crossover shown in Fig. 1 and generate  $x_{off}$ 
{** Elite Selection **}
[ $winner, loser$ ] = compete ( $x_{off}, x_e$ )
if  $x_{off} == winner$  then
     $x_e = x_{off}$ 
end if
{** PV Update **}
 $\mu^{t+1} = \mu^t + \frac{1}{N_p} (winner - loser)$ 
 $\sigma^{t+1} = \sqrt{(\sigma^t)^2 + (\mu^t)^2 - (\mu^{t+1})^2 + \frac{1}{N_p} (winner^2 - loser^2)}$ 
{** Perturbation of the PV **}
if  $rand(0,1) < M_p$  then
     $\mu^{t+1} = \mu^{t+1} + 2\tau \cdot rand(0,1) - \tau$ 
     $\sigma^{t+1} = \sqrt{(\sigma^{t+1})^2 + \tau \cdot rand(0,1)}$ 
end if
 $t = t + 1$ 
inject  $x_e$  into the supervision unit

```

Fig. 3. Pseudo-code at the generic compact unit

```

receive  $N_c$  elite solutions
for  $i = 1$  to  $N_c$  do
    {**  $F_i$  update **}
    generate  $rand_1$  and  $rand_2$ 
     $F_i = \begin{cases} F_l + F_u \cdot rand_1, & \text{if } rand_2 < \tau_1 \\ F_i, & \text{otherwise} \end{cases}$ 
    {**mutation**}
    perform standard DE/rand/1 mutation  $x'_{off} = x_t + F_i(x_r - x_s)$ 
    {**  $CR_i$  update **}
    generate  $rand_3$  and  $rand_4$ 
     $CR_i = \begin{cases} rand_3, & \text{if } rand_4 < \tau_2 \\ CR_i, & \text{otherwise} \end{cases}$ 
    {**crossover**}
    perform standard binomial crossover with the  $CR_i$  calculated
    {**selection**}
    perform the standard DE one-to-one spawning selection
end for
inject each elite solution into its corresponding compact unit by updating the elite value

```

Fig. 4. Pseudo-code at the supervision unit

$f_{12}$  Schwefel's Problem 2.13

$f_{13}$  Expanded Extended Griewanks plus Rosenbrocks Function (F8F2)

$f_{14}$  Shifted Rotated Expanded Scaffers F6

In order to understand the scalability of the proposed approach, the test problems above have been considered for  $n = 30$  and  $n = 100$ . Thus, totally 28 test problems are included in this study. The proposed ScDE has been run with  $N_c = 20$  compact units, each of them being characterized by a virtual population size  $N_p = 20$ . Within each unit, the control parameters  $F$  and  $CR$  have been set equal to 0.5 and 0.9, respectively. The perturbation mechanism is activated with a probability  $M_p = 0.001$  and a weight factor  $\tau = 0.1$ . The supervision unit is a jDE with  $F_l = 0.1$ ,  $F_u = 0.9$ , and  $\tau_1 = \tau_2 = 0.1$ , see [17]. ScDE has been compared with three modern DE based algorithms representing the state-of-the-art for DE. The algorithms are listed in the following:

- j-Differential Evolution (jDE) with  $F_l = 0.1$ ,  $F_u = 0.9$ , and  $\tau_1 = \tau_2 = 0.1$ , see [17].
- J-Adaptive Differential Evolution (JADE) with  $c = 0.1$ , see [18].

TABLE II  
WILCOXON TEST FOR 30D TEST PROBLEMS

Test Problem	jDE	JADE	SADE
$f_1$	=	=	=
$f_2$	+	-	=
$f_3$	+	-	-
$f_4$	-	-	-
$f_5$	-	-	-
$f_6$	=	-	=
$f_7$	-	=	+
$f_8$	+	+	+
$f_9$	+	-	+
$f_{10}$	+	=	+
$f_{11}$	-	+	=
$f_{12}$	+	+	+
$f_{13}$	+	+	+
$f_{14}$	+	=	=

- Self Adaptive Differential Evolution (SADE), see [3].

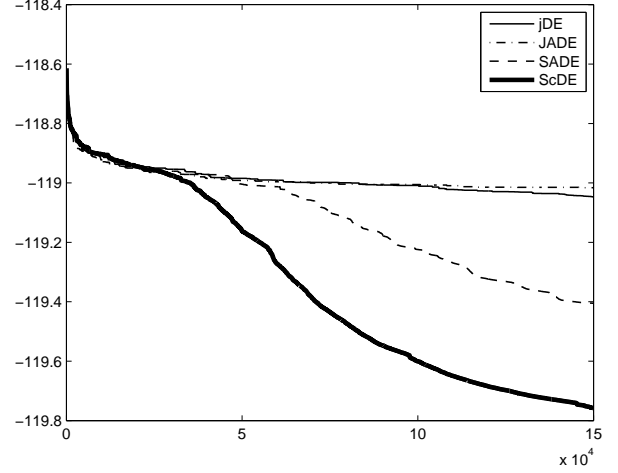
We intentionally decided to leave out from the result display basic, simple, and not so promising versions of DE in order to put our ScDE in the most challenging conditions. For each algorithm and test problem, 30 independent runs have been performed. The budget of each single run has been fixed equal to  $5000 \times n$  fitness evaluations. The population size of the population-based algorithms has been fixed equal to the dimensionality of the corresponding problem. This choice has been performed on the basis of the experimental results reported in [28] where it is shown that, in DE based algorithms, relatively small population size values often lead to a better performance with respect to traditional size values several times bigger than the dimensionality. Table I shows the average of the final results detected by each algorithm under investigation  $\pm$  the corresponding standard deviation values, for the 30-dimension test problems. The best results are highlighted in bold face.

In order to strengthen the statistical significance of the results, the Wilcoxon Rank-Sum test has also been applied according to the description given in [29], where the confidence level has been fixed to 0.95. Table II summarizes the results of the Wilcoxon test. A “+” indicates the case in which ScDE statistically outperforms, for the corresponding test problem, the other algorithm considered for comparison; a “=” indicates that a pairwise comparison leads to success of the Wilcoxon Rank-Sum test, i.e., the two algorithms have the same performance; a “-” indicates that ScDE is outperformed.

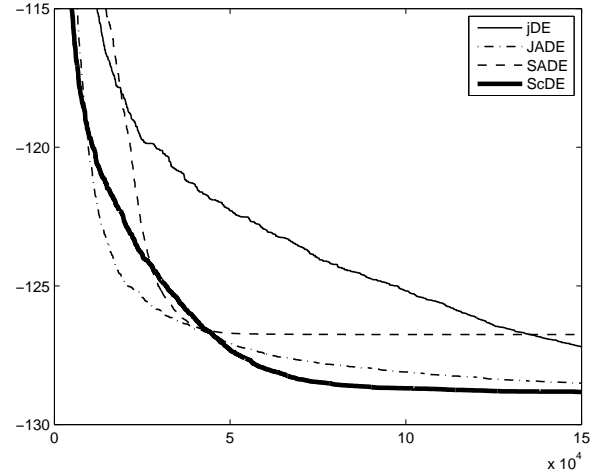
Figures 5 show some examples of average performance trends of ScDE and the other DE based algorithms for 30-dimensional test problems.

Numerical results related to 100D test problems are reported in Table III, Table IV, and Figures 6.

Numerical results show that for both the dimensionality cases, 30 and 100 dimensions, the proposed ScDE displays a respectable performance against the advanced algorithms considered in this study. More specifically, ScDE appears to be



(a)  $f_8$  with  $n = 30$



(b)  $f_{12}$  with  $n = 30$

Fig. 5. Performance trends for 30D test problems

TABLE IV  
WILCOXON TEST FOR 100D TEST PROBLEMS

Test Problem	jDE	JADE	SADE
$f_1$	=	=	=
$f_2$	+	-	+
$f_3$	=	-	-
$f_4$	=	-	=
$f_5$	-	-	-
$f_6$	+	+	+
$f_7$	-	-	+
$f_8$	+	+	+
$f_9$	+	-	+
$f_{10}$	+	-	+
$f_{11}$	-	+	+
$f_{12}$	+	+	+
$f_{13}$	+	+	+
$f_{14}$	+	=	+

TABLE I  
AVERAGE FINAL FITNESS  $\pm$  STANDARD DEVIATION FOR 30D PROBLEMS

Test Problem	jDE	JADE	SADE	ScDE
$f_1$	<b>0.000e+00</b> $\pm$ <b>0.000e+00</b>	<b>0.000e+00</b> $\pm$ <b>0.000e+00</b>	<b>0.000e+00</b> $\pm$ <b>0.000e+00</b>	<b>0.000e+00</b> $\pm$ <b>0.000e+00</b>
$f_2$	4.219e+00 $\pm$ 4.91e+00	<b>2.842e-13</b> $\pm$ <b>2.99e-13</b>	1.131e-01 $\pm$ 2.42e-01	8.795e-02 $\pm$ 8.38e-02
$f_3$	1.656e+06 $\pm$ 6.91e+05	<b>4.408e+04</b> $\pm$ <b>3.15e+04</b>	3.393e+05 $\pm$ 1.73e+05	1.286e+06 $\pm$ 9.94e+05
$f_4$	2.708e+02 $\pm$ 2.86e+02	<b>3.830e+00</b> $\pm$ <b>5.91e+00</b>	3.885e+02 $\pm$ 4.47e+02	9.539e+02 $\pm$ 5.97e+02
$f_5$	1.437e+03 $\pm$ 6.78e+02	<b>6.276e+02</b> $\pm$ <b>4.56e+02</b>	1.997e+03 $\pm$ 7.35e+02	2.890e+03 $\pm$ 6.55e+02
$f_6$	3.438e+01 $\pm$ 3.00e+01	<b>7.489e+00</b> $\pm$ <b>3.00e+01</b>	2.209e+01 $\pm$ 2.36e+01	3.108e+01 $\pm$ 3.35e+01
$f_7$	<b>1.139e-02</b> $\pm$ <b>6.87e-03</b>	1.498e-02 $\pm$ 9.99e-03	4.696e+03 $\pm$ 2.08e-07	1.764e-02 $\pm$ 1.29e-02
$f_8$	2.095e+01 $\pm$ 6.10e-02	2.098e+01 $\pm$ 3.72e-02	2.060e+01 $\pm$ 3.78e-01	<b>2.024e+01</b> $\pm$ <b>2.16e-01</b>
$f_9$	8.291e-01 $\pm$ 6.98e-01	<b>0.000e+00</b> $\pm$ <b>0.000e+00</b>	1.526e+01 $\pm$ 1.43e+01	7.299e-01 $\pm$ 3.58e+00
$f_{10}$	6.218e-01 $\pm$ 6.44e-01	<b>0.000e+00</b> $\pm$ <b>0.000e+00</b>	1.884e+01 $\pm$ 1.52e+01	<b>0.000e+00</b> $\pm$ <b>0.000e+00</b>
$f_{11}$	<b>1.978e+01</b> $\pm$ <b>4.58e+00</b>	2.690e+01 $\pm$ 1.73e+00	2.219e+01 $\pm$ 3.21e+00	2.314e+01 $\pm$ 4.49e+00
$f_{12}$	7.917e+04 $\pm$ 2.54e+04	3.470e+04 $\pm$ 5.66e+03	3.314e+04 $\pm$ 2.95e+04	<b>8.943e+03</b> $\pm$ <b>7.54e+03</b>
$f_{13}$	2.803e+00 $\pm$ 1.03e+00	1.495e+00 $\pm$ 1.01e-01	3.246e+00 $\pm$ 9.55e-01	<b>1.177e+00</b> $\pm$ <b>2.08e-01</b>
$f_{14}$	1.293e+01 $\pm$ 4.05e-01	<b>1.239e+01</b> $\pm$ <b>2.42e-01</b>	1.243e+01 $\pm$ 5.82e-01	1.250e+01 $\pm$ 3.46e-01

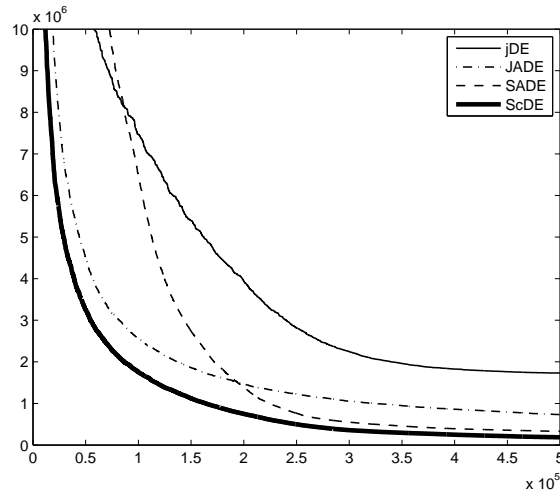
TABLE III  
AVERAGE FINAL FITNESS  $\pm$  STANDARD DEVIATION FOR 100D PROBLEMS

Test Problem	jDE	JADE	SADE	ScDE
$f_1$	<b>5.684e-14</b> $\pm$ <b>8.13e-14</b>	2.274e-13 $\pm$ 3.25e-13	2.842e-13 $\pm$ 3.04e-13	4.547e-13 $\pm$ 5.14e-13
$f_2$	6.957e+03 $\pm$ 2.09e+03	<b>4.890e+00</b> $\pm$ <b>2.34e+01</b>	1.716e+03 $\pm$ 7.42e+02	2.630e+02 $\pm$ 8.35e+01
$f_3$	8.799e+06 $\pm$ 2.46e+06	<b>1.844e+06</b> $\pm$ <b>3.67e+05</b>	5.825e+06 $\pm$ 1.82e+06	9.972e+06 $\pm$ 3.43e+06
$f_4$	1.006e+05 $\pm$ 3.05e+04	<b>5.000e+04</b> $\pm$ <b>8.68e+03</b>	9.405e+04 $\pm$ 2.25e+04	8.773e+04 $\pm$ 2.56e+04
$f_5$	<b>9.694e+03</b> $\pm$ <b>1.20e+03</b>	1.072e+04 $\pm$ 1.77e+03	1.682e+04 $\pm$ 2.28e+03	2.217e+04 $\pm$ 3.07e+03
$f_6$	1.335e+02 $\pm$ 4.75e+01	1.292e+02 $\pm$ 4.52e+01	1.698e+02 $\pm$ 4.92e+01	<b>1.109e+02</b> $\pm$ <b>2.38e+01</b>
$f_7$	<b>4.181e-03</b> $\pm$ <b>5.73e-03</b>	6.466e-03 $\pm$ 6.73e-03	1.398e+04 $\pm$ 1.10e+03	1.686e-02 $\pm$ 1.06e-02
$f_8$	2.126e+01 $\pm$ 2.12e-01	2.126e+01 $\pm$ 2.44e-01	2.037e+01 $\pm$ 2.90e-01	<b>2.026e+01</b> $\pm$ <b>2.25e-01</b>
$f_9$	3.384e+01 $\pm$ 7.27e+00	<b>5.684e-14</b> $\pm$ <b>6.91e-14</b>	1.661e+02 $\pm$ 4.85e+01	5.671e+00 $\pm$ 2.78e+01
$f_{10}$	3.826e+01 $\pm$ 7.34e+00	<b>4.146e-02</b> $\pm$ <b>2.03e-01</b>	1.660e+02 $\pm$ 5.62e+01	1.513e+01 $\pm$ 5.29e+01
$f_{11}$	<b>8.800e+01</b> $\pm$ <b>8.65e+00</b>	1.316e+02 $\pm$ 3.08e+00	1.220e+02 $\pm$ 8.64e+00	1.136e+02 $\pm$ 1.11e+01
$f_{12}$	1.732e+06 $\pm$ 4.14e+05	7.326e+05 $\pm$ 7.55e+04	3.299e+05 $\pm$ 1.50e+05	<b>1.870e+05</b> $\pm$ <b>2.49e+05</b>
$f_{13}$	1.007e+01 $\pm$ 1.90e+00	7.625e+00 $\pm$ 5.60e-01	1.814e+01 $\pm$ 2.74e+00	<b>4.100e+00</b> $\pm$ <b>6.44e-01</b>
$f_{14}$	4.679e+01 $\pm$ 5.41e-01	<b>4.576e+01</b> $\pm$ <b>5.46e-01</b>	4.649e+01 $\pm$ 8.55e-01	4.601e+01 $\pm$ 7.53e-01

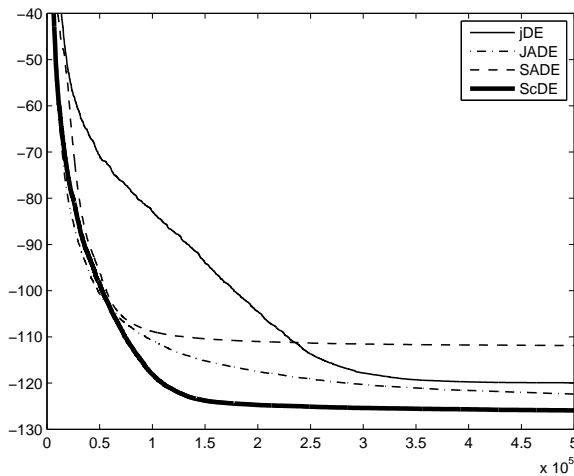
significantly more promising than SADE. Results also show that ScDE has, on average, a better performance than jDE. If we consider that the supervision unit is jDE based, we can conclude that the proposed algorithmic structure has actually some potential. The comparison with JADE shows that our ScDE is slightly outperformed, especially in the 30D case. The performance of ScDE with respect to the other algorithms contained in this study appears more promising in 100D rather than 30D cases. Despite the fact that further investigations are necessary in order to draw proper conclusions on the scalability of the proposed approach, it is interesting to remark that on the basis of the preliminary results the proposed scheme appears to be a promising alternative for large scale problems. This conjecture can be justified by considering ScDE as a distributed algorithm where instead of actual sub-populations there are virtual populations and where instead of a classical migration scheme there is the supervision unit for propagating the most promising solutions.

#### IV. CONCLUSION

This paper proposes a novel algorithmic structure for solving continuous optimization problems. This novel structure employs cDE units as local search components and a global optimizer as a supervising structure. More specifically, the cDE units evolve independently, then feed with their elites the population of a global optimizer which processes the solutions returning a new set of elite solutions for the cDE units. The supervision unit has thus the role of performing the global search and promoting the propagation of the most promising genotypes between the various compact units. A preliminary implementation of this idea is presented in this paper, where the supervision unit is a jDE. The proposed algorithm has been tested on a set of popular test problems with two levels of dimensionality and has been compared with the-state-of-the-art DE based algorithms. Numerical results show that the preliminary implementation included in this paper displays a comparable performance with respect to advanced DE based algorithms. This allows us to conclude that the algorithmic



(a)  $f_{12}$  with  $n = 100$



(b)  $f_{13}$  with  $n = 100$

Fig. 6. Performance trends for 100D test problems

idea here proposed has a good potential. It is interesting to remark that the most promising results have been obtained in the higher dimensional case.

Further development of this work will be considered in order to better exploit the idea of supervision unit for a set of simultaneous cDEs. A more extensive testing will be performed in order to verify the potential for highly dimensional problems. In addition, other structures will be considered. More specifically, further work will focus on communication strategies between the various compact units. Communication strategies involving not only the elite solutions but also virtual populations will be taken into account as well as the design of ad-hoc global optimizers for playing the role of supervision units.

#### ACKNOWLEDGMENT

This research 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. This research is also supported by Tekes - the Finnish Funding Agency for Technology and Innovation, grant 40214/08 (Dynergia).

#### REFERENCES

- [1] F. Neri and V. Tirronen, "Recent advances in differential evolution: A review and experimental analysis," *Artificial Intelligence Review*, vol. 33, no. 1, pp. 61–106, 2010.
- [2] H.-Y. Fan and J. Lampinen, "A trigonometric mutation operation to differential evolution," *Journal of Global Optimization*, vol. 27, no. 1, pp. 105–129, 2003.
- [3] 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, pp. 398–417, 2009.
- [4] S. Rahnamayan, H. R. Tizhoosh, and M. M. Salama, "Opposition-based differential evolution," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 1, pp. 64–79, 2008.
- [5] 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.
- [6] J. Brest, B. Bošković, S. Greiner, V. Žumer, and M. S. Maučec, "Performance comparison of self-adaptive and adaptive differential evolution algorithms," *Soft Computing*, vol. 11, no. 7, pp. 617–629, 2007.
- [7] J. Brest, A. Zamuda, B. Bošković, M. S. Maucec, and V. Žumer, "High-dimensional real-parameter optimization using self-adaptive differential evolution algorithm with population size reduction," in *Proceedings of the IEEE World Congress on Computational Intelligence*, 2008, pp. 2032–2039.
- [8] F. Neri and V. Tirronen, "Scale factor local search in differential evolution," *Memetic Computing Journal*, vol. 1, no. 2, pp. 153–171, 2009.
- [9] 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.
- [10] 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, no. 8, pp. 811–831, 2009.
- [11] Y. Gao and Y.-J. Wang, "A memetic differential evolutionary algorithm for high dimensional functions' optimization," in *Proceedings of the Third International Conference on Natural Computation*, 2007, pp. 188–192.
- [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] D. K. Tasoulis, N. G. Pavlidis, V. P. Plagianakos, and M. N. Vrahatis, "Parallel differential evolution," in *Proceedings of the IEEE Congress on Evolutionary Computation*, 2004, pp. 2023–2029.
- [14] 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.
- [15] M. Weber, F. Neri, and V. Tirronen, "Distributed differential evolution with explorative-exploitative population families," *Genetic Programming and Evolvable Machines*, vol. 10, no. 4, pp. 343–371, 2009.
- [16] S. Das, A. Konar, and U. K. Chakraborty, "Two improved differential evolution schemes for faster global search," in *Proceedings of the 2005 conference on Genetic and evolutionary computation*. ACM, 2005, pp. 991–998.
- [17] 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.
- [18] J. Zhang and A. C. Sanderson, "Jade: Adaptive differential evolution with optional external archive," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 5, pp. 945–958, 2009.

- [19] E. Mininno, F. Neri, F. Cupertino, and D. Naso, "Compact differential evolution," *IEEE Transactions on Evolutionary Computation*, 2011, to appear.
- [20] E. Mininno, F. Cupertino, and D. Naso, "Real-valued compact genetic algorithms for embedded microcontroller optimization," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 2, pp. 203–219, 2008.
- [21] 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.
- [22] W. Gautschi, "Error function and fresnel integrals," in *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*, M. Abramowitz and I. A. Stegun, Eds., 1972, ch. 7, pp. 297–309.
- [23] W. J. Cody, "Rational chebyshev approximations for the error function," *Mathematics of Computation*, vol. 23, no. 107, pp. 631–637, 1969.
- [24] K. V. Price, R. Storn, and J. Lampinen, *Differential Evolution: A Practical Approach to Global Optimization*. Springer, 2005.
- [25] I. Rechemberg, *Evolutionstrategie: Optimierung Technischer Systeme nach prinzipien des Biologischen Evolution*. Fromman-Holzboog Verlag, 1973.
- [26] K. Ohkura, Y. Matsumura, and K. Ueda, "Robust evolution strategies," *Applied Intelligence*, vol. 15, no. 3, pp. 153–169, 2001.
- [27] 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.
- [28] F. Neri and V. Tirronen, "On memetic differential evolution frameworks: a study of advantages and limitations in hybridization," in *Proceedings of the IEEE World Congress on Computational Intelligence*, pp. 2135–2142.
- [29] F. Wilcoxon, "Individual comparisons by ranking methods," *Biometrics Bulletin*, vol. 1, no. 6, pp. 80–83, 1945.