Ensemble Strategies in Compact Differential Evolution

Rammohan Mallipeddi, Giovanni Iacca, Ponnuthurai Nagaratnam Suganthan, Ferrante Neri, Ernesto Mininno
School of Electrical and Electronics Engineering, Nanyang Technological university, Singapore, 639798.
E-mail: mallipeddi.ram@gmail.com, epnsugan@ntu.edu.sg
Department of Mathematical Information Technology, P.O. Box 35 (Agora), University of Jyväskylä, Finland
E-mail: giovanni.iacca@jyu.fi, ferrante.neri@jyu.fi, ernesto.mininno@jyu.fi

Abstract—Differential Evolution is a population based stochastic algorithm with less number of parameters to tune. However, the performance of DE is sensitive to the mutation and crossover strategies and their associated parameters. To obtain optimal performance, DE requires time consuming trial and error parameter tuning. To overcome the computationally expensive parameter tuning different adaptive/self-adaptive techniques have been proposed. Recently the idea of ensemble strategies in DE has been proposed and favorably compared with some of the state-of-the-art self-adaptive techniques. Compact Differential Evolution (cDE) is modified version of DE algorithm which can be effectively used to solve real world problems where sufficient computational resources are not available. cDE can be implemented on devices such as micro controllers or Graphics Processing Units (GPUs) which have limited memory. In this paper we introduced the idea of ensemble into cDE to improve its performance. The proposed algorithm is tested on the 30D version of 14 benchmark problems of Conference on Evolutionary Computation (CEC) 2005. The employment of ensemble strategies for the cDE algorithms appears to be beneficial and leads, for some problems, to competitive results with respect to the-state-of-the-art DE based algorithms.

Keywords—Compact Differential Evolution; Ensemble; Global Optimization; parameter adaptation; mutation strategy.

I. INTRODUCTION

Differential Evolution (DE) is a simple yet powerful population-based stochastic search technique widely used for solving global optimization over continuous spaces. The performance [1, 2] of the DE algorithm is sensitive to the mutation strategy, crossover strategy and control parameters such as the population size (NP), crossover rate (CR) and the scale factor (F). The best settings for the control parameters can be different for different optimization problems and the same functions with different requirements for consumption time and accuracy. Therefore, to successfully solve a specific optimization problem, it is generally necessary to perform a time-consuming trial-and-error search for the most appropriate combination of strategies and their associated parameter values. However, such a trial-and-error search process suffers from high computational costs. The population of DE may evolve through different regions in the search space, within which different strategies [3] with different parameter settings may be more effective than others. To overcome the time consuming trial and error parameter tuning different partial adaptation schemes have been proposed [3-7]. In [8] we proposed and demonstrated the superior performance of the ensemble strategy proposed using DE as the base algorithm.

DE start with a population of individuals, which communicate and exchange information through natural processes like mutation, recombination and selection, to evolve increasingly fitter new individuals to a particular environment. Therefore significant resources are required to store the individuals of the population in DE. In practical applications, such as commercial vacuum cleaner robot, powerful computing devices may be unavailable due to cost and/or space limitations which restricts the usage of DE. In order to overcome this problem compact DE was proposed [9, 10]. cDE do not store and process an entire population of solutions instead uses its probabilistic representation in order to perform the optimization process. Therefore, cDE implementation requires a much smaller number of parameters to be stored. Hence, a run of cDE requires much less memory compared to standard DE.

cDE contains an evolving distribution which is sampled in every generation to produce candidate solutions which undergo mutation and crossover. Therefore, cDE reproduces the search logic of the standard DE algorithm. The main drawback of cDE 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 modeling the population, the search tends to quickly narrow around the most promising candidate solutions. Therefore, the performance of cDE as compared to the classical degrades as the dimensionality increases. Therefore to improve the performance of cDE we propose an ensemble cDE (EnDE).

The remainder of the paper is organized in the following way. Section II presents a brief literature review on the classical and compact DE algorithms. Section III presents the proposed EnDE algorithm. Section IV presents the experimental results and Section V concludes the paper.
II. LITERATURE REVIEW

A. Differential Evolution

Differential Evolution (DE) is a parallel direct search algorithm which utilizes \( NP \) \( D \)-dimensional parameter vectors, \( X_{i,G} = [x_{i,1}^G, x_{i,2}^G, ..., x_{i,D}^G] \), \( i = 1, ..., NP \) uniformly distributed within the search space constrained by minimum and maximum parameter bounds \( X_{\text{min}} = [x_{\text{min},1}, x_{\text{min},2}, ..., x_{\text{min},D}] \) and \( X_{\text{max}} = [x_{\text{max},1}, x_{\text{max},2}, ..., x_{\text{max},D}] \). \( G \) is the generation count. The initial population undergoes evolution by communication between them through operations such as mutation, crossover and selection.

Through mutation operation each individual \( X_{i,G} \) in the current population generates a mutant vector \( V_{i,G} = [v_{i,1}^G, v_{i,2}^G, ..., v_{i,D}^G] \). Some of the most frequently used mutation strategies are

\[
\text{DE/best/1} [11]: \quad V_{i,G} = X_{\text{best},G} + F.(X_{i,G} - X_{\text{rand},G}) \tag{1}
\]

\[
\text{DE/best/2} [11]: \quad V_{i,G} = X_{\text{best},G} + F.(X_{i,G} - X_{\text{rand},G}) + F.(X_{\text{best},G} - X_{\text{rand},G}) \tag{2}
\]

\[
\text{DE/rand/1} [11]: \quad V_{i,G} = X_{\text{best},G} + F.(X_{i,G} - X_{\text{rand},G}) \tag{3}
\]

\[
\text{DE/current-to-rand/1} [12]: \quad V_{i,G} = X_{i,G} + K.(X_{\text{best},G} - X_{\text{rand},G}) \tag{4}
\]

The indices \( r_1, r_2, r_3, r_4 \) are mutually exclusive integers randomly generated within the range \([1, NP] \), which are also different from the index \( i \). These indices are randomly generated once for each mutant vector. The scale factor \( F \) is a positive control parameter for scaling the difference vector. \( X_{\text{best},G} \) is the best individual vector with the best fitness value in the population at generation \( G \). \( K \) is randomly chosen within the range \([0, 1] \).

In crossover operation each pair of the target vector \( X_{i,G} \) and its corresponding mutant vector \( V_{i,G} \) combine to generate a trial vector \( U_{i,G} = [u_{i,1}^G, u_{i,2}^G, ..., u_{i,D}^G] \). Basic DE employs the binomial (uniform) crossover defined as follows [13]:

\[
u_{i,j}^G = \begin{cases} v_{i,j}^G & \text{if } \text{rand}(0,1) \leq CR \text{ or } j = j_{\text{rand}} \\ x_{i,j}^G & \text{otherwise} \end{cases} \quad \text{for } j = 1, 2, ..., D \tag{5}
\]

In equation (5), the crossover rate \( CR \) is a user-specified constant within the range \([0, 1] \), which controls the fraction of parameter values copied from the mutant vector. \( j_{\text{rand}} \) is a randomly chosen integer in the range \([1, D] \).

In exponential crossover [14], an integer \( n \in [1, D] \) is chosen randomly, which acts as a starting point in the target vector, from where the crossover or exchange of components with the mutant vector starts. \( L \in [1, D] \) denotes the number of components that are contributed by the mutant vector to the target vector. The integer \( L \) is drawn from \([1, D] \) depending on the crossover probability \( (CR) \), according to the following pseudo-code:

\[
L = 0; \quad \text{DO} \quad L = L + 1; \quad \text{WHILE} \quad (\text{rand}(0,1) < CR \text{ and } L < D); \tag{6}
\]

After choosing \( n \) and \( L \), the trial vector is obtained as:

\[
u_{i,j}^G = \begin{cases} v_{i,j}^G & \text{for } j = (n)_j, (n+1)_j, ..., (n+L-1)_j \\ x_{i,j}^G \text{ for all other } j \in [1, D] \end{cases} \tag{7}
\]

where the angular brackets \( \langle \cdot \rangle_D \) denote a modulo function with modulus \( D \).

In exponential crossover operator, the parameters of trial vector \( U_{i,G} \) are inherited from the corresponding mutant vector \( V_{i,G} \) starting from a randomly chosen parameter index \( (n) \) till the first time \( \text{rand}(0,1) < CR \). The remaining parameters of the trial vector \( U_{i,G} \) are copied from the corresponding target vector \( X_{i,G} \). DE’s exponential crossover operator is functionally equivalent to the circular two-point crossover operator [3].

Once the objective function values of all trial vectors are evaluated a selection operation is performed. The objective function value of each trial vector \( f(U_{i,G}) \) is compared to that of its corresponding target vector \( f(X_{i,G}) \) in the current population. If the trial vector has less or equal objective function value (in a minimization problem) than the corresponding target vector, the trial vector will replace the target vector and enter the population of the next generation. Otherwise, the target vector will remain in the population for the next generation. The selection operation can be expressed as follows:

\[
X_{i,G+1} = \begin{cases} U_{i,G} \text{ if } f(U_{i,G}) \leq f(X_{i,G}) \\ X_{i,G}, \text{ otherwise} \end{cases} \tag{7}
\]

The above 3 steps are repeated generation after generation until a termination criterion (reaching the maximum number of function evaluations set) is satisfied. The algorithmic description of the DE is summarized in Table I.

In DE, the mutation strategy, crossover strategy and control parameters such as the population size \( (NP) \), crossover rate \( (CR) \) and the scale factor \( (F) \) can affect the performance [1, 2] of the DE algorithm. The best settings for the control parameters can be different for different
Step 2

We propose and demonstrated that the superior performance of the ensemble strategy proposed, compared to solutions obtained using single adaptation schemes. Although different partial adaptation schemes have been proposed [3-7] to overcome the time consuming trial-and-error procedure, in [8] we proposed and demonstrated the superior performance of the ensemble strategy proposed.

TABLE I: The Standard DE Algorithm

**Step 1** Set the generation number \( G = 0 \), and initialize a population of \( NP \) individuals \( P_0 = \{X_{0,1}, \ldots, X_{0,NP}\} \) with \( X_{i,0} = [x_{i,0}^1, \ldots, x_{i,0}^D] \), \( i = 1, \ldots, NP \) uniformly distributed in the search space, within which different strategies [3] with different parameter settings may be more effective than others. Although different partial adaptation schemes have been proposed [3-7] to overcome the time consuming trial-and-error search for the most appropriate combination of strategies and their associated parameter values is necessary to successfully solve a specific optimization problem. However, such a trial-and-error search process suffers from high computational costs. The population of DE may evolve through different regions in the search space, within which different strategies [3] with different parameter settings may be more effective than others. Although different partial adaptation schemes have been proposed [3-7] to overcome the time consuming trial-and-error search for the most appropriate combination of strategies and their associated parameter values is necessary to successfully solve a specific optimization problem. However, such a trial-and-error search process suffers from high computational costs. The population of DE may evolve through different regions in the search space, within which different strategies [3] with different parameter settings may be more effective than others.

**Step 2** WHILE stopping criterion is not satisfied DO

1. **Step 2.1 Mutation step**
   
   /*Generate a mutant vector \( U_{i,G} = [u_{i,1}^1, u_{i,2}^2, \ldots, u_{i,D}^D] \) for each target vector \( X_{i,G} \)
   
   */

   FOR \( i = 1 \) to \( NP \)

   Generate a mutant vector \( U_{i,G} = [u_{i,1}^1, u_{i,2}^2, \ldots, u_{i,D}^D] \) corresponding to the target vector \( X_{i,G} \) via one of the equations (2)-(6).

   END FOR

2. **Step 2.2 Crossover step**

   /*Generate a trial vector \( U_{i,G} = [u_{i,1}^1, u_{i,2}^2, \ldots, u_{i,D}^D] \) for each target vector \( X_{i,G} \)
   
   */

   /*Binomial crossover*/

   FOR \( i = 1 \) to \( NP \)

   FOR \( j = 1 \) to \( D \)

   if \( \text{rand}[0,1] \leq CR \) or \( (j = j_{rand}) \)

   \( u_{i,j}^D = v_{i,j}^D \)

   otherwise

   \( u_{i,j}^D = v_{i,j}^D \)

   END FOR

   END FOR

3. **Step 2.3 Selection step**

   /*Selection*/

   FOR \( i = 1 \) to \( NP \)

   /* Evaluate the trial vector \( U_{i,G} \)

   */

   IF \( f(U_{i,G}) \leq f(X_{i,G}) \), THEN

   \( X_{i,G+1} = U_{i,G} \)

   ELSE

   \( X_{i,G+1} = X_{i,G} \)

   END IF

END IF

END FOR

**Step 3** Increment the generation count \( G = G + 1 \)

**Step 3** END WHILE

B. Compact Differential Evolution

cDE consists of a \( (2 \times D) \) probability vector \( (PV) \) where each of the 2 \( (1 \times D) \) vectors corresponding to mean (\( \mu \)) and standard deviation (\( \sigma \)) values respectively. The \( \mu \) values are set equal to 0 while \( \sigma \) values are set equal to a large positive number \( \Delta \). A solution is sampled from \( PV \) and plays the role of elite. Subsequently, at each generation \( G \), some solutions are sampled on the basis of the selected mutation scheme. For example, if a DE/rand/1 mutation (equation 3) is selected; three individuals \( X_{i,G} \), \( X_{j,G} \), and \( X_{k,G} \) are sampled from \( PV \).

More specifically, the sampling mechanism of a design variable \( X_{i,[l]} \) associated to a generic candidate solution \( X_{i,G} \) 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 [9]:

\[
PDF_{\text{truncNorm}}(x) = \frac{e^{-\frac{(x-\mu[i])^2}{2\sigma[i]^2}}}{\sqrt{2\pi} \sigma[i]} \quad \text{for } x \in [\mu[i]-3\sigma[i], \mu[i]+3\sigma[i]]
\]

where \( \text{erf} \) is the error function.

From the PDF, the corresponding Cumulative Distribution Function (CDF) is constructed by means of Chebyshev polynomials according to the procedure described in [33]. It must be observed that the co domain of CDF is [0, 1]. In order to sample the design variable \( X_{i,[l]} \) from \( PV \) a random number \( \text{rand}(0, 1) \) is sampled from a uniform distribution. The inverse function of CDF, in correspondence of \( \text{rand}(0, 1) \), is then calculated. This latter value is \( X_{i,[l]} \).

The mutation is then performed and the provisional offspring is generated. A crossover between the elite and the provisional offspring is performed in order to generate the
offspring. The fitness value of the offspring is then computed and compared with that of the elite individual. The comparison allows the definition of winner and loser solutions. The following equations are then applied to update the PV for the subsequent solution generations. If the offspring outperforms the elite individual, the offspring replaces the elite.

\[
\mu^{G+1} = \mu^G + \frac{1}{N_p} (\text{winner} - \text{loser})
\]

\[
(\sigma^{G+1})^2 = (\sigma^G)^2 + \frac{1}{N_p} (\text{winner}^2 - \text{loser}^2 - 1)
\]

Table II: Compact Differential Evolution

<table>
<thead>
<tr>
<th>Generate (Np) individuals of the initial population pseudo-randomly</th>
</tr>
</thead>
<tbody>
<tr>
<td>while budget condition do</td>
</tr>
<tr>
<td>for (k = 1 : Np) do</td>
</tr>
<tr>
<td>compute (f(x_k))</td>
</tr>
<tr>
<td>end for</td>
</tr>
<tr>
<td>for (k = 1 : Np) do</td>
</tr>
<tr>
<td>{** Mutation **}</td>
</tr>
<tr>
<td>select three individuals (X_{i1}, X_{i2}) and (X_{i3})</td>
</tr>
<tr>
<td>Generate mutant vector (V_k)</td>
</tr>
<tr>
<td>{** Crossover **}</td>
</tr>
<tr>
<td>Apply crossover operation on (V_k) and (X_{i1}) to get (U_k)</td>
</tr>
<tr>
<td>{** Survivor Selection **}</td>
</tr>
<tr>
<td>if (f(U_k) \leq f(x_k)) then</td>
</tr>
<tr>
<td>save index for replacement (x_k = U_k)</td>
</tr>
<tr>
<td>end if</td>
</tr>
<tr>
<td>end for</td>
</tr>
<tr>
<td>perform replacements and update (\mu) and (\sigma) values</td>
</tr>
<tr>
<td>end while</td>
</tr>
</tbody>
</table>

Table III: Ensemble Compact Differential Evolution

<table>
<thead>
<tr>
<th>Generate (Np) individuals of the initial population pseudo-randomly</th>
</tr>
</thead>
<tbody>
<tr>
<td>while budget condition do</td>
</tr>
<tr>
<td>for (k = 1 : Np) do</td>
</tr>
<tr>
<td>compute (f(x_k))</td>
</tr>
<tr>
<td>end for</td>
</tr>
<tr>
<td>for (k = 1 : Np) do</td>
</tr>
<tr>
<td>{** Mutation **}</td>
</tr>
<tr>
<td>select three individuals (X_{e1}, X_{e2}) and (X_{e3})</td>
</tr>
<tr>
<td>Generate mutant vector (V_i)</td>
</tr>
<tr>
<td>{** Crossover **}</td>
</tr>
<tr>
<td>Apply crossover operation on (V_i) and (X_{e1}) to get (U_i)</td>
</tr>
<tr>
<td>{** Survivor Selection **}</td>
</tr>
<tr>
<td>if (f(U_i) \leq f(x_k)) then</td>
</tr>
<tr>
<td>save index for replacement (x_k = U_i)</td>
</tr>
<tr>
<td>end if</td>
</tr>
<tr>
<td>end for</td>
</tr>
<tr>
<td>perform replacements and update (\mu) and (\sigma) values</td>
</tr>
<tr>
<td>if (\text{rand} &lt; \text{migration probability}) then</td>
</tr>
<tr>
<td>if (f(X_{j1}) \leq f(X_{j+1})) then</td>
</tr>
<tr>
<td>(X_{j+1} = X_{j1})</td>
</tr>
<tr>
<td>(f(X_{j+1}) = f(X_{j1}))</td>
</tr>
<tr>
<td>end if</td>
</tr>
<tr>
<td>end if</td>
</tr>
<tr>
<td>end while</td>
</tr>
</tbody>
</table>

III. ENSEMBLE COMPACT DIFFERENTIAL EVOLUTION

ENcDE starts with a \(Nc\) (2 x D)-matrices. Parameter \(Nc\) stands for number of compact units unlike cDE which contains only one distribution. Each matrix \(PV_m\) \((m = 1, 2, \ldots, Nc)\) is called Probability Vector. More specifically, \(PV_m\) is a 2 x D matrix:

\[
PV_{m}^{G} = [\mu^G, \sigma^G]
\]

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 \(G\) indicates the generation count.

In each of the \(Nc\) distributions the mean (\(\mu\)) and standard deviation (\(\sigma\)) values are initialized as described in Section II (b). Then, one individual is sampled from each \(PV_m\). These individuals are indicated as elite \(X_{e}^{w}\) and are the elite individual related to the \(m^\text{th}\) probability vector \(PV_{m}\).

ENcDE contains a pool of mutation strategies, crossover strategies, \(F\) and \(CR\) values. Each of the \(Nc\) units is assigned a mutation and crossover strategy and its associated parameter values randomly picked from the pool. Each unit undergoes mutation, crossover and selection. After selection the respective \(PV\) vector is updated.

In ENcDE, the \(Nc\) units are arranged in a ring topology and communicate by exchanging the elite member. The communication is done based on a probability called migration probability. The algorithm description is presented in Table III.

IV. EXPERIMENTAL RESULTS

The benchmark functions used in this study are taken from CEC 2005 [15]. More specifically, the following test functions have been included:

- \(f_1\) Shifted sphere function.
Crossover Strategies: Binomial and Exponential
Mutation Strategies: Given by equations 3 & 4

Parameter values employed in the study are:
- Population size,
- Compact units, each of them being characterized by a virtual
- \( f \) on the Bounds

Shifted Schwefel’s Problem 1.2
Shifted Rotated High Conditioned Elliptic Function
Shifted Schwefel’s Problem 1.2 with Noise
Shifted Schwefel’s Problem 2.6 with Global Optimum on Bounds
Shifted Rosenbrocks Function
Shifted rotated Griewank’s Function with Bounds
Shifted rotated Ackley’s Function with Global Optimum on the Bounds
Shifted Rastrigin’s Function:

All the test problems considered have dimensionality of 30. The proposed ENcDE has been run with 14 adaptive DE called JADE [16]. In JADE the parameter \( c = 0.1 \).

### Table IV: Summary of Results for benchmark problems

<table>
<thead>
<tr>
<th>Function</th>
<th>30D JADE</th>
<th>30D ENcDE</th>
<th>( t )-test (( p ))</th>
</tr>
</thead>
<tbody>
<tr>
<td>( f_1 )</td>
<td>Mean</td>
<td>Std</td>
<td>Mean</td>
</tr>
<tr>
<td>( f_2 )</td>
<td>8.59E-28</td>
<td>4.19E-28</td>
<td>3.25E-05</td>
</tr>
<tr>
<td>( f_3 )</td>
<td>7.96E-03</td>
<td>3.88E-03</td>
<td>7.56E-06</td>
</tr>
<tr>
<td>( f_4 )</td>
<td>2.45E-02</td>
<td>8.40E-02</td>
<td>3.47E-03</td>
</tr>
<tr>
<td>( f_5 )</td>
<td>7.53E-02</td>
<td>3.68E-02</td>
<td>5.23E-03</td>
</tr>
<tr>
<td>( f_6 )</td>
<td>1.03E+01</td>
<td>2.72E+01</td>
<td>2.16E+01</td>
</tr>
<tr>
<td>( f_7 )</td>
<td>1.56E-02</td>
<td>1.31E-02</td>
<td>1.53E-02</td>
</tr>
<tr>
<td>( f_8 )</td>
<td>2.08E+01</td>
<td>2.46E+01</td>
<td>2.02E+01</td>
</tr>
<tr>
<td>( f_9 )</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>( f_{10} )</td>
<td>2.73E+01</td>
<td>5.70E+00</td>
<td>1.21E+02</td>
</tr>
<tr>
<td>( f_{11} )</td>
<td>2.68E+01</td>
<td>2.03E+00</td>
<td>2.62E+01</td>
</tr>
<tr>
<td>( f_{12} )</td>
<td>4.92E+03</td>
<td>3.97E+03</td>
<td>5.11E+03</td>
</tr>
<tr>
<td>( f_{13} )</td>
<td>1.67E+00</td>
<td>3.04E-02</td>
<td>1.06E+00</td>
</tr>
<tr>
<td>( f_{14} )</td>
<td>1.24E+01</td>
<td>3.27E-01</td>
<td>1.25E+01</td>
</tr>
</tbody>
</table>

To each algorithm is run 25 times for maximum function evaluations of 300000. The mean and standard deviation values are presented in Table IV. To compare the algorithms we applied statistical \( t \)-test with a confidence level of 95%. The results of the \( t \)-test are summarized in the last column of Table IV. For ENcDE vs JADE comparison, -1, 0, 1 indicate that ENcDE is worst than, equal to and better than JADE.

From the results it can be observed that the performance of ENcDE is comparable or better than JADE algorithm on 8 cases. In the remaining 6 cases JADE outperforms ENcDE. The better performance of JADE over ENcDE can be attributed to the diversity maintained by the population in JADE compared to the distributions in ENcDE. However the memory requirements for a run of ENcDE are much less than that of the JADE algorithm.

### V. Conclusions

In this paper we tried to integrate two different ideas, the ensemble strategies and compact DE. Ensemble idea showed an improved performance compared to most of the self-adaptive techniques. cDE is a compact version of DE which can be employed in applications where sufficient computational resources are not available. By integrating the idea of ensemble strategies with cDE, improved performance of cDE can be observed. All in all, this paper can be seen as a preliminary study which combines the potentials of compact computing with ensemble strategies. Preliminary results are encouraging and indicate two future research directions. The first one will consider, in a memetic fashion, the definition of compact local search and their integration within the ensemble structure. The second will investigate multiple population models, i.e., employing not only truncated Gaussian probabilistic models but also other schemes inspired, for example, by Cauchy distribution.

### References


