2014 IEEE Congress on Evolutionary Computation (CEC)
July 6-11, 2014, Beijing, China

Micro-Differential Evolution with Vectorized Random Mutation Factor

Hojjat Salehinejad, *Student Member, IEEE,* Shahryar Rahnamayan, *Senior Member, IEEE,*
Hamid R. Tizhoosh, *Member, IEEE*, and Stephen Y. Chen, *Member, IEEE*

***Abstract*—One of the main disadvantages of population-based evolutionary algorithms (EAs) is their high computational cost due to the nature of evaluation, specially when the population size is large. The micro-algorithms employ a very small number of individuals, which can accelerate the convergence speed of algorithms dramatically, while it highly increases the stagnation risk. One approach to overcome the stagnation problem can be increasing the diversity of the population. To do so, a micro-differential evolution with vectorized random mutation factor (MDEVM) algorithm is proposed in this paper, which utilizes the small size population benefit while preventing stagnation through diversification of the population. The proposed algorithm is tested on the 28 benchmark functions provided at the IEEE congress on evolutionary computation 2013 (CEC-2013). Simulation results on the benchmark functions demonstrate that the proposed algorithm improves the convergence speed of its parent algorithm.**

I. Introduction

EVOLUTIONARY and swarm intelligence algorithms work on a population set. The population set size is one of the most important factors, which crucially affects the algorithm performance [1]. Generally speaking, a population-based algorithm with a large population size has lower risk of premature convergence as well as stagnation. Unfortunately, an algorithm with a large populations size needs more function evaluations and therefore shows a lower convergence velocity [2]. The stagnation is not the same as premature convergence. During stagnation, the population remains unconverged but divert and the optimization process does not progress. A large population size offers a more diversified pool of individuals whose recombination offers higher likelihood to locate the global solution [2]-[4]. Therefore, reducing the population size while raising the diversity of the population is a key point to achieve a faster convergence speed while maintaining a low risk of premature convergence or stagnation.

A population-based algorithms with a small population size is called a micro-algorithm, some times denoted by μ-algorithm [4]. The micro-algorithms due to their low populations size and less hardware demand than their parent algorithms can be utilized to solve many real-time problems such as in wireless communication networks [3], [5], or vehicle navigation systems [6]. Many research works have been conducted to develop efficient micro-algorithms. One of the earliest research works in this direction was a genetic algorithm (GA) with five chromosomes in its population [28]. The strategy in this micro-GA approach is to copy the best found string in the current population to the next generation's population. This work was tested on low-dimensional problems, which resulted in a faster convergence speed compared to the traditional GA [29]. Particle swarm optimization (PSO) was another algorithm tested with the micro concept [7],[30]. By using the micro-PSO algorithm, a parallel master-slave model of cooperative micro-PSO is introduced [7], in which the original search space is decomposed into subspaces with smaller dimensions. Then, five individuals are considered in each subspace to identify suboptimal partial solution components. Its performance was assessed on a set of five widely used test problems with significant improvements in solution quality, compared to the original PSO algorithm [7]. In [30] the conducted simulations are for five high-dimensional benchmark functions. The results demonstrated superior performance of micro-PSO versus the standard PSO.

The micro-algorithms also have been employed to solve multi-objective optimization (MOO) problems. The improved version of nondominated sorting genetic algorithm (NSGA-II) with a specific population initialization strategy are embedded into the standard micro-GA to solve the MOO problems [10]. In [21] a multi-objective micro genetic extreme learning machine (G-ELM) was proposed, which provides the appropriate number of hidden nodes in the machine for solving the problem as well as the tuned values of weights and biases, which minimizes the mean square error (MSE) of results.

The differential evolution (DE) algorithm is one of the state-of-the-art global optimization algorithms, which is popular due to its simplicity and reliability. This algorithm works based on the scaled difference between two individuals of a population's set, where the scaling factor is called the mutation factor. Due to fast implementation as well as reliability and simplicity of the DE algorithm, it has been employed in many science and engineering areas, such as solving large capacitor placement problems [16] and synthesis of spaced antenna arrays [17]. Many research works have been conducted to enhance the DE algorithm, such as opposition-based differential evolution (ODE) [14], enhanced differential evolution using center-based sampling [15], and opposition-based adaptive differential evolution [16]. Some

Hojjat Salehinejad and Shahryar Rahnamayan are with the Department of Electrical, Computer, and Software Engineering, University of Ontario Institute of Technology, 2000 Simcoe Street North, Oshawa, ON L1H 7K4, Canada (emails: {hojjat.salehinejad, shahryar.rahnamayan}@uoit.ca).

Hamid R. Tizhoosh is with the Department of Systems Design Engineering, University of Waterloo, 200 University Avenue West, Waterloo, ON N2L 3G1, Canada (email: tizhoosh@uwaterloo.ca).

Stephen Y. Chen is with the School of Information Technology, York University, Toronto, ON M3J 1P3 Canada (email: sychen@yorku.ca).

978-1-4799-1488-3/14/$31.00 ©2014 IEEE

approaches toward reducing computational cost of DE-based algorithms by reducing the population size have been proposed, [8]-[13]. A gradually reducing population size method is proposed in [8]. This method is examined on 13 benchmark functions, where the results have demonstrated a higher robustness as well as efficiency compared to standard DE (SDE) [8]. In another approach [9], small size cooperative sub-populations are employed to find sub-components of the original problem concurrently. During cooperation of sub-populations, the sub-components are combined to construct the complete solution of the problem. Performance evaluation of this method has been done on high-dimensional instances of five sample test problems with encouraging results reported in [9]. As a micro-DE application, it is employed for evolving an indirect representation of the bin packing problem, where acceptable performance is reported in [11]. The idea of self-adaptive population size was carried out to test absolute encoding and relative encoding methods for DE [12]. The reported simulation results on 20 benchmark problems denote that in terms of the average performance and stability, the self-adaptive population size using relative encoding outperforms the absolute encoding method and the standard DE algorithm [12]. The idea of micro-ODE was proposed and evaluated for an image thresholding case study [13]. Performance of the proposed method was compared with the Kittler algorithm and the standard micro-DE (SMDE). The micro-ODE method has outperformed these algorithms on 16 challenging test images and has demonstrated faster convergence speed due to embedding the opposition-based population initialization scheme [13].

In this paper, an enhanced version of the micro-DE algorithm, i.e., micro-differential evolution using vectorized random mutation factor (MDEVM), is proposed based on vectorization of the random mutation amplification factor in the DE algorithm. In the proposed algorithm, in contrast to the traditional DE algorithm where the mutation factor (i.e. F) is constant, the mutation factor is randomized and vectorized. This technique increases the diversity of the population to avoid stagnation.

In the next section, a brief review of the DE algorithm and its micro scheme is presented. In Section III, diversity enhancement in micro-differential evolution using vectorized random mutation factor is reviewed in detail. The proposed MDEVM algorithm is presented in Section IV and its performance is evaluated in Section V. Finally, the paper is concluded in Section VI.

II. Differential Evolution

Generally speaking, during solving a black-box problem, an optimizer has no knowledge about the shape of the landscape as it tries to find optimal decision variables to minimize/maximize an objective function. The DE algorithm is one of the state-of-the-art evolutionary algorithms, which, similar to other algorithms in its category, starts its search procedure with some uniform random initial vectors and tries to improve them generation by generation toward an optimal solution. The population $\mathbf{P} = \{\mathbf{X}_1, ..., \mathbf{X}_{NP}\}$ consists of NP vectors in generation g, where $\mathbf{X}_i$ is a D-dimensional vector defined as $\mathbf{X}_i = (x_{i,1}, ..., x_{i,D})$. A simple DE algorithm consists of the following three major operations: mutation, crossover, and selection.

Mutation: This step selects three vectors randomly from the population such that $i_1 \neq i_2 \neq i_3 \neq i$, where $i \in \{1, ..., NP\}$ and $NP \geq 4$ for each vector $\mathbf{X}_i$. The mutant vector is calculated as

$$\mathbf{V}_i = \mathbf{X}_{i_1} + F(\mathbf{X}_{i_2} - \mathbf{X}_{i_3}), \tag{1}$$

where the factor $F \in (0, 2]$ is a real constant number, which controls the amplification of the added differential variation of $(\mathbf{X}_{i_2} - \mathbf{X}_{i_3})$. The exploration of DE increases by selecting higher values for F.

Crossover: The crossover operation increases diversity of the population by shuffling the mutant and parent vector as follows:

$$U_{i,d} = \begin{cases} V_{i,d}, & rand_d(0,1) \leq Cr \text{ or } d_{rand} = d \\ x_{i,d}, & otherwise \end{cases}, \tag{2}$$

where $d = 1, ..., D$, $Cr \in [0, 1]$ is the crossover rate parameter, and $rand(a, b)$ generates a random number in the interval $[a, b]$ with a uniform distribution. Therefore, the trial vector $\mathbf{U}_i \; \forall i \in \{1, ..., NP\}$ can be generated:

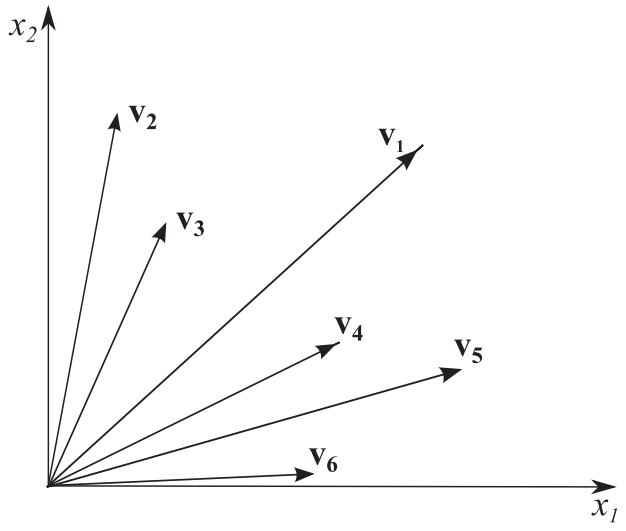
$$\mathbf{U}_i = (U_{i,1}, ..., U_{i,D}). \tag{3}$$

Selection: The $\mathbf{U}_i$ and $\mathbf{X}_i$ vectors are evaluated and compared with respect to their fitness values; the one with better fitness is selected for the next generation.

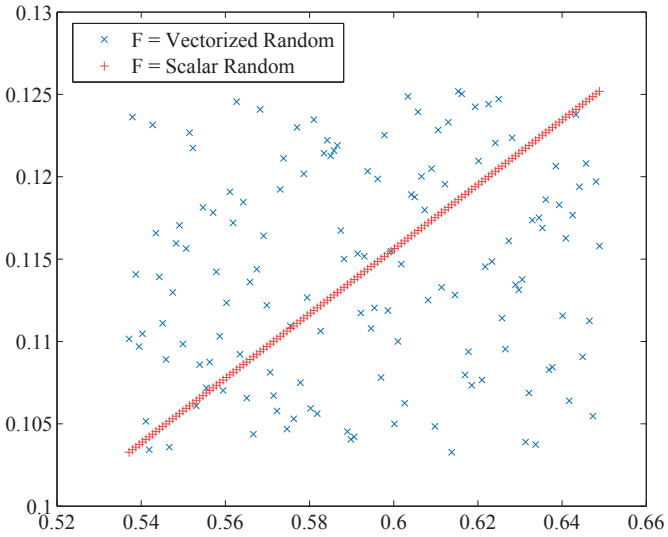
III. Proposed Diversity Enhancement in Micro-Differential Evolution Using Vectorized Random Mutation Factor

The transition from a scalar constant F to a scalar random F to a vectorized random F in DE has an interesting inverse in PSO. In original PSO [22], "velocities were adjusted according to their difference, per dimension, from best locations", and this design guideline is further solidified in the 2007 standard for PSO which states that the attraction vectors are multiplied by the weights ε_1 and ε_2 which "are independent random numbers uniquely generated at every update for each individual dimension $d = 1$ to D" [23]. However, an interesting divergence has occurred to create a version with scalar random weights likely starting with "$rand()$ and $Rand()$ are two random functions in the range [0,1]" in [24] which becomes, for example, "r_1 and r_2 are independent random numbers between 0 and 1" in [25]. As an extreme similar to scalar constant F, there is also "Canonical Deterministic PSO" in which the update equation "system does not contain stochastic factors" [26]. Conversely, since "the above dimension by dimension method is biased", a "geometrical" form of update weights has also been proposed [27]. Overall, the role and effect of update weights in both DE and PSO is an interesting area for continued and coordinated study.

In our proposed algorithm, the population size is considered very small compared to the SDE algorithm. Reducing

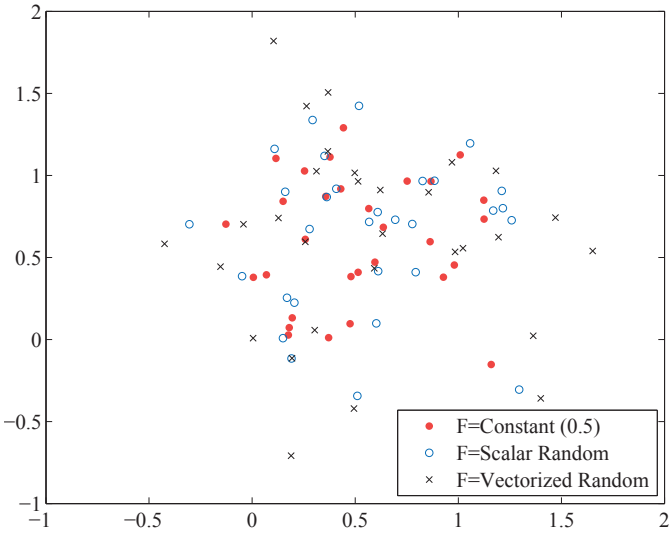


(a) Mutation vector distribution.

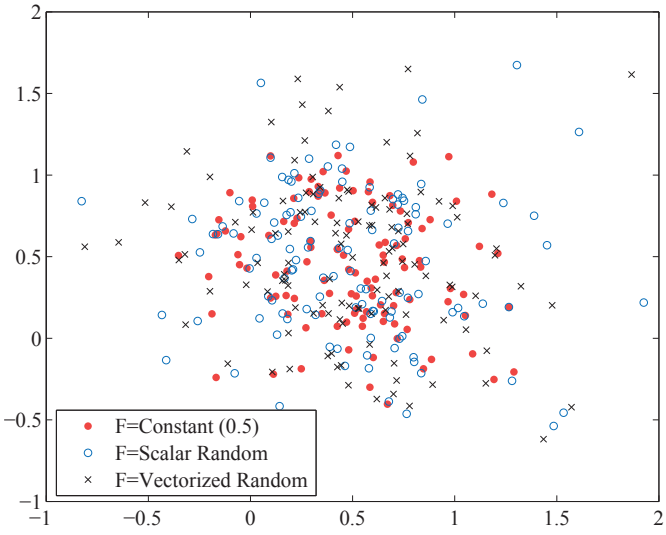


(b) Monte-Carlo simulation.

Fig. 1. Mutation vector diversity for a 2-D individual.



(a) $NP = 4$.



(b) $NP = 5$.

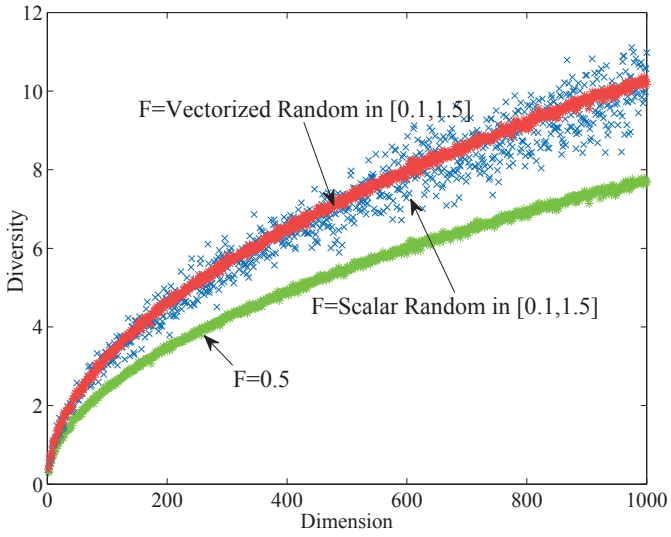
Fig. 2. Population diversity for different mutation factor schemes with populations sizes $NP \in \{4, 5\}$.

the population size results in faster convergence but also a higher risk of stagnation. It is possible to decrease the stagnation risk by increasing the population diversity [2], [29]. The mutation factor F is one of the most significant control parameters for the DE algorithm, which can affect its performance significantly. Therefore, proper selection of F value is critical. In the DE algorithm, F has a constant mutation factor generally set to $F = 0.5$ [2], [14]. This factor can also be selected randomly from the interval $[0.1, 1.5]$ for each individual i in the population vector, $F_i = rand(0.1, 1.5)$, [29]. We call this approach the micro-differential evolution with scalar random mutation factor (MDESM) algorithm if the population size is very small such as $NP \in \{4, 5\}$. In the SMDE algorithm, in order to increase the population diversity, we propose the idea of utilizing a random vector (and not scalar) $\mathbf{F}$ for each individual in the population. This approach is called the MDEVM algorithm. Therefore, the mutation factor can be defined for each individual i as follows:

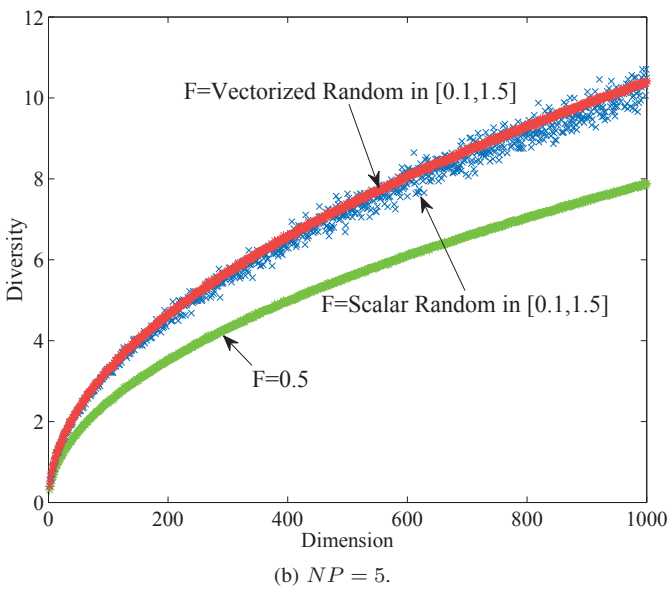
$$\mathbf{F}_i = \{F_{i,1}, ..., F_{i,D}\},\ \forall i \in 1, ..., NP, \tag{4}$$

where $F_{i,1} = rand(0.1, 1.5)$ [29].

In Fig. 1.a, the mutation vector distribution for a 2-D individual is presented. In the case of having a constant mutation factor, diversity of the generated mutation vector is limited to one static point on the direction of vector $\mathbf{V}_1$ (i.e. $F \times \mathbf{V}_1$). By considering an identical random F for all parameters of an individual, the diversity of mutation vector is along the line which is indicated by the vector $\mathbf{V}_1$ (i.e., $F \times \mathbf{V}_1$, F is a uniform random scalar number). Conversely, by randomizing F for each parameter of each individual using a random vector $\mathbf{F}$, its diversity covers the whole plane containing the

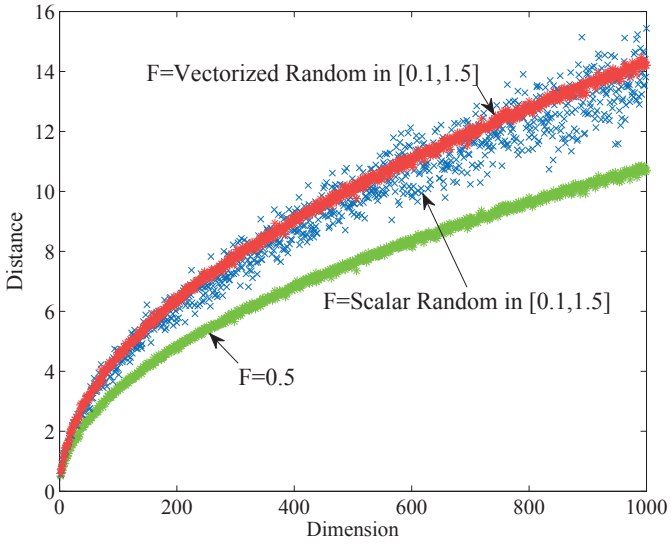


(a) $NP = 4$.

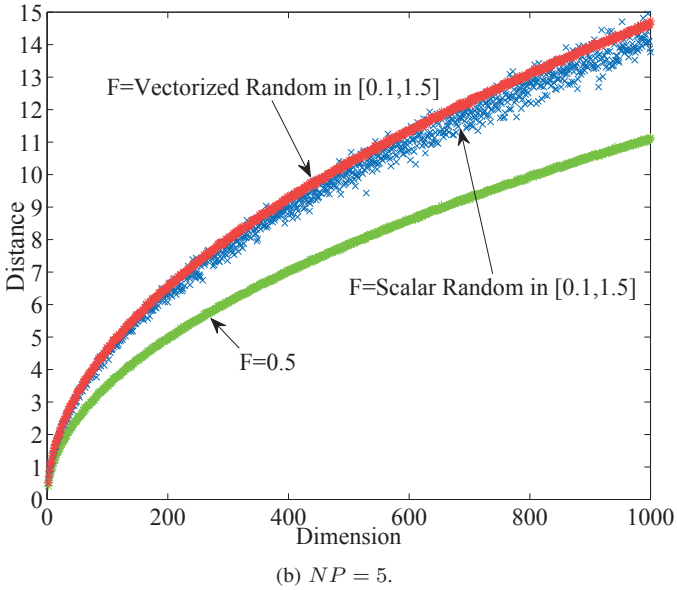


(b) $NP = 5$.

Fig. 3. Boundary coverage calculated based on the average of distances among population centroid and individuals for different mutation factor schemes.



(a) $NP = 4$.



(b) $NP = 5$.

Fig. 4. Population diversity calculated based on the average of distances between individuals pair for different mutation factor schemes.

example vectors $\{\mathbf{V}_2, ..., \mathbf{V}_6\}$. In Fig. 1.b, this difference in covering the search space is investigated by using Monte-Carlo simulation. Since we have considered $NP = 5$ and $r = 3$ randomly selected population vectors, in total, we have $\binom{NP-1}{r} \times r! \times NP = 4 \times 6 \times 5 = 120$ distinct possibilities to generate mutant vectors. The simulation illustrates that the vectorized random **F** supports a higher diversity than the random scalar one (i.e. F_i), where its diversity is limited to the points on a line. In other words, if all variables in the individual vector i are multiplied by a random scalar number, other points on the same direction of the line which is indicated by vector $\mathbf{V}_1$ are generated. In fact, in this scheme F is generating points on the same direction as $\mathbf{V}_1$. If the relationship among the variables (variables' interaction) are linear, the mutation vector is doing fine (which is a very exceptional case, especially during solving real-world problems). However, when F is vectorized randomly, the mutation vector has no restriction to explore any point on the search space with no line restriction which was the case for random scalar F. This discussion is valid for higher dimensions, where the line needs to be replaced with a plane or hyperplane.

The population diversity for all possible mutations of individuals for $NP = 4$ and $NP = 5$ are presented in Fig. 2.a and Fig.2.b, respectively. In order to be fair, for two randomized F schemes (i.e., scalar and vector), the same number of the mutant points (equal to all possible mutations for the F=constant) have been generated. Both

Algorithm 1 Micro-Differential Evolution with Vectorized Mutation (MDEVM)

Procedure MDEVM
$g = 0$
//Initial Population Generation
for $i = 1 \rightarrow NP$ **do**
  for $d = 1 \rightarrow D$ **do**
    $\mathbf{X}_{i,d} = x_d^{min} + rand(0,1) \times (x_d^{max} - x_d^{min})$
  end for
  $\mathbf{P}_i^g = \mathbf{X}_i$
end for
//End of Initial Population Generation
while $(|BFV - VTR| > EVTR$ & $NFC < NFC_{Max})$ **do**
  for $i = 1 \rightarrow NP$ **do**
  //Mutation
    Select three random population vectors from $\mathbf{P}^g$ *where* $(i_1 \neq i_2 \neq i_3 \neq i)$
    for $d = 1 \rightarrow D$ **do**
      $F = rand(0.1, 1.5)$
      $\mathbf{V}_{i,d} = \mathbf{X}_{i_1,d} + F(\mathbf{X}_{i_2,d} - \mathbf{X}_{i_3,d})$
    end for
  //End of Mutation
  //Crossover
    for $d = 1 \rightarrow D$ **do**
      if $rand(0,1) < Cr$ or $d_{rand} = d$ **then**
        $U_{i,d} = V_{i,d}$
      else
        $U_{i,d} = x_{i,d}$
      end if
    end for
  //End of Crossover
  //Selection
    if $f(\mathbf{U}_i) \leq f(\mathbf{X}_i)$ **then**
      $\mathbf{X}'_i = \mathbf{U}_i$
    else
      $\mathbf{X}'_i = \mathbf{X}_i$
    end if
  //End of Selection
  end for
  $\mathbf{X}_i = \mathbf{X}'_i, \forall i \in \{1, ..., NP\}$
  $g = g + 1$
  $\mathbf{P}^g = \{\mathbf{X}_1, ..., \mathbf{X}_{NP}\}$
end while

plots demonstrate that the vectorized random mutation factor supports higher diversity in the population than the scalar random and constant mutation factors.

The population diversity for $NP = 4$ and $NP = 5$ are measured for varying search space dimensions as shown in Fig. 3. The boundary coverage (BC) of populations are measured by averaging the distances among the centroid of population and individuals

$$BC_D = \frac{1}{NP} \sum_{i=1}^{NP} \sqrt{(x_{i,1} - x_1^c)^2 + ... + (x_{i,D} - x_D^c)^2}, \quad (5)$$

where the centroid of the population is $\mathbf{X}^c = (x_1^c, ..., x_D^c)$, computed as

$$x_d^c = \frac{1}{NP} \sum_{i=1}^{NP} x_{i,d}, \ \forall d \in \{1, ..., D\}. \quad (6)$$

As Fig.3.a demonstrates for $NP = 4$, the diversity of vectorized random mutation factor in $[0.1, 1.5]$ is more concentrated, with higher values for the range of dimensions than the scalar random mutation factor in $[0.1, 1.5]$. By considering $NP = 5$ as in Fig. 3.b, all schemes have more robustness than in the $NP = 4$ case study. However, the scalar random mutation factor scheme has lower diversity than the vectorized random mutation factor. The constant mutation factor, i.e. $F = 0.5$, has the least diversity among the other schemes. The population diversity (PD) is computed by averaging the distances between all individual pairs of population

$$PD_D = \frac{\sum_{i=1}^{NP} \sum_{\substack{j=1 \\ i \neq j}}^{NP} \sqrt{(x_{i,1} - x_{j,1})^2 + ... + (x_{i,D} - x_{j,D})^2}}{NP \times (NP - 1)}. \quad (7)$$

This population diversity is presented in Fig. 4 for $NP = 4$ and $NP = 5$, where almost the identical behaviour as shown in Fig. 3 is apparent. Our measurements show that the average of diversity for scalar random F is less than the vectorized approach across the range of dimensions.

IV. Proposed Micro-Differential Evolution Using Vectorized Random Mutation Factor Algorithm

The pseudocode of the proposed MDEVM approach is described in Algorithm 1. After generation of initial population, the mutation vector is computed by using the proposed Eq. (4). Then, the *crossover* and *mutation* procedures are conducted similar to the DE algorithm and the next population is generated. The termination criterion is met when the difference between best fitness value (BFV) and fitness value to reach (VTR) is less than fitness error-value-to-reach ($EVTR$), or the searching procedure exceeds the maximum number of function calls NFC_{Max}, i.e., $NFC \geq NFC_{Max}$. As mentioned, the only difference between DE and MDEVM is in the mutation amplification factor, F; which is a constant number in the DE and a uniform random vector in the proposed MDEVM algorithm.

V. Simulation Results

In this section, the proposed MDEVM algorithm is compared with the SMDE and MDESM algorithms. In the next subsection, the parameter setting and used benchmark functions are described. Then, the comparison strategies and metrics for performance evaluations are presented. Later, the detailed simulations and visualizations are illustrated.

A. Parameter Setting and Benchmark Functions

All the experiments have been conducted on the CEC-2013 testbed [19], which is comprised of 28 benchmark functions and is an improved version of CEC-2005 [20] counterpart with additional test functions and modified formula of the composition functions, oscillations, and symmetric-breaking transforms. This testbed is divided into three categories which are uni-modal functions ($f_1 - f_5$), multi-modal functions ($f_6 - f_{20}$), and composition functions ($f_{21} - f_{28}$) [19]. Parameter setting for all the experiments are presented in

TABLE I
PARAMETER SETTING FOR ALL CONDUCTED EXPERIMENTS

Parameter Name	Parameter Description	Parameter Value
D	Problems Dimension	100
NP	Population Size	5
Cr	Crossover Probability Constant	0.9
NFC_{Max}	Maximum Number of Function Calls	$1e4 \times D$
$EVTR$	Objective Function Error Value to Reach	1e-8
$NRun$	Number of Runs	30

TABLE II
PERFORMANCE COMPARISON OF SMDE, MDESM, AND PROPOSED MDEVM ALGORITHMS ON CEC-2013 BENCHMARK FUNCTIONS. RESULTS OF THE SUCCESSFUL ALGORITHM FOR EACH BENCHMARK FUNCTION IS HIGHLIGHTED IN BOLDFACE.

F	SMDE		MDESM		MDEVM		Best Value Improvement	
	Best	Std	Best	Std	Best	Std	SMDE	MDEVM
1	2.49E+05	3.45E+04	1.03E+05	3.62E+04	**1.31E+04**	1.29E+05	**1.80E+03%**	**6.86E+02%**
2	7.33E+09	4.35E+09	9.08E+08	6.35E+08	**1.57E+08**	1.03E+08	**4.58E+03%**	**4.79E+02%**
3	3.57E+19	3.26E+25	7.73E+14	1.84E+19	**1.74E+11**	2.15E+12	**2.05E+10%**	**4.43E+05%**
4	3.23E+05	2.18E+05	3.76E+05	5.93E+05	**2.49E+05**	6.12E+05	**2.97E+01%**	**5.10E+01%**
5	5.21E+04	5.57E+04	8.36E+04	5.40E+04	**2.33E+04**	4.75E+04	**1.23E+02%**	**2.58E+02%**
6	5.41E+04	2.08E+04	1.42E+04	9.21E+03	**1.21E+03**	2.36E+03	**4.36E+03%**	**1.07E+03%**
7	7.07E+06	1.56E+10	1.38E+06	1.05E+08	**2.03E+04**	2.14E+06	**3.47E+04%**	**6.70E+03%**
8	**2.11E+01**	6.60E-02	2.12E+01	4.00E-02	2.13E+01	2.36E-02	-5.75E-01%	-4.30E-01%
9	**1.49E+02**	4.60E+00	1.57E+02	4.20E+00	1.56E+02	1.91E+00	-4.13E+00%	**1.22E+00%**
10	3.98E+04	9.15E+03	1.17E+04	5.01E+03	**2.94E+03**	1.75E+03	**1.25E+03%**	**3.00E+02%**
11	3.99E+03	8.14E+02	3.51E+03	1.03E+03	**2.39E+03**	8.02E+02	**6.73E+01%**	**4.70E+01%**
12	3.56E+03	7.73E+02	3.57E+03	5.57E+02	**2.06E+03**	6.39E+02	**7.27E+01%**	**7.28E+01%**
13	3.75E+03	9.25E+02	3.41E+03	8.21E+02	**2.44E+03**	4.66E+02	**5.38E+01%**	**3.98E+01%**
14	2.76E+04	1.27E+03	2.04E+04	1.72E+03	**1.28E+04**	1.04E+03	**1.15E+02%**	**5.92E+01%**
15	2.75E+04	1.18E+03	**2.38E+04**	2.30E+03	3.14E+04	7.96E+02	-1.25E+01%	-2.44E+01%
16	**3.25E+00**	2.52E-01	3.47E+00	2.17E-01	3.43E+00	2.31E-01	-5.39E+00%	**9.96E-01%**
17	5.99E+03	8.70E+02	5.48E+03	1.11E+03	**4.65E+03**	4.51E+03	**2.87E+01%**	**1.78E+01%**
18	6.02E+03	1.05E+03	5.63E+03	1.38E+03	**4.70E+03**	4.59E+03	**2.81E+01%**	**1.97E+01%**
19	1.39E+07	2.08E+07	8.64E+06	2.58E+07	**5.81E+05**	3.50E+07	**2.29E+03%**	**1.39E+03%**
20	5.00E+01	0.00E+00	5.00E+01	0.00E+00	**4.95E+01**	2.40E-01	**1.01E+00%**	**1.01E+00%**
21	1.33E+04	4.35E+03	9.39E+03	1.58E+03	**3.26E+03**	1.31E+03	**3.08E+02%**	**1.88E+02%**
22	2.92E+04	1.49E+03	2.27E+04	1.44E+03	**1.35E+04**	1.36E+03	**1.16E+02%**	**6.76E+01%**
23	2.82E+04	1.47E+03	**2.31E+04**	2.28E+03	3.21E+04	6.28E+02	-1.21E+01%	-2.80E+01%
24	7.20E+02	1.13E+03	**5.74E+02**	7.61E+02	5.91E+02	4.56E+02	**2.19E+01%**	-2.81E+00%
25	**5.69E+02**	2.27E+01	5.81E+02	6.14E+00	5.90E+02	5.57E+00	-3.54E+00%	-1.58E+00%
26	6.87E+02	2.49E+03	4.53E+02	4.64E+01	**2.49E+02**	8.39E+01	**1.76E+02%**	**8.17E+01%**
27	4.42E+03	1.31E+03	**4.02E+03**	9.69E+01	4.27E+03	3.54E+01	**3.46E+00%**	-5.96E+00%
28	2.63E+04	1.85E+04	3.08E+04	3.73E+04	**1.94E+04**	1.30E+04	**3.55E+01%**	**5.86E+01%**

TABLE III
SUCCESSES PERCENTAGE OVER BENCHMARK FUNCTIONS FOR SMDE, MDESM, AND MDEVM ALGORITHMS REGARDING THE *Best* METRIC. THE BEST SUCCESS PERCENTAGE IS HIGHLIGHTED IN BOLD.

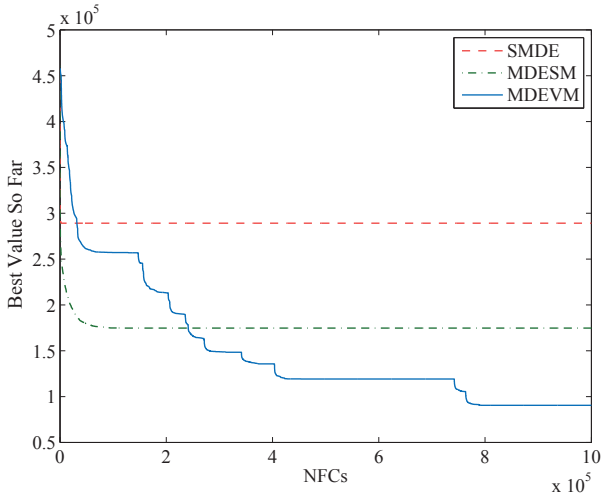
	SMDE	MDESM	MDEVM
$Success\ Percentage$	14.3%	14.3%	**71.4%**

Table I adapted from the literature, [14], [19], [29], unless a change is mentioned. The reported values are averaged for 30 independent runs per function per algorithm to minimize the effect of the stochastic nature of the algorithms on the results.

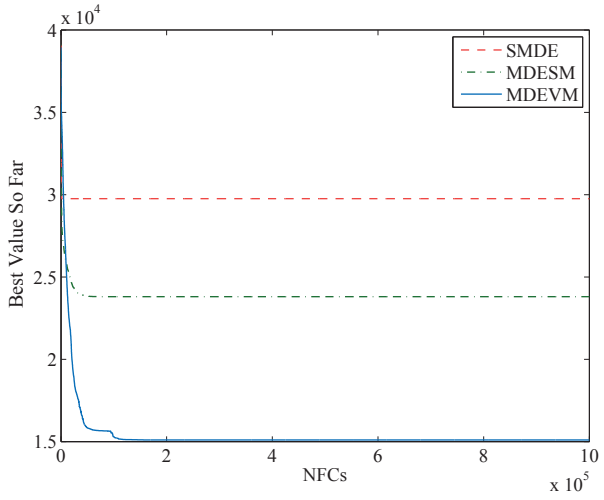
B. Simulation Results Analysis

In Table II, the values (*Best*) and standard deviation (*Std*) of function error values for the SMDE, MDESM, and MDEVM algorithms are presented. In this table, the value that the MDEVM algorithm has improved the *Best* value of the SMDE and MDESM algorithms is denoted under the best value improvement (BVI) as a percentage. For each benchmark function, the best results for each algorithm is highlighted in boldface. In Table III, the total number of times each algorithm has outperformed the other algorithms regarding the *Best* metric over each benchmark function is presented as a percentage. The results clearly confirm that the performance of the proposed MDEVM algorithm is higher than the SMDE and MDESM algorithms' with a success percentage of 71.4%. It is also interesting to mention that for the benchmark functions in which the proposed MDEVM method is outperformed, the *BVI* is not high. In fact, the average for the SMDE *BVI* and SMDE *BVI* in which the proposed MDEVM algorithm is outperformed are $-3.28E+01\%$ and $-6.31E+01\%$, respectively.

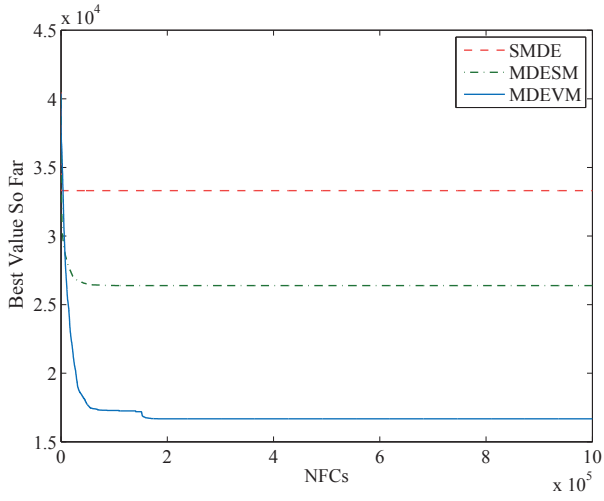
Performance of the SMDE, MDESM, and MDEVM algorithms regarding best value so far versus $NFCs$ are



(a) Benchmark function f_1.



(b) Benchmark function f_{14}.



(c) Benchmark function f_{22}.

Fig. 5. Some sample performance graphs: Best value so far versus number of funcion calls for SMDE, MDESM, and MDEVM algorithms.

presented in Fig. 5 for uni-modal function f_1, multi-modal function f_{14}, and composition function f_{22}. The results demonstrate that the SMDE and MDESM algorithms have stagnated or been trapped in local optimum in early stages while the proposed MDEVM algorithm, thanks to its vectorized random mutation factor, can explore the problem landscape more with lower stagnation or prematurity risk.

VI. Conclusion Remarks

The mutation factor as well as population size play important roles in determining the performance level of the differential evolution (DE) algorithm. A large population size results in a higher diversity of population but it is computationally expensive, although it has a lower chance of stagnation and premature convergence due to its high exploration capability. In the micro-DE algorithm, where the population size is small, convergence to a solution is faster than standard DE algorithm, however with much higher chance of stagnation and premature convergence. In this situation, increasing the diversity of the population while keeping the convergence speed of algorithm can be the key to achieving a high performance for the micro-DE algorithm. The *crossover* module helps to diversify the population but the mutation vector should also provide higher probabilities to do so.

In this paper, we have proposed an enhanced version of the micro-DE algorithm based on the important capability of the mutation factor to provide diversity in the population, i.e. the micro-differential evolution using vectorized random mutation factor (MDEVM) algorithm. In this approach, in contrast to the standard micro DE, the mutation factor F is selected randomly for each variable of each individual in the population. In this case, the population can provide much higher diversity during the search process. In order to analyze the performance of the proposed MDEVM algorithm, we have conducted experiments for different styles of the mutation factor in the micro-DE algorithm, which are constant mutation factor (standard DE-algorithm), scalar random mutation factor (randomized mutation factor for each individual), and the proposed vectorized random mutation factor (MDEVM algorithm). The simulation results clearly demonstrate performance superiority of the MDEVM algorithm.

In order to design fast but reliable optimization algorithms to tackle with real-time applications, mostly in embedded systems, micro-algorithms can be one of the promising approaches. Therefore, more research works are required to be done regarding performance and reliability enhancement of such algorithms.

References

[1] F. Caraffini, F. Neri, and I. Poikolainen, "Micro-differential evolution with extra moves along the axes," in *Proc. IEEE Symposium on Differential Evolution,* 2013, pp. 46-53.

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

[3] H. Salehinejad, S. Talebi, and F. Pouladi, "A metaheuristic approach to spectrum assignment for opportunistic spectrum access," in *Proc. IEEE 17th International Conference on Telecommunications,* 2010, pp. 234-238.

[4] F. Viveros-Jimnez1, E. Mezura-Montes, and A. Gelbukh, "Empirical analysis of a micro-evolutionary algorithm for numerical optimization," *International Journal of Physical Sciences*, vol. 7(8), pp. 1235-1258, 2012.

[5] H. Salehinejad and S. Talebi, "PAPR Reduction of OFDM Signals by Novel Global Harmony Search in PTS Scheme," *International Journal of Digital Multimedia Broadcasting*, vol. 2012, Article ID 940849, 7 pages, 2012.

[6] H. Salehinejad and S. Talebi, "Dynamic Fuzzy Logic-Ant Colony System-Based Route Selection System," *Applied Computational Intelligence and Soft Computing*, vol. 2010, Article ID 428270, 13 pages, 2010.

[7] K. E. Parsopoulos, "Parallel cooperative micro-particle swarm optimization: A masterslave model," *Appl. Soft Comput.*, vol. 12, no. 11, pp. 3552-3579, Nov. 2012.

[8] J. Brest and M. Sepesy Mauec, "Population size reduction for the differential evolution algorithm," *Appl. Intell.*, vol. 29, no. 3, pp. 228-247, Sep. 2007.

[9] K. E. Parsopoulos, "Cooperative micro-differential evolution for high-dimensional problems," in P*roc. 11th Annual conference on Genetic and evolutionary computation*, 2009.

[10] Choo Jun Tan, Chee Peng Lim, and Yu-N Cheah, "A Modified micro Genetic Algorithm for undertaking Multi-Objective Optimization Problems," *Journal of Intelligent and Fuzzy Systems*, vol. 24, no. 3, pp.483-495, 2013.

[11] M. A. Sotelo-figueroa, H. Jos, P. Soberanes, J. M. Carpio, H. J. F. Huacuja, L. C. Reyes, J. Alberto, and S. Alcaraz, "Evolving Bin Packing Heuristic Using Micro-Differential Evolution with Indirect," Recent Advances on Hybrid Intelligent Systems, 2013, pp. 349-359.

[12] N. S. Teng, J. Teo, and M. H. a. Hijazi, "Self-adaptive population sizing for a tune-free differential evolution," *Soft Comput.*, vol. 13, no. 7, pp. 709-724, Jul. 2009.

[13] S. Rahnamayan and H. R. Tizhoosh, "Image thresholding using micro opposition-based Differential Evolution (Micro-ODE)," in *Proc. IEEE Congress on Evolutionary Computation*, 2008, pp. 1409-1416.

[14] S. Rahnamayan, H. R. Tizhoosh, and M. M. A. Salama, "Opposition-Based Differential Evolution," *IEEE Trans. Evol. Comput.*, vol. 12, no. 1, pp. 64-79, Feb. 2008.

[15] A. Esmailzadeh and S. Rahnamayan, "Enhanced Differential Evolution Using Center-Based Sampling," in *Proc. IEEE Congress on Evolutionary Computation*, 2011, pp. 2641-2648.

[16] X. Zhang and S. Y. Yuen, "Opposition-based adaptive differential evolution," in *Proc. IEEE Congress on Evolutionary Computation*, 2012, pp. 1-8.

[17] J.-P. Chiou, C.-F. Chang, and C.-T. Su, "Ant Direction Hybrid Differential Evolution for Solving Large Capacitor Placement Problems," *IEEE Trans. Power Syst.*, vol. 19, no. 4, pp. 1794-1800, Nov. 2004.

[18] D. G. Kurup, M. Himdi, and a. Rydberg, "Synthesis of uniform amplitude unequally spaced antenna arrays using the differential evolution algorithm," *IEEE Trans. Antennas Propag.*, vol. 51, no. 9, pp. 2210-2217, Sep. 2003.

[19] J. J. Liang, B. Y. Qu, P. N. Suganthan, and A. G. Hernndez-daz, "Problem Definitions and Evaluation Criteria for the CEC 2013 Special Session on Real-Parameter Optimization," no. January, 2013.

[20] 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 Tech. Univ., Singapore and KanGAL, Kanpur Genetic Algorithms Lab., IIT, Kanpur, India, Tech. Rep., No.2005005, May 2005.

[21] D. Lahoz, B. Lacruz, and P. M. Mateo, "A multi-objective micro genetic ELM algorithm," *Neurocomputing* vol. 111, pp. 90-103, 2013.

[22] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," in *Proc. IEEE International Conference on Neural Networks*, 1995, pp. 1942-1948.

[23] D. Bratton and J. Kennedy, "Defining a standard for particle swarm optimization," in *Proc. IEEE Swarm Intelligence Symposium*, 2007, pp. 120-127.

[24] Y. Shi, and R. C. Eberhart, "A Modified Particle Swarm Optimizer," in *Proc. IEEE World Congress on Computational Intelligence*, 1998, pp. 69-73.

[25] G. Venter and J. Sobieszczanski-Sobieski, "Particle Swarm Optimization," *AIAA Journal*, vol. 41, pp. 1583-1589, Aug 2003.

[26] K. Jinno and T. Shindo, "Analysis of dynamical characteristic of canonical deterministic PSO," in *Proc. IEEE World Congress on Computational Intelligence*, 2010, pp. 1105-1110.

[27] M. Clerc, "Standard Particle Swarm Optimisation," Particle Swarm Central, Tech. Rep., 2012.

[28] K. Krishnakumar, "micro-genetic algorithms for stationary and non-stationary function optimization," *Intell. Control Adapt. Syst.*, vol. 1196, pp. 289-296, 1989.

[29] M. Olguin-Carbajal, E. Alba, and J. Arellano-verdejo, "Micro-Differential Evolution with Local Search for High Dimensional Problems," in *Proc. IEEE Congress on Evolutionary Computation*, 2013, pp. 48-54.

[30] T. Huang and A. S. Mohan, "Micro-particle swarm optimizer for solving high dimensional optimization problems (PSO for high dimensional optimization problems)," *Appl. Math. Comput.*, vol. 181, no. 2, pp. 1148-1154, Oct. 2006.