# Balancing the exploration and exploitation capabilities of the Differential Evolution Algorithm

M.G. Epitropakis, V.P. Plagianakos, and M.N. Vrahatis

*Abstract*— **The hybridization and composition of different Evolutionary Algorithms to improve the quality of the solutions and to accelerate execution is a common research practice. In this paper we propose a hybrid approach that combines Differential Evolution mutation operators in an attempt to balance their exploration and exploitation capabilities. Additionally, a self-balancing hybrid mutation operator is presented, which favors the exploration of the search space during the first phase of the optimization, while later opts for the exploitation to aid convergence to the optimum. Extensive experimental results indicate that the proposed approaches effectively enhance DE's ability to accurately locate solutions in the search space.**

## I. Introduction

Evolutionary algorithms (EAs) are nature inspired problem solving optimization algorithms. The broad class of EAs has demonstrated numerous methods that have been effectively and successfully applied to many difficult real-life optimization tasks [1]. All these algorithms share the common conceptual base of simulating the evolution of the individuals that form the population using a predefined set of operators. Commonly two kinds of operators are used: the *selection* and the *search* operators. The most widely used search operators are *mutation* and *recombination*.

For the rest of the paper we consider the minimization problem of finding global minimizers of a continuous nonlinear, (possibly) nondifferentiable, multimodal objective function $f$. More specifically, our goal is to locate *global minimizers* $x_t^*$ of the real–valued objective function $f\colon \mathcal{E} \to \mathbb{R}$:

$$f(x_t^*) \leqslant f(x), \quad \forall\, x \in \mathcal{E},$$

where $t = 1, 2, \ldots,$ and the compact set $\mathcal{E} \subseteq \mathbb{R}^n$ is a $n$–dimensional scaled translation of the unit hypercube.

We attempt to tackle the above minimization problem using the Differential Evolution (DE) Algorithm [2], [3]. DE is one of the most promising and widely used EA, and is capable of handling non-differentiable, nonlinear and multimodal objective functions. It has been designed as a stochastic parallel direct search method and typically requires few, easily chosen, control parameters.

Experimental results have shown that DE has good convergence properties and outperforms other well known evolutionary algorithms [2], [4], [5], [6]. However, not all of the DE mutation operators are equally efficient. In this paper we demonstrate that some of the DE mutation operators favor the exploration of the search space, while some other operators favor its fast exploitation. The explorative mutation operators have a greater possibility of locating the minima of the objective function, but generally need more iterations (generations). On the other hand, the exploitive mutation operators rapidly converge to a minimum of the objective function. In this case exists the risk of premature convergence to a suboptimal solution.

To combine the exploration and exploitation capabilities of DE, we propose a new hybrid scheme that utilizes an explorative and an exploitive mutation operator, in an attempt to balance their effects. Additionally, a self-balancing hybrid mutation operator is presented. This mutation operator favors the exploration, during the first phase of optimization, in order to locate the most promising regions of the search space, while later applies almost exclusively the exploitive mutation operator to aid convergence to the optimum. Extensive experimental results indicate that the proposed approaches are promising.

The rest of the paper is organized as follows. In Section II the DE algorithm is briefly described, while in Section III its exploration and exploitation capabilities are analyzed. In Section IV we propose a hybrid Differential Evolution Algorithm, while in Section V, we present a self-balancing mutation scheme. Section VI is devoted to the presentation and the discussion of the experimental results. The paper ends with concluding remarks and some pointers for future work.

## II. The Differential Evolution Algorithm

Differential Evolution [2] has been designed as a stochastic parallel direct search method, which utilizes concepts borrowed from the broad class of EAs. The method typically requires few, easily chosen, control parameters. Experimental results have shown that DE has good convergence properties and outperforms other well known EAs [2], [4], [7].

More specifically, DE is a population–based stochastic algorithm that exploits a population of potential solutions, *individuals*, to effectively probe the search space. The population of individuals is randomly initialized in the optimization domain with *NP*, $n$–dimensional, vectors following a uniform probability distribution. Individuals evolve over successive iterations to explore the search space and locate the minima of the objective function. Throughout the execution process, the population size, *NP*, is fixed. At each iteration, called *generation*, new vectors are derived by the combination of

All the authors are with Computational Intelligence Laboratory (CI Lab), Department of Mathematics, University of Patras Artificial Intelligence Research Center (UPAIRC), University of Patras, GR–26110 Patras, Greece. (email: {mikeagn, vpp, vrahatis}@math.upatras.gr).

This work was partially supported by an "Empirikion Foundation" award that helped the acquisition and the implementation of the distributed computer cluster.

randomly chosen vectors from the current population. This operation in our context can be referred to as *mutation*, while the outcoming vectors as *mutant individuals*. Each mutant individual is then mixed with another, predetermined, vector – the *target* vector – through an operation called *recombination*. This operation yields the so–called *trial* vector. Finally, the trial vector undergoes the *selection* operator, according to which it is accepted as a member of the population of the next generation only if it yields a reduction in the value of the objective function $f$ relative to that of the target vector. Otherwise, target vector is retained in the next generation.

The search operators efficiently shuffle information among the individuals, enabling the search for an optimum to focus on the most promising regions of the solution space. Next, we briefly describe the search operators that were considered in this paper.

### A. Original Mutation Operators

Here we describe the original mutation operators proposed in [2]. Specifically, for each individual $x_g^i$, $i = 1, 2, \ldots,$ *NP*, where $g$ denotes the current generation, the mutant individual $v_{m,g+1}^i$ is generated according to one of the following equations:

$$v_{1,g+1}^i = x_g^{\text{best}} + F(x_g^{r1} - x_g^{r2}), \quad (1)$$
$$v_{2,g+1}^i = x_g^{r1} + F(x_g^{r2} - x_g^{r3}), \quad (2)$$
$$v_{3,g+1}^i = x_g^i + F(x_g^{\text{best}} - x_g^i) + F(x_g^{r1} - x_g^{r2}), \quad (3)$$
$$v_{4,g+1}^i = x_g^{\text{best}} + F(x_g^{r1} - x_g^{r2}) + F(x_g^{r3} - x_g^{r4}), \quad (4)$$
$$v_{5,g+1}^i = x_g^{r1} + F(x_g^{r2} - x_g^{r3}) + F(x_g^{r4} - x_g^{r5}), \quad (5)$$

where $m = 1, 2, \ldots, 5$ denotes the mutation operator applied, $x_g^{\text{best}}$ is the best member of the previous generation, $r_1, r_2, r_3, r_4, r_5 \in \{1, 2, \ldots, i-1, i+1, \ldots, NP\}$, are random integers mutually different and not equal to the running index $i$, and $F > 0$ is a real parameter, called *mutation constant*. The *mutation constant*, controls the amplification of the difference between two individuals, and is used to prevent the risk of stagnation, of the search process.

Trying to rationalize the above equations, we observe that Eq. (2) is similar to the crossover operator employed by some Genetic Algorithms; while Eq. (1) is derived from Eq. (2), by substituting the best member of the previous generation, $x_g^{\text{best}}$, for the random individual $x_g^{r1}$. Eqs. (3), (4) and (5) are modifications obtained by the combination of Eqs (1) and (2). It is clear that more mutation operators can be generated using the above ones as building blocks, such as the trigonometric mutation operator [8] and recently proposed genetically programmed mutation operators [9].

For the rest of the paper, we call $\text{DE}_1$, $\text{DE}_2$, …, $\text{DE}_5$ the DE algorithm that uses Eq. (1), Eq. (2), …, Eq. (5) as the mutation operator, respectively.

### B. Recombination and Selection Operators

Having performed mutation, the recombination operator is subsequently applied to further increase the diversity of the population. To this end, the mutant individuals are combined with other predetermined individuals, called the target individuals. Specifically, for each component $l$ ($l = 1, 2, \ldots, n$) of the mutant individual $v_{m,g+1}^i$, we randomly choose a real number $r$ in the interval $[0, 1]$. Then, we compare this number with the *recombination constant*, $CR$. If $r \leqslant CR$, then we select, as the $l$–th component of the trial individual $u_{g+1}^i$, the $l$–th component of the mutant individual $v_{m,g+1}^i$. Otherwise, the $l$–th component of the target vector $x_g^i$ becomes the $l$–th component of the trial vector. This operation yields the trial individual.

Finally, the trial individual is accepted for the next generation only if it reduces the value of the objective function (selection operator).

## III. DE Exploration vs. Exploitation

An issue when applying EAs is to determine a set of control parameters that balances the exploration and the exploitation capabilities of the given algorithm. There is always a trade off between the efficient exploration of the search space and its effective exploitation. In extreme cases, inadequate choice of the parameter values can hinder the algorithm's ability to locate the optimum. For example, if the mutation rate is too high, much of the space will be explored, but there is a high probability of losing promising solutions; the algorithm has difficulty to converge to an optimum due to insufficient exploitation.

Recently, the exploration and exploitation capabilities of different mutation strategies were studied. Following the methodology presented in [10], we show that not all DE search operators have the same impact on the exploration of the search space. Thus, the choice of the most efficient mutation operator and/or optimal parameter values can be troublesome. To illustrate this we utilize the following simple multimodal 2–dimensional function:

$$f(x_1, x_2) = \sin(x_1)^2 + \sin(x_2)^2,$$

where $(x_1, x_2) \in \mathbb{R}^2$. This function has an infinite number of global minimizers in $\mathbb{R}^2$, with function value equal to zero, at the points $(\kappa\pi, \lambda\pi)$, where $\kappa, \lambda \in \mathbb{Z}$. Restricted in the hypercube $[-5, 5]^2$ the function $f$ has 9 global minimizers. In Figure 1 a surface plot of the function $f$ is exhibited.

The original DE variants described above are applied to compute a global minimizer of the objective function $f$. Experimental results indicate that $\text{DE}_1$ exhibits very fast convergence to one of the global minima of $f$. On the contrary, $\text{DE}_2$ explores a large portion of the search space before converging to a solution. This fact is illustrated in Figures 2 and 3, where (for visualization purposes) a population consisting of 1000 individuals is plotted after 1, 5, 10, 20 generations of $\text{DE}_1$ and $\text{DE}_2$, respectively.

A closer look at Equations (1) and (2) reveals that $\text{DE}_1$ uses the best individual as a starting point for the computation of the mutant vector, thus constantly pushing the population closer to the location of the best computed point. On the other hand, since $\text{DE}_2$ utilizes three randomly chosen individuals for the computation of the mutant one, its exploration

[redacted]

mutation op… of the search space (a… brid approach that combines an ex… mutation operator, in an attempt to balance… new operator is a linear combination of an explorative a… exploitive operator. More specifically, the mutant individual $w^i_{g+1}$ is generated using the following equation:

$$w^i_{g+1} = \xi \cdot (v^i_{a,g+1}) + (1-\xi) \cdot (v^i_{b,g+1}), \qquad (6)$$

where $\xi \in [0,1]$ determines the influence of the explorative over the exploitive mutation operator, $a$ denotes the explorative operator ($v^i_{a,g+1}$), while $b$ denotes the exploitive mutation operator. For $\xi = 1$, Eq. (6) is equivalent to the explorative operator. Similarly, for $\xi = 0$, Eq. (6) is equal to the exploitive operator. For all the intermediate values of $\xi$, the resulting operator combines the exploration and the exploitation operators. Notice that the above combination has shown promising results to other stochastic optimization methods, such as the Particle Swarm Optimization (PSO) algorithm, resulting the Unified PSO (UPSO) [11], [12], [13], [14], and a UPSO-based variant of DE that utilizes the concept of the local neighborhood of each population vector [15].

The above approach creates new classes of DE algorithms that depend on the selection of the explorative and exploitive mutation operators. For example, one can choose $\mathrm{DE}_2$ or

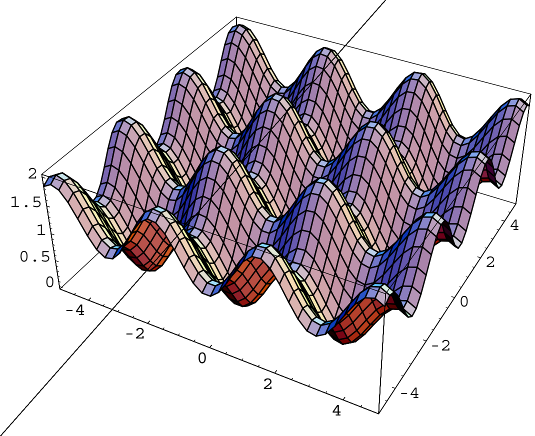

Fig. 1. 3–D Plot of the $\sin(x_1)^2 + \sin(x_2)^2$ function

DE

is

$\xi$

$\xi$

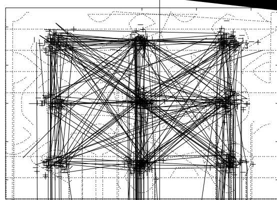

Fig. 3. $\mathrm{DE}_2$

$\xi$. However, this value of $\xi$ … DE

$\mathrm{DE}_5$ for the explorative operator, and $_1$ or $_4$ for the exploitive mutation operator. As a rule of thumb, mutation operators that are based on the best individual of the current population perform better exploitation. On the other hand, operators that combine randomly selected individuals to compute the mutant one, are better suited for the exploration of the search space.

Extensive experimental results indicate that the hybrid mutation operator usually outperforms the other DE mutation operators, for some value of $\xi$

problem dependent.

## V. A S ELF-BALANCING MUTATION SCHEME

To alleviate the problem of selecting a proper value of $\xi$

$\xi$

for each problem, we propose a self-balancing scheme. This is similar to the temperature decrease of the Simulated Annealing method [10], [17]. More specifically, the parameter $\xi$ is computed by one the following equations:

(8)

(7)

$\rho) \cdot 2$

$_1$ $\rho \cdot 2(1 + \quad ^{-c^g},$

$_2$ $\xi = \quad ^{-c^g},$

$_3$ $= \dfrac{_1 + \xi_2}{2},$ (9)

where $g$ is the current generation, $\rho \in [0, 1]$ is a random number from the uniform distribution, and $c$ is the noise decay constant. In our experiments $c$ had a value equal to the one tenth of the maximum allowed number of generations. Furthermore, to restrict the values of $\xi$ in the range $[0, 1]$, if $\xi > 1$ we assign $\xi = 1$; similarly if $\xi < 0$ we assign $\xi = 0$.

The values of $\xi$ computed using Eq. (7) initially favor exploration. In order to favor exploitation during the first phase of the optimization process, we compute the value of $\xi$ using Eq. (8). Finally, Eq. (9) is the average of the two previous cases and balances their effects. In Figure 4 we plot the values of $\xi_1$, $\xi_2$, and $\xi_3$ to exhibit their behavior (the maximum allowed generation number used to plot this figure is 1000). Notice that the values of $\xi$ approach zero, when the number of generations exceeds the half of the maximum allowed generation number. During that phase of optimization only the exploitation operator is used in an attempt to help the DE algorithm to converge.

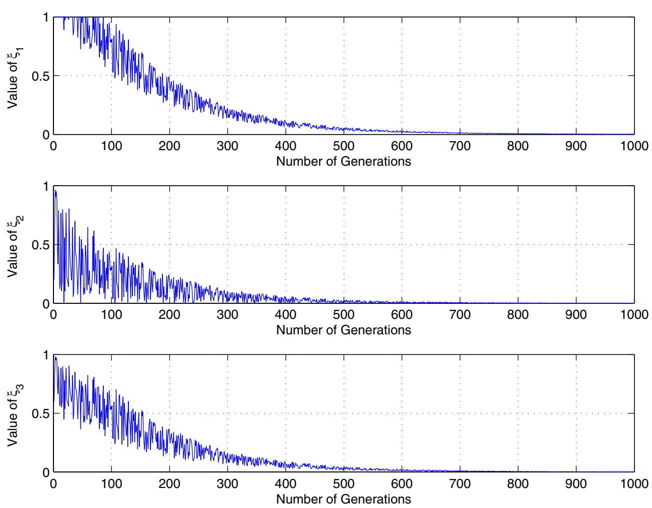


Fig. 4. Plot of the values of $\xi_1$ (Top), $\xi_2$ (Middle), and $\xi_3$ (Bottom)

## VI. Experimental Results

We implemented and tested the proposed hybrid DE algorithm and the self-balancing mutation scheme on a large number of optimization benchmarks. In this study we report experimental results from eight well–known minimization test functions.

The computational experiments were performed utilizing a DE interface developed in C++, using GNU compiler collection (gcc) version 3.4.6 on a Debian GNU Linux operating system. For each test function and each mutation operator, we have conducted 1000 independent runs and have used the fixed values of $F = 0.5$ and $CR = 0.7$ as the DE mutation and crossover constants respectively. Furthermore, in Table I the parameter setup used in the numerical experiments conducted is summarized. Specifically, $D$ denotes the dimensionality of the problem, *NP* stands for the size of the population used for each function, while $MaxGen$ is the maximum number of generations allowed. The noise decay constant of the self-balancing scheme had the value $c = 10/MaxGen$.

| No | Test function | $D$ | *NP* | $MaxGen$ |
|---|---|---|---|---|
| 1 | Sphere function | 5 | 30 | 1000 |
| 2 | Rosenbrock's saddle | 2 | 30 | 1000 |
| 3 | Step function | 5 | 20 | 1000 |
| 4 | Quartic function | 30 | 100 | 2000 |
| 5 | Shekel's foxholes | 2 | 30 | 1000 |
| 6 | Corana's parabola | 4 | 15 | 2000 |
| 7 | Griewangk's function | 10 | 50 | 10000 |
| 8 | Levy No.5 function | 2 | 40 | 1000 |

TABLE I
PARAMETER SETUP VALUES

Next, we will briefly report the benchmark optimization functions used along with their global minima and minimizers in the search space.

### A. Test Functions

More information about the eight test functions selected, appear in [2], [18], [19], [20], [21], [22].

*1) Sphere:*

$$f_1(x) = \sum_{j=1}^{5} x_j^2, \qquad x_j \in [-5.12, 5.12]. \qquad (10)$$

The sphere test function is a considered to be a simple minimization problem. The minimum is $f_1^*(0, \ldots, 0) = 0$.

*2) Rosenbrock's Saddle:*

$$\begin{aligned} f_2(x) &= 100 \cdot (x_1^2 - x_2)^2 + (1 - x_1)^2, \\ & \quad x_j \in [-2.048, 2.048]. \end{aligned} \qquad (11)$$

This is a two–dimensional test function, which is known to be relatively difficult to minimize. The minimum is $f_2^*(1, 1) = 0$.

*3) Step Function:*

$$f_3(x) = 30 + \sum_{j=1}^{5} \lfloor x_j \rfloor, \quad x_j \in [-5.12, 5.12]. \qquad (12)$$

The minimum of this function is $f_3^*(-5-\xi, \ldots, -5-\xi) = 0$, where $\xi \in [0, 0.12]$. This function exhibits many flat regions that can cause search stagnation.

*4) Quartic Function:*

$$f_4(x) = \sum_{j=1}^{30} \left( j \cdot x_j^4 + \eta \right), \qquad (13)$$

where $x_j \in [-1.28, 1.28]$. This is test function is designed to evaluate the behavior of minimization algorithms in the presence of noise. To this end, $\eta$ is a random variable following the uniform distribution in the range $[0, 1]$. The inclusion of $\eta$ makes $f_4$ more difficult to optimize. The functional minimum of the function is $f_4^*(0, \ldots, 0) \leqslant 30 \cdot E[\eta] = 15$, where $E[\eta]$ is the expectation of $\eta$.

*5) Shekel's Foxholes:*

$$f_5(x) = \frac{1}{0.002 + \psi_1(x)},\ x_j \in [-65.536, 65.536], \tag{14}$$

where, $\psi_1(x) = \sum_{i=0}^{24} \frac{1}{1+i+\sum_{j=1}^{2}(x_j - a_{ij})^6}$. The parameters for this function are:

$$\begin{aligned} a_{i1} &= \{-32, -16, 0, 16, 32\},\ \text{where} \\ & \quad i = \{0, 1, 2, 3, 4\} \text{ and } a_{i1} = a_{i \bmod 5, 1} \\ a_{i2} &= \{-32, -16, 0, 16, 32\}, \text{where} \\ & \quad i = \{0, 5, 10, 15, 20\} \text{ and} \\ & \quad a_{i2} = a_{i+k,2},\ k = \{1, 2, 3, 4\}. \end{aligned}$$

The global minimum of $f_5^*(-32, -32) = 0.998004$.

*6) Corana Parabola:*

$$f_6(x) = \sum_{j=1}^{4} \begin{cases} \psi_2(x_j), \text{ if } |x_j - z_j| < 0.05, \\ \psi_3(x_j), \text{ otherwise.} \end{cases} \tag{15}$$

where $\psi_2(x_j) = 0.15\,(z_j - 0.05\text{sign}(z_j))^2\, d_j$, $\psi_3(x_j) = d_j x_j^2$, $z_j = \lfloor 5|x_j| + 0.49999 \rfloor \text{sign}(x_j) 0.2$ and $d_j = \{1, 1000, 10, 100\}$. The function is characterized by a multitude of local minima, increasing in depth as one moves closer to the origin. The global minimum of the function is $f_6^*(x) = 0$, for $x_j^* \in (-0.05, 0.05)$.

*7) Griewangk's Function:*

$$\begin{aligned} f_7(x) &= \sum_{j=1}^{10} \frac{x_j^2}{4000} - \prod_{j=1}^{10} \cos\left(\frac{x_j}{\sqrt{j}}\right) + 1, \\ & \quad x_j \in [-400, 400]. \end{aligned} \tag{16}$$

This test function is riddled with local minima. The global minimum of the function is $f_7^*(0, \dots, 0) = 0$.

*8) Levy No.5 Function:*

$$f_8(x) = \sigma_1 \sigma_2 + (x_1 + 1.42513)^2 + (x_2 + 0.80032)^2, \tag{17}$$

where $x_i \in [-10, 10], i = 1, 2$, and $\sigma_1$ and $\sigma_2$ are given by:

$$\sigma_1 = \sum_{i=1}^{5} \Big[ i\ \cos\big((i-1)x_1 + i\big) \Big],$$

$$\sigma_2 = \sum_{j=1}^{5} \Big[ j\ \cos\big((j+1)x_2 + j\big) \Big].$$

There exist about 760 local minima and one global minimum with function value $f_8^*(x) = -176.1375$, located at $x^* = (1.3068, 1.4248)$. The large number of local optimizers makes it difficult for any method to locate the global minimizer.

### *B. Presentation of the Results*

We have conducted two independent sets of experiments. During the first set we tested the hybrid mutation operators. Next, the self-balancing scheme was extensively studied.

To test the proposed hybrid mutation operator two different classes of methods were studied. More specifically, we used $\text{DE}_2$ as the explorative and $\text{DE}_1$ as the exploitive component ($v^i_{a,g+1}$ and $v^i_{b,g+1}$, respectively) of the new hybrid mutation operator. We call this class of DE algorithms $\text{DE}_{2,1}$ and we conducted extensive experiments for different values of $\xi$. Similarly, we define as $\text{DE}_{5,4}$ the hybrid mutation operator that incorporates $\text{DE}_5$ and $\text{DE}_4$.

| Mutation Strategy | $\xi$ | Min | Mean | Max | St.D. | Success | Total Generations | EFE |
|---|---|---|---|---|---|---|---|---|
| $\text{DE}_1$ | - | 48 | 74.65 | 184 | 14.92 | 61.3 | 819760 | 18.27 |
| $\text{DE}_2$ | - | 81 | 139.57 | 194 | 13.83 | 98.8 | 161899 | 21.19 |
| $\text{DE}_3$ | - | 83 | 119.62 | 286 | 15.93 | 95.3 | 207996 | 18.83 |
| $\text{DE}_4$ | - | 85 | 120.61 | 172 | 12.54 | 96.5 | 186388 | 18.75 |
| $\text{DE}_5$ | - | 142 | 197.89 | 262 | 18.61 | 100 | 197891 | 29.68 |
| $\text{DE}_{2,1}$ | **1.0** | 85 | 138.59 | 228 | 13.99 | 99.6 | **146036** | 20.87 |
| | **0.9** | 71 | 114.67 | 242 | 12.91 | 97.9 | **154262** | **17.57** |
| | 0.8 | 67 | 96.11 | 170 | 11.29 | 91.1 | 265557 | **15.83** |
| | 0.7 | 54 | 82.15 | 146 | 10.83 | 68.8 | 680518 | **17.91** |
| | 0.6 | 46 | 70.94 | 170 | 10.93 | 44.9 | 1133853 | 23.7 |
| | 0.5 | 43 | 64.33 | 132 | 11.76 | 28.9 | 1440590 | 33.39 |
| | 0.4 | 45 | 63.02 | 235 | 18.81 | 19.7 | 1618414 | 47.98 |
| | 0.3 | 40 | 60.48 | 195 | 15.45 | 21 | 1592700 | 43.2 |
| | 0.2 | 42 | 65.82 | 221 | 19.54 | 29.7 | 1425548 | 33.24 |
| | 0.1 | 42 | 70.32 | 186 | 17.48 | 48.8 | 1058316 | 21.61 |
| | 0.0 | 46 | 74.15 | 148 | 13.65 | 67.1 | 707753 | **16.58** |
| $\text{DE}_{5,4}$ | 1.0 | 141 | 197.34 | 306 | 19.44 | 100 | 197342 | 29.6 |
| | 0.9 | 101 | 160.52 | 208 | 15.37 | 99.8 | 164194 | 24.13 |
| | **0.8** | 91 | 131.07 | 167 | 12.17 | 99.6 | **138547** | 19.74 |
| | **0.7** | 74 | 108.93 | 312 | 13.39 | 99.3 | **122170** | **16.46** |
| | 0.6 | 67 | 94.93 | 164 | 10.95 | 96 | 171130 | **14.83** |
| | 0.5 | 57 | 86 | 236 | 11.86 | 91.6 | 246775 | **14.08** |
| | 0.4 | 52 | 83.23 | 197 | 13.41 | 86.9 | 334331 | **14.37** |
| | 0.3 | 57 | 83.84 | 212 | 11.72 | 89.6 | 283118 | **14.04** |
| | 0.2 | 58 | 89.2 | 150 | 10.61 | 90.3 | 274545 | **14.82** |
| | 0.1 | 61 | 100.63 | 158 | 11.81 | 95.2 | 191802 | **15.86** |
| | 0.0 | 79 | 120.59 | 191 | 13.35 | 97 | 176970 | 18.65 |

TABLE II

COMPARATIVE RESULTS FOR THE CORANA TEST FUNCTION, FOR DIFFERENT VALUES OF $\xi$

We performed 1000 independent runs for each algorithm and each problem. Here, due to space limitations we report results from three test problems. Tables II, III, and IV exhibit the results for the $\text{DE}_{2,1}$ and $\text{DE}_{5,4}$ algorithms on the Corana, Quartic, and Levy No.5 test functions respectively. The following notation is used in the Tables: $Min$ indicates the minimum number of generations for the experiments that reached a solution; $Max$ is the maximum number of generations; $Mean$ is the average generation number and $St.D.$ is the standard deviation; $Success$ is percentage of experiments that reached a solution. Next, $TotalGenerations$ indicates the total number of generation for all 1000 experiments, including the generations when an algorithm fails to locate an optimum of the objective function. Note that this number depends on the maximum number of generations allowed for each experiment. Finally, the last column, presents the expected number of function evaluations (EFE) [23], which is defined as:

$$\text{EFE} = \frac{(NP) \cdot (Mean)}{(Success)}.$$

The experimental results on the 8 test functions indicate that the hybrid mutation operators outperform the other DE

| Mutation Strategy | $\xi$ | Min | Mean | Max | St.D. | Success | Total Generations | EFE |
|---|---|---|---|---|---|---|---|---|
| $\mathrm{DE}_1$ | - | 42 | 75.48 | 124 | 10.59 | 100 | 75481 | 75.48 |
| $\mathrm{DE}_2$ | - | 71 | 126.68 | 182 | 17.88 | 100 | 126678 | 126.68 |
| $\mathrm{DE}_3$ | - | 35 | 73.43 | 106 | 10.12 | 100 | 73430 | 73.43 |
| $\mathrm{DE}_4$ | - | 69 | 125.01 | 198 | 19.07 | 100 | 125005 | 125.01 |
| $\mathrm{DE}_5$ | - | 94 | 178.78 | 274 | 26.45 | 100 | 178779 | 178.78 |
| $\mathrm{DE}_{2,1}$ | 1.0 | 65 | 127.3 | 185 | 17.85 | 100 | 127297 | 127.3 |
| | 0.9 | 56 | 94.53 | 136 | 12.58 | 100 | 94526 | 94.53 |
| | **0.8** | 41 | 72.25 | 99 | 8.94 | 100 | **72246** | **72.25** |
| | **0.7** | 28 | 57.08 | 78 | 7.07 | 100 | **57079** | **57.08** |
| | **0.6** | 23 | 47.36 | 64 | 5.98 | 100 | **47355** | **47.36** |
| | **0.5** | 23 | 42.21 | 60 | 5.61 | 100 | **42205** | **42.21** |
| | **0.4** | 22 | 41.07 | 58 | 5.58 | 100 | **41066** | **41.07** |
| | **0.3** | 25 | 42.78 | 61 | 6.02 | 100 | **42776** | **42.78** |
| | **0.2** | 29 | 48.94 | 68 | 6.73 | 100 | **48943** | **48.94** |
| | **0.1** | 30 | 58.47 | 83 | 8.13 | 100 | **58474** | **58.47** |
| | 0.0 | 36 | 74.71 | 109 | 10.42 | 100 | 74714 | 74.71 |
| $\mathrm{DE}_{5,4}$ | 1.0 | 93 | 179.82 | 261 | 26.26 | 100 | 179822 | 179.82 |
| | 0.9 | 73 | 136.31 | 196 | 19.05 | 100 | 136308 | 136.31 |
| | 0.8 | 61 | 103.39 | 146 | 13.59 | 100 | 103391 | 103.39 |
| | 0.7 | 47 | 80.82 | 111 | 10.17 | 100 | 80819 | 80.82 |
| | **0.6** | 40 | 67.49 | 99 | 8.51 | 100 | **67490** | **67.49** |
| | **0.5** | 32 | 60.4 | 91 | 7.8 | 100 | **60400** | **60.4** |
| | **0.4** | 38 | 59.81 | 84 | 7.41 | 100 | **59809** | **59.81** |
| | **0.3** | 34 | 64.84 | 94 | 9.05 | 100 | **64835** | **64.84** |
| | 0.2 | 41 | 76.5 | 111 | 10.63 | 100 | 76495 | 76.5 |
| | 0.1 | 55 | 95.25 | 136 | 13.43 | 100 | 95249 | 95.25 |
| | 0.0 | 64 | 125 | 192 | 18.41 | 100 | 125003 | 125 |

TABLE III

COMPARATIVE RESULTS FOR THE QUARTIC TEST FUNCTION, FOR DIFFERENT VALUES OF $\xi$

| Mutation Strategy | $\xi$ | Min | Mean | Max | St.D. | Success | Total Generations | EFE |
|---|---|---|---|---|---|---|---|---|
| $\mathrm{DE}_1$ | - | 18 | 33.07 | 56 | 5.4 | 57.5 | 444018 | 23.01 |
| $\mathrm{DE}_2$ | - | 38 | 75.05 | 110 | 9.68 | 99.8 | 76900 | 30.08 |
| $\mathrm{DE}_3$ | - | 46 | 82.67 | 579 | 31.79 | 96.9 | 111105 | 34.12 |
| $\mathrm{DE}_4$ | - | 43 | 69.1 | 148 | 16.32 | 91.3 | 150084 | 30.27 |
| $\mathrm{DE}_5$ | - | 73 | 119.32 | 207 | 19.65 | 100 | 119319 | 47.73 |
| $\mathrm{DE}_{2,1}$ | **1.0** | 48 | 76.38 | 110 | 9.76 | 100 | **76375** | 30.55 |
| | **0.9** | 41 | 65.28 | 106 | 8.62 | 99.9 | **66214** | 26.14 |
| | **0.8** | 38 | 56.42 | 101 | 7.7 | 99.9 | **57359** | **22.59** |
| | **0.7** | 30 | 49.92 | 109 | 8.66 | 99.7 | **52771** | **20.03** |
| | 0.6 | 27 | 44.7 | 108 | 9.47 | 95 | 92469 | **18.82** |
| | 0.5 | 24 | 39.67 | 142 | 10.27 | 86.7 | 167394 | **18.3** |
| | 0.4 | 20 | 36.86 | 128 | 10.35 | 76.1 | 267047 | **19.37** |
| | 0.3 | 21 | 33.95 | 107 | 9.13 | 66.4 | 358540 | **20.45** |
| | 0.2 | 17 | 32.47 | 93 | 8.72 | 61.7 | 403036 | **21.05** |
| | 0.1 | 20 | 31.85 | 74 | 7.16 | 58.9 | 429758 | **21.63** |
| | 0.0 | 20 | 33.02 | 54 | 5.16 | 58.5 | 434318 | **22.58** |
| $\mathrm{DE}_{5,4}$ | 1.0 | 70 | 119.36 | 188 | 19.12 | 100 | 119362 | 47.74 |
| | 0.9 | 63 | 101.6 | 186 | 17.61 | 100 | 101596 | 40.64 |
| | 0.8 | 50 | 84.17 | 219 | 14.42 | 100 | 84169 | 33.67 |
| | **0.7** | 45 | 71.58 | 162 | 12.96 | 100 | **71583** | 28.63 |
| | **0.6** | 38 | 64.62 | 222 | 14.02 | 99.9 | **65553** | 25.87 |
| | 0.5 | 35 | 60.4 | 214 | 18.3 | 96.7 | 91403 | 24.98 |
| | 0.4 | 34 | 58.23 | 201 | 19.15 | 92.1 | 132634 | 25.29 |
| | 0.3 | 33 | 57.43 | 246 | 19.79 | 87.8 | 172424 | 26.16 |
| | 0.2 | 33 | 59.05 | 335 | 23.55 | 85.7 | 193610 | 27.56 |
| | 0.1 | 32 | 65.17 | 241 | 22.02 | 87.9 | 178288 | 29.66 |
| | 0.0 | 42 | 69.17 | 176 | 17 | 89.9 | 163182 | 30.78 |

TABLE IV

COMPARATIVE RESULTS FOR THE LEVY NO.5 TEST FUNCTION, FOR DIFFERENT VALUES OF $\xi$

mutation operators for some values of $\xi$ (in bold). However, these values may differ for each optimization problem.

During the second set of experiments we studied the self-balancing schemes. To this end, 1000 independent runs were performed for each scheme and each problem. Tables V–XII summarize the results (in bold are the best performing algorithms). We denote $\mathrm{DE}_{2,1,\xi_1}$, $\mathrm{DE}_{2,1,\xi_2}$, and $\mathrm{DE}_{2,1,\xi_3}$ the hybrid algorithm that uses the $\xi_1$, $\xi_2$, and $\xi_3$ self-balancing schemes, respectively. Similarly we define the $\mathrm{DE}_{5,4,\xi_1}$, $\mathrm{DE}_{5,4,\xi_2}$, and $\mathrm{DE}_{5,4,\xi_3}$ algorithms. It is evident that the proposed self-balancing schemes exhibit excellent success rates and, in general, are among the most efficient algorithms of this study. The most promising self-balancing schemes are $\xi_1$ and $\xi_3$. To this end, for an unknown optimization problem the application of the self-balancing mutation schemes is highly recommended.

## VII. CONCLUSIONS

In this study we proposed a hybrid Differential Evolution mutation operator that is a linear combination of two other mutation operators (an explorative and an exploitive operator) in an attempt to balance their effects. The new operator depends on the user-defined parameter $\xi$. To alleviate the problem of selecting a proper value of $\xi$ for each problem, we proposed a self-balancing mutation scheme. This scheme favors the exploration of the search space during the first phase of the optimization, while later, opts for the exploitation to aid convergence to the optimum.

Extensive experimental results indicate that the proposed approaches enhance DE's ability to accurately locate solutions in the search space. The use of the self-balancing mutation scheme can lead to reliable optimization of unknown objective function, since it alleviates problems generated by poor selection of the user–defined parameters, such as decreased rate of convergence, or even divergence and premature saturation. Thus, with the application of the self-balancing mutation scheme locating optima becomes feasible on a first–time basis for a given unknown problem.

However, exhaustive experimental results and comparisons are needed. In a future communication we intend to study more hybrid mutation operators that are combinations of different (original and recently proposed) DE operators. Additionally, we will investigate the parallel implementation of the proposed approaches, as well as their performance on difficult high–dimensional real–life problems encountered in bioinformatics, medical applications and neural network training.

## REFERENCES


[1] T. Baeck, D. B. Fogel, and Z. Michalewicz, Eds., *Handbook of Evolutionary Computation*. Oxford University Press, 1997.

[2] R. Storn and K. Price, "Differential evolution – a simple and efficient adaptive scheme for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, pp. 341–359, 1997.

[3] K. Price, R. M. Storn, and J. A. Lampinen, *Differential Evolution: A Practical Approach to Global Optimization (Natural Computing Series)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2005.

| Mutation Strategy | Min | Mean | Max | St.D. | Success | Total Generations | EFE |
|---|---|---|---|---|---|---|---|
| $DE_1$ | 18 | 33.07 | 56 | 5.4 | 57.5 | 444018 | **23.01** |
| $DE_2$ | 38 | 75.05 | 110 | 9.68 | 99.8 | **76900** | 30.08 |
| $DE_3$ | 46 | 82.67 | 579 | 31.79 | 96.9 | 111105 | 34.12 |
| $DE_4$ | 43 | 69.1 | 148 | 16.32 | 91.3 | 150084 | 30.27 |
| $DE_5$ | 73 | 119.32 | 207 | 19.65 | 100 | 119319 | 47.73 |
| $DE_{2,1,\xi_1}$ | 60 | 81.64 | 121 | 8.83 | 100 | 81636 | 32.65 |
| $DE_{2,1,\xi_2}$ | 25 | 40.84 | 106 | 9.35 | 92 | 117569 | **17.75** |
| $DE_{2,1,\xi_3}$ | 37 | 54.84 | 114 | 7.21 | 100 | **54835** | **21.93** |
| $DE_{5,4,\xi_1}$ | 78 | 119.28 | 199 | 18.97 | 100 | 119279 | 47.71 |
| $DE_{5,4,\xi_2}$ | 22 | 63.16 | 155 | 14.95 | 99.4 | **68781** | 25.42 |
| $DE_{5,4,\xi_3}$ | 43 | 78.55 | 146 | 11.67 | 100 | 78546 | 31.42 |

TABLE V

COMPARATIVE RESULTS FOR THE LEVY NO.5 TEST FUNCTION, FOR $\xi_1$, $\xi_2$, AND $\xi_3$ SELF-BALANCING SCHEMES

| Mutation Strategy | Min | Mean | Max | St.D. | Success | Total Generations | EFE |
|---|---|---|---|---|---|---|---|
| $DE_1$ | 42 | 75.48 | 124 | 10.59 | 100 | 75481 | 75.48 |
| $DE_2$ | 71 | 126.68 | 182 | 17.88 | 100 | 126678 | 126.68 |
| $DE_3$ | 35 | 73.43 | 106 | 10.12 | 100 | **73430** | **73.43** |
| $DE_4$ | 69 | 125.01 | 198 | 19.07 | 100 | 125005 | 125.01 |
| $DE_5$ | 94 | 178.78 | 274 | 26.45 | 100 | 178779 | 178.78 |
| $DE_{2,1,\xi_1}$ | 84 | 130.75 | 159 | 11.26 | 100 | 130745 | 130.75 |
| $DE_{2,1,\xi_2}$ | 34 | 51.71 | 71 | 6.54 | 100 | **51713** | **51.71** |
| $DE_{2,1,\xi_3}$ | 43 | 71.71 | 94 | 8.36 | 100 | **71712** | **71.71** |
| $DE_{5,4,\xi_1}$ | 78 | 160.28 | 196 | 13.51 | 100 | 160278 | 160.28 |
| $DE_{5,4,\xi_2}$ | 34 | 75.51 | 105 | 9.99 | 100 | 75514 | 75.51 |
| $DE_{5,4,\xi_3}$ | 55 | 96.13 | 128 | 10.52 | 100 | 96126 | 96.13 |

TABLE VI

COMPARATIVE RESULTS FOR THE QUARTIC TEST FUNCTION, FOR $\xi_1$, $\xi_2$, AND $\xi_3$ SELF-BALANCING SCHEMES

| Mutation Strategy | Min | Mean | Max | St.D. | Success | Total Generations | EFE |
|---|---|---|---|---|---|---|---|
| $DE_1$ | 207 | 363.2 | 592 | 83.94 | 7.5 | 9277240 | 2421.33 |
| $DE_2$ | 566 | 835.95 | 1140 | 97.3 | 100 | **835950** | **417.98** |
| $DE_3$ | 486 | 1087.56 | 5673 | 400.07 | 91.6 | 1836208 | 593.65 |
| $DE_4$ | 594 | 1044.77 | 1520 | 145.61 | 91.8 | 1779100 | **569.05** |
| $DE_5$ | 1190 | 1713.58 | 2165 | 149.06 | 100 | **1713577** | 856.79 |
| $DE_{2,1,\xi_1}$ | 869 | 1692.45 | 2788 | 309.47 | 99.9 | 1700756 | 847.07 |
| $DE_{2,1,\xi_2}$ | 193 | 519.17 | 1313 | 196 | 25.3 | 7601350 | 1026.03 |
| $DE_{2,1,\xi_3}$ | 359 | 870.22 | 2043 | 244.27 | 97.9 | **1061942** | **444.44** |
| $DE_{5,4,\xi_1}$ | 2880 | 4315.46 | 5138 | 306.76 | 100 | 4315461 | 2157.73 |
| $DE_{5,4,\xi_2}$ | 541 | 1763.28 | 2839 | 383.48 | 94.7 | 2199830 | 930.98 |
| $DE_{5,4,\xi_3}$ | 1300 | 3156.25 | 4482 | 557.73 | 100 | 3156247 | 1578.12 |

TABLE VII

COMPARATIVE RESULTS FOR THE GRIEWANGK TEST FUNCTION, FOR $\xi_1$, $\xi_2$, AND $\xi_3$ SELF-BALANCING SCHEMES

| Mutation Strategy | Min | Mean | Max | St.D. | Success | Total Generations | EFE |
|---|---|---|---|---|---|---|---|
| $DE_1$ | 48 | 74.65 | 184 | 14.92 | 61.3 | 819760 | 18.27 |
| $DE_2$ | 81 | 139.57 | 194 | 13.83 | 98.8 | **161899** | 21.19 |
| $DE_3$ | 83 | 119.62 | 286 | 15.93 | 95.3 | 207996 | 18.83 |
| $DE_4$ | 85 | 120.61 | 172 | 12.54 | 96.5 | 186388 | 18.75 |
| $DE_5$ | 142 | 197.89 | 262 | 18.61 | 100 | 197891 | 29.68 |
| $DE_{2,1,\xi_1}$ | 111 | 143.17 | 188 | 9.75 | 98.3 | 174733 | 21.85 |
| $DE_{2,1,\xi_2}$ | 41 | 70.41 | 176 | 14.49 | 54 | 958024 | 19.56 |
| $DE_{2,1,\xi_3}$ | 66 | 92.65 | 561 | 18.2 | 86.5 | 350138 | **16.07** |
| $DE_{5,4,\xi_1}$ | 142 | 179.01 | 222 | 11.15 | 99.9 | 180827 | 26.88 |
| $DE_{5,4,\xi_2}$ | 63 | 95.34 | 185 | 10.31 | 96.5 | **162006** | **14.82** |
| $DE_{5,4,\xi_3}$ | 85 | 119.5 | 149 | 9.41 | 98.7 | **143946** | **18.16** |

TABLE VIII

COMPARATIVE RESULTS FOR THE CORANA TEST FUNCTION, FOR $\xi_1$, $\xi_2$, AND $\xi_3$ SELF-BALANCING SCHEMES

| Mutation Strategy | Min | Mean | Max | St.D. | Success | Total Generations | EFE |
|---|---|---|---|---|---|---|---|
| $DE_1$ | 5 | 22.77 | 43 | 5.4 | 63.6 | 378481 | **10.74** |
| $DE_2$ | 6 | 62.07 | 99 | 11.01 | 98.8 | **73329** | 18.85 |
| $DE_3$ | 5 | 40.16 | 431 | 15.71 | 91.6 | 120790 | **13.15** |
| $DE_4$ | 8 | 48.46 | 87 | 9.69 | 99.1 | **57021** | **14.67** |
| $DE_5$ | 3 | 88.74 | 129 | 15.11 | 100 | **88740** | 26.62 |
| $DE_{2,1,\xi_1}$ | 4 | 99 | 163 | 19.85 | 99.2 | 106210 | 29.94 |
| $DE_{2,1,\xi_2}$ | 9 | 50.18 | 111 | 16.3 | 56.4 | 464301 | 26.69 |
| $DE_{2,1,\xi_3}$ | 16 | 82.08 | 138 | 17.55 | 84.3 | 226190 | 29.21 |
| $DE_{5,4,\xi_1}$ | 11 | 143.83 | 227 | 27 | 100 | 143831 | 43.15 |
| $DE_{5,4,\xi_2}$ | 26 | 86.75 | 154 | 20.73 | 97.5 | 109582 | 26.69 |
| $DE_{5,4,\xi_3}$ | 1 | 116.59 | 197 | 24.65 | 99.4 | 121894 | 35.19 |

TABLE IX

COMPARATIVE RESULTS FOR THE SHEKEL TEST FUNCTION, FOR $\xi_1$, $\xi_2$, AND $\xi_3$ SELF-BALANCING SCHEMES

| Mutation Strategy | Min | Mean | Max | St.D. | Success | Total Generations | EFE |
|---|---|---|---|---|---|---|---|
| $DE_1$ | 0 | 6.56 | 27 | 3.28 | 94.1 | 65171 | **1.39** |
| $DE_2$ | 6 | 21.89 | 105 | 6.72 | 100 | 21885 | 4.38 |
| $DE_3$ | 1 | 24 | 119 | 11.49 | 65.4 | 361697 | 7.34 |
| $DE_4$ | 0 | 6.14 | 22 | 3.33 | 99.8 | **8124** | **1.23** |
| $DE_5$ | 2 | 18.81 | 41 | 6.16 | 100 | **18814** | 3.76 |
| $DE_{2,1,\xi_1}$ | 2 | 19.49 | 43 | 6.12 | 100 | 19493 | 3.9 |
| $DE_{2,1,\xi_2}$ | 1 | 15.16 | 35 | 5.91 | 46.2 | 545003 | 6.56 |
| $DE_{2,1,\xi_3}$ | 3 | 23.61 | 54 | 7.07 | 88.6 | 134919 | 5.33 |
| $DE_{5,4,\xi_1}$ | 1 | 17.5 | 39 | 6.12 | 100 | **17503** | 3.5 |
| $DE_{5,4,\xi_2}$ | 0 | 14.44 | 36 | 5.95 | 96.4 | 49918 | **3** |
| $DE_{5,4,\xi_3}$ | 3 | 19.63 | 44 | 6.51 | 99.9 | 20614 | 3.93 |

TABLE X

COMPARATIVE RESULTS FOR THE STEP TEST FUNCTION, FOR $\xi_1$, $\xi_2$, AND $\xi_3$ SELF-BALANCING SCHEMES

| Mutation Strategy | Min | Mean | Max | St.D. | Success | Total Generations | EFE |
|---|---|---|---|---|---|---|---|
| $DE_1$ | 9 | 25.55 | 54 | 3.81 | 100 | **25546** | **7.66** |
| $DE_2$ | 27 | 63.51 | 243 | 16.58 | 98.4 | 78489 | 19.36 |
| $DE_3$ | 27 | 57.13 | 127 | 10.36 | 100 | 57128 | 17.14 |
| $DE_4$ | 9 | 45.85 | 75 | 6 | 100 | 45850 | 13.76 |
| $DE_5$ | 51 | 87.53 | 127 | 12.05 | 100 | 87532 | 26.26 |
| $DE_{2,1,\xi_1}$ | 41 | 72.4 | 431 | 39.36 | 100 | 72403 | 21.72 |
| $DE_{2,1,\xi_2}$ | 18 | 31.68 | 73 | 7.13 | 100 | **31679** | **9.5** |
| $DE_{2,1,\xi_3}$ | 28 | 63.99 | 409 | 66.3 | 100 | 63994 | 19.2 |
| $DE_{5,4,\xi_1}$ | 42 | 86.88 | 115 | 8.62 | 100 | 86883 | 26.06 |
| $DE_{5,4,\xi_2}$ | 24 | 43.35 | 69 | 5.88 | 100 | **43347** | **13** |
| $DE_{5,4,\xi_3}$ | 33 | 58.46 | 89 | 7.37 | 100 | 58462 | 17.54 |

TABLE XI

COMPARATIVE RESULTS FOR THE ROSENBROCK TEST FUNCTION, FOR $\xi_1$, $\xi_2$, AND $\xi_3$ SELF-BALANCING SCHEMES

| Mutation Strategy | Min | Mean | Max | St.D. | Success | Total Generations | EFE |
|---|---|---|---|---|---|---|---|
| $DE_1$ | 41 | 51.9 | 64 | 3.54 | 100 | **51902** | **15.57** |
| $DE_2$ | 88 | 115.68 | 138 | 7 | 100 | 115680 | 34.7 |
| $DE_3$ | 61 | 75.77 | 87 | 4.17 | 100 | 75770 | 22.73 |
| $DE_4$ | 71 | 90.11 | 106 | 5.38 | 100 | 90114 | 27.03 |
| $DE_5$ | 107 | 156.28 | 186 | 9.22 | 100 | 156282 | 46.88 |
| $DE_{2,1,\xi_1}$ | 84 | 99.47 | 112 | 3.88 | 100 | 99470 | 29.84 |
| $DE_{2,1,\xi_2}$ | 40 | 50.6 | 60 | 3.29 | 100 | **50598** | **15.18** |
| $DE_{2,1,\xi_3}$ | 56 | 68.07 | 101 | 3.8 | 100 | 68072 | **20.42** |
| $DE_{5,4,\xi_1}$ | 106 | 119.12 | 133 | 4.42 | 100 | 119120 | 35.74 |
| $DE_{5,4,\xi_2}$ | 54 | 71.61 | 83 | 4.31 | 100 | **71612** | 21.48 |
| $DE_{5,4,\xi_3}$ | 72 | 85.69 | 98 | 4.28 | 100 | 85694 | 25.71 |

TABLE XII

COMPARATIVE RESULTS FOR THE SPHERE TEST FUNCTION, FOR $\xi_1$, $\xi_2$, AND $\xi_3$ SELF-BALANCING SCHEMES


[4] R. Storn, "System design by constraint adaptation and differential evolution," *IEEE Transactions on Evolutionary Computation*, vol. 3, pp. 22–34, 1999.

[5] V. P. Plagianakos and M. N. Vrahatis, "Neural network training with constrained integer weights," in *IEEE Congress of Evolutionary Computation (CEC'99)*, P. Angeline, Z. Michalewicz, M. Schoenauer, X. Yao, and A. Zalzala, Eds. Washington D.C., U.S.A.: IEEE Press, 1999, pp. 2007–2013.

[6] ——, "Parallel evolutionary training algorithms for 'hardware–friendly' neural networks," *Natural Computing*, vol. 1, pp. 307–322, 2002.

[7] J. Vesterstrom and R. Thomsen, "A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems," in *IEEE Congress on Evolutionary Computation (CEC 2004)*, vol. 2, 2004, pp. 1980–1987.

[8] H. Y. Fan and J. Lampinen, "A trigonometric mutation operation to differential evolution," *Journal of Global Optimization*, vol. 27, pp. 105–129, 2003.

[9] N. G. Pavlidis, D. K. Tasoulis, V. P. Plagianakos, and M. N. Vrahatis, "Human designed vs. genetically programmed differential evolution operators," in *IEEE Congress on Evolutionary Computation (CEC 2006)*, 2006, pp. 1880–1886.

[10] D. K. Tasoulis, V. P. Plagianakos, and M. N. Vrahatis, "Clustering in evolutionary algorithms to efficiently compute simultaneously local and global minima," in *IEEE Congress on Evolutionary Computation (CEC 2005)*, vol. 2, Sept 2005, pp. 1847–1854.

[11] K. E. Parsopoulos and M. N. Vrahatis, "UPSO: A unified particle swarm optimization scheme," in *Lecture Series on Computer and Computational Sciences, Proceedings of the International Conference of "Computational Methods in Sciences and Engineering" (ICCMSE 2004)*, vol. 1, 2004, pp. 868–873.

[12] ——, "Unified particle swarm optimization for tackling operations research problems," in *IEEE 2005 Swarm Intelligence Symposium (SIS 2005)*, Pasadena (CA), USA, 2005, pp. 53–59.

[13] ——, "Unified particle swarm optimization in dynamic environments," *Lecture Notes in Computer Science (LNCS)*, vol. 3449, pp. 590–599, 2005.

[14] ——, "Parameter selection and adaptation in unified particle swarm optimization," *Mathematical and Computer Modelling*, vol. 46, no. 1-2, pp. 198–213, 2007.

[15] U. K. Chakraborty, S. Das, and A. Konar, "Differential evolution with local neighborhood," in *IEEE Congress on Evolutionary Computation (CEC 2006)*, 2006, pp. 2042–2049.

[16] S. Kirkpatrick, C. D. G. Jr., and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, pp. 671–680, 1983.

[17] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, "Equation of state calculations by fast computing machines," *The Journal of Chemical Physics*, vol. 21, no. 6, pp. 1087–1092, 1953.

[18] A. Corana, M. Marchesi, C. Martini, and S. Ridella, "Minimizing multimodal functions of continuous variables with the "simulated annealing" algorithm," *ACM Transactions Mathematical Software*, vol. 13, no. 3, pp. 262–280, 1987.

[19] K. D. Jong, "An analysis of the behavior of a class of genetic adaptive systems," Ph.D. dissertation, University of Michigan, 1975.

[20] A. O. Griewank, "Generalized descent for global optimization," *Journal of optimization theory and applications*, vol. 34, no. 1, pp. 11–39, 1981.

[21] A. Levy, A. Montalvo, S. Gomez, and A. Galderon, *Topics in Global Optimization*. Springer-Verlag, New York, 1981.

[22] D. K. Tasoulis, N. G. Pavlidis, V. P. Plagianakos, and M. N. Vrahatis, "Parallel differential evolution," in *IEEE Congress on Evolutionary Computation (CEC 2004)*, vol. 2, June 2004, pp. 2023–2029.

[23] I. C. Trelea, "The particle swarm optimization algorithm: convergence analysis and parameter selection," *Information Processing Letters*, vol. 85, no. 6, pp. 317–325, 2003.