DE with Random Vector based Mutatiton for High Dimensional Problems

Mg. Sebastián Hernández[1], Dr. Efrén Mezura-Montes[2], and Dr. Guillermo Leguizamón[3]

[1]National University of Southern Patagonia,
[2]Artificial Intelligence Research Center, University of Veracruz
[3]National University of San Luis
sebastian.unpa@gmail.com
emezura@uv.mx
legui@unsl.edu.ar

Abstract. Metaheuristic techniques are the current standard for solving optimization problems. Differential Evolution (DE) is one of the most used because all operations are on real floating point numbers and does not require extra coding. However, the performance shown by DE could decay when applied in problems of high dimensionality. In this paper we present RLSDE, a modified version of DE, based on a random vector as a scaling factor for the differential mutation and the application of a local search operator. These modifications constitute an algorithm capable of solving 100D problems using few computational resources. RLSDE is compared against the results obtained with the classic version of DE and ELSDE (Enchanced Local Search Differential Evolution), showing the performance of the proposal.

Keywords: differential evolution, high-dimensional optimization problem, local search

1 Introduction

As technology advances, it is necessary to create and solve increasingly complex mathematical models in order to provide more precise solutions. In turn, these models should involve a large number of variables, which makes the problem of optimization in a problem of high dimensionality (100 or more variables). It should also be considered that they are generally non-linear functions and therefore their resolution is not simple. All these conditions make that optimal search methods require a large computational cost and often fail in their objective due, among other factors, to the exponential growth of the search space and the complexity of the problem. That is why new optimization techniques should be proposed with simple but powerful conditions [1], [4], [6], [13], capable of facing and solving high dimensional problems. In this article a modified and hybrid version of DE is presented. The main modification was made on the differential mutation: the population was classified into three different groups based on their

performance and then this classification was taken into account when choosing the vectors to generate the mutant vector. The other modification that was applied in the generation of the mutant vector is the use of a variable scale factor. Finally, this proposal was hybridized with a random local search engine whose operation is based on the classification of the population used in the differential mutation. The performance of this proposal is evaluated with classical functions of the literature, the results obtained are compared against those presented in [12], obtained by the ELSDE algorithm with the same number of function evaluations. The results obtained by RLSDE show the quality of this proposal. The rest of this paper is organized as follows, the second section describe the basic DE algorithm, the third section presents our version, the RLSDE algorithm. Its performance is analyzed and compared against DE y ELSDE in the fourth section. Finally, the conclusions are given in the fifth section.

2 Differential Evolution

Differential evolution was presented in 1996 by Storn and Price [11]. IIt is a population metaheuristic whose operation is based on the application of operators to the individuals (vectors) of the population, through a time of evolution:

- **Initialization**. Taking into account that each variable of the search space is confined to a certain region, real values within this range are generated in a random way. At generation zero (initial value), the jth component of the ith individual is defined as:

$$\mathbf{x}_{j,i,0} = rand_j\,[0,1) \cdot (b_{j,U} - b_{j,L}) + b_{j,L},$$

  where $[b_{j,L}, b_{j,U}]$ is the definition interval of the jth variable.
- **Differential mutation**. Like most population metaheuristics, DE applies an operator to generate new individuals, disrupting existing ones in the population. This is done by randomly selecting three individuals from the population (targets vectors) and creating the *donor vector* with one of them as *base* and the other two in the form of *scaled difference*:

$$\mathbf{v}_{i,g} = \mathbf{x}_{r_0,g} + F \cdot (\mathbf{x}_{r_1,g} - \mathbf{x}_{r_2,g})\,,$$

  where g represents the evolution time, F is the scale factor applied in the vector difference and $\mathbf{x}_{r_i}, i = 0, 1, 2$ are the randomly selected vectors, and the indices r_1, r_2, r_3 are mutually exclusive. This operator is considered the most important of DE, since it adds population diversity taking into account the characteristics of individuals present in the population.
- **Crossover**. This operator, also called *discrete recombination*, generates the *trial vector* by randomly mixing components of the target and donor vectors, whose indices in the population are the same:

$$u_{j,i,g} = \begin{cases} v_{j,i,g} \text{ if } rand_j\,[0,1) \leq C_r \\ x_{j,i,g} \text{ othercase} \end{cases},$$

where C_r is the crossover rate that will be compared to a random value to decide who will bring the component to the *trial vector*.

– **Selection**. The next step of the algorithm calls for *selection* to determine whether the target or the trial vector survives to the next generation. When comparing the fitness of target vector ($\mathbf{x}_{i,g}$) with the trial vector ($\mathbf{u}_{i,g}$), it is decided whether the target remains at least one more generation in the population or if it is replaced by the trial vector:

$$\mathbf{x}_{i,g+1} = \begin{cases} \mathbf{u}_{i,g} \text{ if } f(\mathbf{u}_{i,g}) < f(\mathbf{x}_{i,g}) \\ \mathbf{x}_{i,g} \text{ othercase} \end{cases}$$

3 State of the art

DE is a well-known and used population metaheuristic due to its ability to obtain quality solutions. But as it is also affected by the curse of dimensionality, algorithms based on DE are often presented in order to effectively solve high-dimensional problems. In 2014 article [3] was presented. The authors suggest that the strategies used in low dimension (mutation, population size choice, crossover) are not adequate to solve problems in high dimension. In addition, with the change of dimension it is known that the search space increases exponentially but the amount of function evaluations used generally only grows linearly. In 2017 the articles [5] and [2] were presented. In the first of them, the center of gravity of three randomly chosen individuals was used as the base vector for differential mutation, and in the other article the authors used the population covariance matrix because they observe that the traditional approach to do facing the problem of increasing dimension is to increase the evolution time. According to the authors, it is not taken into account that both magnitudes are not modified proportionally and therefore a good performance is not achieved. In 2018 Meselhi et al. present [8], where they propose the use of the Enhanced Differential Grouping (EDG) method, capable of discovering the dependency relationships between variables and then grouping the independent variables in the same subproblem. In 2019, Cai et al. publish cite Cai2019. In this article the authors apply to algorithms of high dimension an algorithm that makes a prediction of the global minimum and from that prediction the direction that the differential mutation should take in search of that optimum is guided.

4 Proposed modifications

DE, like most optimization algorithms, suffers from the *curse of dimensionality* due to the exponential increase of the search space when considering increases in the dimensionality of the problem. Many modifications have been proposed in order to mitigate the effects of increasing the size of the search space [2], [8]. In this case, the proposed modifications are simple and show a significant improvement in performance compared to the efficiency of the original DE version.

4.1 Population partitioning

The population is classified, according to its fitness value, into three groups. For this, the individual with the minimum fitness is $k = 1$ and the individual with the maximum fitness is $k = n$. Then, A group contains individuals whose k position on the fitness scale verifies $k < 0.25 \times popsize$, that is, the group of the best individuals; group C contains those where $k > 0.75 \times popsize$ (the worst individuals) and group B includes the remaining individuals of the population. When differential mutation is applied, three individuals are selected in a random manner so that the base vector does not belong to the same group as the two to be used in the difference, in order to favor diversity.

4.2 Vector scale factor

The scale factor F is responsible for *smoothing* the disturbance generated by the vector difference when adding to the base vector in the differential mutation. The lower the value of F, the smaller the size of the steps performed by the mutation and therefore it will take longer to achieve convergence. Larger values of F facilitate exploration, but may cause the algorithm to exceed some optimal values due to its passage width. The classical version of DE uses a fixed scale factor for the vector difference applied in the differential mutation. In this article we propose to use a vector as a scale factor, so that each of the variables of the difference vector is disturbed differently. This new vector is generated from a *center* C and a expansion radius r, where the components are random within the interval $(C - r, C + r)$:

$$\mathbf{F} = [f_1, f_2, \ldots, f_D]\,,$$

where the components of the vector are $f_i = C + 2 \cdot r \cdot rand() - r$

4.3 Local search

It is common to hybridize the algorithms of global optimization with local search algorithms [7], [10], [9]. Local search algorithms intensify the exploration of individuals located around a particular individual, generally of good performance. In our case, local search is applied to the best individual in each generation. The local search operator generates six new individuals for each generation of evolution, from the best individual in the population ($\mathbf{x}_{best}$), from the average of population ($\mathbf{x}_\mu$), and from the average individual in group A ($\mathbf{x}_\mu^A$). It's like that $\mathbf{x}_\mu^A$ and $\mathbf{x}_\mu$ are obtained respectivelly as follows:

- The jth component of $\mathbf{x}_\mu^A$ is obtained from calculating the average of the jth components of the individuals in group A, without taking into account $\mathbf{x}_{best}$.
- The jth component of $\mathbf{x}_\mu$ is obtained from calculating the average of the jth components of all individuals in the population, including $\mathbf{x}_{best}$.

Then, six new individuals are generated with different characteristics and with different disturbed variables that will be compared, one at a time, against $\mathbf{x}_{best}$. The only new individual that emerges from disturbing all its components is $\mathbf{x}_c$, called *centroid individual*, with a very simple mathematical formulation:

$$\mathbf{x}_c = \frac{\mathbf{x}_{best} + \mathbf{x}_\mu^A + \mathbf{x}_\mu}{3}$$

where each individual of the population contributes with its components in a different proportion, which will then be scaled up to a third:

- $\mathbf{x}_{best}$ contributes in full form from the original sum and proportionally to $1/popsize$ from $\mathbf{x}_\mu$.
- The individuals of group A, contribute with proportion $1/\left(popsize_A - 1\right)$ from the original sum and then with the proportion $1/popsize$ from $\mathbf{x}_\mu$.
- The individuals of groups B and C, contribute with the proportion $1/popsize$ from $\mathbf{x}_\mu$.

Then two normalized individuals are generated, according to the infinite norm of vectors:

$$\mathbf{A} \in \mathbb{R}^D, \|\mathbf{A}\|_\infty = \max_{i \in [1,D]} |A_i|,$$

denominated $\mathbf{d}_\mu^A$ and $\mathbf{d}_\mu$:

$$\mathbf{d}_\mu^A = \frac{\left|\mathbf{x}_{best} - \mathbf{x}_\mu^A\right|}{\left\|\mathbf{x}_{best} - \mathbf{x}_\mu^A\right\|_\infty}, \qquad \mathbf{d}_\mu = \frac{|\mathbf{x}_{best} - \mathbf{x}_\mu|}{\|\mathbf{x}_{best} - \mathbf{x}_\mu\|_\infty},$$

where its minimum and maximum components can be 0 and 1 respectively. Only fifteen percent of the components of the *distance vectors* will be taken into account, then a permutation is generated on the vector $\mathbf{v} = [1, D]$, in our case $\mathbf{v} = [1, 2, 3, \ldots, 100]$. Thus, components $\mathbf{v}(1)$ to $\mathbf{v}(15)$ indicate which $\mathbf{d}_\mu^A$ values will be conserved and components $\mathbf{v}(16)$ to $\mathbf{v}(30)$ shows which components of $\mathbf{d}_\mu$ will be used. The other components are transformed to zero. With the new versions of $\mathbf{d}_\mu^A$ and $\mathbf{d}_\mu$, four new individuals are created in two stages. In the first one, vectors $\mathbf{x}_{best}$, $\mathbf{d}_\mu^A$ and $\mathbf{d}_\mu$ are taken:

$$\begin{aligned} \mathbf{x}_{D_1} &= \mathbf{x}_{best} - F_d \cdot \left(\mathbf{d}_\mu^A * \mathbf{x}_\mu^A\right) \\ \mathbf{x}_{D_2} &= \mathbf{d}_\mu^A * \mathbf{x}_\mu^A - F_d \cdot \mathbf{x}_{best} \end{aligned}$$

with

$$F_d = \frac{f\left(\mathbf{x}_{best}\right)}{f\left(\mathbf{x}_\mu^A\right)}$$

and in second stage:

$$\begin{aligned} \mathbf{x}_{D_3} &= \mathbf{x}_{best} - F_d \cdot \left(\mathbf{d}_\mu * \mathbf{x}_\mu\right) \\ \mathbf{x}_{D_4} &= \mathbf{d}_\mu * \mathbf{x}_\mu - F_d \cdot \mathbf{x}_{best} \end{aligned}$$

with

$$F_d = \frac{f\left(\mathbf{x}_{best}\right)}{f\left(\mathbf{x}_\mu\right)}$$

In both cases the symbol $*$ represents the product component to component between two vectors.

Finally, the last of the individuals is generated: $\mathbf{x}_{best'}$, who exchanges internally five of its variables. This exchange is done taking into account that some of the functions selected for optimization are not separable and this process can help the convergence of those functions.

Example From a hypothetical population are obtained:

- $\mathbf{x}_{best} = [-0.0654166; -0.0804726; 0.098836; -0.0992805; \ldots; -0.0438895]$, $f(\mathbf{x}_{best}) = 0.0481651791782$
- $\mathbf{x}_{\mu}^{A} = [-0.122108; 0.0187508; -0.0661062; -0.0786949; \ldots; -0.124477]$, $f\left(\mathbf{x}_{\mu}^{A}\right) = 0.0703484791916$
- $\mathbf{x}_{\mu} = [-0.0570289; -0.0897206; -0.109161; 0.0178861; \ldots; 0.0386951]$, $f(\mathbf{x}_{\mu}) = 0.093300675794$

Then, the *centroid individual* is created from the three vectors described above:

- $\mathbf{x}_c = [-0.0815178; -0.0504808; -0.0254772; -0.0533631; \ldots; -0.0432238]$, $f(\mathbf{x}_c) = 0.0317538969857$

Since $\mathbf{x}_c$ has better performance than $\mathbf{x}_{best}$ then the local search is considered successful, $\mathbf{x}_{best}$ is replaced by $\mathbf{x}_c$ although the local search process continues until the six explorations are performed, in case it is possible to further improve $\mathbf{x}_{best}$.

The permutation vector $\mathbf{v}$ is created randomly, according to this example with $D = 100$, where one of the possible combinations for the first fifteen components could be:

$$[7, 10, 1, 12, 11, \ldots, 3, 4],$$

these being the positions of the vector $\mathbf{d}_{\mu}^{A}$ that do not become zero. So $\mathbf{d}_{\mu}^{A}$, for our example, is a vector with $D = 100$, where only the components $7, 10, 1, 12, 11, \ldots, 3, 4$ are nonzero. Continuing with hypotetical positions 16 to 30 of the permutation vector $\mathbf{v}$:

$$[29, 20, 27, 16, 17, \ldots, 19, 26],$$

therefore those will be the $\mathbf{d}_{\mu}$ positions that will not be transformed into the zero value.

Applying the distance formula and then generating four new disturbed vectors $\mathbf{x}_D$:

- $\mathbf{x}_{D_1} = [-0.0915425; -0.0504808; -0.0309095; -0.0573951; \ldots; -0.0636804]$, $f(\mathbf{x}_{D_1}) = 0.036670111539$
- $\mathbf{x}_{D_2} = [-0.074396; -0.00829604; -0.037242; -0.0333039; \ldots; -0.13158]$, $f(\mathbf{x}_{D_2}) = 0.0276253117511$
- $\mathbf{x}_{D_3} = [-0.0815178; -0.0555942; -0.0387451; -0.0533631; \ldots; -0.0386199]$, $f(\mathbf{x}_{D_3}) = 0.035218104998$

– $\mathbf{x}_{D_4} = [-0.00990799; -0.0482058; -0.112258; -0.00648596; \ldots; 0.0326252]$, $f(\mathbf{x}_{D_4}) = 0.031244911621$

Since $\mathbf{x}_{D_2}$ has a better performance than $\mathbf{x}_{best}$, then it replaces it. Then, in the last step five components of $\mathbf{x}_{D_2}$ ($\mathbf{x}_{best}$) are exchanged:

– $\mathbf{x}_{best} = [-0.074396; -0.00829604; -0.037242; -0.0333039; \ldots; -0.13158]$, $f(\mathbf{x}_{D_2}) = 0.0276253117511$
– $\mathbf{x}_{best'} = [-0.0333039; -0.00829604; -0.037242; -0.074396; \ldots; 0.031471]$, $f(\mathbf{x}_{D_2}) = 0.0292658878856$

Therefore, the local search operation was successful and the performance of $\mathbf{x}_{best}$ extracted from the population ($\mathbf{x}_c$ and $\mathbf{x}_{D_2}$) was improved twice, which returns to continue with the evolution as a new individual.

5 Experiments

An experimental study was conducted, with nine classic functions of the literature, to evaluate the efficiency of the proposed algorithm against the classic DE and ELSDE. The ELSDE algorithm is a modified version of DE where a novel local search operation was presented. This local operation combines both advantage of orthogonal crossover and opposition-based search learning strategy. The authors apply this new local search engine only to an individual in the population, chosen randomly. Finnaly, concluded that ELSDE is an efficient method for the high-dimensional optimization problems.

5.1 Scalable functions

The set of functions to which RLSDE was applied is the same as that used in [12]. These functions are continuous and have different characteristic. They also have many local extreme points and high optimizing complexity.

1. **Sphere**. Continuous, differentiable, separable and multimodal. Defined in $[-100, 100]^D$:
$$f(\mathbf{x}) = \sum_{i=1}^{D} x_i^2$$
2. **Rosenbrock**. Continuous, non-separable, multimodal and non-convex. Defined in $[-100, 100]^D$:
$$f(\mathbf{x}) = \sum_{i=1}^{D} \left[100\left(x_{i+1} - x_i^2\right)^2 + (1 - x_i)^2\right]$$
3. **Ackley**. Continuous, non-convex and multimodal. Defined in $[-32, 32]^D$:
$$f(\mathbf{x}) = -20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}\right) - \exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i)\right) + 20 + \exp(1)$$

4. **Griewank**. Continuous, unimodal and not convex. Defined in $[-600, 600]^D$:

$$f_1(\mathbf{x}) = 1 + \sum_{i=1}^{D} \frac{x_i^2}{4000} - \prod_{i=1}^{D} \cos\left(\frac{x_i}{\sqrt{i}}\right)$$

5. **Rastrigin**. Continuous, separable, multimodal and convex. Defined in $[-5, 5]^D$:

$$f(\mathbf{x}) = 10D + \sum_{i=1}^{D} \left[x_i^2 - 10\cos(2\pi x_i)\right]$$

6. **Schwefel 2.26**. Continuous, differentiable, separable and multimodal. Defined in $[-500, 500]^D$:

$$f(\mathbf{x}) = -\frac{1}{D} \sum_{i=1}^{D} x_i \sin\left(\sqrt{|x_i|}\right)$$

7. **Salomon**. Continuous, non-separable, multimodal and non-convex. Defined in $[-100, 100]^D$:

$$f(\mathbf{x}) = 1 - \cos\left(2\pi \sqrt{\sum_{i=1}^{D} x_i^2}\right) + 0.1 \sqrt{\sum_{i=1}^{D} x_i^2}$$

8. **Generalized Penalized Function 1**. Continuous, non-separable and multimodal. Defined in $[-50, 50]^D$:

$$f(\mathbf{x}) = \frac{\pi}{D}\left[10\sin^2(\pi y_1) + \sum_{i=1}^{D-1} (y_i - 1)^2 \left[1 + 10\sin^2(\pi y_{i+1})\right] + (y_n - 1)^2\right] + \sum_{i=1}^{D} u_i$$

with

$$u_i = \begin{cases} 100(x_i - 10)^4 & x_i > 10 \\ 0 & -10 \leq x_i \leq 10 \\ 100(-x_i - 10)^4 & x_i < -10 \end{cases}, \qquad y_i = 1 + \frac{x_i + 1}{4}$$

9. **Generalized Penalized Function 2**. Continuous, non-separable and multimodal. Defined in $[-50, 50]^D$:

$$f(\mathbf{x}) = \sum_{i=1}^{D} u_i + 0.1\left[\sin^2(3\pi x_1) + (x_D - 1) + \sum_{i=1}^{D-1} (x_i - 1)^2 \left[1 + \sin^2(3\pi x_{i+1})\right]\right]$$

with

$$u_i = \begin{cases} 100(x_i - 5)^4 & x_i > 5 \\ 0 & -5 \leq x_i \leq 5 \\ 100(-x_i - 5)^4 & x_i < -5 \end{cases}$$

5.2 Execution parameters

This new proposed algorithm (RLSDE) was executed with the following parameter setting:

- Time of evolution: 3500 generations.
- Population size: 50 individuals.
- Dimension of the search space: $D = 100$.
- Scale factor center: $C = 0.4$.
- Scale factor radius: $r = 0.25$.
- Crossover probability: 0.5.
- Number of executions per experiment of each function: 30.

Considering the six new individuals evaluated in each generation by the local search, the total number of function evaluations per experiment is $3500\,(50 + 6) = 1.96E5$.

5.3 Results and comparison

The results obtained are shown in the same way that the authors presented the results of ELSDE, showing the average and the standard deviation of each function. One difference to highlight is that in ELSDE they used $1.00E6$ function evaluations for each experiment, whereas RLSDE (the proposed algorithm) used only $1.95E5$ function evaluations.

Table 1 shows the results obtained by DE, ELSDE, and RLSDE for the nine functions of the test suited considered with $D = 100$.

Fun	DE (mean ± std)	ELSDE (mean ± std)	RLSDE (mean ± std)
f_1	4.21E+03 ± 8.65E+02	2.07E-16 ± 1.06E-16	**1.18E-36 ± 2.62E-36**
f_2	**5.50E+01 ± 8.53E + 00**	8.21E+01 ± 7.95E-01	9.66E+01 ± 3.10E-01
f_3	2.63E+05 ± 2.01E+04	1.79E-09 ± 4.62E-10	**3.00E-15 ± 1.63E-15**
f_4	6.20E+01 ± 4.56E+00	**0.00E+00 ± 0.00E+00**	**0.00E+00 ± 0.00E+00**
f_5	2.40E+06 ± 1.38E+06	2.46E-10 ± 3.50E-10	**0.00E+00 ± 0.00E+00**
f_6	4.46E+03 ± 1.24E+03	**1.45E+01 ± 7.93E+01**	1.03E+02 ± 2.48E+02
f_7	4.19E+00 ± 1.81E+00	9.99E-02 ± 1.41E-06	**9.99E-02 ± 2.71E-12**
f_8	8.55E+02 ± 6.89E+01	1.07E-10 ± 5.22E-11	**4.71E-33 ± 0.00E+00**
f_9	8.96E+00 ± 6.68E-01	3.39E-04 ± 2.34E-03	**1.35E-32 ± 5.59E-48**

Table 1. Comparison of results with dimension $D = 100$

When the mean values obtained by ELSDE and RLSDE are statistically compared through the Wilcoxon Signed-Rank test, the difference between the value of the both algorithms and expected difference μ_0 is not big enough to be statistically significant. Despite this, it is observed that the overall performance of RLSDE is equivalent or better than that of ELSDE, due to the difference between the number of evaluations for both algorithms. Figures 1 a 9 shows the

output of one experiment on each of the functions. It should be noted that the functions f_4 and f_5 do not reach the 3500 generations in their graphic domain since the presented scales are logarithmic and the value obtained is zero. The quality of the solution obtained in F_1 can improve, it is necessary to increase the evolution time or improve the setting of the RLSDE parameters. The functions F_2, F_6 and F_7 show a rapid convergence and then stagnation, possibly the performance of the local search engine should be improved. Finally, in F_3, F_4, F_5, F_8 and F_9, the evolution time is well used, achieving quality convergence within the scheduled evolution time. This shows the power of RLSDE in high dimensions.

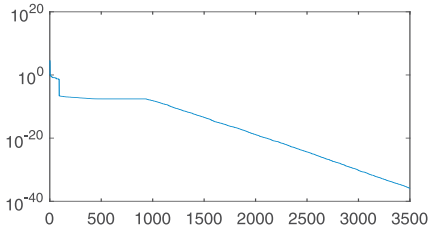

Fig. 1. f_1 - $\mathbf{x}_{best}$: $1.07E-36$

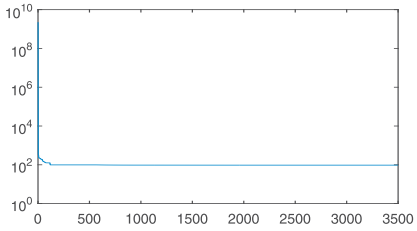

Fig. 2. f_2 - $\mathbf{x}_{best}$: $9.59E+01$

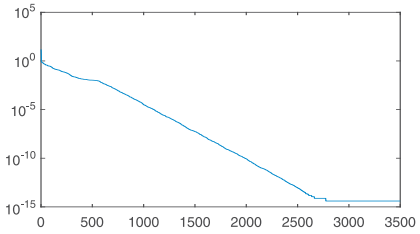

Fig. 3. f_3 - $\mathbf{x}_{best}$: $4.44E-16$

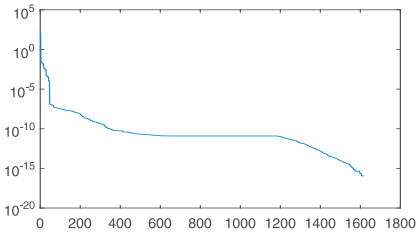

Fig. 4. f_4 - $\mathbf{x}_{best}$: $0.00E+00$

6 Conclusions

RLSDE, a new modified version of DE was presented. The differential mutation operator was modified, classifying the population, in each generation, into three groups according to their fitness. In this way, the base vector was chosen randomly but the vectors used for the difference could not belong to the same group as the base. In this way, the exploration of the search space is favored by

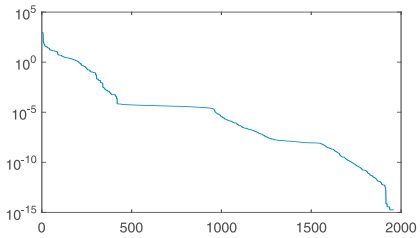

Fig. 5. f_5 - $\mathbf{x}_{best}$: $0.00E+00$

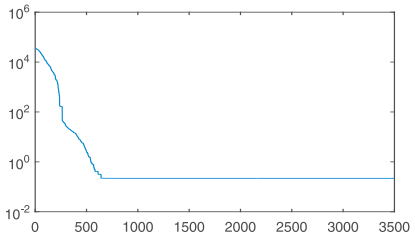

Fig. 6. f_6 - $\mathbf{x}_{best}$: $1.27E-03$

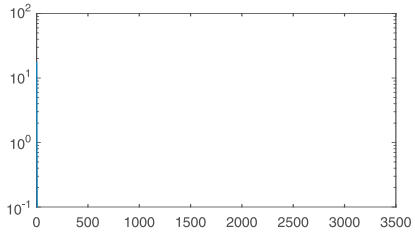

Fig. 7. f_7 - $\mathbf{x}_{best}$: $9.98E-02$

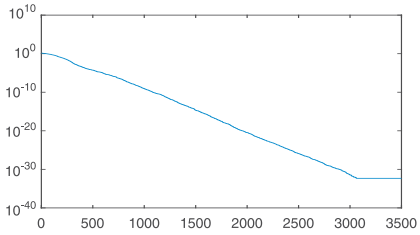

Fig. 8. f_8 - $\mathbf{x}_{best}$: $4.71E-33$

preventing three individuals of similar quality from interacting to generate the trial vector, assuming that these individuals possess very few differences from each other. Another improvement, also within the differential mutation operator, was the application of a random vector as a scale factor. By not applying a uniform value, each of the components was affected to a different extent, so the search space exploration was also favored.

Finally, to favor the exploitation of promising regions, a random local search engine based on distances was applied to the best individual of each generation.

The results obtained are interesting because they are comparable to those obtained by other similar algorithms, but with much less function evaluations.

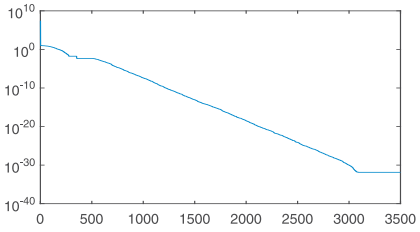

Fig. 9. f_9 - $\mathbf{x}_{best}$: $1.34E-32$

As future work we will try to solve some of the multiple CEC (IEEE) tests and we will try to improve the performance of this version by adjusting the execution parameters to obtain a better performance.

References

1. Cao, B., Zhao, J., Lv, Z., Liu, X., Yang, S., Kang, X., Kang, K.: Distributed parallel particle swarm optimization for multi-objective and many-objective large-scale optimization. IEEE Access **5**, 8214–8221 (2017). https://doi.org/10.1109/access.2017.2702561
2. Caraffini, F., Neri, F., Iacca, G.: Large scale problems in practice: The effect of dimensionality on the interaction among variables. In: Applications of Evolutionary Computation, pp. 636–652. Springer International Publishing (2017). https://doi.org/10.1007/978-3-319-55849-3-41
3. Chen, S., Montgomery, J., Bolufé-Röhler, A.: Measuring the curse of dimensionality and its effects on particle swarm optimization and differential evolution. Applied Intelligence **42**(3), 514–526 (nov 2014). https://doi.org/10.1007/s10489-014-0613-2
4. Deng, C., Dong, X., Yang, Y., Tan, Y., Tan, X.: Differential evolution with novel local search operation for large scale optimization problems. In: Advances in Swarm and Computational Intelligence, pp. 317–325. Springer International Publishing (2015). https://doi.org/10.1007/978-3-319-20466-6
5. Hiba, H., Mahdavi, S., Rahnamayan, S.: Differential evolution with center-based mutation for large-scale optimization. In: 2017 IEEE Symposium Series on Computational Intelligence (SSCI). IEEE (nov 2017). https://doi.org/10.1109/ssci.2017.8280938
6. Lin, H.: A differential evolution algorithm based on local search and boundary reflection for global optimization. 2015 11th International Conference on Computational Intelligence and Security (CIS) pp. 253–257 (2015)
7. Locatelli, M., Maischberger, M., Schoen, F.: Differential evolution methods based on local searches. Computers & OR **43**, 169–180 (2014)
8. Meselhi, M.A., Sarker, R.A., Essam, D.L., Elsayed, S.M.: Enhanced differential grouping for large scale optimization. In: Proceedings of the 10th International Joint Conference on Computational Intelligence. SCITEPRESS - Science and Technology Publications (2018). https://doi.org/10.5220/0006938902170224
9. Michiels, W., Aarts, E., Korst, J.: Theoretical Aspects of Local Search. Springer-Verlag GmbH (2006)
10. Noman, N., Iba, H.: Accelerating differential evolution using an adaptive local search. IEEE Transactions on Evolutionary Computation **12**, 107–125 (2008)
11. Price, K., Storn, R., Lampinen, J.: Differential evolution - A practical approach to global optimization. Springer-Verlag (2005)
12. Xiao-Gang, D., Chang-Shou, D., Zhang, Y., Yu-Cheng, T.: Enhancing local search of differential evolution algorithm for high dimensional optimization problem. In: 2015 34th Chinese Control Conference (CCC). IEEE (jul 2015). https://doi.org/10.1109/chicc.2015.7260973
13. Zhang, Q., Cheng, H., Ye, Z., Wang, Z.: A competitive swarm optimizer integrated with cauchy and gaussian mutation for large scale optimization. In: 2017 36th Chinese Control Conference (CCC). IEEE (jul 2017). https://doi.org/10.23919/chicc.2017.8028924