Proceedings of the IASTED International Conference
Artificial Intelligence and Applications (AIA 2014)
February 17 - 19, 2014 Innsbruck, Austria

USING RANDOM LOCAL SEARCH HELPS IN AVOIDING LOCAL OPTIMUM IN DIFFERENTIAL EVOLUTION

Miguel Leon Ortiz, Ning Xiong
School of Innovation, Design and Engineering
Mälardalen University
Västeras, Sweden
Email: miguel.leonortiz@mdh.se, ning.xiong@mdh.se

ABSTRACT
Differential Evolution is a stochastic and meta-heuristic technique that has been proved powerful for solving real-valued optimization problems in high-dimensional spaces. However, Differential Evolution does not guarantee to converge to the global optimum and it is easily to become trapped in a local optimum. In this paper, we aim to enhance Differential Evolution with Random Local Search to increase its ability to avoid local optimum. The proposed new algorithm is called Differential Evolution with Random Local Search (DERLS). The advantage of Random Local Search used in DERLS is that it is simple and fast in computation. The results of experiments have demonstrated that our DERLS algorithm can bring appreciable improvement for the acquired solutions in difficult optimization problems.

KEY WORDS
Differential Evolution, random local search, local search, evolutionary algorithms, local optimum, global optimization

1 Introduction

Optimization is an important issue in solving many real world problems in industrial and scientific applications. Generally, numerical optimization techniques can be divided into two categories: point-based transition and population-based transition [1]. Population-based transition works with a population of possible solutions, and new generations are created from old populations to progressively improve the precision of the results. Over the last years, various models of genetic algorithms [2] and memetic algorithms [3], [4] were proposed as representatives of the population-based schemes, which have been shown to be powerful means to solve many real-world optimization problems.

Differential Evolution (DE) is another category of the population-based approaches for optimization. It is a stochastic and meta-heuristic evolutionary algorithm, first proposed in 1997 [5]. After that a lot of works have been done by many researchers to further improve its performance. Previous researches have demonstrated that DE represents a powerful tool to solve problems with real-valued parameters [6], [7], and also in high-dimensional spaces [8].

However, there is one problem that DE sometimes converges too fast. This would be good for the majority of unimodal functions yet a risk for multi-modal functions. The search process with DE is very likely to get stuck into a local optimum in the multimodal situations and there is no mechanism in classic DE to reduce the chance of this to happen.

In this paper we aim to create a new function into DE to increase its ability to avoid local optimum, in our case, local minimum. Our idea is to combine Random Local Search (RLS) into a classic DE algorithm. Although this topic has been addressed by other researchers (see examples in [9], [10], [11], [12], and [13]), the local search strategies used in other works are different from ours and are more computational expensive. The advantages of the RLS strategy employed in our DE algorithm is that it is simple and fast in computation. The results of experiments have demonstrated that the simple RLS strategy combined with DE can lead to substantial improvement of results in difficult optimization problems.

The rest of the paper is organized as follows. Section 2 briefly describes some related works. Classic DE algorithm and a proposed algorithm of DE combined with RLS are explained in Section 3. Section 4 presents and compares the results. Finally, in Section 5 the paper concludes with remarks on conclusion and future works.

2 Related Work

Since the first proposal of DE in 1997 [14], a lot of works have been done to improve the search ability of this algorithm.

Ali, Pant and Nagar [9] proposed two different local search algorithms, namely Trigonometric Local Search and Interpolated Local Search, which were applied to refine the best solution and two random solutions in every generation respectively.

Local search differential evolution was developed in [10] where a new local search operator was used on every individual in the population with a probability. The search strategy attempted to find a random better solution between trial vector and the best solution in the generation.

DOI: 10.2316/P.2014.816-021

Dai, Zhou, Zhang and Jiang [11] combined Orthogonal Local Search with DE in the so-called OLSDE (Orthogonal Local Search Differential Evolution) algorithm. Therein two individuals were randomly selected from the population at each iteration for local refinements.

Pei-chong, Xu and Xiao-hong [12] proposed a DE with Chaos Local Search (CLS). CLS was applied on 20% of the population after running of original DE to obtain superior global convergence and robustness in solving global optimization problem.

Poikolainen and Neri [13] proposed a DE with Deterministic Local Search; this algorithm is called Differential evolution with concurrent fitness based local search (DEcfbLS). They apply Local Search in the most promising solutions.

The random local search method developed for this paper differs from the aforementioned works in that it inherently is a limited hill climbing search. Random exploration is performed successively on different single variables and only a small subset of the variables are randomly selected to be involved in the local search process. This local search method is very attractive in being well scalable to large optimization problems. It is naturally adequate to solve separable problems. However, in cooperation with DE, it is capable of exploring along all dimensions of the search space at the same time.

3 Proposed Algorithm to Avoid Local Optimum

In this Section we will first introduce the classic DE approach and then present our proposed algorithm that combines classic DE with RLS.

3.1 Classic Differential Evolution

DE is an algorithm based on population with N_p individuals. Every individual in the population represents a possible solution to the problem to be solved. One individual in the population is represented by $X_{i,g}$ with $i = 1, 2, \ldots, N_p$ and g is the index of the generation. DE has three main operators: mutation, recombination and selection. The explanation of these operators is given below:

MUTATION. This operation creates a different solution for every individual using other individuals in the same population. The vector for the mutated solution is called noisy vector and it is represented like $V_{i,g}$. There are a few possible ways to mutate the current population [15] and the meaning of the name is DE/x/y/z, where x means the vector to be mutated, y is the number of difference vectors used in the equation and z denotes the recombination used, we always uses binomial and it will be explained below; three of them are listed below

- DE/rand/1/bin:

$$V_{i,g} = X_{r_1,g} + F \times (X_{r_2,g} - X_{r_3,g}) \qquad (1)$$

- DE/best/1/bin:

$$V_{i,g} = X_{best,g} + F \times (X_{r_1,g} - X_{r_2,g}) \qquad (2)$$

- DE/current-to-best/1/bin:

$$V_{i,g} = X_{i,g} + F_1 \times (X_{best,g} - X_{i,g}) + F_2 \times (X_{r1,g} - X_{r2,g}) \qquad (3)$$

where $r1, r2, r3 \in \{1, 2, \ldots, N_p\}$ are randomly created integers and F, F_1 and F_2 are constant factors in the interval (0,2]. $X_{best,g}$ means the best solution in the generation g and $X_{i,g}$ represents individual i in the generation g. The Eq. (1) means for every individuals in the population we choose one randomly individual and we plus this one with the difference between others two randomly individuals multiply for a constant. The Eq. (2) is the same than Eq. (1) but we change the first randomly individual for the best individual in that generation. In Eq. (3) we plus the current value with a difference between the best individual and we plus this with difference between two random individuals, those two differences need to be multiply for constants.

RECOMBINATION. This operation combines all the individuals in the population with the noisy vectors created in the mutation stage. Such new solutions created are called trial vectors and we use $T_{i,g}$ to represent the trial vector corresponding to individual i in the generation g. The parameters in the trial vector are decided according to Eq. (4).

$$T_{i,g}[j] = \begin{cases} V_{i,g}[j] & \text{if } rand[0,1] < Pr \ \ or \ \ j = j_{rand} \\ X_{i,g}[j] & \text{otherwise} \end{cases} \qquad (4)$$

where j represents the index of parameters in a vector, j_{rand} is a random number between 1 and N_p to ensure that at least one value from the noisy vector will be delivered to the trial vector and Pr is the probability of recombination. Eq. (4) means that we choose randomly every component between the individual $X_{i,g}$ and the noisy vector $V_{i,g}$.

SELECTION. By this operation we compare a trial vector and its parent solution in the population to decide the winner to enter the next generation. Hence, if the problem of interest is minimization, the individuals in the new generation are generated using Eq. (5) as follows:

$$X_{i,g+1} = \begin{cases} T_{i,g} & \text{if } f(T_{i,g}) < f(X_{i,g}) \\ X_{i,g} & \text{otherwise} \end{cases} \qquad (5)$$

where $f(T_{i,g})$ means the fitness of $T_{i,g}$ and $f(X_{i,g})$ denotes the fitness of $X_{i,g}$. the Eq. (5) chooses the best individual between $X_{i,g}$ and $T_{i,g}$ comparing their fitness values.

The Pseudocode of differential evolution is giving below:

Differential Evolution

1. Initialize the population with randomly created individuals.
2. Calculate the fitness values of all individuals in the population
3. While the termination condition is not satisfied do
4. Create noisy vectors using a mutation strategy in Eq. (1), (2), (3)
5. Create trial vectors by recombining noisy vectors with parent vectors according to Eq. (4).
6. Evaluate trial vectors with their fitness values
7. Select winning vectors according to Eq. (5) as individuals in the new generation
8. End while.

3.2 Differential Evolution Enhanced with Random Local Search (DERLS)

In order to increase the capability to escape from a local optimum, we propose to incorporate a random local search (RLS) strategy into the classic DE algorithm. The RLS strategy will be applied every time when a new generation is created for further improvement. The idea of our algorithm is use RLS in each generation of DE. When one generation of DE ends, then, we apply RLS only to the best solution, trying to avoid the local optimum.

The basic idea of local search is to explore in the closest neighborhood. There are actually a number of alternatives for local search, such as Gradient search [1] and Simplex method [16], but in our paper, we adopt RLS strategy to allow more randomness while still pressing down extra computational cost.

With our RLS strategy, we attempt to avoid the local optimum by doing small random "jumps" into more promising regions in the space of solution. We could use this strategy to the whole population but the problem is that, if we do that, we spend a lot of time and resources trying to improve bad solution. Therefore our RLS tries to find better solution from the best solution in every generation. To do that, this RLS choose 10% elements of the best solution, changing these values to others values randomly in the range of the problem, creating a new solution. If the RLS does not improve the solution, it tries again with the same process M tries without improvement. If RLS improve the actual best solution, then we restart the tries and we change the best solution for this new solution.

The pseudocode of RLS is given below:

RLS (applied to the best individual):

1. Set $i = 1$;
2. while ($i <= M$) Do
3. $Candidates = \{1, 2, \ldots, dimension\}$
4. Set $j = 1$
5. while $j < \alpha \times dimension$ Do
6. Randomly selection k from Candidates
7. Assign a random possible value to parameter k of the vector
8. Remove k from Candidates;
9. Set $j = j + 1$
10. End while (line 5)
11. If this new solution is better than the parent, then
12. Replace the parent solution with the new one, set $i = 1$
13. else set $i = i + 1$
14. End if
15. end while (line 2)

in the RLS algorithm, M is equals to the numbers of times that we are trying to improve the current best solution and α is the percent of the totally values of one individual that we need to change.

The proposed DE algorithm with combined RLS is formulated with the pseudocode in the following. First we need to mutate the population, and a few possible ways are available for this operation. In the experiments shown later, we only used the mutation strategies in Eq. (1) and (2). After mutation, we do crossover to recombine noisy and parent vectors according to Eq. (4). Then the winning solutions are selected based on comparison of the fitness values of the trial and parent vectors, according to Eq. (5). The winning solutions form a new generation and the best individual from it is identified. Subsequently RLS is executed on the best solution of the population for possible improvements. If a new better solution is found by RLS, it is introduced into the population as replacement of an old individual.

The working procedure of DERLS is formulated as follows

DERLS:

1. Initialize population with random individuals
2. Evaluate the individuals in the population with fitness values
3. while the termination condition is not met Do
4. Mutate the population according Eq. (1), (2) or (3)
5. Recombine noisy vectors with parent vectors according Eq. (4)
6. Select the winning vectors as individuals of the new generation according to Eq. (5)
7. Choose the best individual from the population
8. Apply RLS to the best individual
9. Actualize the best individual
10. End while

Table 1. The eight functions used in the experiments

FUNCTION	RANGE	MINIMUM
f1(x) = $\sum_{i=1}^{n} x_i^2$	[-100,100]	0
f2(x) = $\sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	[-10,10]	0
f3(x) = $\sum_{i=1}^{n} (\sum_{j=1}^{i} x_j)^2$	[-100,100]	0
f4(x) = $\max_i \{\lvert x_i \rvert, 1 \leq i \leq n\}$	[-100,100]	0
f5(x) = $\sum_{i=1}^{n-1} [100 \times (x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	[-30,30]	0
f6(x) = $\sum_{i=1}^{n} (x_i + 0.5)^2$	[-100,100]	0
f7(x) = $\sum_{i=1}^{n} i \times x_i^4 + random[0,1)$	[-1.28,1.28]	0
f8(x) = $\sum_{i=1}^{n} -x_i \times sin(\sqrt{\lvert x_i \rvert})$	[-500,500]	-12569.5
f9(x) = $\sum_{i=1}^{n} [x_i^2 - 10 \times cos(2 \times \pi \times x_i) + 10]$	[-5.12,5.12]	0
f10(x) = -20 $\times exp(-0.2 \times \sqrt{\frac{1}{n} \times \sum_{i=1}^{n} x_i^2}) - exp(\frac{1}{n} \times \sum_{i=1}^{n} cos(2\pi x_i)) + 20 + e$	[-32,32]	0
f11(x) = $\frac{1}{4000} \times \sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} cos(\frac{x_i}{\sqrt{i}}) + 1$	[-600,600]	0

4 Experiments and Results

To examine the capability of the proposed algorithm in solving real-valued optimization problems, we used the eleven mathematical functions [17] listed in Table 1 in the experiments to compare the results found by our algorithm and true global optimal solutions of these functions. Functions 1, 2, 3, 4, 5, 6 and 7 in the table are unimodal and functions 8, 9, 10 and 11 are multi-modal. All these functions contain 30 parameters/variables. We ran our algorithm 50 times on each function in attempts to find best solutions for them. The numbers of evaluations performed in optimizing the respective functions are given in Table 2. We had this number of evaluation because we ran the experiments with the same number of generation than [17]. In our experiments we used $M = 5$, $\alpha = 0.1$, $N_p = 60$, $F = 0.9$ and $P_r = 0.85$.

The experiments were conducted as follows. First, we compared classic DE with our DERLS algorithm both using DE/rand/1/bin as the mutation strategy. Secondly, we employed the DE/best/1/bin mutation strategy to the classic DE and DERLS algorithm for comparison. Finally, we compared both mutation strategies (DE/rand/1/bin and DE/best/1/bin) in classic DE and then demonstrated how the inferior strategy could be enhanced by integration RLS. The results in boldface mean that these variants are the best option to that function

4.1 DERLS versus Classic DE with DE/rand/1/bin Strategy

First, we intended to compare classic DE and DERLS under the DE/rand/1/bin mutation strategy. The experiment results obtained on the eight test functions are illustrated in Table 3. This table shows us the mean errors and standard deviations that were obtained with experiments on all the functions. Throughout the paper we use error to denote the difference between the best result acquired and the true global optimum value.

Table 2. Evaluations per function

FUNCTION	NUM. EVALUATION
F1	90000
F2	120000
F3	300000
F4	300000
F5	1200000
F6	90000
F7	180000
F8	540000
F9	300000
F10	90000
F11	120000

We can see in this table that DERLS acquired better results in functions 3, 4, 5, 6, 8, 10 and 11. Particularly, in the most challenging task of function 8, the improvement with DERLS is highly significant, decreasing mean error from 1470 to 270. On functions 1, 2, 7, and 9 DERLS did not dominate over classic DE, but the results were quite similar with trivial differences. Therefore we can conclude that DERLS generally is stronger than classic DE if both adopt the DE/rand/1/bin mutation strategy.

4.2 DERLS versus Classic DE with DE/best/1/bin Strategy

In this subsection, we shall compare DERLS and classic DE using DE/best/1/bin as the mutation strategy. Our purpose is to check, with the other mutation strategy, whether we can obtain the same conclusion as that we got before with the DE/rand/1/bin strategy. The experiments produced the results as listed in Table 4.

.

Table 3. Comparison of results under DE/rand/1/bin strategy

FUNCTION	DE		DERLS		DE-DERLS
	ERROR	Std.DEV	ERROR	Std.DEV	
F1	**2,28E-02**	1,90E-02	1,36E-01	5,49E-02	-1,13E-01
F2	**1,49E-02**	4,53E-03	3,69E-02	1,77E-01	-2,20E-02
F3	6,06E+01	3,30E+01	**3,08E+01**	1,75E+01	2,98E+01
F4	4,62E+00	4,36E+00	**2,41E+00**	2,84E+00	2,21E+00
F5	3,99E-01	1,21E+00	**1,60E-01**	7,90E-01	2,39E-01
F6	2,31E-01	1,17E-01	**1,92E-01**	3,18E-01	3,90E-02
F7	**2,14E-02**	6,11E-03	2,59E-02	1,51E-02	-4,50E-03
F8	1,47E+03	4,08E+02	**2,70E+02**	1,59E+02	1,20E+03
F9	**1,31E+01**	3,83E+00	1,86E+01	7,22E+00	-5,50E+00
F10	1,99E+01	1,36E-02	**1,71E+00**	4,46E+00	1,82E+00
F11	3,45E-05	2,35E-05	**2,86E-05**	3,12E-05	5,90E-06

Table 4. Comparison of results under DE/best/1/bin strategy

FUNCTION	DE		DERLS		DE-DERLS
	ERROR	Std.DEV	ERROR	Std.DEV	
F1	**2,04E-06**	2,23E-06	3,20E+00	1,43E+01	-3,20E+00
F2	**3,53E-05**	1,65E-04	8,61E-01	1,74E+00	-8,61E-01
F3	5,00E+02	1,82E+03	**1,59E-02**	1,42E-02	5,00E+02
F4	**1,62E-02**	2,87E-02	1,66E-02	2,15E-02	-4,00E-04
F5	**1,12E+00**	1,81E+00	2,67E+00	5,93E+00	-1,55E+00
F6	**1,59E-06**	1,64E-06	6,52E+01	1,51E+02	-6,52E+01
F7	**1,70E-02**	5,29E-03	9,73E-02	2,12E-01	-8,03E-02
F8	3,09E+03	8,77E+02	**4,70E+02**	1,95E+02	2,62E+03
F9	4,43E+01	1,17E+01	**2,61E+01**	8,08E+00	1,82E+01
F10	1,43E+01	8,88E+00	**3,19E+00**	4,93E+00	1,11E+01
F11	**0,00E+00**	0,00E+00	5,46E-03	1,71E-02	-5,46E-03

Table 5. Comparison of results of two mutation strategies

FUNCTION	DE/best/1/bin		DE/rand/1/bin		DE b-DE r
	ERROR	Std.DEV	ERROR	Std.DEV	
F1	**2,04E-06**	2,23E-06	2,28E-02	1,90E-02	-2,28E-02
F2	**3,53E-05**	1,65E-04	1,49E-02	4,53E-03	-1,49E-02
F3	5,00E+02	1,82E+03	**6,06E+01**	3,30E+01	4,39E+02
F4	**1,62E-02**	2,87E-02	4,62E+00	4,36E+00	-4,60E+00
F5	1,12E+00	1,81E+00	**3,99E-01**	1,21E+00	7,21E-01
F6	**1,59E-06**	1,64E-06	2,31E-01	1,17E-01	-2,31E-01
F7	**1,70E-02**	5,29E-03	2,14E-02	6,11E-03	-4,40E-03
F8	3,09E+03	8,77E+02	**1,47E+03**	4,08E+02	1,62E+03
F9	4,43E+01	1,17E+01	**1,31E+01**	3,83E+00	3,12E+01
F10	**1,43E+01**	8,88E+00	1,99E+01	1,36E-02	-5,60E+00
F11	**0,00E+00**	0,00E+00	3,45E-05	2,35E-05	-3,45E-05

Table 6. Comparison DERLS best strategy with DE rand strategy

FUNCTION	DERLS with DE/best/1/bin		DE/rand/1/bin		DERLS b-DE r
	ERROR	Std.DEV	ERROR	Std.DEV	
F1	3,20E+00	1,43E+01	**2,28E-02**	1,90E-02	3,18E+00
F2	8,61E-02	1,74E+00	**1,49E-02**	4,53E-03	7,12E-02
F3	**1,59E-02**	1,42E-02	6,06E+01	3,30E+01	-6,06E+01
F4	**1,66E-02**	2,15E-02	4,62E+00	4,36E+00	-4,60E+00
F5	2,67E+00	5,93E+00	**3,99E-01**	1,21E+00	2,27E+00
F6	6,52E+01	1,51E+02	**2,31E-01**	1,17E-01	6,50E+01
F7	9,73E-02	2,12E-01	**2,14E-02**	6,11E-03	7,59E-02
F8	**4,70E+02**	1,95E+02	1,47E+03	4,08E+02	-1,00E+03
F9	2,61E+01	8,08E+00	**1,31E+01**	3,83E+00	1,30E+01
F10	**3,19E+00**	4,93E+00	1,99E+01	1,36E-02	-1,67E+01
F11	5,46E-03	1,71E-02	**3,45E-05**	2,35E-05	5,43E-03

We can observe the results of these algorithms on Table 4. First, we are going to compare both algorithms on unimodal functions (functions 1 to 7). Although classic DE performed better in 6 of the 7 (unimodal) functions, the differences in three of them are actually minor with the difference values being 3, 1.5 and 0.00546. Considerable differences between DERLS and classic DE appeared on functions 3 and 6. In function 3, DERLS almost found the true optimum while the classic DE got a big, unacceptable error of 500. On function 6, the results from DERLS were still acceptable though they were inferior to those of classic DE. From the above comparisons, we can claim that DERLS would be a more robust alternative if considering a wide variety of unimodal functions.

Next we attempt to compare both algorithms in multimodal functions. We can see in the table that DERLS performed better than classic DE in three of the four functions (functions 8, 9 and 10). Moreover, it is important to note that the improvement acquired by DERLS on function 8 as the most difficult problem is extremely significant, with the difference value being 2620. Finally, in function 11, the solution found by DERLS is so close to the true optimum that its difference with that of classic DE is completely ignorable. Based on the above analysis, we can point out that DERLS would be more appealing to apply to remain robust in various situations and, in particular, to escape from local optimum in multimodal functions.

4.3 DERLS with DE/best/1/bin Strategy versus Classic DE with DE/rand/1/bin Strategy

In this subsection, we want to compare the two mutation strategies: DE/rand/1/bin and DE/best/1/bin. Then RLS will be integrated with the inferior mutation strategy to see if performance improvement would be possible by means of local search.

First, we compare both strategies without local search and the results are illustrated in Table 5. In Table 5 the DE/best/1/bin mutation strategy seems to be better in seven functions than DE/rand/1/bin, however the differences are very small and actually unimportant. On the other side, the DE/rand/1/bin strategy is much better in function 3, 8 and 9 bringing significant improvements. So we can conclude, with these results, that classic DE with DE/rand/1/bin strategy outperforms that with the other strategy

With the conclusion drawn above, we are now going to improve DE with DE/best/1/bin mutation strategy by integrating it with RLS . Then we compare the results of DERLS with DE/best/1/bin strategy against classic DE with DE/rand/1/bin mutation strategy. The results are given in Table 6.

With the first glimpse, we see that classic DE with DE/rand/1/bin mutation strategy is better than DERLS with DE/best/1/bin strategy in seven of the eleven functions.

However, the more important in our analysis is not if it is better or not on the surface, but whether it dominates its counterpart significantly. With this view in mind, we shall compare the results on the unimodal functions and multimodal functions respectively in the following.

In unimodal functions, classic DE is better in five of the seven functions. There are 2 big differences in function 3 and function 6. DERLS is much better than DE in function 3, while in function 6 classic DE obviously dominates. Hence we can claim that classic DE with DE/rand/1/bin mutation strategy is likely to perform better than DERLS with DE/best/1/bin strategy in most unimodal functions.

In multimodal functions, classic DE seems to outperform in two of the four functions. But now we would like to consider more on the magnitude of the differences. In function 8 DERLS is much better than classic DE, with the mean error being 470 that is much smaller than the error of 1470 from classic DE. We should also notice that function 8 is a very difficult problem, on which the classic DE with DE/best/1/bin mutation strategy obtained a large error at 3090. So we can see an extremely important improvement here by inducing the local search. Also in function 10

Table 7. DERLS versus Evolutionary Programming

FUNCTION	DERLS		CEP[17]		FEP[17]	
	ERROR	Std.DEV	ERROR	Std.DEV	ERROR	Std.DEV
F8	2,70E+02	1,59E+02	4,65E+03	6,35E+02	**1,50E+01**	5,26E+01
F9	1,86E+01	7,22E+00	8,90E+01	2,31E+01	**4,60E-02**	1,20E-02
F10	1,71E+00	4,66E+00	9,20E+00	2,80E+00	**1,80E-02**	2,10E-03
F11	**2,86E-05**	3,12E-05	8,60E-02	1,20E-01	1,60E-02	2,20E-02

Table 8. DERLS versus others algorithms

FUNCTION	DERLS		PSO[18]		arPSO[18]		SEA[18]	
	ERROR	Std.DEV	ERROR	Std.DEV	ERROR	Std.DEV	ERROR	Std.DEV
F8	**2,70E+02**	1,59E+02	5,38E+03	6,72E+02	3,97E+03	2,07E+03	9,01E+02	2,34E+02
F9	1,86E+01	7,22E+00	4,29E+01	1,62E+01	2,15E+00	4,91E+00	**7,18E-01**	9,22E-01
F10	1,71E+00	4,66E+00	1,40E+00	7,91E-01	1,87E-07	7,15E-08	**1,05E-02**	9,08E-04
F11	**2,86E-05**	3,12E-05	2,35E-02	3,42E-02	9,23E-02	3,41E-01	4,64E-03	3,96E-03

DERLS is better than DE with a difference of 16.7. In function 9 classic DE seems better, but difference between the results is only 13. Again, in function 11, the difference is tiny and both DERLS and classic DE got results extremely close to the true global minimum. Therefore we can claim that DERLS with DE/best/1/bin mutation strategy has stronger ability than DE with DE/rand/1/bin mutation strategy to deal with difficult multimodal functions.

4.4 DERLS versus Others Algorithms

In this last subsection we intend to compare DERLS with several other evolutionary computing methods. This comparison will be done only on the multimodal functions (functions 8, 9, 10 and 11), since our main goal is to help DE in avoiding local optimum, which merely occurs in multi-modal cases. We divide the experiments in two different parts: first we compare DERLS against classical evolutionary programming (CEP) and fast evolutionary programming (FEP) methods, and secondly we compare the performance of DERLS with that of particle swarm optimization (PSO), attractive and repulsive PSO (arPSO) and simple evolutionary algorithm (SEA). The results of CEP and FEP on functions 8, 9, 10 and 11 were reported in[17] and the results of PSO, arPSO and SEA on the same functions were given in [18].

The results of the first comparison are illustrated in Table 7. In this table we can see that DERLS is always better than CEP on all the four functions. Compared with FEP, DERLS is better in function 11 yet inferior on the other three functions. The superiority of FEP on most multimodal functions may possibly be explained by the Cauchy mutation function used, which enables very good search ability in a large neighborhood.

The results for the second part of comparison are shown in Table 8. We can see that DERLS obviously dominates PSO, arPSO and SEA on functions 8 and 11. Especially, on function 8, the differences of results between DERLS and the other compared methods are quite large. On function 9, DERLS outperforms standard PSO but behaves less good than arPSO and SEA. Finally, on function 10, the result acquired by DERLS is comparable to those of others with only minor differences.

Based on the above analysis, we would like to rank FEP as the best and DERLS as the second best for tackling multimodal functions.In particular, DERLS performs much better than the others (except FEP) in function 8, which is the most difficult problem to solve in the experiments.

5 Conclusion

In this paper, we propose DERLS as a new DE algorithm which embeds random local search in its structure. We examined the capability of DERLS in solving real-valued optimization problems in experiments and the results we obtained are favorable. It has been shown that DERLS is stronger in multimodal functions than classic DE. Moreover, DERLS is not difficult to implement and it takes the same time in calculation as classic DE. On the other hand, in unimodal functions, DERLS is not able to bring meaningful improvement.

Further work for improvement of DERLS will be carried out in non-fixed neighborhood for local search as well as the adaptive control of the intensity of random local search during the evolutionary process. As direct applications, we are going to employ DERLS to solve various industrial design problems such as parameter optimization for analog circuits. We will also apply and test DERLS in machine learning tasks such as automatic learning of fuzzy knowledge bases [19], [20] and similarity model construction [21], [22] in case-based reasoning.

Acknowledgements

The work is funded by the Swedish Knowledge Founda-

tion (KKS) grant (project no 16317). The authors are also grateful to ABB FACTS, Prevas and VG Power for their co-financing of the project.

References

[1] N. Xiong and M. Leon, Principles and state-of-the-art of engineering optimization techniques, *Proc. The Seventh International Conference on Advanced Engineering Computing and Applications in Sciences, ADVCOMP 2013, Porto, Portugal*, 2013, 36–42.

[2] F. Herrera and M. Lozano, Two-loop real-coded genetic algorithms with adaptive control of mutation step size, *Applied Intelligence*, 13, 2000, 187–204.

[3] D. Molina, M. Lozano, A. M. Sanchez, and F. Herrera, Memetic algorithms based on local search chains for large scale continuous optimization prob-lems: Massw-chains, *Soft Computing*, 15, 2011, 2201–2220.

[4] D. Molina, M. Lozano, C. Garcia-Martinez, and F. Herrera, Memetic algorithms for continuous optimization based on local search chains, *Evolutionary Computation*, 18, 2010, 27–63.

[5] R. Storn and K. Price, Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces, *Journal of Global Optimization*, 11(4), 1997, 341 – 359.

[6] X. Song and L. Tang, A novel hybrid differential evolution-estimation of distribution algorithm for dynamic optimization problem, *Proc. 2013 IEEE Congress Congress on Evolutionary Computation (CEC), Cancun, Mexico*, 2013, 1720–1717.

[7] A. Mandal, A.K. Das, P. Mukherjee, and S. Das, Modified differential evolution with local search algorithm for real world optimization, *Proc. 2011 IEEE congress on Evolutionary Computation (CEC), New Orleans, USA*, 2011, 1565–1572.

[8] C. Worasucheep, High-dimensional function optimization with a self adaptative differential evolution, *Proc. IEEE International Conference on Intelligent Computing and Intelligent Systems, Shanghai*, 2009, 668 – 673.

[9] M. Ali, M. Pant, and A. Nagar, Two local search strategies for differential evolution, *Proc. Bio-Inspired Computing: Theories and Applications (BIC-TA), 2010 IEEE Fifth International Conference, Changsha, China*, 2010, 1429 – 1435.

[10] Gu Jirong and Gu Guojun, Differential evolution with a local search operator, *Proc. 2010 2nd International Asia Conference on Informatics in Control, Automation and Robotics (CAR), Wuhan, China*, 2010, 480 – 483.

[11] Z. Dai and A. Zhou, A diferential ecolution with an orthogonal local search, *Proc. 2013 IEEE congress on Evolutionary Computation (CEC), Cancun, Mexico*, 2013, 2329 – 2336.

[12] W. Pei-chong, Q. Xu, and H. Xiao-hong, A novel differential evolution algorithm based on chaos local search, *Proc. International conference on information Engineering and Computer Science, ICIECS 2009, Wuhan, China*, 2009, 1–4.

[13] I. Poikolainen and F. Neri, Differential evolution with concurrent fitness based local search, *Proc. 2013 IEEE Congress on Evolutionary Computation (CEC), Cancun, Mexico*, 2013, 384–391.

[14] R. Storn and K. Price, Differential evolution - a simple and efficient adaptive scheme for global optimiza-tion over continuous spaces, Tech rep. tr-95-012, Comput. Sci. Inst., Berkeley, CA, USA, 1995.

[15] W. Gong and Z. Cai, Differential evolution with ranking-based mutation operators, *IEEE Transactions on Cybernetics*, 2013, 2066-2081.

[16] J.S. Arora, O.A. Elwakeil, and A.I. Chahande, Global optimization methods for engineering applications: a review, *Structural optimization*, 9, 1995, 137–159.

[17] X. Yao, Y. Liu, and G. Lin, Evolutionary programming made faster, *IEEE Transactions on Evolutionary Computation*, 3, 1999, 82–102.

[18] J. Vesterstrom and R. Thomsen, A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems, Proc. *Congress on Evolutionary Computation, CEC2004*, 2004, 1980–1987.

[19] N. Xiong and P. Funk, Construction of fuzzy knowledge bases incorporating feature selection, *Soft Computing,* 10(9), 2006, 796–804.

[20] N. Xiong, Evolutionary learning of rule premises for fuzzy modeling, *International Journal of Systems Science*, 32(9), 2001, 1109–1118.

[21] N. Xiong and P. Funk, Combined feature selection and similarity modeling in case-based reasoning using hierarchical memetic algorithm, *Proc. of the IEEE World Congress on Computational Intelligence*, 2010, 1537–1542.

[22] N. Xiong, Learning fuzzy rules for similarity assessment in case-based reasoning, *Expert Systems and Applications*, 38, 2011, 10780–10786.