Differential Evolution with (μ+λ)-Selection Technique

Miao-miao Liu[1], Pan Huo[2] and Wen-yin Gong[3,*]

[1] School of Computer Science, China University of Geosciences, Wuhan, 430074, P. R. China

[2] School of Computer Science, China University of Geosciences, Wuhan, 430074, P. R. China

[3] School of Computer Science, China University of Geosciences, Wuhan, 430074, P. R. China

[*] Corresponding author

Abstract

Differential Evolution (DE) is a global numerical optimization algorithm which is robust, easy to use, and lends itself very well to parallel computation. The original DE adopts one-to-one tournament selection to select the individuals surviving in the next generation, which maybe lead to the individuals with lower object function values abandoned. In the evaluation strategy, (μ+λ)-selection has been proved to be an effective selection. In this selection method, we compare all the individuals mixed with the parents and the offspring, and the individuals with lower fitness must survive in the next generation. Inspired by this, we proposed an improve method to improve the speed of convergence of DE in the paper. We combine this selection with five DE mutation strategies, and testing them with the 13 benchmark functions, and most of them have better performance compared with the original DE in the same condition.

Keywords: *DE, (μ+λ)-selection, (μ+λ)-DE, replacement*

1. Introduction

Differential Evolution (DE) is proposed by Price and Storn in 1995 [1-2], which is an simple yet effective global numerical optimization algorithm. DE algorithm has obtained better performance than most of optimization algorithms in terms of robustness and the speed of convergence over common benchmark functions and real-world problems[3-6]. It has been successfully applied to diverse domains of science and engineering, such as mechanical engineering design, signal processing, chemical engineering, machine intelligence, and pattern recognition, etc. More details related DE applications can be found in [7-13], and the references therein.

It's already proved that the original DE can be improved by many authority literature, and improved algorithms have been put forward, such as MDE[14], ODE[15], jDE[16], JADE[17],and more. But they all use one-to-one tournament selection to select individuals, which is to compare offspring with its parents, so the individuals which have better fitness may be abandoned in the evolution process, and the diversity of the population can not be guaranteed. The origial DE with (μ+λ)-selection ((μ+λ)-DE for short) can solve these problems well, because it selects the individuals from a mixed population mixed with all the parents and offspring, and the elitism must survive in the next generation. We combine the (μ+λ)-selection with five mutation strategies, and compare them with the original algorithm in the same condition. And we set high and low dimension conditions to test the performance of the original DE and the (μ+λ)-DE, and discuss it for different population size. According to the results, we can conclude that (μ+λ)-DE has faster speed of convergence and the variety of the population can also be guaranteed, especially for the high dimension and large population size problems.

The rest of this paper is organized as follows: section 2 makes a brief the original DE and the (μ+λ)-ES. In section 3, we will explain our improvement idea and method in details. The specific experiment settings and results are show in section 4, and we also make analysis of the experiment results in this section. In the last section, we make a summarization to (μ+λ)-DE.

2. Prerations

DE is a simple yet effective global algorithm. It mainly consists of four operations, which are initialization, mutation, crossover and selection. The (μ+λ)-selection is originally proposed in (μ+λ)-ES, which selects the individuals from the parent and offspring population.

2.1 DE

We make a brief instruction to original DE algorithm. DE is a simple yet effective algorithm. The original DE mainly

includes four operations: initialization, mutation, crossover and selection, and their relations are as Fig.1:[18]

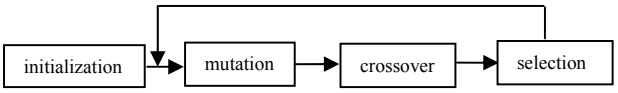

Fig.1 Four major steps in DE

2.1.1 Initialization

DE adopts random initialization. The i^{th} individual in the population is represented by $X_{i,G}$, and

$$X_{i,G} = \left\{x_{i,G}^1, x_{i,G}^2, ..., x_{i,G}^D\right\} \quad (1)$$

where i=1, 2, ..., NP, NP is the size of population, and G denotes the generation that the population belongs to. $x_{i,G}^j$ is a real number and randomly chosen from the range [l_j, u_j], namely

$$x_{i,G}^j = rndreal\left(l_j, u_j\right) \quad (2)$$

where $j = 1, 2, ..., D$, D is the number of decision variables.

2.1.2. Mutation:

In the DE algorithm, the core operator is the differential mutation operator. The mutation operator is to create the mutant vector $V_{i,G}$ for each target vector $X_{i,G}$ in the current population. There are many mutaiton operators have been proposed in [3,19], and the classical one is the "DE/rand/1/bin". In DE, some well-known mutation operators are listed as follows.

DE/rand/1/bin:

$$V_{i,G} = X_{r_1,G} + F \bullet \left(X_{r_2,G} - X_{r_3,G}\right) \quad (3)$$

DE/rand/2/bin:

$$V_{i,G} = X_{r_1,G} + F \bullet \left(X_{r_2,G} - X_{r_3,G}\right) + F \bullet \left(X_{r_4,G} - X_{r_5,G}\right) \quad (4)$$

DE/best/2/bin:

$$V_{i,G} = X_{best,G} + F \bullet \left(X_{r_1,G} - X_{r_2,G}\right) + F \bullet \left(X_{r_3,G} - X_{r_4,G}\right) \quad (5)$$

DE/rand-to-best/2/bin:

$$V_{i,G} = X_{r_1,G} + F \bullet \left(X_{best,G} - X_{r_1,G}\right) + F \bullet \left(X_{r_2,G} - X_{r_3,G}\right) + F \bullet \left(X_{r_4,G} - X_{r_5,G}\right) \quad (6)$$

DE/current-to-best/2/bin:

$$V_{i,G} = X_{i,G} + F \bullet \left(X_{best,G} - X_{r_1,G}\right) + F \bullet \left(X_{r_1,G} - X_{r_2,G}\right) + F \bullet \left(X_{r_3,G} - X_{r_4,G}\right) \quad (7)$$

where the index r_1, r_2, r_3, r_4 and r_5 are integers which are randomly selected from the range $\{1, 2, ..., NP\}$, and meet the condition of $r_1 \neq r_2 \neq r_3 \neq r_4 \neq r_5 \neq i$. The $X_{best,G}$ is the individual which has the best fitness function value in the current generation G.

2.1.3. Crossover:

After mutation, a binomial crossover operations will be used to create the trial vector $U_{i,G}$. Its creating scheme is as follows:

$$u_{i,G}^j = \begin{cases} v_{i,G}^j, & \text{if } rand(0,1) < Cr \text{ or } j_{rand} == j \\ x_{i,G}^j, & otherwise \end{cases} \quad (8)$$

where the index j is from 1 to D, j_{rand} is a integer randomly selected from $\{1, 2, ..., D\}$, namely

$$j_{rand} = \text{randint}(1, D) \quad (9)$$

and the crossover probability $Cr \in [0,1]$

D. Selection:

After the mutation and crossover operations, the offspring population has been created. Afterwards, the selection operator is to determine the parent individual or the offspring individual will survive in the next generation by comparing their fitness to the function, and the rule is as follows:

$$X_{i,G+1} = \begin{cases} U_{i,G}, & if \left(f\left(U_{i,G}\right) \leq f\left(X_{i,G}\right)\right) \\ X_{i,G}, & otherwise \end{cases} \quad (10)$$

where the $f(X_{i,G})$ is the fitness value of the i^{th} individual $X_{i,G}$ in the generation G . This method is called one-to-one tournament selection, and it can guarantee that the elitism individual can survive in the next generation. The steps of the original DE based on "DE/rand/1/bin" are as Algorithm 1:

Algorithm 1: the steps of the original DE

```
1: randomly create the population P
2: evaluate every individual of the initial population
3: while (evaluations < max_evaluations)
4:     for i=1 to NP
5:         randomly select the integer index r1 ≠ r2 ≠ r3 ≠ i
6:         j_rand = randint(1, D)
7:         for i=1 to D
8:             v_{i,G}^j = x_{r1,G}^j + F•(x_{r2,G}^j − x_{r3,G}^j)
9:             if (rand(0,1) < Cr or j_rand == j)
10:                u_{i,G}^j = v_{i,G}^j
11:            else
12:                u_{i,G}^j = x_{i,G}^j
13:            end if
14:        end for
15:    end for
16:    for i=1 to NP
17:        evaluate the offspring population U_{i,G}
18:        if (f(U_{i,G}) ≤ f(X_{i,G}))
19:            X_{i,G+1} = U_{i,G}
20:        else
21:            X_{i,G+1} = X_{i,G}
22:        end if
23:    end for
24: end while
```

2.2 (μ+λ)-selection

(μ+λ)-selection is originally proposed in the evolution strategies [20, 21], where the μ is the parent population size, and the λ is the size of the offspring. Its main idea is that λ offspring are created in each generation, and then select μ individuals with lower object function value from the popuation mixed with the offspring and parent.

ES also faces the problem of reducing strength of mutation, like other evolution algorithm. Thus the (μ+λ)-ES is introduced to help with it. Because the worst individuals is abandoned in the process of the generation, the (μ+λ)-ES can assure the convergence. Its basic steps are as TABLE 1.

Table 1: the steps of (μ+λ)-ES

step 1: creating the initiation population
step 2: mutation and crossover to create λ offspring
step 3: selecting μ individuals into next generation from the set of μ parent and λ offspring individuals, like Fig.2
step 4: letting the selected individuals replace the parent individuals and repeat the step 2 to 3.

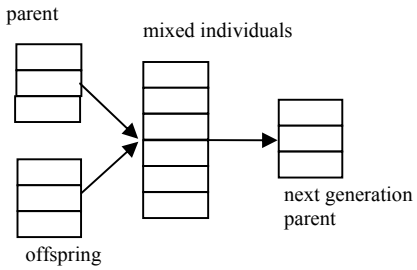

Fig.2: (μ+λ)-selection

3. Our Approach

The original DE adopts one-to-one tournament selection, which one-to-one compares the offspring with its parent, and selects the individuals with lower object function value to survive in the next generation. Hence, the individuals which are better than the others may be abandoned. Inspired by the (μ+λ)-ES, we can borrow the idea to the original DE to solve this problem.

3.1 Combine parents and offspring

In order to store the individuals for selecting, we create a mixed population mixed with parents and offspring. Its individuals is $M_{i,G}$, and

$$M_{i,G} = \left\{ m_{i,G}^1, m_{i,G}^2, ..., m_{i,G}^D \right\} \quad (11)$$

where $i = \{1, 2, ..., NP, NP+1, ..., 2NP\}$.

For the individuals in the mixed parent before NP, it is the offspring individuals. And for the next NP individuals, it is the parent individuals. Namely:

$$M_{i,G} = \begin{cases} U_{i,G} & if\ (1 \le i \le NP) \\ X_{i,G} & otherwise \end{cases} \quad (12)$$

3.2 Select individuals

Before selecting the individuals to survive, we need to sort the mixed population individuals according to their object function value from low to high. Then the individuals ranking before NP can survive in the next generation, as follows:

$$X_{i,G+1} = M_{i,G} \quad (13)$$

where $i = \{1,2,...,NP\}$, and $M_{i,G}$ is the individuals from the sorted mixed population.

Algorithm 2: the steps of the (μ+λ)-DE

```
1: randomly create the population P
2: evaluate every individual of the initial population
3: while (evaluations < max_evaluations)
4:     for i=1 to NP
5:         randomly select the integer index r1 ≠ r2 ≠ r3 ≠ i
6:         j_rand = randint(1, D)
7:         for i=1 to D
8:             v_{i,G}^j = x_{r1,G}^j + F•(x_{r2,G}^j − x_{r3,G}^j)
9:             if (rand(0,1)<Cr or j_rand == j)
10:                u_{i,G}^j = v_{i,G}^j
11:            else
12:                u_{i,G}^j = x_{i,G}^j
13:            end if
14:        end for
15:    end for
16:    for i=1 to NP
17:        evaluate the offspring population U_{i,G}
18:        M_{i,G} = U_{i,G}
19:    end for
20:    for i=NP+1 to 2*NP
21:        M_{i,G} = X_{i−NP,G}
22:    end for
23:    rank the mixed population with the object function value
from low to high
24:    for i=1 to NP
25:        X_{i,G+1} = M_{i,G}
26:    end for
27:end while
```

Table 2: Comparison on the error values between DE and its corresponding (μ+λ)-DE variant for functions F01-F13 at D=30 NP=100

Prob	*DE/rand/1/bin* *Mean(Std Dev)*			*DE/rand/2/bin* *Mean(Std Dev)*			*DE/best/2/bin* *Mean(Std Dev)*		
	DE	(μ+λ)-DE		DE	(μ+λ)-DE		DE	(μ+λ)-DE	
F01	5.71E-14 (4.90E-14)	7.84E-20 (7.68E-20)	+	1.30E+02 (3.53E+01)	1.02E+01 (2.95E+00)	+	6.27E-32 (6.80E-32)	3.59E-42 (1.60E-41)	+
F02	3.47E-07 (1.37E-07)	4.94E-10 (3.19E-10)	+	3.09E+01 (6.79E+00)	3.59E+00 (2.77E+00)	+	1.47E-15 (1.71E-15)	3.21E-21 (2.95E-21)	+
F03	4.48E-01 (2.86E-01)	4.67E-03 (4.12E-03)	+	1.05E+04 (1.35E+03)	6.51E+03 (1.25E+03)	+	3.67E-06 (3.55E-06)	2.51E-11 (3.10E-11)	+
F04	1.39E-01 (3.45E-01)	7.29E-01 (1.19E+00)	-	3.14E+01 (2.64E+00)	2.29E+01 (2.67E+00)	+	2.99E-04 (3.80E-04)	2.50E-02 (3.81E-02)	-
F05	1.68E+01 (8.06E-01)	2.22E+01 (1.04E+00)	-	2.27E+04 (7.88E+03)	1.74E+03 (7.13E+02)	+	7.19E-01 (1.55E+00)	8.21E+00 (5.04E+00)	-
F06	0.00E+00 (0.00E+00)	0.00E+00 (0.00E+00)	=	1.37E+02 (3.09E+01)	1.59E+01 (3.68E+00)	+	3.60E-01 (5.98E-01)	2.20E+00 (1.54E+00)	-
F07	1.28E-02 (2.93E-03)	9.27E-03 (2.56E-03)	+	6.82E+05 (2.65E+05)	1.71E+04 (9.30E+03)	+	1.06E-02 (3.82E-03)	1.04E-02 (3.34E-03)	=
F08	-5.30E+03 (3.57E+02)	-5.56E+03 (3.97E+02)	+	-4.99E+03 (2.93E+02)	-5.02E+03 (2.41E+02)	=	-5.12E+03 (3.50E+02)	-5.16E+03 (3.00E+02)	=
F09	1.74E+02 (1.09E+01)	1.67E+02 (1.39E+01)	=	2.34E+02 (1.06E+01)	2.28E+02 (1.19E+01)	+	1.88E+02 (1.48E+01)	1.85E+02 (1.21E+01)	=
F10	6.37E-08 (2.07E-08)	7.10E-11 (2.64E-11)	+	4.53E+00 (3.01E-01)	2.70E+00 (2.11E-01)	+	4.99E-02 (2.48E-01)	1.23E-01 (3.74E-01)	-
F11	1.48E-04 (1.05E-03)	4.93E-04 (1.99E-03)	+	2.23E+00 (2.83E-01)	1.10E+00 (3.12E-02)	+	8.27E-03 (8.81E-03)	9.06E-03 (1.07E-02)	=
F12	5.08E-15 (5.71E-15)	4.00E-21 (4.43E-21)	+	6.61E+01 (2.01E+02)	9.79E+00 (3.79E+00)	+	6.85E-02 (1.58E-01)	1.28E-01 (3.21E-01)	=
F13	2.86E-14 (2.75E-14)	2.68E-20 (2.93E-20)	+	6.77E+03 (1.09E+04)	2.32E+01 (7.82E+00)	+	8.79E-04 (3.01E-03)	3.27E-02 (2.25E-01)	=
w/t/l	9/2/2			12/1/0			3/6/4		

Prob	*DE/rand-to-best/2/bin* *Mean(Std Dev)*			*DE/current-to-best/2/bin* *Mean(Std Dev)*		
	DE	(μ+λ)-DE		DE	(μ+λ)-DE	
F01	1.27E+04 (1.37E+03)	2.03E+03 (4.83E+02)	+	1.33E-21 (7.36E-22)	1.56E-39 (1.55E-39)	+
F02	6.62E+01 (5.41E+00)	2.65E+01 (7.55E+00)	+	2.86E-09 (1.26E-09)	2.75E-18 (1.81E-18)	+
F03	2.24E+04 (2.61E+03)	1.62E+04 (1.83E+03)	+	1.69E-03 (8.02E-04)	1.01E-10 (9.51E-11)	+
F04	5.22E+01 (2.24E+00)	4.43E+01 (3.20E+00)	+	1.02E-03 (3.33E-04)	4.76E-01 (5.84E-01)	-
F05	1.45E+07 (3.33E+06)	7.70E+05 (2.96E+05)	+	9.70E-02 (5.67E-01)	2.09E+01 (2.13E+00)	-
F06	1.29E+04 (1.44E+03)	2.22E+03 (4.60E+02)	+	0.00E+00 (0.00E+00)	0.00E+00 (0.00E+00)	=
F07	5.30E+08 (1.31E+08)	3.40E+07 (1.32E+07)	+	9.53E-03 (2.68E-03)	4.74E-03 (1.55E-03)	+
F08	-4.92E+03 (2.35E+02)	-5.00E+03 (2.51E+02)	=	-4.89E+03 (2.56E+02)	-4.92E+03 (2.86E+02)	=
F09	2.57E+02 (1.30E+01)	2.49E+02 (1.19E+01)	+	1.87E+02 (1.01E+01)	1.74E+02 (1.17E+01)	+
F10	1.65E+01 (3.58E-01)	1.05E+01 (6.61E-01)	+	1.78E-11 (5.53E-12)	6.84E-15 (1.53E-15)	+
F11	1.14E+02 (1.47E+01)	2.06E+01 (4.67E+00)	+	3.15E-03 (6.31E-03)	1.33E-03 (3.60E-03)	+
F12	1.17E+07 (4.23E+06)	3.89E+05 (4.39E+05)	+	1.85E-20 (3.20E-20)	1.57E-32 (1.38E-47)	+
F13	3.97E+07 (1.45E+07)	1.99E+06 (1.41E+06)	+	2.00E-20 (1.33E-20)	1.36E-32 (6.97E-34)	+
w/t/l	12/1/0			9/2/2		

⋆ "+", "-", and "=" indicate our approach is respectively better than, worse than, or similar to its competitor according to the Wilcoxon signed-rank test at $\alpha = 0.05$

3.3 DE with (μ+λ)-selection

Combining (μ+λ)-selection with the original DE, the (μ+λ)-selection with DE algorithm are present. The pseudo-code of (μ+λ)-DE with "DE/rand/1/bin" mutation is shown in Algorithm 2. We can see the main difference between Algorithm 1 and 2 is the selection operation. Algorithm 2 selects the individuals from mixed population. In this way, the elitism individuals can survive. And the individuals surviving can not from the same ancestor, so the strength of mutation can also be guaranteed.

4. Experiment and Analysis

In this section, we perform comprehensive experiments to verify the performance of the proposed (μ+λ)-DE algorithm. We use 13 benchmark functions presented in the CEC-2005 competition [21] on real-parameter optimization as the test suite. Functions f_1-f_5 are unimodal. Function f_6 is the step function which has one minimum and is discontinuous. Function f_7 is a noisy quadratic function. Functions f_8-f_{13} are multimodal functions where the number of local minima increases exponentially with the problem dimension [16], [23], [24]. More details for these functions can be found in [22].

4.1 Parameter settings

In order to compare the performance of (μ+λ)-DE and the original DE, we make the same settings for them, according to [14], [15] and [17]. For each function, we set the probability of crossover Cr=0.9, the scale factor F=0.5, maximal number of fitness function evaluations max_evaluations = $5{,}000 \times D$ [23], and get the statistics from 50 runs independently. We adopts five mutations to test the performance separately, and combine them with the (μ+λ)-selection to test.

4.2 Comparison DE with different mutation strategies

The proposed (μ+λ)-DE roots in original DE, so we compare the performance of them with different mutations. In Table 3, we record the results of them at D=30 and NP=100. The "w/t/l" represents the comparison results of the benchmark functions. From Table 2, we can see that the (μ+λ)-DE has lower fitness value. And for the most benchmark functions, (μ+λ)-DE are superior to the original DE to the final results. Only in the "DE/best/2/bin" mutation, the (μ+λ)-DE's number of win is less than the original DE. In other mutations, (μ+λ)-DE is over, or at least equal to, the original DE for most function.

The Fig.3 shows comparison of (μ+λ)-DE and the original DE with the benchmark function 1 evolution process in different mutation strategies. In the process of evolution, (μ+λ)-DE and the original has the same start points, and has similar process of convergence, but (μ+λ)-DE has faster convergence speed.

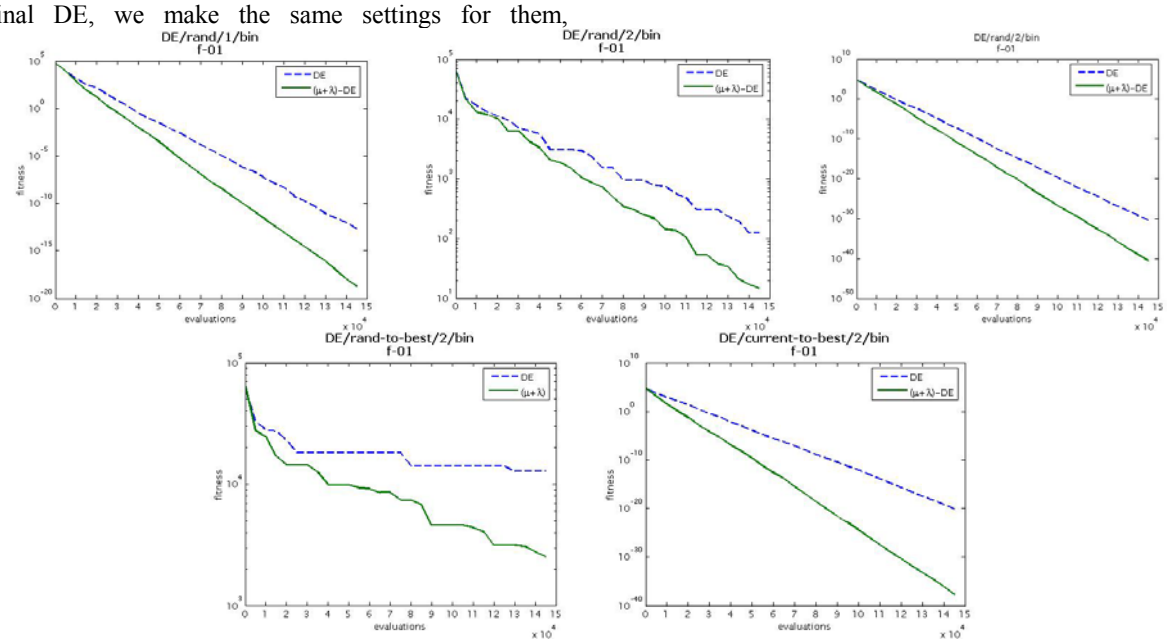

Fig.3: the comparison of (μ+λ)-DE and the original DE for the process of evolution with different mutation in the condition 1.

Table 3: Comparison on the error values between DE and its corresponding (μ+λ)-DE variant for functions F01-F13 at D=30

Pro	*NP=50* *Mean(Std Dev)*			*NP=100* *Mean(Std Dev)*			*NP=200* *Mean(Std Dev)*			*NP=200* *Mean(Std Dev)*		
	DE	(μ+λ)-DE		DE	(μ+λ)-DE		DE	(μ+λ)-DE		DE	(μ+λ)-DE	
F01	2.72E-43 (9.71E-43)	5.09E-56 (1.62E-55)	+	5.71E-14 (4.90E-14)	7.84E-20 (7.68E-20)	+	8.45E-03 (3.29E-03)	5.35E-05 (2.40E-05)	+	6.51E+01 (1.35E+01)	8.50E+00 (1.60E+00)	+
F02	1.94E-23 (1.66E-23)	6.48E-31 (6.72E-31)	+	3.47E-07 (1.37E-07)	4.94E-10 (3.19E-10)	+	1.25E-01 (2.51E-02)	9.05E-03 (3.02E-03)	+	9.03E+00 (1.12E+00)	2.81E+00 (7.90E-01)	+
F03	1.94E-05 (4.59E-05)	1.05E+00 (3.68E+00)	-	4.48E-01 (2.86E-01)	4.67E-03 (4.12E-03)	+	7.06E+02 (2.27E+02)	1.50E+02 (4.38E+01)	+	7.20E+03 (1.42E+03)	4.68E+03 (7.81E+02)	+
F04	8.08E+00 (3.31E+00)	1.15E+01 (4.58E+00)	-	1.39E-01 (3.45E-01)	7.29E-01 (1.19E+00)	-	2.28E+00 (4.06E-01)	7.08E-01 (1.45E-01)	+	1.92E+01 (1.82E+00)	1.22E+01 (1.35E+00)	+
F05	2.68E+01 (1.29E+01)	4.39E+01 (4.69E+01)	-	1.68E+01 (8.06E-01)	2.22E+01 (1.04E+00)	-	2.92E+01 (9.24E-01)	2.41E+01 (4.43E-01)	+	5.03E+03 (1.70E+03)	6.98E+02 (1.80E+02)	+
F06	0.00E+00 (0.00E+00)	2.00E-01 (6.39E-01)	-	0.00E+00 (0.00E+00)	0.00E+00 (0.00E+00)	=	0.00E+00 (0.00E+00)	0.00E+00 (0.00E+00)	=	6.70E+01 (1.03E+01)	1.15E+01 (1.88E+00)	+
F07	8.08E-03 (3.88E-03)	3.46E+03 (1.77E+04)	-	1.28E-02 (2.93E-03)	9.27E-03 (2.56E-03)	+	1.97E-01 (7.08E-02)	5.05E-02 (1.23E-02)	+	9.57E+04 (2.96E+04)	3.74E+03 (1.82E+03)	+
F08	-1.00E+04 (1.74E+03)	-1.19E+04 (4.75E+02)	+	-5.30E+03 (3.57E+02)	-5.56E+03 (3.97E+02)	+	-5.10E+03 (3.25E+02)	-5.18E+03 (2.69E+02)	=	-4.99E+03 (2.70E+02)	-5.00E+03 (2.66E+02)	=
F09	2.10E+01 (1.21E+01)	1.36E+01 (3.77E+00)	+	1.74E+02 (1.09E+01)	1.67E+02 (1.39E+01)	=	1.94E+02 (1.49E+01)	1.93E+02 (9.41E+00)	=	2.12E+02 (9.93E+00)	2.08E+02 (1.03E+01)	=
F10	4.14E-15 (0.00E+00)	5.33E-06 (3.74E-05)	-	6.37E-08 (2.07E-08)	7.10E-11 (2.64E-11)	+	3.32E-02 (8.15E-03)	2.38E-03 (4.80E-04)	+	3.62E+00 (1.86E-01)	2.15E+00 (1.80E-01)	+
F11	2.61E-03 (5.32E-03)	2.02E-03 (4.92E-03)	=	1.48E-04 (1.05E-03)	4.93E-04 (1.99E-03)	+	3.28E-02 (2.29E-02)	1.52E-04 (1.27E-04)	+	1.57E+00 (9.81E-02)	1.07E+00 (1.59E-02)	+
F12	1.57E-32 (1.38E-47)	1.09E+01 (7.69E+01)	-	5.08E-15 (5.71E-15)	4.00E-21 (4.43E-21)	+	1.65E-03 (1.07E-03)	6.70E-06 (3.95E-06)	+	8.04E+00 (1.55E+00)	2.19E+00 (5.40E-01)	+
F13	2.20E-04 (1.55E-03)	8.97E+02 (6.28E+03)	-	2.86E-14 (2.75E-14)	2.68E-20 (2.93E-20)	+	8.51E-03 (3.88E-03)	3.72E-05 (1.94E-05)	+	2.57E+01 (5.41E+00)	6.16E+00 (1.96E+00)	+
w/t/l	4/1/8			9/2/2			10/3/0			11/2/0		

⋆ "+", "-", and "=" indicate our approach is respectively better than, worse than, or similar to its competitor according to the Wilcoxon signed-rank test at $\alpha = 0.05$

4.3 Influence of the population size

In our experiments, different parameters settings have different optimization performance. For population size, the individuals stick into local convergence easily, when it is small. However, when it is large, the individuals can not get good optimization results. Thus we make experiments at the same problem dimensionality D=30 and different population size NP=50, 100, 200 and 400. The comparison of the original DE and our (μ+λ)-DE results are recorded in the Table 3. At population size NP=50, the performance of (μ+λ)-DE does not get better optimization results. But when the population size is large, the (μ+λ)-DE performs better than, or at least equal to, the original DE from the number of the win function. With our improved idea, only the function 8 does not get obvious optimization, but the others have better performance, when the population size is large.

4.4 Scalability study

In our above experiments, we set all the dimension of the benchmark functions D=30. In this section, we make scalability study, and the dimensions are scaled at D=30, 100 and 200. For higher dimension problems, larger population size is a must. Thus we set the population size NP=4×D with "DE/rand/1/bin" mutation strategy for both the original DE and the (μ+λ)-DE. The results are represented in the Table 4. For most benchmark functions, (μ+λ)-DE still has better optimization performance. And comparing the results of the "w/t/l", we can see the results in higher dimensions are better than that in lower. Only the F08 doesn't get obvious optimization in all the conditions, but for the others, the (μ+λ)-DE all gets better optimization. In high dimension benchmark function, (μ+λ)-DE are better or at least equal to the original DE. So, we can conclude that (μ+λ)-DE has better performance for the high-dimensional problems.

Table 4: Comparison on the error values between DE and its corresponding (μ+λ)-DE variant for functions F01-F13

Prob	*D=30* *Mean(Std Dev)*			*D=100* *Mean(Std Dev)*			*D=200* *Mean(Std Dev)*		
	DE	(μ+λ)-DE		DE	(μ+λ)-DE		DE	(μ+λ)-DE	
F01	8.08E-10 (5.49E-10)	2.09E-14 (2.16E-14)	+	4.15E+02 (8.40E+01)	6.17E+01 (1.53E+01)	+	2.20E+04 (2.33E+03)	1.07E+04 (1.27E+03)	+
F02	4.24E-05 (1.88E-05)	2.35E-07 (1.19E-07)	+	1.01E+02 (2.09E+01)	4.78E+01 (1.20E+01)	+	2.91E+02 (1.89E+01)	1.46E+02 (5.95E+01)	+
F03	6.03E+00 (2.81E+00)	1.53E-01 (1.19E-01)	+	1.07E+08 (1.44E+07)	7.39E+07 (1.30E+07)	+	8.78E+05 (6.99E+04)	8.49E+05 (6.50E+04)	=
F04	4.61E-02 (5.48E-02)	7.40E-01 (1.81E+00)	=	4.42E+01 (3.20E+00)	3.28E+01 (3.01E+00)	+	8.25E+01 (1.60E+00)	8.16E+01 (2.16E+00)	=
F05	1.92E+01 (6.79E-01)	2.02E+01 (1.19E+00)	-	3.71E+04 (1.32E+04)	3.40E+03 (9.16E+02)	+	1.08E+07 (1.87E+06)	2.93E+06 (6.79E+05)	+
F06	0.00E+00 (0.00E+00)	0.00E+00 (0.00E+00)	=	7.54E+02 (1.41E+02)	1.22E+02 (2.89E+01)	+	2.20E+04 (2.94E+03)	1.04E+04 (1.53E+03)	+
F07	1.98E-02 (5.41E-03)	1.27E-02 (3.10E-03)	+	6.69E+08 (2.05E+08)	3.35E+07 (1.50E+07)	+	3.13E+09 (6.02E+08)	8.39E+08 (1.85E+08)	+
F08	-5.12E+03 (2.69E+02)	-5.41E+03 (2.35E+02)	+	2.95E+04 (7.73E+02)	2.93E+04 (7.31E+02)	=	-1.35E+04 (5.89E+02)	-1.36E+04 (6.51E+02)	=
F09	1.86E+02 (1.10E+01)	1.81E+02 (1.14E+01)	+	1.28E+03 (4.03E+01)	1.22E+03 (3.88E+01)	+	2.05E+03 (5.02E+01)	1.99E+03 (4.25E+01)	+
F10	8.36E-06 (2.39E-06)	4.14E-08 (1.47E-08)	+	5.40E+00 (3.82E-01)	3.24E+00 (1.86E-01)	+	1.17E+01 (3.67E-01)	9.30E+00 (3.57E-01)	+
F11	2.46E-04 (1.74E-03)	1.48E-04 (1.05E-03)	+	6.15E+02 (5.42E+01)	2.83E+02 (4.30E+01)	+	1.96E+02 (2.18E+01)	9.59E+01 (1.04E+01)	+
F12	9.61E-11 (9.86E-11)	1.62E-15 (1.67E-15)	+	1.63E+01 (2.89E+00)	5.58E+00 (1.46E+00)	+	2.78E+06 (1.57E+06)	2.39E+05 (1.73E+05)	+
F13	5.80E-10 (6.86E-10)	1.22E-14 (1.19E-14)	+	8.97E+02 (1.61E+03)	5.69E+01 (1.23E+01)	+	1.41E+07 (3.89E+06)	2.54E+06 (9.71E+05)	+
w/t/l	10/2/1			12/1/0			10/3/0		

⋆ "+", "-", and "=" indicate our approach is respectively better than, worse than, or similar to its competitor according to the Wilcoxon signed-rank test at $\alpha = 0.05$

5. Conclution

DE is a simple yet effective global optimization algorithm. And there are already many improved methods [14-17] have been improved, but most of them still adopt one-to-one tournament selection, but superior individuals may be abandoned in the process of evolution. Inspired by the (μ+λ)-ES, we propose (μ+λ)-DE, which combines the original DE with the (μ+λ)-selection. (μ+λ)-DE combine the parent population and the offspring population into the mixed population, and sorts its individuals with object function value from low to high, then the individuals before NP can survive in the next generation.

We tested the performance of the (μ+λ)-DE with five mutation strategies in two conditions, and they all get better results for most benchmark functions, especially in high dimension and large population size conditions. All elitism individuals must survive in each generation, and the individuals in each generation can not from the same ancestor. Therefore, the (μ+λ)-DE has stronger mutation ability compared with the original DE, and can get faster convergence speed.

(μ+λ)-DE has been proved that it has better performance than the original DE， especially for high dimension and the large population size problems. In this paper, we conduct experiment for five mutation startegies, and set different parameter to research our (μ+λ)-DE performance. For further works, we can combine (μ+λ)-selection with other proposed DE improvement strategies, and study its performance of optimization.

Acknowledgement

This work was partly supported by the National Natural Science Foundation of China under Grant No. 61203307, the Fundamental Research Funds for the Central Universities at China University of Geosciences (Wuhan) under Grant No. CUG130413, and the Research Fund for

the Doctoral Program of Higher Education under Grant No. 20110145120009.

References

[1] R. Storn and K. Price, "Differential Evolution—A Simple and Efficient Adaptive Scheme for Global Optimization over Continuous Spaces", Berkeley, CA, Tech. Rep. TR-95-012. 1995.

[2] R. Storn and K. Price, "Differential Evolution—A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces", in Journal of Global Optimization 11. Norwell, MA: Kluwer, 1997, pp. 341–359.

[3] K. Price, R. M. Storn, and J. A. Lampinen, "Differential evolution: a practical Approach to Global Optimization" (Natural Computing Series),1st ed. New York: Springer, 2005, ISBN: 3540209506.

[4] J. Vesterstroem and R. Thomsen, "A Comparative Study of Differential Evolution, Particle Swarm Optimization, and Evolutionary Algorithms on Numerical Benchmark Problems", Proc. Congr. Evol. Comput., vol. 2, 2004, pp. 1980–1987.

[5] J. Andre, P. Siarry, and T. Dognon, "An Improvement of the Standard Genetic Algorithm Fighting Premature Convergence in Continuous Optimization", Advance in Engineering Software 32, 2001, pp. 49–60.

[6] O. Hrstka and A. Kucerová, "Improvement of Real Coded Genetic Algorithm Based on Differential Operators Preventing PrematureConvergence",Advance in Engineering Software 35, 2004, pp. 237–246.

[7] T. Rogalsky, R. W. Derksen, and S. Kocabiyik "Differential evolution in aerodynamic optimization", in Proc. 46th Annu. Conf. Can. Aeronautics Space Inst. 29–36. 1999.

[8] R. Joshi and A. C. Sanderson. "Minimal Representation Multi-sensor Fusion Using Differential Evolution", IEEE Trans. Syst. Man. Cybern. 1999, Part A, 29, 1. 63–76.

[9] S. Das and A. Konar. "Design of Two Dimensional IIR Filters with Modern Search Heuristics: A Comparative Study", Int. J. Comput. Intell. Applicat. 2006, 6, 3, 329–355,

[10] F. S.Wang and H. J. Jang , "Parameter Estimation of A Bio-Reaction Model by Hybrid Differential Evolution", in Proc. IEEE Congr. Evol. Comput. 2000, 1. Piscataway, NJ: IEEE Press 410–417.

[11] J. Lampinen. "A Bibliography of Differential Evolution Algorithm", Lappeenranta University of Technology. Department of Information Technology, Laboratory of Information Processing, Tech. Report [Online]. Available: http://www.lut.fi/jlampine/debiblio.htm, 1999.

[12] M. Omran, A. P. Engelbrecht, and A. Salman. "Differential Evolution Methods for Unsupervised Image Classification", in Proc. 7th Congr. Evol. Comput. (CEC-2005), 2005,2. Piscataway, NJ: IEEE Press, 966–973.

[13] S. Das, A. Abraham, and A. Konar. "Adaptive Clustering Using Improved Differential Evolution Algorithm", IEEE Trans. Syst., Man, Cybern. A, 2008,38, 1, 218–237.

[14] M. Ali, M. Pant and A. Abraham. "Improving Differential Evolution Algorithm by Synergizing Different Improvement Mechanisms", ACM Transactions on Autonomous and Adaptive Systems (TAAS), 2012, Vol. 7 Issue 2, No. 20.

[15] Shahryar Rahnamayan, Hamid R. Tizhoosh, and Magdy M. A. Salama, "Opposition-Based Differential Evolution",IEEE Transactions on evolutionary computation, 2008,Volume: 12 , Issue: 1 Page(s): 64 - 79

[16] Janez Brest. Saso Greiner, Borko Boskovic, Marjan Mernik and Viljem Zumer "Self-Adapting Control Parameters in Differential Evolution: a Comparative Study on Numerical Benchmark Problems", IEEE Transactions on evolutionary computation , Vol. 10 , Issue: 6, 2006, pp. 646 - 657

[17] Jingqiao Zhang and Arthur C. Sanderson, "JADE: Adaptive Differential Evolution with Optional External Archive", IEEE Transactions on evolutionary computation, Vol.13 , Issue: 5, 2009, pp. 945 - 958

[18]Nikhil Padhye Piyush Bhardawaj and Kalyanmoy Deb, 2012, "Improving Differential Rvolution Through a Unified Approach", Journal of Global Optimization, 2012, DOI 10.1007/s10898-012-9897-0

[19]R.Storn and K.Price, "Home page of differential evolution," 2010.[Online]. Available: http://www.ICSI.Berkeley.edu/storn/code.html

[20]Manolis Papadrakakis and Nikolaos D. Lagaros, "Advanced Solution Methods in Structural Optimization Based on Evolution Strategies", Engineering Computations, Vol. 15 No. 1, 1998, pp. 12-34,

[21]Hans-Georg Beyer and Hans-Paul Schwefel, "Evolution strategies", Natural Computing, Vol.1, Issue 1, 2002, pp. 3-52

[22]P. N. Suganthan, N. Hansen, J. J. Liang, K. Deb, Y.-P. Chen, A.Auger, and S.Tiwari, "Problem definitions and evaluation criteria for the CEC2005 special session on real-parameter optimization", 2005, [Online]. Available: http://www.ntu.edu.sg/home/EPNSugan

[23] X. Yao, Y. Liu, and G. Lin. "Evolutionary Programming Made Faster", IEEE Trans. Evol. Comput., Vol.3, No.2, 1999, pp. 82.

[24] A. Törn and A.Zilinskas, "Global Optimization", in Lecture Notes Computer Science. Heidelberg, Germany: Spring-Verlag, 1989, vol. 350, pp. 1–24.

Miaomiao Liu is still a student in China University of Geosciences. She started the study of DE in 2011, and current research interest is differential evolution.

Pan Huo is still a student in China University of Geosciences. She started the study of DE in 2011, and current research interest is differential evolution.

Dr. Wenyin Gong
Address: No. 388, Lumo Road,
School of Computer Science,
China University of Geosciences,
Wuhan 430074, P.R. China
Tel: 86-27-67883716 Fax: 86-27-67883716
Homepage: http://cs.cug.edu.cn/teacherweb/gwy/