

# Novel Composition Test Functions for Numerical Global Optimization

*J. J. Liang<sup>1</sup>, P. N. Suganthan<sup>1</sup> and K. Deb<sup>2</sup>*

<sup>1</sup>School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore 639798

<sup>2</sup>Department of Mechanical Engineering, Indian Institute of Technology, Kanpur, 208 016, INDIA

liangjing@pmail.ntu.edu.sg, epnsugan@ntu.edu.sg, deb@iitk.ac.in

## ABSTRACT

In the evolutionary optimization field, there exist some algorithms taking advantage of the known property of the benchmark functions, such as local optima lying along the coordinate axes, global optimum having the same values for many variables and so on. Multiagent genetic algorithm (MAGA) [1] is an example for this class of algorithms. In this paper, we identify shortcomings associated with the existing test functions. Novel hybrid benchmark functions whose complexity and properties can be controlled easily, are introduced and several evolutionary algorithms are evaluated with the novel test functions.

## 1. INTRODUCTION

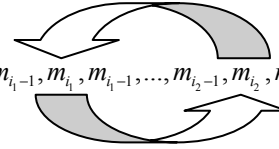
Many real-world problems can be formulated as optimization problems most of which can be converted to the following form:

Minimize  $f(x)$ ,  $x = [x_1, x_2, \dots, x_D]$ , where  $x \in [\underline{x}, \bar{x}]$

Many novel algorithms are introduced to solve the above global optimization problem. In order to compare and evaluate different algorithms, various benchmark functions with various properties have been proposed. Many of these popular benchmark functions possess some properties that have been exploited by some algorithms to achieve excellent results. According to our experience, some of these problems are:

1. Global optimum having the same parameter values for different variables\dimensions: Most of the popular benchmark functions have the same parameter values for different dimensions at the global optimum because of their symmetry. For example, Rastrigin's function's and Griewank's function's global optima are  $[0, 0, 0, \dots, 0]$  and Rosenbrock's function's global

optimum are  $[1, 1, 1, \dots, 1]$ . In this situation, if there exist some operators to copy one dimension's value to the other dimensions, the global optimum may be found rapidly. For example, the neighborhood competition operator in [1] defined as follows:

$$l = (m_1, \dots, m_{i_1-1}, m_{i_1}, m_{i_1+1}, \dots, m_{i_2-1}, m_{i_2}, m_{i_2+1}, \dots, m_D)$$


$$= (m_1, \dots, m_{i_1-1}, m_{i_2}, m_{i_2-1}, \dots, m_{i_1+1}, m_{i_1}, m_{i_2+1}, \dots, m_D)$$

where  $m$  is the best solution in the population and  $l$  is the new generated solution,  $i_1$  and  $i_2$  are two

random integers and  $1 < i_1 < i_2 < D$ ,  $D$  is the

dimensionality of the problem. Hence, if the algorithm has found the globally optimal coordinates for some dimensions, they will be easily copied to the other dimensions. But, this operator is not useful if the global optimum does not have the same value for many dimensions. According to our experiments, although it generated excellent results on the standard benchmark functions, when we shifted the global optimum to make the optimum to have different values for different dimensions, the performance of the MAGA algorithm significantly deteriorated. When we solve real-world problems, the global optimum is unlikely to have the same value for different dimensions.

2. Global optimum at the origin [1]: In this case, the global optimum  $o$  is equal to  $[0, 0, 0, \dots, 0]$ . Zhong et al. [1] proposed the following function

$[l \cdot (1 - sRadius), l \cdot (1 + sRadius)]$ , where  $l$  is the search center and  $sRadius$  is the local search radius, to perform the local search. We can observe that the

local search range is much smaller when  $l$  is near the origin than when  $l$  is far from the origin. This operator is not effective if the global optimum is not at the origin. Hence, this operator is specifically designed to exploit this common property of many benchmark functions.

3. Global optimum lying in the center of the search range: Some algorithms have the potential to converge to the center of the search range. The mean-centric crossover operator is just a good example for this type. When we randomly generate the initial population uniformly, the mean-centric method will have a trend to lead the population to the center of the search range.
4. Global optimum on the bounds: This situation is encountered in some Multi-objective optimization algorithms as some algorithms set the dimensions moving out of the search range to the bounds [2]. If the global optimum is on the bounds, as in some multi-objective benchmark functions, the global optimum will be easily found. However, if there are some local optima near the bounds, it will be easy to fall into the local optima and fail to find the global optimum.
5. Local optima lying along the coordinate axes or no linkage among the variables / dimensions: Most of the benchmark functions, especially high dimensional functions, always have their symmetrical grid structure and local optima are always along the coordinate axes. In this case, the information of the local optima could be used to locate the global optimum. Further, for some functions it is possible to locate the global optimum by using just  $D$  one-dimensional searches for a  $D$  dimensional problem. Some co-evolutionary algorithms [3] and the one dimensional mutation operator [1][4] just use these properties to locate the global optimum rapidly.

By analyzing these problems, we recommend that the researchers should use the following methods to avoid these problems when they use the benchmark functions suffering from these problems, to test a novel algorithm.

1. Shift the global optimum to a random position as shown below to make the global optimum to have different parameter values for different dimensions for benchmark functions suffering from problems 1 to 3:

$$F(x) = f(x - o_{new} + o_{old})$$

where  $F(x)$  is the new function,  $f(x)$  is old function,  $o_{old}$  is the old global optimum and  $o_{new}$  is the new setting global optimum which has different values for different dimensions and not in the center of the search range.

2. For Problem 4, considering there are real problems has the global optimum on the bounds, it is an acceptable method for bounds handling to set the population to the near bounds when they are out of the search range. However, we suggest using different kinds of benchmark functions to test the algorithms. For example, we can use some problems with the global optimum on bounds, not on bounds and some problems with local optima on bounds. We may not just test one algorithm that uses this bounds handling method, on benchmark functions with the global optimum on bounds, and conclude the algorithm to be good.

3. Rotate the functions with problem 5 as below:

$$F(x) = f(R * x)$$

where  $\mathbf{R}$  is an orthogonal rotation matrix obtained using Salmon's method [5]. In this way, we can avoid local optima lying along the coordinate axes and retain the benchmark functions' properties at the same time.

When we test a novel algorithm, except the global optimum's position need be shifted, functions having different properties should be included. For example, continuous functions, non-continuous functions, global optimum on the bounds, global optimum not on the bounds, unrotated functions, rotated functions, function with no clear structure in the fitness landscape, narrow global basin of attraction and so on. Some functions with different properties can be found in our CEC'05 functions.

## 2. NOVEL COMPOSITION TEST FUNCTIONS

Based on these considerations, we propose a general framework<sup>1</sup> to construct novel and challenging composition test functions possessing many desirable properties. The idea is to compose the standard benchmark functions to construct a more challenging function with a randomly located global optimum and several randomly located deep local optima. Gaussian functions are used to combine these benchmark functions and blur the individual function's structure. The details are described below and Pseudo Code is given in Figure 1.

$F(x)$  : new composition function.

$f_i(x)$  :  $i$ th basic function used to construct the composition function.

$n$  : number of basic functions. The bigger  $n$  is, the more complex  $F(x)$  is.

$D$  : dimension.

$[X \min, X \max]^D$  :  $F(x)$  's search range

$[x \min_i, x \max_i]^D$  :  $f_i(x)$  's search range

$M_i$  : orthogonal rotation matrix for each  $f_i(x)$

$o_i$  : new shifted optimum position for each  $f_i(x)$

$o_{iold}$  : old optimum position for each  $f_i(x)$

$F(x) = \sum_{i=1}^n \{w_i * [f_i'((x - o_i + o_{iold}) / \lambda_i * M_i) + bias_i]\} + f\_bias$

$w_i$  : weight value for each  $f_i(x)$ , calculated as below:

$$w_i = \exp\left(-\frac{\sum_{k=1}^D (x_k - o_{ik} + o_{ikold})^2}{2D\sigma_i^2}\right),$$

$$w_i = \begin{cases} w_i & \text{if } w_i = \max(w_i) \\ w_i * (1 - \max(w_i) \wedge 10) & \text{if } w_i \neq \max(w_i) \end{cases}$$

then normalize the weight  $w_i = w_i / \sum_{i=1}^n w_i$

$\sigma_i$  : used to control each  $f_i(x)$  's coverage range, a small  $\sigma_i$  gives a narrow range for  $f_i(x)$ .

$\lambda_i$  : used to stretch or compress the function,  $\lambda_i > 1$  means stretch,  $\lambda_i < 1$  means compress. Since different basic function has different search range, in order to make full use of the basic function, we usually set

$$\lambda_i = \sigma_i * \frac{X \max - X \min}{x \max_i - x \min_i}.$$

$o_i$  define the global and local optima's position,  $bias_i$  define which optimum is global optimum. The smallest  $bias_i$  corresponds to the global optimum. Using  $o_i$ ,  $bias_i$ , a global optimum can be placed anywhere.

If  $f_i(x)$  are different functions, different functions have different properties and height, in order to get a better mixture, we estimate the biggest function value  $f_{\max_i}$  for 10 functions  $f_i(x)$ , then normalize each basic function to similar height as below:

$f_i'(x) = C * f_i(x) / |f_{\max_i}|$ ,  $C$  is a predefined constant.

$|f_{\max_i}|$  is estimated using  $|f_{\max_i}| = f_i((z / \lambda_i) * M_i)$ ,

$z = X \max$

---

Define  $f_i, \sigma_i, \lambda_i, bias_i, o_i, M_i$ ,  $i = 1, 2, \dots, n$  and constant  $C$ ,  $z = X \max$

For  $i = 1:n$

$$w_i = \exp\left(-\frac{\sum_{k=1}^D (x_k - o_{ik} + o_{ikold})^2}{2D\sigma_i^2}\right),$$

$$fit_i = f_i(((x - o_i + o_{iold}) / \lambda_i) * M_i)$$

$$f_{\max_i} = f_i((z / \lambda_i) * M_i),$$

$$f_i = C * fit_i / f_{\max_i}$$

EndFor

$$SumW = \sum_{i=1}^n w_i$$

$$MaxW = \max(w_i)$$

For  $i = 1:n$

$$w_i = \begin{cases} w_i & \text{if } w_i = \max(w_i) \\ w_i * (1 - \max(w_i) \wedge 10) & \text{if } w_i \neq \max(w_i) \end{cases}$$

$$w_i = w_i / SumW$$

---

<sup>1</sup> We developed novel benchmark functions possessing several desirable properties as a part of our invited session on function optimization at CEC 2005. The details can be found at <http://www.ntu.edu.sg/home/EPNSugan>.

EndFor

$$F(x) = f\_bias + \sum_{i=1}^n \{w_i * [f_i + bias_i]\}$$

Figure 1. Pseudo Code

### 3. EXPERIMENT

From the descriptions in Section 2, we can observe that by controlling  $f_i$ ,  $\sigma_i$ ,  $\lambda_i$ ,  $bias_i$ ,  $o_i$  and  $M_i$ , we obtain different composition functions with different desired properties. In our experiments, we define and use six different composition functions and test six different optimization algorithms on this group of composition functions.

#### Part I. Parameters settings for the composition functions

Basic function number  $n=10$

Dimension  $D=10$

$C=2000$ ,

Search range:  $[-5, 5]^D$

$f\_bias=0$

$bias = [0, 100, 200, 300, 400, 500, 600, 700, 800, 900]$ .

Hence, the first function  $f_1(x)$  is always the function with the global optimum, as its bias is zero always.  $o_1, o_2, \dots, o_9$  are all generated randomly in the search range, except  $o_{10}$  is set  $[0, 0, \dots, 0]$  for trapping algorithms which have a potential to converge to the center of the search range.  $M_1, M_2, \dots, M_n$  are  $D \times D$  orthogonal rotation matrixes obtained by using Salmon's method [5]

Basic functions:

Sphere Function

$$f(x) = \sum_{i=1}^D x_i^2, x \in [-100, 100]^D$$

Rastrigin's Function

$$f(x) = \sum_{i=1}^D (x_i^2 - 10 \cos(2\pi x_i) + 10), x \in [-5, 5]^D$$

Weierstrass Function

$$f(x) = \sum_{i=1}^D \left( \sum_{k=0}^{k_{\max}} [a^k \cos(2\pi b^k (x_i + 0.5))] \right)$$

$$-D \sum_{k=0}^{k_{\max}} [a^k \cos(2\pi b^k \cdot 0.5)]$$

$$a=0.5, b=3, k_{\max}=20, x \in [-0.5, 0.5]^D$$

Griewank's Function

$$f(x) = \sum_{i=1}^D \frac{x_i^2}{4000} - \prod_{i=1}^D \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1, x \in [-100, 100]^D$$

Ackley's Function

$$f(x) = -20 \exp\left(-0.2 \sqrt{\frac{1}{D} \sum_{i=1}^D x_i^2}\right)$$

$$- \exp\left(\frac{1}{D} \sum_{i=1}^D \cos(2\pi x_i)\right) + 20 + e, x \in [-32, 32]^D$$

$f_i, \sigma_i, \lambda_i$  settings for these six composition functions are given below:

*Composition Function 1 (CF1):*

$f_1, f_2, \dots, f_{10}$ : Sphere Function

$[\sigma_1, \sigma_2, \dots, \sigma_{10}] = [1, 1, \dots, 1]$

$[\lambda_1, \lambda_2, \dots, \lambda_{10}] = [5/100, 5/100, \dots, 5/100]$

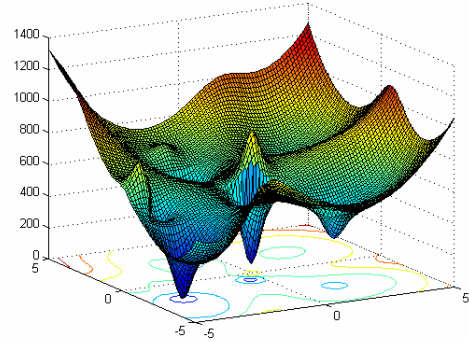


Figure 2. 2-D Composition Function 1

*Composition Function 2 (CF2):*

$f_1, f_2, \dots, f_{10}$ : Griewank's Function

$[\sigma_1, \sigma_2, \dots, \sigma_{10}] = [1, 1, \dots, 1]$

$[\lambda_1, \lambda_2, \dots, \lambda_{10}] = [5/100, 5/100, \dots, 5/100]$

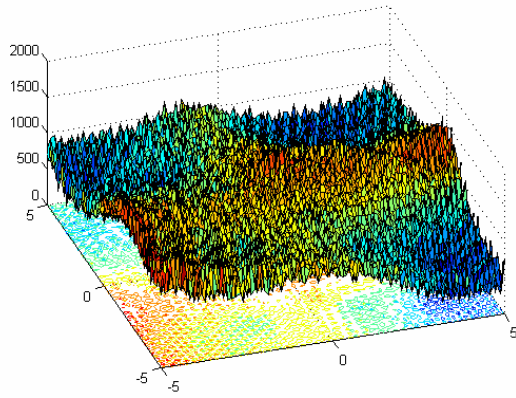


Figure 3. 2-D Composition Function 2

*Composition Function 3 (CF3):*

$f_1, f_2, \dots, f_{10}$  : Griewank's Function

$[\sigma_1, \sigma_2, \dots, \sigma_{10}] = [1, 1, \dots, 1]$

$[\lambda_1, \lambda_2, \dots, \lambda_{10}] = [1, 1, \dots, 1]$

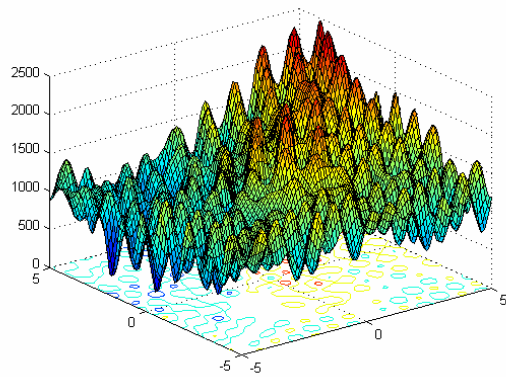


Figure 4. 2-D Composition Function 3

*Composition Function 4 (CF4):*

$f_{1-2}(x)$  : Ackley's Function

$f_{3-4}(x)$  : Rastrigin's Function

$f_{5-6}(x)$  : Weierstrass Function

$f_{7-8}(x)$  : Griewank's Function

$f_{9-10}(x)$  : Sphere Function

$[\sigma_1, \sigma_2, \dots, \sigma_{10}] = [1, 1, \dots, 1]$

$[\lambda_1, \lambda_2, \dots, \lambda_{10}] = [5/32, 5/32, 1, 1, 5/0.5, 5/0.5, 5/100, 5/100, 5/100, 5/100]$

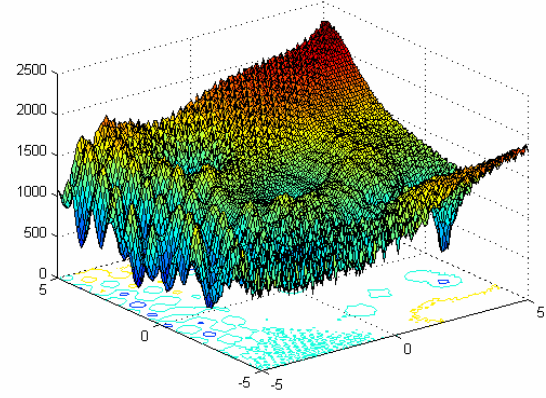


Figure 5. 2-D Composition Function 4

*Composition Function 5 (CF5):*

$f_{1-2}(x)$  : Rastrigin's Function

$f_{3-4}(x)$  : Weierstrass Function

$f_{5-6}(x)$  : Griewank's Function

$f_{7-8}(x)$  : Ackley's Function

$f_{9-10}(x)$  : Sphere Function

$[\sigma_1, \sigma_2, \dots, \sigma_{10}] = [1, 1, \dots, 1]$

$[\lambda_1, \lambda_2, \dots, \lambda_{10}] = [1/5, 1/5, 5/0.5, 5/0.5, 5/100, 5/100, 5/32, 5/32, 5/100, 5/100]$

Here  $\lambda_1, \lambda_2$  are set 1/5 in order to get a more complex landscape for the global optimum's area.

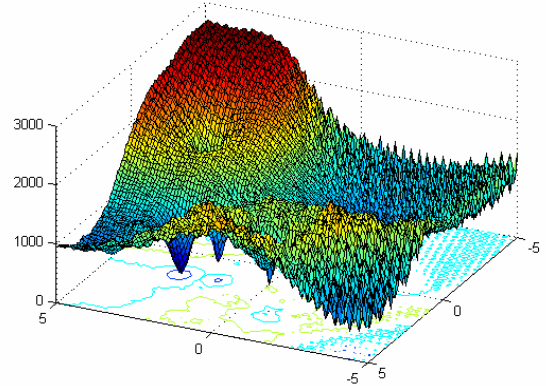


Figure 6. 2-D Composition Function 5

*Composition Function 6 (CF6):*

$f_{1-2}(x)$  : Rastrigin's Function

$f_{3-4}(x)$  : Weierstrass Function

$f_{5-6}(x)$  : Griewank's Function

$f_{7-8}(x)$  : Ackley's Function

$f_{9-10}(x)$ : Sphere Function

$[\sigma_1, \sigma_2, \dots, \sigma_{10}] = [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1]$   
 $[\lambda_1, \lambda_2, \dots, \lambda_{10}] = [0.1*1/5, 0.2*1/5, 0.3*5/0.5, 0.4*5/0.5, 0.5*5/100, 0.6*5/100, 0.7*5/32, 0.8*5/32, 0.9*5/100, 1*5/100]$

CF5 and CF6 use the same optima's position  $o$  and the same orthogonal matrixes  $M_1, M_2, \dots, M_n$ . The difference between CF5 and CF6 is the values of  $\sigma$  and  $\lambda$ , which makes CF6 has a narrower coverage area for the basic function with the global optimum and a flatter coverage area for the basic function with the local optima. In this way, the complexity of the function is increased. We can also observe this in Table 1.

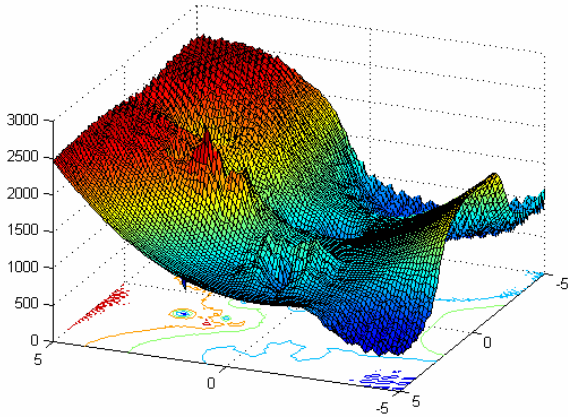


Figure 7. 2-D Composition Function 6

## Part II. Experiment Result

Algorithms used and associate parameter settings:

- Particle Swarm Optimizor (PSO) [7]:  
population size=20
- Cooperative PSO (CPSO) [3]:  
population size=10, number of groups= $D=10$
- Comprehensive Learning PSO (CLPSO) [8]:  
population size=10,  $Pc=0.05$
- Evolution Strategy with Covariance Matrix Adaptation (CMA-ES) [9]:  
x0: random position in the search space,  
expected initial distance to optimum per coordinate  
Coordist=5
- G3 model with PCX crossover (G3-PCX) [10]:

population size=150,  $\mu=3$ ,  $\lambda=2$ , replace number=1

- Differential Evolution (DE) [11]:  
population size=50, strategy: rand/1/exp

For each test function, each algorithm is run 20 times and the maximum fitness evaluations are set at 50,000 for all algorithms. The mean and standard deviation values of the results are recorded in Table 1.

Table 1: Results achieved by using the six algorithms on six composition functions

	PSO	CPSO	CLPSO
CF1: Mean	1.0000e+002	1.5626e+002	<b>5.7348e-008</b>
Std.	8.1650e+001	1.3427e+002	1.0352e-007
CF2: Mean	1.5591e+002	2.4229e+002	<b>1.9157e+001</b>
Std.	1.3176e+002	1.4895e+002	1.4748e+001
CF3: Mean	1.7203e+002	3.6264e+002	<b>1.3281e+002</b>
Std.	3.2869e+001	1.9631e+002	2.0027e+001
CF4: Mean	<b>3.1430e+002</b>	5.2237e+002	3.2232e+002
Std.	2.0066e+001	1.2209e+002	2.7461e+001
CF5: Mean	8.3450e+001	2.5556e+002	<b>5.3705e+000</b>
Std.	1.0111e+002	1.7563e+002	2.6056e+000
CF6: Mean	8.6142e+002	8.5314e+002	5.0116e+002
Std.	1.2581e+002	1.2798e+002	7.7800e-001

Table 1 (Cont.): Results achieved by using the six algorithms on six composition functions

	CMA-ES	G3-PCX	DE
CF1: Mean	1.0000e+002	6.0000e+001	6.7459e-002
Std.	1.8856e+002	6.9921e+001	1.1057e-001
CF2: Mean	1.6199e+002	9.2699e+001	2.8759e+001
Std.	1.5100e+002	9.9067e+001	8.6277e+000
CF3: Mean	2.1406e+002	3.1980e+002	1.4441e+002
Std.	7.4181e+001	1.2519e+002	1.9401e+001
CF4: Mean	6.1640e+002	4.9296e+002	3.2486e+002
Std.	6.7192e+002	1.4249e+002	1.4784e+001
CF5: Mean	3.5853e+002	2.6021e+001	1.0789e+001
Std.	1.6826e+002	4.1579e+001	2.6040e+000
CF6: Mean	9.0026e+002	7.7208e+002	<b>4.9094e+002</b>
Std.	8.3186e-002	1.8939e+002	3.9461e+001

From the results achieved by different algorithms, we can observe all algorithms generated their best results for CF1, since it is constructed using the unimodal sphere basic function. With 10 sphere functions, CF1 is a multimodal function with 1 global optimum and 9 local optima. Good areas are easier to find and when the global area has been found, the global optimum is not difficult to achieve. But for CF2 and CF3, they are constructed with more complex multimodal Griewank's and Rastrigin's functions. Hence, they have more local optima thereby increasing their complexity. The global area is not easy to find, and even when the global area has been found, the real optimum is difficult to reach.

CF4, CF5 and CF6 are all constructed with different basic functions. We may call them hybrid composition functions. Each function has different properties. Ackley's function, which has a narrow optimum basin, occupies the global area of CF4, and hides the global optimum in a bad fitness area. While in CF5 and CF6, the global areas are occupied by Rastrigin's functions whose local optima are so many that it is difficult to reach the global optimum. Based on CF5, CF6's  $\sigma$  and  $\lambda$  are adjusted to obtain a narrower global optimum area and flatter local optimum areas. From the results, we can observe that CF6 has become more complex than CF5. Our objective of controlling the complexity of the functions using these parameters is achieved successfully. Among these six algorithms, which have shown their excellent performance on the standard benchmark functions failed on all the composition functions though CLPSO gives comparatively better results. Since most real world problems has no clear structure as the standard benchmark functions, it is clear that further research on evolutionary algorithms to solve this class of problems is needed.

#### 4. CONCLUSION

In this paper, we analyze some problems in the benchmark functions frequently used in testing the efficiency of global optimization algorithms. Consequently, we suggest when we test a novel optimization algorithm, we should shift the global optimum's position and test the algorithm on

functions with different properties.

Novel composition functions are introduced to achieve randomly distributed landscapes and controllable properties. By controlling  $f_i, \sigma_i, \lambda_i, bias_i, o_i, M_i$ , users can construct test functions with the properties they need to test the algorithm's performances on different problem scenarios. Except the 6 functions defined in this paper, more composition functions with different properties are available at <http://www.ntu.edu.sg/home/EPNSugan/>. If we set  $bias_i$  to 0 for all  $f_i$ s, these composition functions will be good test problems for niching algorithms that attempt to find all the global optima in one run. In the current work,  $\sigma_i$  and  $\lambda_i$  have the same value for all dimensions for a certain basic function, but in the future work, different  $\sigma_i, \lambda_i$  for different dimensions be set at different values to increase the complexity of the composition test functions.

#### 5. REFERENCES

- [1] W. C. Zhong, J. Liu, M. Z. Xue and L. C. Jiao, "A multiagent genetic algorithm for global numerical optimization". *IEEE Trans. on Systems, Man and Cybernetics (Part B)*, vol. 34, pp. 1128-1141, April 2004.
- [2] C. A. Coello Coello, G. T. Pulido, and M. S. Lechuga, "Handling multiple objectives with particle swarm optimization." *IEEE Transactions on Evolutionary Computation*, 8(3):256 - 279, June 2004.
- [3] F. van den Bergh and A. P. Engelbrecht, "A cooperative approach to particle swarm optimization." *IEEE Transactions on Evolutionary Computation*, 8(3):225 - 239, June 2004.
- [4] Y. W. Leung and Y. P. Wang, "An orthogonal genetic algorithm with quantization for global numerical optimization." *IEEE Trans. on Evolutionary Computation*, 5(1):41-53, Feb. 2001.
- [5] R. Salomon, "Reevaluating genetic algorithm performance under coordinate rotation of benchmark functions," *BioSystems*, vol. 39, pp. 263-278, 1996.
- [6] H-P Paul Schwefel, *Evolution and Optimum Seeking*,

John Wiley & Sons Inc, 1993

- [7] Y. Shi and R. C. Eberhart, "A modified particle swarm optimizer". *Proc. of the IEEE Congress on Evolutionary Computation (CEC 1998)*, Piscataway, NJ, pp. 69-73, 1998.
- [8] J. J Liang, A. K. Qin, P. N. Suganthan, and S. Baskar, "Evaluation of Comprehensive Learning Particle Swarm Optimizer." *Lecture Notes in Computer Science*, Vol. 3316, pp. 230-235, 2004
- [9] N. Hansen, S. D. Muller and P. Koumoutsakos, "Reducing the Time Complexity of the Derandomized Evolution Strategy with Covariance Matrix Adaptation (CMA-ES)." *Evolutionary Computation*, 11(1):1-18, 2003
- [10] K. Deb, A. Anand, and D. Joshi, "A Computationally Efficient Evolutionary Algorithm for Real-Parameter Optimization," *KanGAL Report* No. 2002003, April 2002.
- [11] R. Storn, and K. Price, "Differential evolution-A simple and Efficient Heuristic for Global Optimization over Continuous Spaces." *Journal of Global Optimization*, 11:341-359, 1997.