Benchmark Functions for the CEC'2008 Special Session and Competition on Large Scale Global Optimization

K. Tang[1], X. Yao[1, 2], P. N. Suganthan[3], C. MacNish[4], Y. P. Chen[5], C. M. Chen[5], Z. Yang[1]

[1]Nature Inspired Computation and Applications Laboratory (NICAL), Department of Computer Science and Technology, the University of Science and Technology of China, Hefer, Anhui, China
[2]The Center of Excellence for Research in Computational Intelligence and Applications (CERCIA), School of Computer Science, the University of Birminham, Edgbaston, Birmingham B15 2TT, U.K.
[3] School of EEE, Nanyang Technological University, Singapore, 639798
[4] School of Computer Science & Software Engineering, the University of Western Australia, M002, 35 Stirling Highway, Crawley, Western Australia, 6009
[5]Natural Computing Laboratory, Department of Computer Science, National Chiao Tung University, Taiwan

ketang@ustc.edu.cn, x.yao@cs.bham.ac.uk, epnsugan@ntu.edu.sg, cara@csse.uwa.edu.au,

ypchen@nclab.tw, ccming@nclab.tw, zhyuyang@mail.ustc.edu.cn

In the past two decades, different kinds of nature-inspired optimization algorithms have been designed and applied to solve optimization problems, e.g., simulated annealing (SA), evolutionary algorithms (EAs), differential evolution (DE), particle swarm optimization (PSO), Ant Colony Optimisation (ACO), Estimation of Distribution Algorithms (EDA), etc. Although these approaches have shown excellent search abilities when applying to some 30-100 dimensional problems, many of them suffer from the "curse of dimensionality", which implies that their performance deteriorates quickly as the dimensionality of search space increases. The reasons appear to be two-fold. First, complexity of the problem usually increases with the size of problem, and a previously successful search strategy may no longer be capable of finding the optimal solution. Second, the solution space of the problem increases exponentially with the problem size, and a more efficient search strategy is required to explore all the promising regions in a given time budget.

Historically, scaling EAs to large size problems have attracted much interest, including both theoretical and practical studies. The earliest practical approach might be the parallelism of an existing EA. Later, cooperative coevolution appears to be another promising method. However, existing work on this topic are often limited to the test problems used in individual studies, and a systematic evaluation platform is not available in the literature for comparing the scalability of different EAs.

In this report, 6 benchmark functions are given based on [1] and [2] for high-dimensional optimization. All of them are scalable for any size of dimension. The codes in Matlab and C for them are available at **http://nical.ustc.edu.cn/cec08ss.php**. The other benchmark function (Function 7 - FastFractal "DoubleDip") is generated based on [3] [4]. The C code for function 7 has been contributed by Ales Zamuda from the University of Maribor, Slovenia. It uses the GJC / CNI interface to run the Java code from C++. In the package, C code is provided in a separate zip file, named “cec08-f7-cpp.zip”.

The mathematical formulas and properties of these functions are described in Section 2, and the evaluation criteria are given in Section 3.

1. Summary of the 7 CEC’08 Test Functions

- **Unimodal Functions (2):**
  - F_1: Shifted Sphere Function
  - F_2: Shifted Schwefel’s Problem 2.21

- **Multimodal Functions (5):**
  - F_3: Shifted Rosenbrock’s Function
  - F_4: Shifted Rastrigin’s Function
  - F_5: Shifted Griewank’s Function
  - F_6: Shifted Ackley’s Function
  - F_7: FastFractal “DoubleDip” Function

2. Definitions of the 7 CEC'08 Test Functions

2.1 Unimodal Functions:

2.1.1. *F_1: Shifted Sphere Function*

$$F_1(\mathbf{x}) = \sum_{i=1}^{D} z_i^{\,2} + f\_bias_1 ,\ \mathbf{z} = \mathbf{x} - \mathbf{o} ,\ \mathbf{x} = [x_1, x_2, ..., x_D]$$

D: dimensions. $\mathbf{o} = [o_1, o_2, ..., o_D]$: the shifted global optimum.

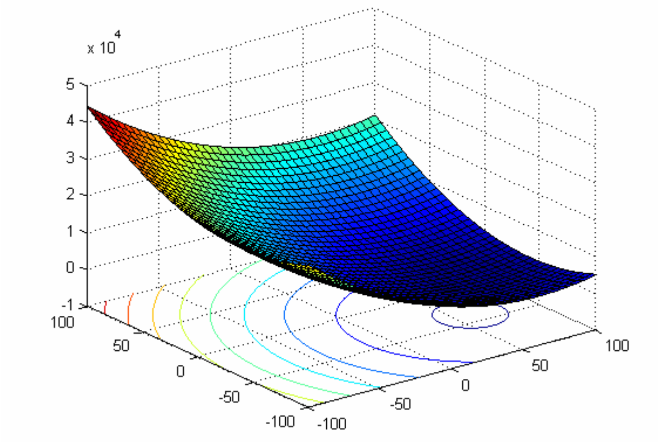

Figure 2-1 3-*D* map for 2-*D* function

Properties:

- Unimodal
- Shifted
- Separable
- Scalable
- Dimension *D* as 100, 500 and 1000
- $\mathbf{x} \in [-100,100]^D$, Global optimum: $\mathbf{x}^* = \mathbf{o}$, $F_1(\mathbf{x}^*) = f\_bias_1 = -450$

Associated Data files:

Name:	sphere__shift_func_data.mat		
Variable:	**o**	1*1000 vector	the shifted global optimum
	When using, cut **o**=**o**(1:*D*) for *D*=100, 500		

2.1.2. F_2: *Schwefel's Problem 2.21*

$F_2(\mathbf{x}) = \max_i \{|z_i|, 1 \le i \le D\} + f\_bias_2$, $\mathbf{z} = \mathbf{x} - \mathbf{o}$, $\mathbf{x} = [x_1, x_2, ..., x_D]$

D: dimensions. $\mathbf{o} = [o_1, o_2, ..., o_D]$: the shifted global optimum.

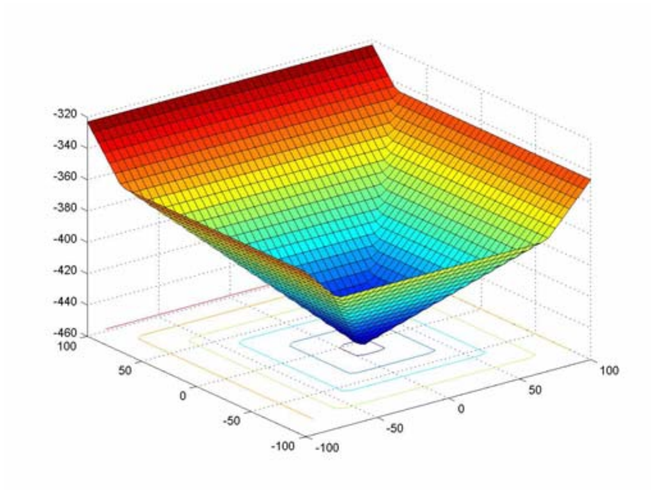

Figure 2-2 3-*D* map for 2-*D* function

Properties:

- Unimodal
- Shifted
- Non-separable
- Scalable
- Dimension *D* as 100, 500 and 1000
- $\mathbf{x} \in [-100,100]^D$, Global optimum: $\mathbf{x}^* = \mathbf{o}$, $F_1(\mathbf{x}^*) =$ *f_bias₁* = - 450

Associated Data files:

Name:	schwefel_shift_func_data.mat		
Variable:	**o**	1*1000 vector	the shifted global optimum
	When using, cut **o**=**o**(1:*D*) for *D*=100, 500		

2.2 Multimodal Functions

2.2.1. F_3: *Shifted Rosenbrock's Function*

$$F_3(\mathbf{x}) = \sum_{i=1}^{D-1}(100(z_i^2 - z_{i+1})^2 + (z_i - 1)^2) + f\_bias_3,\ \mathbf{z} = \mathbf{x} - \mathbf{o} + 1, \mathbf{x} = [x_1, x_2, ..., x_D]$$

D: dimensions

$\mathbf{o} = [o_1, o_2, ..., o_D]$: the shifted global optimum

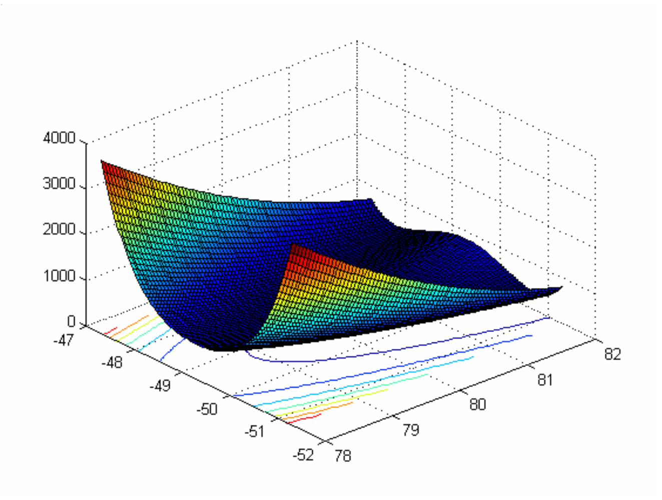

Figure 2-3 3-*D* map for 2-*D* function

Properties:

- Multi-modal
- Shifted
- Non-separable
- Scalable
- Having a very narrow valley from local optimum to global optimum
- Dimension *D* as 100, 500 and 1000
- $\mathbf{x} \in [-100, 100]^D$, Global optimum $\mathbf{x}^* = \mathbf{o}$, $F_3(\mathbf{x}^*) = f\_bias_3 = 390$

Associated Data file:

Name:	rosenbrock_shift_func_data.mat		
Variable:	**o**	1*1000 vector	the shifted global optimum
	When using, cut **o**=**o**(1:*D*) for *D*=100, 500		

2.2.2. F_4: *Shifted Rastrigin's Function*

$$F_4(\mathbf{x}) = \sum_{i=1}^{D} (z_i^2 - 10\cos(2\pi z_i) + 10) + f\_bias_4, \ \mathbf{z} = \mathbf{x} - \mathbf{o}, \ \mathbf{x} = [x_1, x_2, ..., x_D]$$

D: dimensions

$\boldsymbol{o} = [o_1, o_2, ..., o_D]$: the shifted global optimum

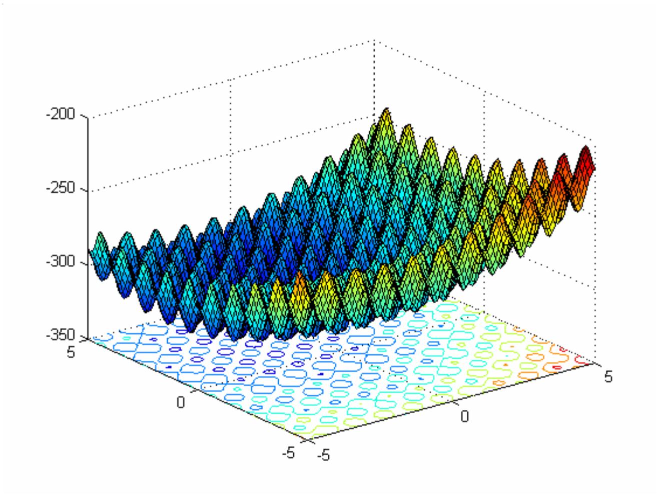

Figure 2-4 3-*D* map for 2-*D* function

Properties:

- Multi-modal
- Shifted
- Separable
- Scalable
- Local optima's number is huge
- Dimension *D* as 100, 500 and 1000
- $\mathbf{x} \in [-5, 5]^D$, Global optimum $\mathbf{x}^* = \mathbf{o}$, $F_4(\boldsymbol{x}^*) = f\_bias_4 = -330$

Associated Data file:

Name:	rastrigin_shifit_func_data.mat		
Variable:	**o**	1*1000 vector	the shifted global optimum
	When using, cut **o**=**o**(1:*D*) for *D*=100, 500		

2.2.3. *F_5: Shifted Griewank's Function*

$$F_5(\mathbf{x}) = \sum_{i=1}^{D} \frac{z_i^{\,2}}{4000} - \prod_{i=1}^{D} \cos\left(\frac{z_i}{\sqrt{i}}\right) + 1 + f\_bias_5 \ , \ \mathbf{z} = (\mathbf{x} - \mathbf{o}), \ \mathbf{x} = [x_1, x_2, ..., x_D]$$

D: dimensions

$\mathbf{o} = [o_1, o_2, ..., o_D]$: the shifted global optimum

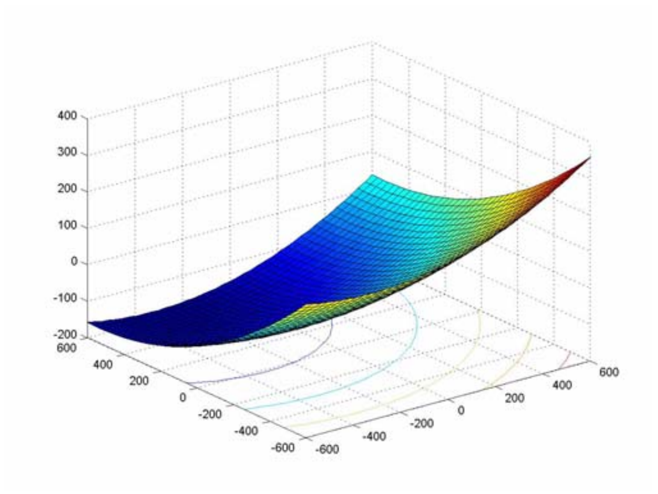

Figure 2-5 3-*D* map for 2-*D* function

Properties:

- Multi-modal
- Shifted
- Non-separable
- Scalable
- Dimension *D* as 100, 500 and 1000
- $\mathbf{x} \in [-600, 600]^D$, Global optimum $\mathbf{x}^* = \mathbf{o}$, $F_5(\mathbf{x}^*) =$ *f_bias*$_5$ = -180

Associated Data file:

Name:	griewank_shfit_func_data.mat		
Variable:	**o**	1*1000 vector	the shifted global optimum
	When using, cut **o**=**o**(1:*D*) for *D*=100, 500		

2.2.4. F_6: *Shifted Ackley's Function*

$$F_6(\mathbf{x}) = -20\exp(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} z_i^2}) - \exp(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi z_i)) + 20 + e + f\_bias_6,\ \mathbf{z} = (\mathbf{x}-\mathbf{o}),$$

$\mathbf{x} = [x_1, x_2, ..., x_D]$, *D:* dimensions

$\mathbf{o} = [o_1, o_2, ..., o_D]$: the shifted global optimum;

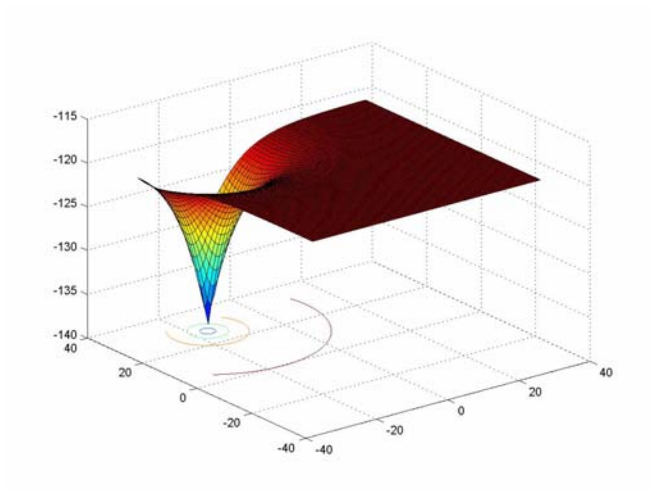

Figure 2-6 3-*D* map for 2-*D* function

Properties:

- Multi-modal
- Shifted
- Separable
- Scalable
- Dimension *D* as 100, 500 and 1000
- $\mathbf{x} \in [-32, 32]^D$, Global optimum $\mathbf{x}^* = \mathbf{o}$, $F_6(\mathbf{x}^*) = f\_bias_6 = -140$

Associated Data file:

Name:	ackley_shift_func_data.mat		
Variable:	**o**	1*1000 vector	the shifted global optimum
	When using, cut **o**=**o**(1:*D*) for *D*=100, 500		

2.2.5. *F*$_7$: *FastFractal "DoubleDip" Function*

Note: To make use of this function in a .m file, please include the following line at the beginning of your file that calls the function:

javaclasspath('FractalFunctions.jar')

$$F_7(\mathbf{x}) = \sum_{i=1}^{D} fractal1D(x_i + twist(x_{(i \bmod D)+1}))$$

$$twist(y) = 4(y^4 - 2y^3 + y^2)$$

$$fractal1D(x) \approx \sum_{k=1}^{3} \sum_{1}^{2^{k-1}} \sum_{1}^{ran2(o)} doubledip(x, ran1(o), \frac{1}{2^{k-1}(2 - ran1(o))})$$

$$doubledip(x,c,s) = \begin{cases} (-6144(x-c)^6 + 3088(x-c)^4 - 392(x-c)^2 + 1)s, -0.5 < x < 0.5 \\ 0, \text{otherwise} \end{cases}$$

$\mathbf{x} = [x_1, x_2, ..., x_D]$, *D:* dimensions

ran1(o): double, pseudorandomly chosen, with seed *o*, with equal probability from the interval [0,1]

ran2(o): integer, pseudorandomly chosen, with seed *o*, with equal probability from the set {0,1,2}
fractal1D(x) is an approximation to a recursive algorithm, it does not take account of wrapping at the boundaries, or local re-seeding of the random generators - please use the executable provided

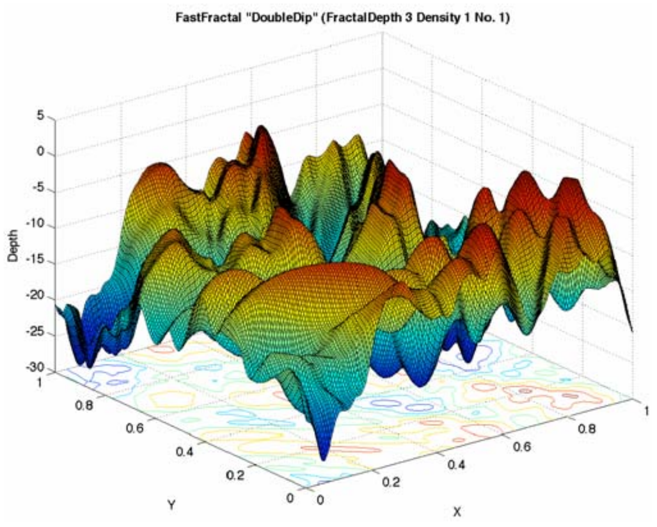

Figure 2-7 3-*D* map for 2-*D* function

Properties:

- Multi-modal
- Non-separable
- Scalable
- Dimension D as 100, 500 and 1000
- $\mathbf{x} \in [-1,1]^D$, Global optimum unknown, $F_7(\mathbf{x}^*)$ unknown

Associated Data file:

Name:	fastfractal_doubledip_data.mat		
Variable:	**o**	integer	seeds the random generators

3. Evaluation Criteria

3.1 Description of the Evaluation Criteria

Problems: 7 minimization problems

Dimensions: *D*=100, 500, 1000

Runs / problem: 25 (**Do not run many 25 runs to pick the best run**)

Max_FES: 5000**D* (Max_FES_100D= 500000; for 500D=2500000; for 1000D=5000000)

Initialization: Uniform random initialization within the search space

Global Optimum: All problems have the global optimum within the given bounds and there is no need to perform search outside of the given bounds for these problems.

Termination: Terminate when reaching Max_FES.

1) Record function error value (f(*x*)-f(*x)) after $\frac{1}{100}$ *FES, $\frac{1}{10}$ *FES and FES at termination (due to Max_FES) for each run.**

For each function, sort the error values in 25 runs from the smallest (best) to the largest (worst**)**

Present the following:

1st (best), 7th, 13th (median), 19th, 25th (worst) function values

Mean and STD for the 25 runs

NOTE: The value of f(*x**) is not available for **function 7** (FastFractal "DoubleDip" Function). In this case, please record the function value f(*x*) directly (i.e., we regard the f(*x**) is **0**).

2) Convergence Graphs (or Run-length distribution graphs)

Convergence Graphs for each problem for ***D*=1000**. The graph would show the median performance of the total runs with termination by the Max_FES. The semi-log graphs should show log10(f(***x***)- f(***x****)) vs FES for each problem.

3) Parameters

We discourage participants searching for a distinct set of parameters for each problem/dimension/etc. Please provide details on the following whenever applicable:

a) All parameters to be adjusted

b) Actual parameter values used.

c) Estimated cost of parameter tuning in terms of number of FEs

d) Corresponding dynamic ranges

e) Guidelines on how to adjust the parameters

4) Encoding

If the algorithm requires encoding, then the encoding scheme should be independent of the specific problems and governed by generic factors such as the search ranges.

FYI: Estimated runtime for the test suite

Dimension: 1000D

Problems: Functions 1-7

Algorithm: Differential Evolution

Runs: Only one run

Max_FES: 5000000

PC: CPU-P4 2.8G, RAM-512M

Runtime: 15 hours

3.2 Example

System: Windows XP (SP1)

CPU: Pentium(R) 4 3.00GHz

RAM: 1 G

Language: Matlab 7.1

Algorithm: Particle Swarm Optimizer (PSO)

Results, *D*=100, Max_FES=500000

Table 3-1 Error Values Achieved for Problems 1-7, with *D*=100

FES \ Prob		1	2	3	4	5	6	7
5.00e+3	1st(Best)							
	7th							
	13th(Median)							
	19th							
	25th (Worst)							
	Mean							
	Std							
5.00e+4	1st(Best)							
	7th							
	13th(Median)							
	19th							
	25th (Worst)							
	Mean							
	Std							
5.00e+5	1st(Best)							
	7th							
	13th(Median)							
	19th							
	25th (Worst)							
	Mean							
	Std							

Results, *D*=500, Max_FES=2500000

Table 3-2 Error Values Achieved for Problems 1-7, with *D*=500

FES \ Prob		1	2	3	4	5	6	7
2.50e+4	1st(Best)							
	7th							
	13th(Median)							
	19th							
	25th (Worst)							
	Mean							
	Std							
2.50e+5	1st(Best)							
	7th							
	13th(Median)							
	19th							
	25th (Worst)							
	Mean							
	Std							
2.50e+6	1st(Best)							
	7th							
	13th(Median)							
	19th							
	25th (Worst)							
	Mean							
	Std							

Results, *D*=1000, Max_FES=5000000

Table 3-3 Error Values Achieved for Problems 1-7, with *D*=1000

FES	Prob	1	2	3	4	5	6	7
5.00e+4	1st(Best)							
	7th							
	13th(Median)							
	19th							
	25th (Worst)							
	Mean							
	Std							
5.00e+5	1st(Best)							
	7th							
	13th(Median)							
	19th							
	25th (Worst)							
	Mean							
	Std							
5.00e+6	1st(Best)							
	7th							
	13th(Median)							
	19th							
	25th (Worst)							
	Mean							
	Std							

Convergence Graphs (1000D)

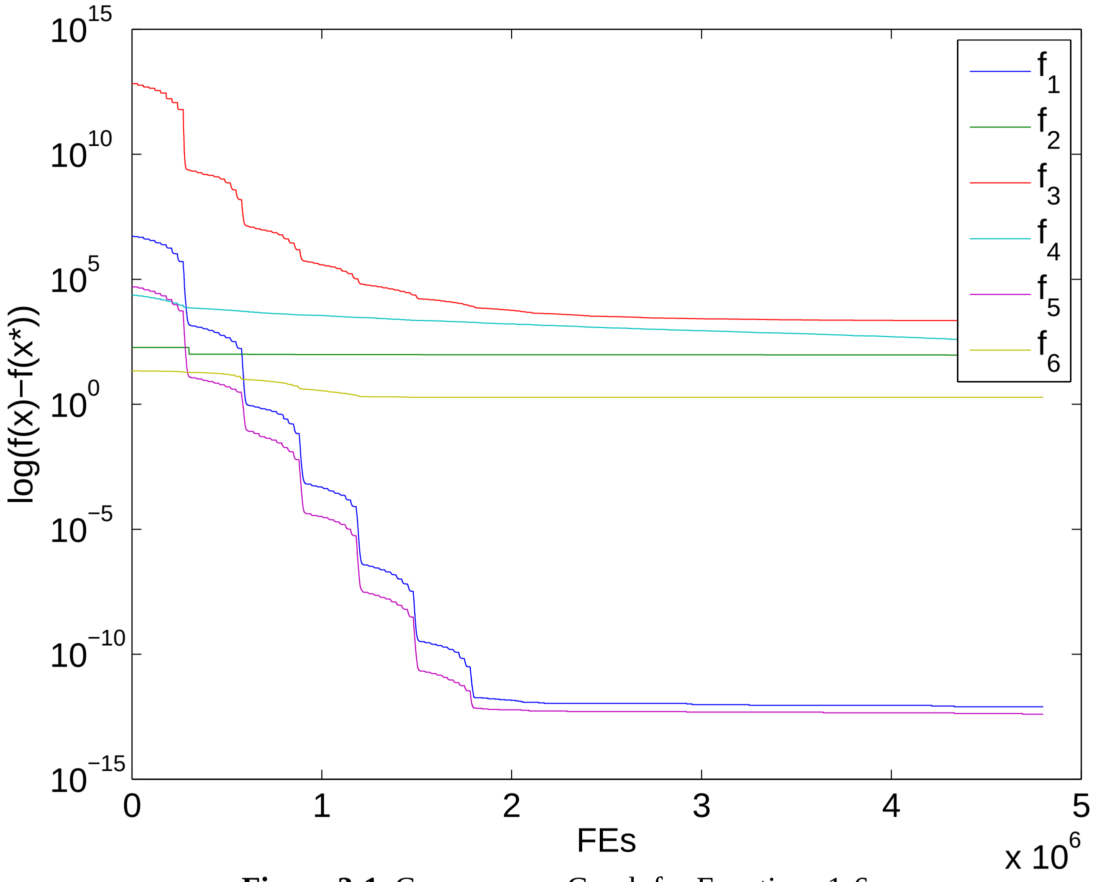

Figure 3-1 Convergence Graph for Functions 1-6

References:

[1] 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 CEC 2005 Special Session on Real-Parameter Optimization," Technical Report, Nanyang Technological University, Singapore, http://www.ntu.edu.sg/home/EPNSugan, 2005.

[2] X. Yao, Y. Liu, and G. Lin, "Evolutionary Programming Made Faster," IEEE Transactions on Evolutionary Computation, vol. 3, no. 2, pp. 82–102, 1999.

[3] MacNish, C.,Towards Unbiased Benchmarking of Evolutionary and Hybrid Algorithms for Real-valued Optimisation, Connection Science, Vol.19, No. 4, December 2007. To appear.

[4] MacNish, C., Benchmarking Evolutionary and Hybrid Algorithms using Randomized Self-Similar Landscapes, Proc. 6th International Conference on Simulated Evolution and Learning (SEAL'06), LNCS 4247, pp. 361-368, Springer, 2006.