Benchmark Generator for CEC'2009 Competition on Dynamic Optimization

C. Li[1], S. Yang[1], T. T. Nguyen[2], E. L. Yu[5], X. Yao[2], Y. Jin[3]
H.-G. Beyer[4], and P. N. Suganthan[5]

October 26, 2008

[1]Department of Computer Science, University of Leicester
University Road, Leicester LE1 7RH, UK
[2]CERCIA, School of Computer Science, University of Birmingham
Edgbaston, Birmingham B15 2TT, U.K.
[3]Honda Research Institute Europe, 63073 Offenbach/Main, Germany
[4]Department of Computer Science, Vorarlberg University of Applied Sciences, Austria
[5]School of EEE, Nanyang Technological University, Singapore, 639798

cl160@le.ac.uk, s.yang@mcs.le.ac.uk, T.T.Nguyen@cs.bham.ac.uk,
YuLing@ntu.edu.sg, x.yao@cs.bham.ac.uk, yaochu.jin@honda-ri.de,
Hans-Georg.Beyer@fhv.at, epnsugan@ntu.edu.sg

Evolutionary algorithms(EAs) have been widely applied to solve stationary optimization problems. However, many real-world applications are actually dynamic. In order to study the performance of EAs in dynamic environments, one important task is to develop proper dynamic benchmark problems.

Over the years, researchers have applied a number of dynamic test problems to compare the performance of EAs in dynamic environments, e.g., the "moving peaks" benchmark (MPB) proposed by Branke [1], the DF1 generator introduced by Morrison and De Jong [6], the single- and multi-objective dynamic test problem generator by dynamically combining different objective functions of exiting stationary multi-objective benchmark problems suggested by Jin and Sendhoff [2], Yang and Yao's exclusive-or (XOR) operator [10, 11, 12], Kang's dynamic traveling salesman problem (DTSP) [3] and dynamic multi knapsack problem (DKP), etc.

Though a number of DOP generators exist in the literature, there is no unified approach of constructing dynamic problems across the binary space, real space and combinatorial space so far. This report uses the generalized dynamic benchmark generator (GDBG) proposed in [4], which construct dynamic environments for all the three solution spaces. Especially, in the real space, we introduce a rotation method instead of shifting the positions of peaks as in the MPB and DF1 generators. The rotation method can overcome the problem of unequal challenge per change for algorithms of the MPB generator, which happens when the peak positions bounce back from the boundary of the landscape.

This report gives two benchmark instances from the GDBG system in the real space. The source code for the two benchmark instances and an test example using the PSO algorithm are available at `http://www.cs.le.ac.uk/people/syang/ECiDUE/DBG.tar.gz` and `http://www.ntu.edu.sg/home/epnsugan/DBG.tar.gz` respectively. The test functions and the best results will also be uploaded to the Evolutionary Computation Benchmark Repository [8] at `http://www.cs.bham.ac.uk/research/projects/ecb/`.

The definition of two benchmark instances are described in Section 2. Section 3 gives the description of seven test problems and performance measurement is given in Section 4.

1 Framework of the GDBG system

DOPs can be defined as follows:

$$F = f(x, \phi, t) \tag{1}$$

where F is the optimization problem, f is the cost function, x is a feasible solution in the solution set $\mathbf{X}$, t is the real-world time, and ϕ is the system control parameter, which determines the solution distribution in the fitness landscape.

In the GDBG system, the dynamism results from a deviation of solution distribution from the current environment by tuning the system control parameters. It can be described as follows:

$$\phi(t+1) = \phi(t) \oplus \Delta\phi \tag{2}$$

where $\Delta\phi$ is a deviation from the current system control parameters. Then, we can get the new environment at the next moment $t+1$ as follows:

$$f(x, \phi, t+1) = f(x, \phi(t) \oplus \Delta\phi, t) \tag{3}$$

There are six change types of the system control parameters in the GDBG system. They are small step change, large step change, random change, chaotic change, recurrent change, and recurrent change with noise. The framework of the six change types are described as follows:

Framework of DynamicChanges

switch(change type)

case small step:

$$\Delta\phi = \alpha \cdot \|\phi\| \cdot r \cdot \phi_{severity} \tag{4-1}$$

case large step:

$$\Delta\phi = \|\phi\| \cdot (\alpha \cdot sign(r) + (\alpha_{max} - \alpha) \cdot r) \cdot \phi_{severity} \tag{4-2}$$

case random:

$$\Delta\phi = N(0,1) \cdot \phi_{severity} \tag{4-3}$$

case chaotic:

$$\phi(t+1) = A \cdot (\phi(t) - \phi_{min}) \cdot (1 - (\phi(t) - \phi_{min})/\|\phi\|) \tag{4-4}$$

case recurrent:

$$\phi(t+1) = \phi_{min} + \|\phi\|(\sin(\frac{2\pi}{P}t + \varphi) + 1)/2 \tag{4-5}$$

case recurrent with noisy:

$$\phi(t+1) = \phi_{min} + \|\phi\|(\sin(\frac{2\pi}{P}t + \varphi) + 1)/2 + N(0,1) \cdot noisy_{severity} \tag{4-6}$$

where $\|\phi\|$ is the change range of ϕ, $\phi_{severity}$ is a constant number that indicates change severity of ϕ, ϕ_{min} is the minimum value of ϕ, $noisy_{severity} \in (0,1)$ is noisy severity in recurrent with noisy change. $\alpha \in (0,1)$ and $\alpha_{max} \in (0,1)$ are constant values, which are set to 0.04 and 0.1 in the GDBG system. A logistics function is used in the chaotic change type, where A is a

positive constant between $(1.0, 4.0)$, if ϕ is a vector, the initial values of the items in ϕ should be different within $\|\phi\|$ in chaotic change. P is the period of recurrent change and recurrent change with noise, φ is the initial phase, r is a random number in $(-1, 1)$, $sign(x)$ returns 1 when x is greater than 0, returns -1 when x is less than 0, otherwise, returns 0. $N(0, 1)$ denotes a normally distributed one dimensional random number with mean zero and standard deviation one.

To simulate some real world problems, where the number of variables changes online over the time processes. For example, in dynamic scheduling problem one machine might break down, or in the routing problem you may have to add a new node into an already-planned route. On the other hand, the performance of EAs may deteriorate quickly as the dimensionality increases due to increasing the problem complexity and search space. In GDBG system, the number of dimensions changes as follows:

$$D(t+1) = D(t) + sign \cdot \Delta D \tag{5}$$

where ΔD is a predefined constant, which the default value of is 1. If $D(t) = Max\_D, sign = -1$; if $D(t) = Min\_D, sign = 1$. Max_D and Min_D are the maximum and minimum number of dimensions. When the number of dimension deceases by 1, just the last dimension is removed from the fitness landscape, the fitness landscape of the left dimensions doesn't change. When the number of dimension increases by 1, a new dimension with random value is added into the fitness landscape. Dimensional change *only happens following* the non-dimensional change.

2 Benchmark instances

The two benchmark instances are: Dynamic rotation peak benchmark generator (DRPBG) and Dynamic composition benchmark generator (DCBG)

2.1 Dynamic rotation peak benchmark generator

The proposed benchmark uses a similar peak-composition structure to those of MPB [1] and DF1[6]. Given a problem $f(x, \phi, t), \phi = (\vec{H}, \vec{W}, \vec{X})$, where $\vec{H}, \vec{W}$ and $\vec{X}$ denote the peak height, width and position respectively. The function of $f(x, \phi, t)$ is defined as follows:

$$f(x, \phi, t) = \max_{i=1}^{m} (\vec{H}_i(t)/(1 + \vec{W}_i(t) \cdot \sqrt{\sum_{j=1}^{n} \frac{(x_j - \vec{X}_j^i(t))^2}{n}})) \tag{6}$$

where m is the number of peaks, n is the number of dimensions.

$\vec{H}$ and $\vec{W}$ change as follows:

$$\vec{H}(t+1) = \text{DynamicChanges}(\vec{H}(t))$$
$$\vec{W}(t+1) = \text{DynamicChanges}(\vec{W}(t))$$

where in the height change,*height_severity* should read $\phi\_h_{severity}$ according to Eq. (4) and $\|\phi\_h\|$ is height range. Accordingly, *width_severity* and width range should read $\phi\_w_{severity}$ and $\|\phi\_w\|$ in the width change.

A rotation matrix[7] $R_{ij}(\theta)$ is obtained by rotating the projection of $\overrightarrow{x}$ in the plane $i-j$ by an angle θ from the i-th axis to the j-th axis. The peak position $\vec{X}$ is changed by the following algorithm:

Step 1. Randomly select l dimensions (l is an even number) from the n dimensions to compose a vector $r = [r_1, r_2, ..., r_l]$.

Step 2. For each pair of dimension $r[i]$ and dimension $r[i+1]$, construct a rotation matrix $R_{r[i],r[i+1]}(\theta(t))$, $\theta(t)$=DynamicChanges($\theta(t-1)$).

Step 3. A transformation matrix $A(t)$ is obtained by:
$A(t) = R_{r[1],r[2]}(\theta(t)) \cdot R_{r[3],r[4]}(\theta(t)) \cdots R_{r[l-1],r[l]}(\theta(t))$

Step 4. $\vec{X}(t+1) = \vec{X}(t) \cdot A(t)$

where the change severity of θ ($\phi\_\theta_{severity}$) is set 1 in Eq. (4), the range of θ should read $\|\phi\_\theta\|$, $\|\phi\_\theta\| \in (-\pi, \pi)$. For the value of l, if n is an even number, $l = n$; otherwise $l = n - 1$.

NOTE: For recurrent and recurrent with noisy change, $\|\phi\_\theta\|$ is within $(0, \pi/6)$.

2.2 Dynamic composition benchmark generator

The dynamic composition functions are extended from the static composition functions developed by Suganthan et al. [5, 9]. The composition function can be described as:

$$F(x, \phi, t) = \sum_{i=1}^{m} (w_i \cdot (f_i'((x - \vec{O}_i(t) + O_{iold})/\lambda_i \cdot \vec{M}_i) + \vec{H}_i(t))) \tag{7}$$

where the system control parameter $\phi = (\vec{O}, \vec{M}, \vec{H})$, $F(x)$ is the composition function, $f_i(x)$ is i-th basic function used to construct the composition function. m is the number of basic functions, $\vec{M}_i$ is orthogonal rotation matrix for each $f_i(x)$, $\vec{O}_i(t)$ is the optimum of the changed $f_i(x)$ caused by rotating the landscape at the time t. O_{iold} is the optimum of the original $f_i(x)$ without any change, the O_{iold} is 0 for all the basic functions used in this report. The weight value w_i for each $f_i(x)$ is calculated as:

$$w_i = exp(-sqrt(\frac{\sum_{k=1}^{n} (x_k - o_i^k + o_{iold}^k)^2}{2n\sigma_i^2}))$$

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

$$w_i = w_i / \sum_{i=1}^{m} w_i$$

where σ_i is the converge range factor of $f_i(x)$, whose default value is 1.0, λ_i is the stretch factor for each $f_i(x)$, which is defined as:

$$\lambda_i = \sigma_i \cdot \frac{X_{max} - X_{min}}{x_{max}^i - x_{min}^i}$$

where $[X_{max}, X_{min}]^n$ is the search range of $F(x)$ and $[x_{max}^i, x_{min}^i]^n$ is the search range of $f_i(x)$.

In Eq. (7), $f_i'(x) = C \cdot f_i(x)/|f_{max}^i|$, where C is a predefined constant, which is set to 2000, and f_{max}^i is the estimated maximum value of $f_i(x)$, which is estimated as:

$$f_{max}^i = f_i(x_{max} \cdot M_i)$$

In the composition DBG, $\vec{M}$ is initialized using the above transformation matrix construction algorithm and then remains unchanged. The dynamism of the system control parameter $\vec{H}$ and $\vec{O}$ are changed as the parameters $\vec{H}$ and $\vec{X}$ in Dynamic rotation peak benchmark generator.

NOTE: For both DRPBG and DCBG, chaotic change of peaks locations directly operates on the value of each dimension instead of using rotation matrix due to simulate chaotic systems in real applications.

Five basic benchmark functions are used in the GDBG system. Table 1 shows the details of the five functions.

Table 1: Details of the basic benchmark functions

name	function	range
Sphere	$f(x) = \sum_{i=1}^{n} x_i^2$	[-100,100]
Rastrigin	$f(x) = \sum_{i=1}^{n} (x_i^2 - 10\cos(2\pi x_i) + 10)$	[-5,5]
Weierstrass	$f(x) = \sum_{i=1}^{n} (\sum_{k=0}^{k_{max}} [a^k \cos(2\pi b^k (x_i + 0.5))]) - n \sum_{k=0}^{k_{max}} [a^k \cos(\pi b^k)]$	
	$a = 0.5, b = 3, k_{max} = 20$	[-0.5,0.5]
Griewank	$f(x) = \frac{1}{4000} \sum_{i=1}^{n} (x_i)^2 - \prod_{i=1}^{n} \cos(\frac{x_i}{\sqrt{i}}) + 1$	[-100,100]
Ackley	$f(x) = -20\exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}) - \exp(\frac{1}{n}\sum_{i=1}^{n} \cos(2\pi x_i)) + 20 + e$	[-32,32]

3 Problem definition and parameters settings

Overview of test functions on real space

F_1: Rotation peak function

F_2: Composition of Sphere's function

F_3: Composition of Rastrigin's function

F_4: Composition of Griewank's function

F_5: Composition of Ackley's function

F_6: Hybrid Composition function

For all test functions:

Dimension: n(fixed)= 10; n(changed)$\in [5, 15]$

Search range: $x \in [-5, 5]^n$

Change frequency: $frequency = 10,000 * n$ FES

The number of changes: $num\_change = 60$

Period: $p = 12$

Severity of recurrent with noisy: $noisy_{severity} = 0.8$

Chaotic constant: $A = 3.67$

Chaotic initialization: If ϕ is a vector, the initial values of the items in ϕ should be randomly generated using uniform distribution within $\|\phi\|$ in Eq. (4)

Step severity: $\alpha = 0.04$

Maximum of α: $\alpha_{max} = 0.1$

Height range: $h \in [10, 100]$

Initial height: $initial\_height = 50$

Height severity: $\phi\_h_{severity} = 5.0$

For all composition functions:

The number of basic function $m = 10$

Converge range factor: $\sigma_i = 1.0, i = 1, 2, \cdots, n$

$C = 2000$

3.1 F_1: Rotation peak function

The number of peaks: $m = 10, 50$

Width range: $w \in [1, 10]$

Width severity: $\phi\_w_{severity} = 0.5$

Initial width: $initial\_width = 5$

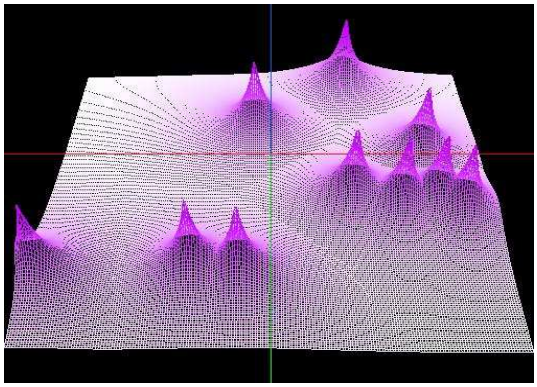

Figure 1: 3-D map for 2-D function of F_1.

Properties

♠ Multi-modal

♠ Scalable

♠ Rotated

♠ The number of local optima are artificially controlled

♠ $x \in [-5, 5]^n$, Global optimum $x^*(t) = \vec{O}_i, F(x^*(t)) = H_i(t), H_i(t) = max_j^m H_j$

3.2 F_2: Composition of Sphere's function

Basic functions: $f_1 - f_{10}$ =Sphere's function

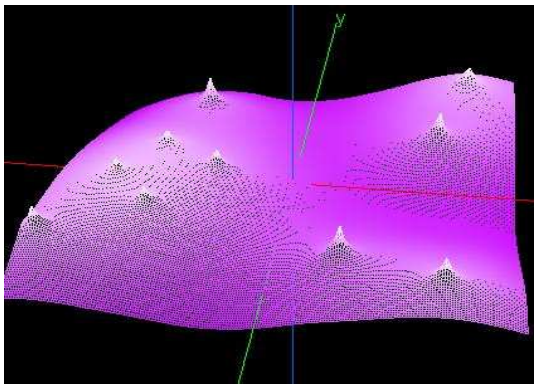

Figure 2: 3-D map for 2-D function of F_2.

Properties

♠ Multi-modal

♠ Scalable

♠ Rotated

♠ 10 local optima

♠ $x \in [-5,5]^n$, Global optimum $x^*(t) = \vec{O}_i, F(x^*(t)) = H_i(t), H_i(t) = min_j^m H_j$

3.3 F_3:Composition of Rastrigin's function

Basic functions: $f_1 - f_{10}$ =Rastrigin's function

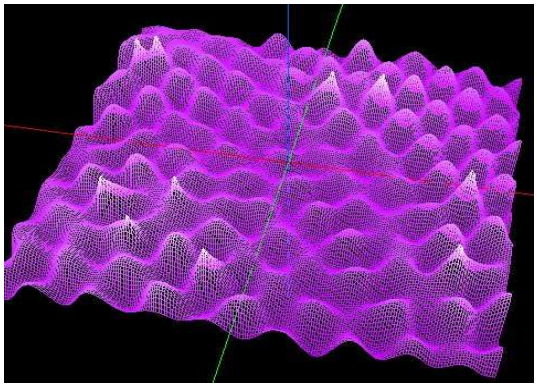

Figure 3: 3-D map for 2-D function of F_3.

Properties

♠ Multi-modal

♠ Scalable

♠ Rotated

♠ A huge number of local optima

♠ $x \in [-5,5]^n$, Global optimum $x^*(t) = \vec{O}_i, F(x^*(t)) = H_i(t), H_i(t) = min_j^m H_j$

3.4 F_4:Composition of Griewank's function

Basic functions: $f_1 - f_{10}$ =Griewank's function

Properties

♠ Multi-modal

♠ Scalable

♠ Rotated

♠ A huge number of local optima

♠ $x \in [-5,5]^n$, Global optimum $x^*(t) = \vec{O}_i, F(x^*(t)) = H_i(t), H_i(t) = min_j^m H_j$

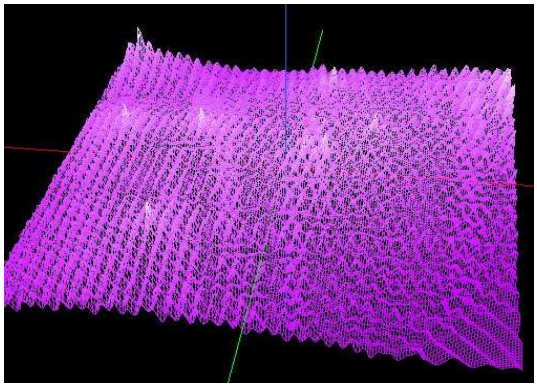

Figure 4: 3-D map for 2-D function of F_4.

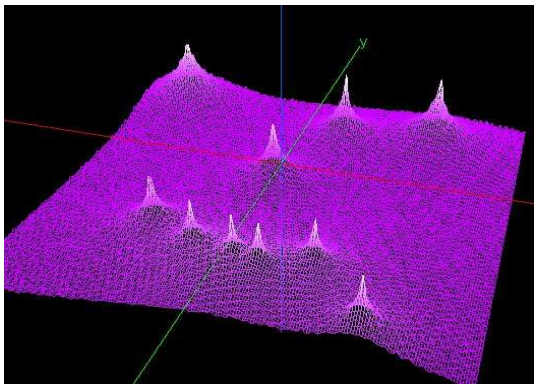

Figure 5: 3-D map for 2-D function of F_5.

3.5 F_5:Composition of Ackley's function

Basic functions: $f_1 - f_{10}$ =Ackley's function

Properties

♠ Multi-modal

♠ Scalable

♠ Rotated

♠ A huge number of local optima

♠ $x \in [-5, 5]^n$, Global optimum $x^*(t) = \vec{O}_i, F(x^*(t)) = H_i(t), H_i(t) = min_j^m H_j$

3.6 F_6:Hybrid Composition function

Basic functions: $f_1 - f_2$ =Sphere's function
$f_3 - f_4$ =Ackley's function $f_5 - f_6$ =Griewank's function
$f_7 - f_8$ =Rastrigin's function $f_9 - f_{10}$ =Weierstrass's function

Properties

♠ Multi-modal

♠ Scalable

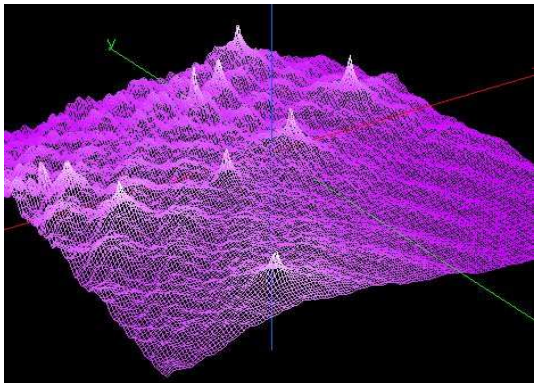

Figure 6: 3-D map for 2-D function of F_6.

♠ Rotated

♠ A huge number of local optima

♠ Different functions properties are mixed together

♠ Sphere Functions give two flat areas for the function

♠ $x \in [-5, 5]^n$, Global optimum $x^*(t) = \vec{O_i}, F(x^*(t)) = H_i(t), H_i(t) = min_j^m H_j$

4 Evaluation Criteria

4.1 Description of the Evaluation Criteria

Problems: Function $F_1 - F_6$

Dimension: $n = 10, [5 - 15]$

Runs/problem/change type: 20 **(Do not run many 20 runs to pick the best run)**

Max_FES/change: $10,000 * n$

Sampling frequency: $s\_f = 100$

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.

Non-dimensional Change Detection: Algorithm should detect the *non-dimensional* change by itself instead of informing the algorithm when a *non-dimensional* change occurs.

Dimensional Change Detection: Algorithm should be informed when a *dimensional* change occurs.

Termination: Terminate when reaching *num_change*.

1)**Record absolute function error value** $E^{last}(t) = |f(x_{best}(t)) - f(x^*(t))|$ **after reaching**

Max_FES/change for each change.
For each change type of each function, present the following values for $x_{best}(t)$ over 20 runs:

Average best, average mean, average worst values and STD.

Average best (Avg_best)=$\sum_{i=1}^{runs} Min_{j=1}^{num\_change} E_{i,j}^{last}(t)/runs$
Average mean (Avg_mean)=$\sum_{i=1}^{runs} \sum_{j=1}^{num\_change} E_{i,j}^{last}(t)/(runs * num\_change)$
Average worst (Avg_worst)=$\sum_{i=1}^{runs} Max_{j=1}^{num\_change} E_{i,j}^{last}(t)/runs$
STD=$\sqrt{\frac{1}{runs*num\_change-1} \sum_{i=1}^{runs} \sum_{j=1}^{num\_change} (E_{i,j}^{last}(t) - Avg\_mean)^2}$

2) **Convergence Graphs (or Run-length distribution graphs)**
Convergence Graphs for each problem for dimension $n = 10$. The graph would show the median performance of the relative value $r(t)$ of $f(x_{best}(t))$ and $f(x^*(t))$ for total runs with termination by the Total_FES.
NOTE: For maximization function F_1, $r(t) = f(x_{best}(t))/f(x^*(t))$, for minimization function $F_2 - F_6$,$r(t) = f(x^*(t))/f(x_{best}(t))$

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.

5) **Overall performance marking measurement**

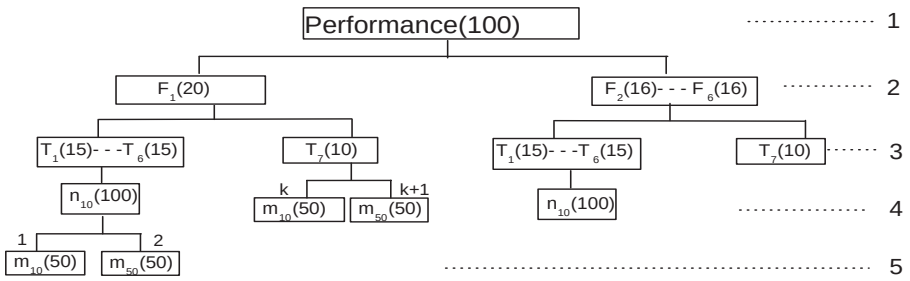

Figure 7: Overall performance marking measurement

$F_1 - F_6$: Function $F_1 - F_6$

$T_1 - T_6$: Change type of small step change, large step change, random change, chaotic change, recurrent change, and recurrent change with noise.

T_7: Random change with changed dimension

n_{10} : The number of dimension of 10.

m_{10} and m_{50}: The number of peaks of 10 and 50.

NOTE: The sum mark of all the nodes within a parent is 100. The mark of leaf node k is calculated by:

$$mark_k = percentage_k * \Sigma_{i=1}^{runs} \Sigma_{j=1}^{num\_change} r_{ij}/(num\_change * runs) \quad (8)$$

where k is the node number in the bottom level, $k = 1, 2, ..., 49$. $r_{ij} = r_{ij}^{last}/(1+\sum_{s=1}^{S}(1 - r_{ij}^{s})/S)$, r_{ij}^{last} is the relative value of the best one to the global optimum after reaching $Max\_FES/change$ for each change. r_{ij}^{s} is the relative value of the best one to the global optimum at the $s-th$ sampling during one change, $S = Max\_FES/change/s\_f$.

The weight of leaf node k is the percentage product of all the nodes from the root to leaf node k,which is obtained by:

$$weight_k = \Pi percentage_{lk} \quad (9)$$

where lk is the parent node of leaf node k in the level l, $l = 1, 2, ..., 5$. All the weights of the 49 leaf nodes are listed in Table (8). The overall algorithm performance is evaluated by:

$$performance = \Sigma_{k=1}^{number\_of\_leaf\_nodes} mark_k * weight_k \quad (10)$$

NOTE: There are totally 49 specific test cases. For each specific case, algorithm independently run 20 times to obtain the mark on the specific case. The overall performance is the sum of all the mark obtained on each test case. Authors should calculate the performance of the algorithms proposed and provide the results of average best, average mean, average worst values and STD of the 49 test cases.

4.2 Example

System: Windows XP (SP1)
CPU: Pentium(R) 4 3.00GHz
RAM: 1 G
Language: C++
Algorithm: Particle Swarm Optimizer (PSO)

References

[1] J. Branke, Memory enhanced evolutionary algorithms for changing optimization problems, *Proc. of the 1999 Congr. on Evol. Comput*, pp. 1875-1882, 1999.

[2] Y. Jin and B. Sendhoff. Constructing dynamic optimization test problems using the multi-objective optimization concept. *EvoWorkshop 2004*, LNCS 3005, pp. 526-536, 2004.

[3] C. Li, M. Yang, and L. Kang. A new approach to solving dynamic TSP, *Proc of the 6th Int. Conf. on Simulated Evolution and Learning*, pp. 236-243, 2006.

[4] C. Li and S. Yang. A Generalized Approach to Construct Benchmark Problems for Dynamic Optimization, *Proc. of the 7th Int. Conf. on Simulated Evolution and Learning*, 2008.

[5] J. J. Liang, P. N. Suganthan and K. Deb. Novel composition test functions for numerical global optimization. *Proc. of IEEE Int. Swarm Intelligence Symp.*, pp. 68-75, 2005.

Table 2: Error Values Achieved for Problems F_1

Dimension(n)	Peaks(m)	Errors	T_1	T_2	T_3	T_4	T_5	T_6
10	10	Avg_best						
		Avg_worst						
		Avg_mean						
		STD						
	50	Avg_best						
		Avg_worst						
		Avg_mean						
		STD						
T_7(5-15)	10	Avg_best	—	—		—	—	—
		Avg_worst	—	—		—	—	—
		Avg_mean	—	—		—	—	—
		STD	—	—		—	—	—
	50	Avg_best	—	—		—	—	—
		Avg_worst	—	—		—	—	—
		Avg_mean	—	—		—	—	—
		STD	—	—		—	—	—

Table 3: Error Values Achieved for Problems F_2

Dimension(n)	Errors	T_1	T_2	T_3	T_4	T_5	T_6
10	Avg_best						
	Avg_worst						
	Avg_mean						
	STD						
T_7(5-15)	Avg_best	—	—		—	—	—
	Avg_worst	—	—		—	—	—
	Avg_mean	—	—		—	—	—
	STD	—	—		—	—	—

Table 4: Error Values Achieved for Problems F_3

Dimension(n)	Errors	T_1	T_2	T_3	T_4	T_5	T_6
10	Avg_best						
	Avg_worst						
	Avg_mean						
	STD						
T_7(5-15)	Avg_best	—	—		—	—	—
	Avg_worst	—	—		—	—	—
	Avg_mean	—	—		—	—	—
	STD	—	—		—	—	—

Table 5: Error Values Achieved for Problems F_4

Dimension(n)	Errors	T_1	T_2	T_3	T_4	T_5	T_6
10	Avg_best						
	Avg_worst						
	Avg_mean						
	STD						
T_7(5-15)	Avg_best	—	—		—	—	—
	Avg_worst	—	—		—	—	—
	Avg_mean	—	—		—	—	—
	STD	—	—		—	—	—

Table 6: Error Values Achieved for Problems F_5

Dimension(n)	Errors	T_1	T_2	T_3	T_4	T_5	T_6
10	Avg_best						
	Avg_worst						
	Avg_mean						
	STD						
T_7(5-15)	Avg_best	—	—		—	—	—
	Avg_worst	—	—		—	—	—
	Avg_mean	—	—		—	—	—
	STD	—	—		—	—	—

Table 7: Error Values Achieved for Problems F_6

Dimension(n)	Errors	T_1	T_2	T_3	T_4	T_5	T_6
10	Avg_best						
	Avg_worst						
	Avg_mean						
	STD						
T_7(5-15)	Avg_best	—	—		—	—	—
	Avg_worst	—	—		—	—	—
	Avg_mean	—	—		—	—	—
	STD	—	—		—	—	—

Table 8: Algorithm Overall Performance

	F1(10)	F1(50)	F2	F3	F4	F5	F6
T_1	( )*0.015	( )*0.015	( )*0.024	( )*0.024	( )*0.024	( )*0.024	( )*0.024
T_2	( )*0.015	( )*0.015	( )*0.024	( )*0.024	( )*0.024	( )*0.024	( )*0.024
T_3	( )*0.015	( )*0.015	( )*0.024	( )*0.024	( )*0.024	( )*0.024	( )*0.024
T_4	( )*0.015	( )*0.015	( )*0.024	( )*0.024	( )*0.024	( )*0.024	( )*0.024
T_5	( )*0.015	( )*0.015	( )*0.024	( )*0.024	( )*0.024	( )*0.024	( )*0.024
T_6	( )*0.015	( )*0.015	( )*0.024	( )*0.024	( )*0.024	( )*0.024	( )*0.024
T_7	( )*0.01	( )*0.01	( )*0.016	( )*0.016	( )*0.016	( )*0.016	( )*0.016
Mark							
Performance(sum the mark obtained for each case and multiply by 100):							

[6] R. W. Morrison and K. A. De Jong. A test problem generator for non-stationary environments, *Proc. of the 1999 Congr. on Evol. Comput.*, pp. 2047-2053, 1999.

[7] R. Salomon, Reevaluating genetic algorithm performance under coordinate rotation of benchmark functions; A survey of some theoretical and practical aspects of genetic algorithms, *BioSystems*, vol. 39, no. 3, pp. 263-278, 1996.

[8] B. Sendhoff, M. Roberts and X. Yao. Evolutionary computation benchmarking repository, *IEEE Computational Intelligence Magazine*, 1(4): 50-51, November 2006.

[9] 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, 2005.

[10] S. Yang. Non-stationary problem optimization using the primal-dual genetic algorithm, *Proc. of the 2003 IEEE Congr. on Evol. Comput.*, pp. 2246-2253, 2003.

[11] S. Yang and X. Yao. Experimental study on population-based incremental learning algorithms for dynamic optimization problems. *Soft Comput.*, 9(11): 815-834, 2005.

[12] S. Yang and X. Yao. Population-based incremental learning with associative memory for dynamic environments. *IEEE Trans. on Evol. Comput.*, 2008.