Benchmark Generator for the IEEE WCCI-2012 Competition on Evolutionary Computation for Dynamic Optimization Problems

Changhe Li[1], Shengxiang Yang[2], David Alejandro Pelta[3]

October 28, 2011

[1]School of Computer Science, China University of Geosciences, Wuhan 430074, China
[2]Department of Information Systems and Computing, Brunel University, UK
[3]Department of Computer Science and A.I., University of Granada, Spain

changhe.lw@gmail.com, shengxiang.yang@brunel.ac.uk, dpelta@decsai.ugr.es

Many realworld optimization problems are dynamic optimization problems (DOPs), where changes may occur over time regarding the objective function, decision variable, and constraints, etc. DOPs raise big challenges to traditional optimization methods as well as evolutionary algorithms (EAs). The last decade has witnessed increasing research efforts on handling dynamic optimization problems using EAs and other metaheuristics, and a variety of methods have been reported across a broad range of application backgrounds. 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 [7], 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 [11, 12, 13], Kang's dynamic traveling salesman problem (DTSP) [3] and dynamic multi knapsack problem (DKP), etc.

Although 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.

Based on our previous benchmark generator for the IEEE CEC'09 Competition on Dynamic Optimization [5], this report updates the two benchmark instances where a new change type has been developed as well as a constraint to the benchmark instance of the dynamic rotation peak benchmark generator. The source code in C++ language for the two benchmark instances is included in the library of EAlib, which is an open platform to test and compare the performances of EAs. The EAlib package is available from the competition website at `http://people.brunel.ac.uk/~csstssy/ECDOP-Competition12.html` and `http://cs.cug.edu.cn/teacherweb/lichanghe/pages/EAlib.html`.

The definition of the two benchmark instances are described in Section 2. Section 3 gives the description of six 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. In addition, a new change type: the number of peaks change, has been developed in this report. The framework of the eight change types are described as follows:

Framework of DynamicChanges

switch(change type)

case small step:

$$T1 : \Delta\phi = \alpha \cdot \|\phi\| \cdot r \cdot \phi_{severity} \tag{4-1}$$

case large step:

$$T2 : \Delta\phi = \|\phi\| \cdot (\alpha \cdot sign(r) + (\alpha_{max} - \alpha) \cdot r) \cdot \phi_{severity} \tag{4-2}$$

case random:

$$T3 : \Delta\phi = N(0,1) \cdot \phi_{severity} \tag{4-3}$$

case chaotic:

$$T4 : \phi(t+1) = A \cdot (\phi(t) - \phi_{min}) \cdot (1 - (\phi(t) - \phi_{min})/\|\phi\|) \tag{4-4}$$

case recurrent:

$$T5 : \phi(t+1) = \phi_{min} + \|\phi\|(\sin(\frac{2\pi}{P}t + \varphi) + 1)/2 \tag{4-5}$$

case recurrent with noisy:

$$T6 : \phi(t+1) = \phi_{min} + \|\phi\|(\sin(\frac{2\pi}{P}t + \varphi) + 1)/2 + N(0,1) \cdot noisy_{severity} \tag{4-6}$$

case number of dimensions change:

$$T7 : D(t+1) = D(t) + sign \cdot \Delta D \tag{4-7}$$

case number of peaks change:

$$T8 : P(t+1) = P(t) + sign \cdot \Delta P \tag{4-8}$$

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. ΔD is a predefined constant, whose the default value 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 does not 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. In the number of peaks change, similar to the dimensional change, ΔP is also a predefined constant of value of 2. If $P(t) = Max\_P, sign = -1$; if $P(t) = Min\_P, sign = 1$. Max_P and Min_P are the maximum and minimum number of peaks. In the number of dimensions change and the number of peaks change, the random change is used.

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[7]. 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:

$$f(x, \phi, t) = \max_{i=1}^{m} (\vec{H}_i(t)/(1 + \vec{W}_i(t) \cdot \sqrt{\sum_{j=1}^{n} \frac{(x_j - \vec{X}_j^i(t))^2}{n}})) \tag{5}$$

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[8] $R_{ij}(\theta)$ 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. [6, 10]. The composition function can be described as:

$$F(x, \phi, t) = \sum_{i=1}^{m} (w_i \cdot (f_i'((x - \vec{O}_i(t) + O_{iold})/\lambda_i \cdot \vec{M}_i) + \vec{H}_i(t))) \tag{6}$$

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)= 5; MIN_D=2, MAX_D=15

The number of peaks: m=10, MIN_P=10, MAX_P=50

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

Total fitness evaluations: 3,000,000

Change frequency: $frequency = 50,000$

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

In order to test an algorithm's performance in hard-to-detect or undetectable dynamic environments, a constraint of changing ratio ($change\_ratio$) was added in F_1 for all change types except the dimensional change (T_7) and the number of peaks change (T_8), where only $m * change\_ratio$ peaks are allowed to change instead of all peaks in the fitness landscape when $change\_ratio$ is less than 1.

The number of peaks: $m = 10$

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

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

Initial width: $initial\_width = 5$

Changing ratio: $change\_ratio$=0.3,0.7,1.0

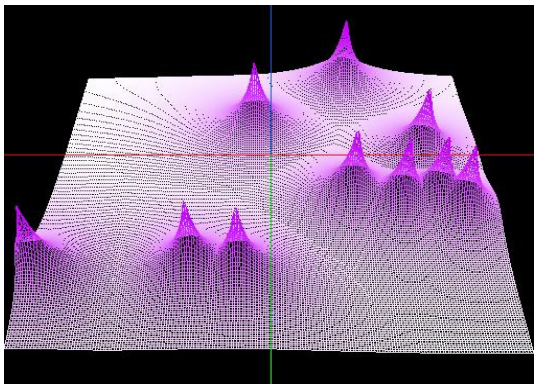

Figure 1: 3-D map for 2-D function of F_1.

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

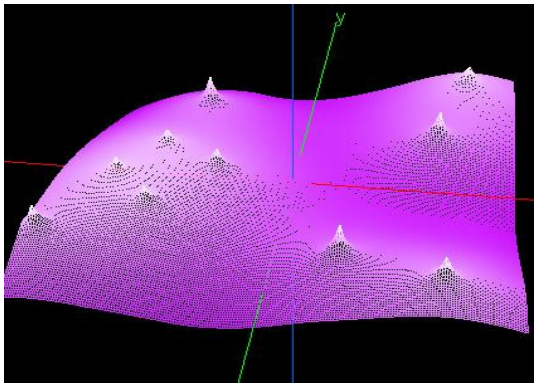

Figure 2: 3-D map for 2-D function of F_2.

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

Properties

♠ Multi-modal

♠ Scalable

♠ Rotated

♠ A huge number of local optima

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

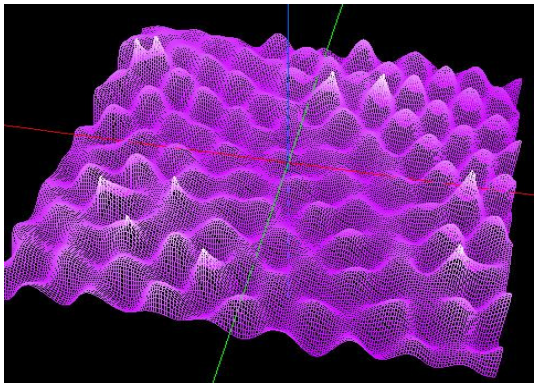

Figure 3: 3-D map for 2-D function of F_3.

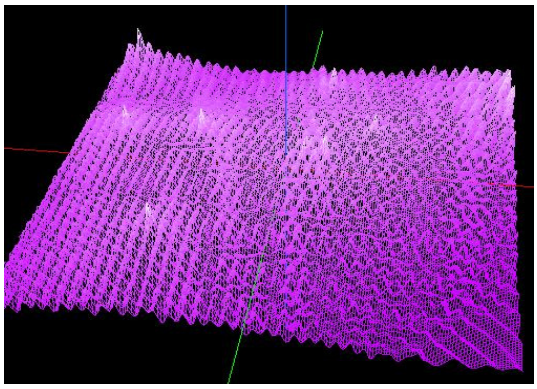

Figure 4: 3-D map for 2-D function of F_4.

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 $x^*(t) = \vec{O_i}, F(x^*(t)) = H_i(t), H_i(t) = min_j^m H_j$

3.5 F_5:Composition of Ackley's function

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

Properties

♠ Multi-modal

♠ Scalable

♠ Rotated

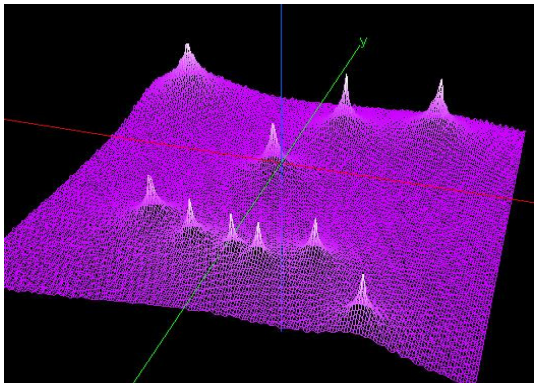

Figure 5: 3-D map for 2-D function of F_5.

♠ 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

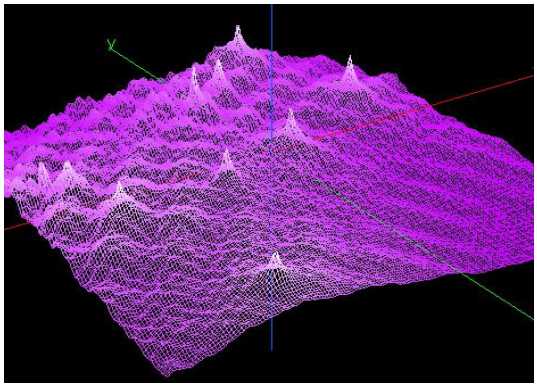

Figure 6: 3-D map for 2-D function of F_6.

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

NOTE: For the number of peaks change in F_6, the Sphere function is added if the number of functions increases to more than 10 functions.

Properties

♠ Multi-modal

♠ Scalable

♠ Rotated

♠ A huge number of local optima

♠ Different functions properties are mixed together

♠ $x \in [-5,5]^n$, 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: $F_1 - F_6$
Change types: $T_1 - T_8$
Dimensions: $n = 5, [2 - 15]$
Peaks or functions: $m = 10, [10 - 50]$
Runs/problem/change type: 20 **(Do not run many 20 runs to pick the best run)**
Max_FES/change: $50,000$
Sampling frequency: $s\_f = 100$
Initialization: Uniformly distributed 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: An algorithm should detect the *non-dimensional* change by itself instead of informing the algorithm when a *non-dimensional* change occurs.
Dimensional Change Detection: An algorithms should be informed when a *dimensional* change occurs.
NOTE: An environmental change should be automatically triggered once the environmental changing criterion is satisfied rather than being called in the framework of an algorithm.
Termination: Terminate when reaching the total fitness evaluations (Max_FES).
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:

Mean and STD.

mean=$\sum_{i=1}^{runs} \sum_{j=1}^{num\_change} E_{i,j}^{last}(t)/(runs * num\_change)$
STD=$\sqrt{\frac{1}{runs*num\_change} \sum_{i=1}^{runs} \sum_{j=1}^{num\_change} (E_{i,j}^{last}(t) - mean)^2}$
2) **Parameters**

We discourage participants searching for a distinct set of parameters for each problem,dimension, or change type. 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

3) **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.

4) **Overall performance marking measurement**

In order to evaluate an algorithm's performance in terms of both convergence speed and solution quality, the performance of an algorithm on test case k is calculated by:

$$performance_k = \Sigma_{i=1}^{runs} \Sigma_{j=1}^{num\_change} r_{ij}/(num\_change * runs) \tag{7}$$

where $r_{ij} = r_{ij}^{last}/(1+\sum_{s=1}^{S}(1 - r_{ij}^s)/S)$, 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, $S = Max\_FES/change/s\_f$. $r_{ij}^s = (f(x_{ij}) + gap)/(f(x_{ij}^*) + gap)$ for the maximization problem F_1 and $r_{ij}^s = (f(x_{ij}^*) +$

Table 2: Mean values and STD for the problem F_1 with changing ratios $change\_ratio$ =0.3,0.7, and 1.0, respectively

Changing ratio	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8
	mean±STD	mean±STD	mean±STD	mean±STD	mean±STD	mean±STD	mean±STD	mean±STD
0.3							—	—
0.7							—	—
1.0								

Table 3: Mean values and STD for the problems F_2-F_6

Problem	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8
	mean±STD	mean±STD	mean±STD	mean±STD	mean±STD	mean±STD	mean±STD	mean±STD
F_2								
F_3								
F_4								
F_5								
F_6								

$gap)/(f(x_{ij}) + gap)$ for the minimization problems $F_2 - F_6$, here gap is used to ensure that $(f(x^*_{ij}) + gap)$ is greater than 0. It was set to $fabs(f(x^*_{ij})) + 1$ in EAlib.

There are totally 60 test cases which are $3*(F_1T_1-F_1T6), F_1T_7, F_1T_8, F_2T_1-F_2T_8, \cdots, F_6T_1-F_6T_8$, respectively. And the overall performance of an algorithm is evaluated by:

$$performance = \Sigma_{k=1}^{60} performance_k \tag{8}$$

NOTE: For a fair competition, participants are encouraged to use the same test platform: EAlib. This is because EAlib is able to generate the exactly same change serials for each particular run for all involved algorithms as well as the same initial population if the same population size is used. What's more, EAlib is also able to calculate an algorithm's performance without participants intervention. EAlib also provides some basic functions which can be used directly to implement algorithms. And it also integrates some algorithms for solving the benchmark problems.

All participants must provide the results for Table 2, Table 3, and Table 4. The winner algorithm will be the highest score obtained in the bottom right cell in Table 4.

4.2 The six problems with the eight change types in EAlib

The six test problems ($F_1 - F_6$) can be generated from the RotationDBG class and CompositionDBG classe, respectively, in EAlib. The following two functions are for generating the six problems with the eight change types ($T_1 - T_8$):

For problem F_1:

RotationDBG::initialize(rDim, rPeaks, rChangeType, rChangingRatio, rFlagDimChange, rFlagNumPeaksChange)

Parameters:

$rDim$: the number of dimensions

$rPeaks$: the number of peaks

$rChangeType$: change type, from 0,$\cdots$,7

$rChangingRatio$: ratio of changing peaks

$rFlagDimChange$: flag of dimensional change

$rFlagNumPeaksChange$: flag of number of peaks change

Table 4: Overall performance

Problem	Changing ratio	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	Sum
F_1	0.3							—	—	
	0.7							—	—	
	1.0									
F_2	1.0									
F_3	1.0									
F_4	1.0									
F_5	1.0									
F_6	1.0									
The overall performance										

Note: For *rChangeType* =0,$\cdots$,5 (T_1,$\cdots$,T_6), both *rFlagDimChange* and *rFlagNumPeaksChange* are false; For *rChangeType*= 6(T_7), *rFlagDimChange* is true and *rFlagNumPeaksChange* is false; For *rChangeType* =7(T_8), *rFlagDimChange* is false and *rFlagNumPeaksChange* is true;

Take the first cell in Table 2 (F_1 with T_1 and *change_ratio*=0.3) as an example. The parameters in the function are: *RotationDBG::initialize(5, 10, 0, 0.3, false,false)*

For problems F_2 -F_6:
CompositionDBG::initialize(rDim, rPeaks, rChangeType, rFunction, rChangingRatio, rFlagDimChange,rFlagNumPeaksChange)
Parameters:
rDim: the number of dimensions
rPeaks: the number of basic functions
rChangeType: change type, from 0,1,$\cdots$,7
rFunction: function ID= 0,$\cdots$,4 ($T_2, \cdots, T_6$)
rChangingRatio: ratio of changing peaks,default value is 1.0
rFlagDimChange: flag of dimensional change
rFlagNumPeaksChange: flag of number of peaks change
Note: For *rChangeType*=0,$\cdots$,5 (T_1,$\cdots$,T_6), both *rFlagDimChange* and *rFlagNumPeaksChange* are false; For *rChangeType*=6(T_7), *rFlagDimChange* is true and *rFlagNumPeaksChange* is false; For *rChangeType*=7(T_8), *rFlagDimChange* is false and *rFlagNumPeaksChange* is true;

Take the first cell in Table 3 (F_2 with T_1) as an example. The parameters in the function are: *CompositionDBG::initialize(5, 10, 0, 0, 1.0, false,false)*

For details of how to run an algorithm on the six problems with the eight change types, please see the examples and the instructions in EAlib.

4.3 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*, LNCS 5361, pp. 391-400, 2008.

[5] C. Li, S. Yang, T. T. Nguyen, E. L. Yu, X. Yao, Y. Jin, H.-G. Beyer, and P. N. Suganthan, "Benchmark Generator for CEC2009 Competition on Dynamic Optimization," Technical Report 2008, Department of Computer Science, University of Leicester, U.K., 2008.

[6] 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.

[7] 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.

[8] R. Salomon. Reevaluating genetic algorithm performance under coordinate rotation of benchmark functions: A survey of some theoretical and practical aspects of genetic algorithms, *BioSystems*, 39(3): 263-278, 1996.

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

[10] 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.

[11] 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.

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

[13] S. Yang and X. Yao. Population-based incremental learning with associative memory for dynamic environments. *IEEE Trans. on Evol. Comput.*, 12(5): 542-561, October 2008.