Continuous Dynamic Problem Generators for Evolutionary Algorithms

Renato Tinós and Shengxiang Yang

***Abstract*— Addressing dynamic optimization problems has attracted a growing interest from the evolutionary algorithm community in recent years due to its importance in the applications of evolutionary algorithms in real world problems. In order to study evolutionary algorithms in dynamic environments, one important work is to develop benchmark dynamic environments. This paper proposes two continuous dynamic problem generators. Both generators use linear transformation to move individuals, which preserves the distance among individuals. In the first generator, the linear transformation of individuals is equivalent to change the direction of some axes of the search space while in the second one it is obtained by successive rotations in different planes. Preliminary experiments were carried out to study the performance of some standard genetic algorithms in continuous dynamic environments created by the proposed generators.**

I. Introduction

Evolutionary algorithms (EAs) have been widely applied for solving stationary optimization problems. However, the environments of real world optimization problems are often dynamic, where the fitness function, design variables, and/or environmental conditions may change over time. In recent years, there has been a growing interest in studying EAs for dynamic optimization problems (DOPs) due to its importance in real world applications [1].

In order to study EAs in dynamic environments, one important work is to develop benchmark dynamic environments. Over the years, researchers have developed a number of DOP generators to create dynamic test environments to compare the performance of EAs in dynamic environments. They can be roughly divided into two types. For the first type, the environment is just switched between several stationary problems or several states of a problem. For example, many researchers have tested their EAs on a time varying knapsack problem where the total capacity of the knapsack changes over time, usually oscillating between two or more fixed values [2], [3]. Cobb and Grefenstette [4] constructed a significantly changing environment that oscillates between two different fitness landscapes. For this type of generator, the environmental dynamics is mainly characterized by the speed of changes, usually measured in EA generations.

The second type of DOP generators starts from a predefined fitness landscape, which will be changed to construct dynamic environments. For example, the stationary landscape can be defined in n-dimensional real space with a number of component peaks, which can change independently [5], [6], [7]. Each peak has its own morphology with such parameters as height, slope and location. The center of the peak with the highest height is taken as the optimum solution of the landscape. From this stationary landscape, dynamic problems can be created through changing the parameters of each peak. In [8], [9] a DOP generator that can generate DOPs from any binary encoded stationary problem was proposed based on an exclusive-or (XOR) operator, see Section II for more details. For this type of generators, the environmental dynamics is characterized by the speed of changes and severity of changes.

This paper proposes two continuous dynamic problem generators. The first generator is an extension of the XOR generator proposed in [8], [9] for binary encoded problems. In this generator, each individual of the current population is moved to a new location in the fitness landscape before being evaluated. In the second generator, the individuals are moved to new locations only when a change occurs. Both generators use linear transformation to move individuals, which preserves the distance among individuals. In the first generator, the linear transformation of individuals is equivalent to change the direction of some axes of the search space while in the second one it is obtained by successive rotations in different planes. Based on the proposed generators, some preliminary experiments were carried out to study some genetic algorithms in continuous dynamic environments.

The rest of this paper is outlined as follows. The next section briefly describes the XOR DOP generator proposed in [8], [9] and shows some analysis regarding its properties. Section III presents the proposed continuous DOP generator for EAs in dynamic environments, which is a simple extension of the XOR DOP generator. Section IV describes another continuous DOP generator proposed in this paper. The experimental study and relevant analysis are presented in Section V. Section VI concludes this paper with discussions on relevant future work.

II. The XOR DOP Generator and its Analysis

The XOR DOP generator proposed in [8], [9] can generate DOPs from any binary encoded stationary problem. Given a stationary problem with fitness function $f(\mathbf{x}(t))$ and $\mathbf{x}(t) \in \{0,1\}^l$, the fitness function $g(\mathbf{x}(t))$ of an environment that is periodically changed every τ generations is formulated as follows:

$$g(\mathbf{x}(t)) = f\big(\mathbf{x}(t) \oplus \mathbf{m}(k)\big), \tag{1}$$

Renato Tinós is with the Department of Physics and Mathematics, FFCLRP, University of São Paulo (USP), 14040-901, Ribeirão Preto, SP, Brazil (email: rtinos@ffclrp.usp.br).

Shengxiang Yang is with the Department of Computer Science, University of Leicester, University Road, Leicester, LE1 7RH, United Kingdon, (email: s.yang@mcs.le.ac.uk).

1-4244-1340-0/07/$25.00

where $\oplus$ is the bitwise exclusive-or (XOR) operator, t is the generation index, $k = \lceil t/\tau \rceil$ is the period index, and $\mathbf{m}(k)$ is a binary mask for period k that is incrementally generated by:

$$\mathbf{m}(k) = \mathbf{m}(k-1) \oplus \mathbf{p}(k), \tag{2}$$

where $\mathbf{p}(k)$ is a binary template randomly created for period k containing $\lfloor \rho \times l \rfloor$ ones, and $\{\rho \in \mathbb{R} \mid 0.0 \leq \rho \leq 1.0\}$ controls the degree of change for the DOP. If $\rho = 0.0$, the problem stays stationary, while if $\rho = 1.0$, the extreme fitness landscape change in the sense of Hamming distance occurs. For the first period, $\mathbf{m}(1)$ is equal to the zero vector.

The main characteristic of the XOR DOP generator is that each individual of the current population is moved to a new location in the fitness landscape before being evaluated. Each individual $\mathbf{x}(t)$ is moved according to the rule:

$$\mathbf{z}(t) = \mathbf{x}(t) \oplus \mathbf{m}(k) \tag{3}$$

If the vector $\mathbf{x}(t) \in [0,1]^l$ is normalized to $\mathbf{x^n}(t) \in [-1,1]^l$, the transformation represented by Eq. 3 can be written as

$$\mathbf{z^n}(t) = \mathbf{A}(k)\mathbf{x^n}(t), \tag{4}$$

where $\mathbf{z^n}(t) \in [-1,1]^l$ is the normalized vector $\mathbf{z}(t) \in [0,1]^l$, and the linear transformation $\mathbf{A}(k)$ is given by

$$\mathbf{A}(k) = \begin{bmatrix} A_1(k) & 0 & \dots & 0 \\ 0 & A_2(k) & \dots & 0 \\ & & \ddots & \\ 0 & 0 & \dots & A_l(k) \end{bmatrix}, \tag{5}$$

where

$$A_i(k) = \begin{cases} 1 & \text{if } m_i(k) = 0 \\ -1 & \text{if } m_i(k) = 1 \end{cases} \tag{6}$$

for $i = 1, \dots, l$, and $m_i(k)$ is the i-th element of the vector $\mathbf{m}(k)$.

It can be observed that the matrix $\mathbf{A}(k)$ is orthogonal, i.e., $\mathbf{A}(k)^{\mathrm{T}}\mathbf{A}(k) = \mathbf{A}(k)\mathbf{A}(k)^{\mathrm{T}} = \mathbf{I}$, where $\mathbf{A}(k)^{\mathrm{T}}$ denotes the transpose of $\mathbf{A}(k)$ and $\mathbf{I}$ is the identity matrix. An orthogonal matrix has some important properties [10], e.g.

i) if $\mathbf{A}$ and $\mathbf{B}$ are orthogonal, then $\mathbf{AB}$ is orthogonal too.
ii) if λ is an eigenvalue of the orthogonal matrix $\mathbf{A}$, then $|\lambda| = 1$.
iii) if $\mathbf{A}$ is an orthogonal matrix, then $\|\mathbf{Ax}\|_2 = \|\mathbf{x}\|_2$ for every vector $\mathbf{x}$, where $\|\mathbf{x}\|_2$ denotes the Euclidean norm of the vector $\mathbf{x}$.

A geometrical interpretation of item (iii) is that the linear transformation $\mathbf{A}(k)$ preserves the angles and magnitudes, i.e., the linear transformation behaves like a rotation in the space.

The following properties of the XOR generator can be observed by analyzing Eqs. 3 and 4 and the properties of an orthogonal matrix:

- The individuals of the population are rotated in order that the i-th gene of each individual is changed when $m_i(k) = 1$. This rotation is equivalent to change the direction of the i-th axis of the fitness space.

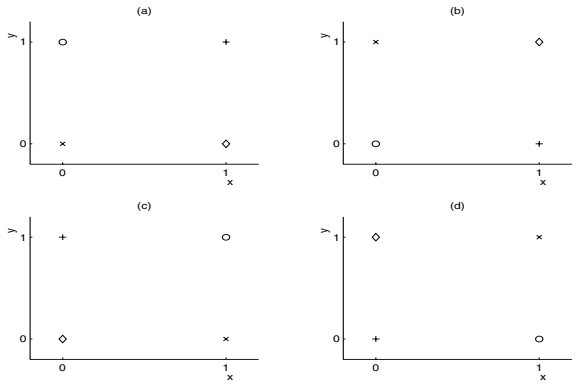

Fig. 1. Individuals (binary encoded) in a planar space with transformations defined by: (a) $\mathbf{m}(k) = [\ 0\ \ 0\ ]^{\mathrm{T}}$ (original individuals); (b) $\mathbf{m}(k) = [\ 0\ \ 1\ ]^{\mathrm{T}}$; (c) $\mathbf{m}(k) = [\ 1\ \ 0\ ]^{\mathrm{T}}$; (d) $\mathbf{m}(k) = [\ 1\ \ 1\ ]^{\mathrm{T}}$.

- The magnitudes of the individuals of the population are preserved after the environmental change.
- The distances among the individuals in the current population remains unaltered after the transformation imposed by Eq. 3. In this way, the environmental change imposed by the XOR Generator preserves the properties of the population before the change.
- The properties of the fitness landscape are not changed after the environmental change.

The last item implies that the properties of the problem are preserved after the environmental changes, which is very interesting when we want to investigate the performance of different algorithms in several kinds of DOPs. However, the use of only the XOR generator is not interesting when we want to investigate the performance of algorithms in DOPs where the fitness landscape changes. It is important to observe, however, that the XOR generator can be combined to other problems generators in order to allow changes in the fitness landscape.

In Fig. 1, the three possible transformations caused by the XOR Generator for all possible individuals in a planar space are illustrated. It can be observed that the individuals are rotated in the space after the environmental changes.

III. Extending the XOR Generator to Continuous Problems

By analyzing the geometrical interpretation of the XOR Generator discussed in last section, it is very easy to extend it to real-valued optimization problems. Given a stationary problem where the fitness function is $f(\mathbf{x}(t))$ and each element of vector $\mathbf{x}$ is $x_i \in \{x_i^{min}, x_i^{max}\}$ for $i = 1, \dots, l$, the fitness function $g(\mathbf{x}(t))$ of an environment that is periodically changed every τ generations is formulated as follows:

$$g(\mathbf{x}(t)) = f\big(\mathbf{z}(t)\big), \tag{7}$$

where the vector $\mathbf{z}(t)$, with elements $z_i \in \{x_i^{min}, x_i^{max}\}$ for $i = 1, \dots, l$, is obtained by unnormalizing the vector

$$\mathbf{z^n}(t) = \mathbf{A}(k)\mathbf{x^n}(t), \tag{8}$$

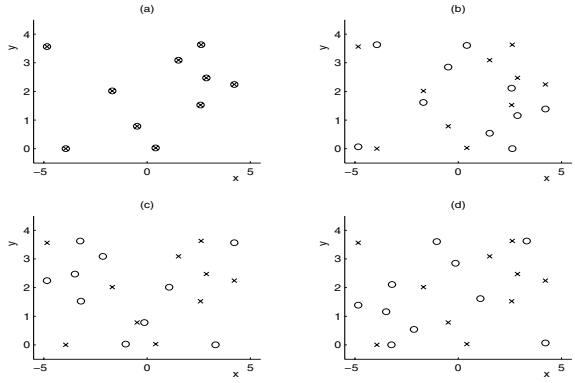

Fig. 2. Individuals (real encoded) in a planar space with transformations defined by: (a) $\mathbf{m}(k) = [\ 0\ \ 0\ ]^{\mathrm{T}}$; (b) $\mathbf{m}(k) = [\ 0\ \ 1\ ]^{\mathrm{T}}$; (c) $\mathbf{m}(k) = [\ 1\ \ 0\ ]^{\mathrm{T}}$; (d) $\mathbf{m}(k) = [\ 1\ \ 1\ ]^{\mathrm{T}}$. The individuals before and after the transformation are respectively represented by 'x' and 'o'.

where $\mathbf{z}^{\mathbf{n}}(t) \in [-1, 1]^l$, $\mathbf{x}^{\mathbf{n}}(t) \in [-1, 1]^l$ is the normalized vector $\mathbf{x}(t)$, and the linear transformation $\mathbf{A}(k)$ is given by Eq. 5 and 6. The binary mask $\mathbf{m}(k)$ is defined as described in Section II.

The XOR dynamic problem generator for real-valued optimization problems has the same properties of the binary version. Like the previous version, the individuals of the population are rotated in order that the i-th gene of each individual is changed when $m_i(k) = 1$. This rotation is equivalent to change the direction of the i-th axis of the fitness space.

However, the generator for real-valued optimization problems has a drawback. When $m_i(k) = 1$, the changes in the vectors close to the center of the i-th axis are smaller than the changes in the vectors far from the center of this axis. In this way, if the individuals of the population are located close to the center of the i-th axis, the changes produced by the XOR generator will be smaller than when they are far from the center of this axis. This property can be observed in Fig. 2, where the three possible transformations caused by the XOR Generator for 10 random individuals in a planar space are illustrated. It can be seen that the individuals are rotated in the space after the environment changes.

IV. A Continuous Dynamic Problem Generator with Control of Rotation of Individuals

It can be observed that other transformation matrices $\mathbf{A}(k)$ can be defined for the dynamic problem generator described in the last section. In order to maintain the property of the invariance of the distances among the individuals after the environmental changes, the matrix $\mathbf{A}(k)$ must be orthogonal. A natural choice for a new matrix is a rotation matrix composed by multiplying simple planar rotation matrices. It is important to observe that the idea of using rotation matrices to generate dynamic problems is not new. In [11], planar rotation matrices were employed to generate rotating environments for EAs.

A simple planar rotation defined by the matrix $\mathbf{R}_{ij}$ is obtained by rotating the projection of $\mathbf{x}(t)$ in the plane $i - j$ by an angle θ from the i-th axis to the j-th axis. After the rotation, the elements of the vector $\mathbf{x}(t)$ remains fixed, with exception of the i-th and j-th elements i and j. The matrix $\mathbf{R}_{ij}$ is obtained as follows [10]:

- replace the element (i, i) of the identity matrix $\mathbf{I}$ by cos(θ).
- replace the element (i, j) of $\mathbf{I}$ by -sin(θ).
- replace the element (j, i) of $\mathbf{I}$ by sin(θ).
- replace the element (j, j) of $\mathbf{I}$ by cos(θ).

For example, the rotation matrix $\mathbf{R}_{13}$ in a three-dimensional space is given by:

$$\mathbf{R}_{13} = \begin{bmatrix} \cos(\theta) & 0 & -\sin(\theta) \\ 0 & 1 & 0 \\ \sin(\theta) & 0 & \cos(\theta) \end{bmatrix} \tag{9}$$

It is easy to check that $\mathbf{R}_{ij}$ is orthogonal.

We propose here the following transformation matrix $\mathbf{A}(k)$ (for simplicity, we consider that l is even):

$$\mathbf{A}(k) = \mathbf{R}_{r_1 r_2}(k)\mathbf{R}_{r_3 r_4}(k) \ldots \mathbf{R}_{r_{l-1} r_l}(k) \tag{10}$$

where the vector $\mathbf{r}^{\mathrm{T}} = [r_1\ r_2 \ldots r_l]$ is obtained by randomizing the vector $\mathbf{r}^{\mathrm{T}} = [1\ 2 \ldots l]$ in the beginning of each period k. In other words, the matrix $\mathbf{A}(k)$ is obtained by successive rotations in different planes (and with different axis). As the rotations are independent, the matrix $\mathbf{A}(k)$ can be directly computed. A method to compute $\mathbf{A}(k)$ in the beginning of each period k is presented in Algorithm 1.

Algorithm 1 Transformation Matrix

1: generate the vector $\mathbf{r}$
2: **for** $i = 1$ to $i = l - 1$ with step $i = i + 2$ **do**
3: replace the element (r_i, r_i) of the identity matrix $\mathbf{I}$ by cos(θ).
4: replace the element (r_i, r_{i+1}) of $\mathbf{I}$ by -sin(θ).
5: replace the element (r_{i+1}, r_i) of $\mathbf{I}$ by sin(θ).
6: replace the element (r_{i+1}, r_{i+1}) of $\mathbf{I}$ by cos(θ).
7: **end for**

Defining $\rho = \theta/180$, where θ is given in degrees, the parameter ρ can be employed to control the degree of change for DOPs. If $\rho = 0.0$, the problem stays stationary, while if $\rho = 1.0$, the extreme changes occur. For small values of ρ, the individuals of the current population are rotated by small angles in the planes defined by $\mathbf{r}$, while they are rotated by large angles when ρ is large. In Fig. 3, four transformations $\mathbf{A}(k) = \mathbf{R}_{12}(k)$ caused by different values of ρ for 10 random individuals in a planar space are illustrated. It can be observed that the transformations presented in Figs. 3 c) and d) are equivalent to those presented in Figs. 2 c) and d).

As in the generator presented in the last section, changes in the vectors close to the middle of the search space are smaller than changes in the vectors far from the center of this axis. This problem must be considered when the generator is applied, which is specially important in problems where the global optimum is located in the middle of the search space.

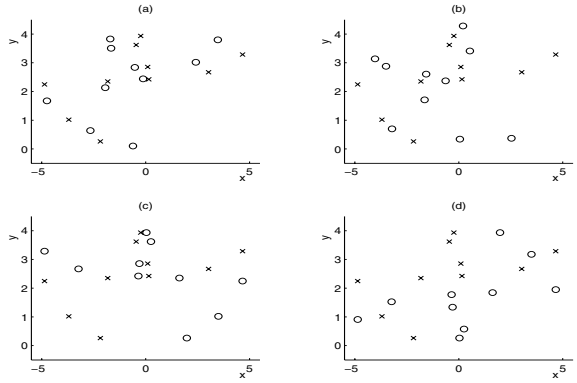

Fig. 3. Individuals (real encoded) in a planar space with transformations defined by: (a) $\rho = 0.1$ (or $\theta = 18^o$); (b) $\rho = 0.3$ (or $\theta = 54^o$); (c) $\rho = 0.5$ (or $\theta = 90^o$); (d) $\rho = 1.0$ (or $\theta = 180^o$).

This DOP generator has the following advantage when compared to the generator presented in the last section.

- all elements of the vector $\mathbf{x}(t)$ are moved by the linear transformation $\mathbf{A}(k)$. In the generator presented in the last section, only $\lfloor \rho \times l \rfloor$ elements are moved. In this new generator, ρ is related to the angle that the planes are moved, and not to the number of changed elements as the generator presented in the last section.

However, this new generator has the following disadvantages:

- the linear transformation $\mathbf{A}(k)$ utilized in this new generator can produce an element x_i of the vector $\mathbf{x}(t)$ out of the range defined by $\{x_i^{min}, x_i^{max}\}$ for $i = 1, \ldots, l$. A procedure must be adopted to deal with this problem.
- the computational implementation of the linear transformation of $\mathbf{x}(t)$ applied in every generation is more expensive for this new generator.

The last disadvantage can be minimized by applying the linear transformation only in the beginning of each period k and not in every generation. In this case, all the population is moved by the linear transformation and the fitness is computed in the usual form. Adopting this strategy means that the individuals are not moved to new positions before being evaluated, but moved only in the beginning of the period. In this way, the fitness function is given by:

$$g(\mathbf{x}(t)) = f(\mathbf{x}(t)) \tag{11}$$

In the beginning of each period k, i.e. when $t/\tau = \lceil t/\tau \rceil$, each individual $\mathbf{x}(t)$ of the current population is moved to the position $\mathbf{z}(t)$ obtained by unnormalizing the vector $\mathbf{z^n}(t)$ given by Eq. 8 with $\mathbf{A}(k)$ computed by Algorithm 1.

As a result of applying the linear transformation only in the beginning of each period k, the computational implementation of this new generator is less expensive because it is not necessary to change the position of each individual before each fitness evaluation, like in the generator presented in Section III.

In comparison to other continuous dynamic problem generators, the continuous dynamic problem generators proposed here have the following properties:

- It is easy to change the speed and degree of the environmental change;
- They do not change the properties of the problem fitness landscape. In this way, they can be used to investigate the performance of different EAs in any real-valued problem, including well studied benchmark optimization problems. In this way, they can be employed to investigate the performance of EAs in problems known by a particular difficulty, like a large number of local optima or with a particular fitness landscape. They can also be applied to investigate the performance of EAs in dynamic problems generated from real-world stationary problems, e.g., they can be employed to investigate the performance of EAs in artificial neural network optimization;
- They can be combined with other dynamic problem generators. When combined with other generators, they can be employed to investigate EAs in changing fitness landscapes;
- All possible solutions, i.e., individiuals located in any place of the seach space, present the same rotation after a change in the problem. In some dynamic problem generators, when the limit values of an axis is reached by the changing global optimum, it may bounce back, which implies in an easier optimization step.

V. Experimental Study

In order to illustrate the work, the problem generators described in Sections III and IV, denoted *Continuous Dynamic Optimization Problem Generator 1* (CDOPG1) and *Continuous Dynamic Optimization Problem Generator 2* (CDOPG2) respectively, were applied in a problem where the fitness function is defined by:

$$f(\mathbf{x}) = \frac{1}{1 + h(\mathbf{x})}, \tag{12}$$

where $h(\mathbf{x})$ is given by the Sphere model

$$h(\mathbf{x}) = \sum_{i=1}^{l} (x_i - c)^2 \tag{13}$$

or by the Generalized Griewank function [12]

$$h(\mathbf{x}) = 1 + \frac{1}{4000} \sum_{i=1}^{l} (x_i - c)^2 - \prod_{i=1}^{l} \cos\left(\frac{x_i - c}{\sqrt{i}}\right) \tag{14}$$

and c is a constant. In the multimodal function given by Eq. 14, the number of local minima increases exponentially with the problem dimension. In this work, we set $l = 30$ and $c = 300$ in order that the global optima is not located in the middle of the search space (see the last section). In the experiments with the Sphere model, we set $l = 30$ and $c = 30$.

TABLE I

MEAN BEST-OF-GENERATION FITNESS OF GAS ON THE DYNAMIC SPHERE MODEL CREATED BY CDOPG1.

τ	50			500			1000		
ρ	0.1	0.5	1.0	0.1	0.5	1.0	0.1	0.5	1.0
SGA	0.00053	0.00005	0.00012	0.50525	0.06230	0.01363	0.73709	0.33225	0.17264
RIGA	0.00061	0.00006	0.00005	0.50462	0.08077	0.03598	0.73400	0.35445	0.24468
HGA	0.00047	0.00007	0.00005	0.23603	0.05024	0.02181	0.52478	0.27431	0.19776

TABLE II

MEAN BEST-OF-GENERATION FITNESS OF GAS ON THE DYNAMIC SPHERE MODEL CREATED BY CDOPG2.

τ	50			500			1000		
ρ	0.1	0.5	1.0	0.1	0.5	1.0	0.1	0.5	1.0
SGA	0.00058	0.00005	0.00013	0.05782	0.05818	0.01263	0.26915	0.32073	0.16799
RIGA	0.00058	0.00006	0.00005	0.06158	0.07907	0.03667	0.28189	0.35032	0.24534
HGA	0.00059	0.00007	0.00005	0.05720	0.04478	0.01898	0.27325	0.27715	0.18927

A. Experimental Design

In the CDOP generators presented in this paper, the fitness function is periodically changed every τ generations. The capability of an algorithm to adapt to dynamic environments under different degrees of convergence can be investigated by setting τ to different values. Based on experiments on stationary problems, environments with three different values of τ were generated. The first two values, $\tau = 50$ and $\tau = 500$, imply changing the fitness function in the early and medium stage of the optimization process respectively. The last value, $\tau = 1000$, implies changing the fitness function in the late stage of the optimization process. Each algorithm was executed for 10 periods of environmental changes in this paper. The degree of change in the dynamic problem generator is controlled by setting the parameter ρ. Environments with three different values of ρ were generated in this paper. These values represent different change levels: light shifting ($\rho = 0.1$), medium variation ($\rho = 0.5$), and severe change ($\rho = 1.0$).

In the experiments presented here, we investigate the performance of the Standard Generational GA (SGA), the Random Immigrants GA (RIGA) [13], and the Hypermutation GA (HGA) [4]. RIGA is inspired by the flux of immigrants of a population in nature [13]. In GAs, the flux of immigrants generally maintains the genetic diversity level of the population, allowing the population to escape from local optima caused by occasional environmental changes. In RIGA, r_r randomly chosen individuals of the current population are replaced by randomly generated individuals in each generation. HGA was proposed to increase the diversity of the population by increasing the mutation rate when the performance of the current best individual of the GA worsens. In this work, when the performance of the current best individual of the HGA worsens, the mutation rate is triggered from p_m to p_{hm}, and remains in this value for 5 generations.

Each algorithm was executed 30 times (with 30 random seeds) for each of the twelve combinations of the environmental dynamics parameters τ and ρ. For each run of an algorithm on a DOP, the individuals of the initial population were randomly chosen with uniform distribution in the range [-600, 600] in the experiments on the Generalized Griewank function and in the range [-100, 100] in the experiments on the Sphere model. In each generation, two individuals of the population were selected according to elitism and the remaining individuals were selected according to roulette wheel sampling. Gaussian mutation with rate $p_m = 0.01$ and two-point crossover with rate $p_c = 0.7$ were employed. The population size was set to 100. Within RIGA, $r_r = 20$ and within HGA, $p_{hm} = 0.3$.

B. Experimental Results

The experimental results of the best-of-generation fitness averaged over 30 runs in the experiments on the dynamic Sphere model created by CDOPG1 and CDOPG2 are presented in Tables I and II respectively. Fig. 4 shows the results of the mean best-of-generation fitness in the experiments with CDOPG1 for $\tau = 50$ and $\tau = 1000$, and Fig. 5 shows the corresponding results for CDOPG2. The experimental results of the best-of-generation fitness averaged over 30 runs in the experiments on the dynamic Generalized Griewank function created by CDOPG1 and CDOPG2 are presented in Tables III and IV respectively. Fig. 6 shows the results of the mean best-of-generation fitness in the experiments with CDOPG1 for $\tau = 50$ and $\tau = 1000$, and Fig. 7 shows the corresponding results with CDOPG2. In these figures, SGA is plotted by a solid line, RIGA by a dashed line, and HGA by a dotted line.

From these tables and figures, several results can be observed and are analyzed as follows.

First, it can be observed that the mean best-of-generation fitness of GAs generally decreases when the value of ρ is increased and when the value of τ is decreased with both DOP generators. This fact can be observed more clearly in the experiments on the Sphere model, which has only one optimum in the fitness landscape. This result is easy to understand. When a change occurs, the individual with the best fitness is moved to a new position with a distance proportional to the value of ρ. The bigger the value of ρ, the severer the change and the greater the challenge to GAs. On the other hand, smaller τ means faster changes and hence GAs have fewer time to explore a new environment.

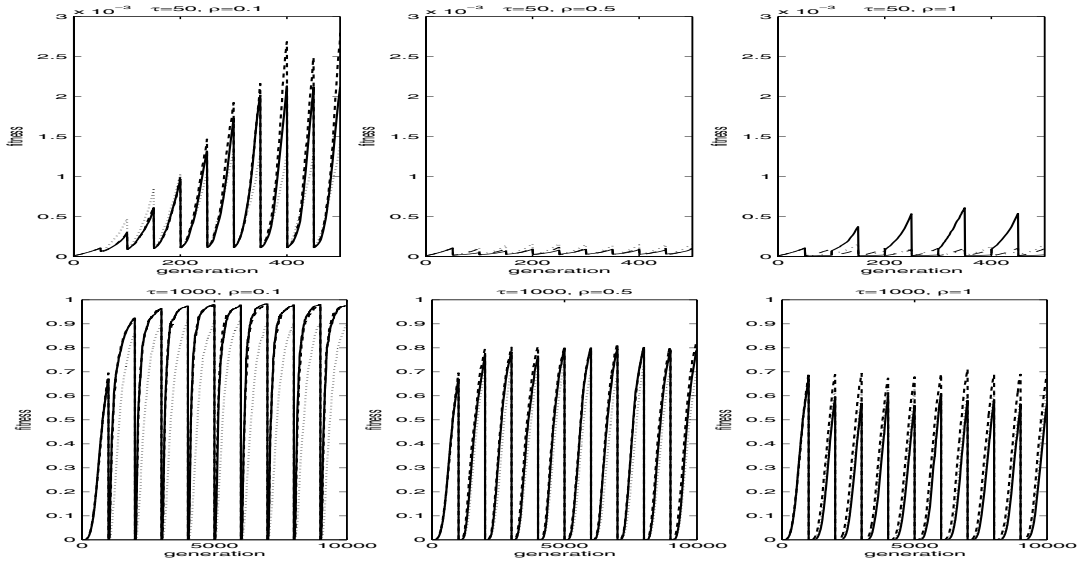

Fig. 4. Mean best-of-generation fitness of GAs on the dynamic Sphere model created by CDOPG1 with $\tau = 50$ and $\tau = 1000$ (SGA: solid line; RIGA: dashed line; HGA: dotted line).

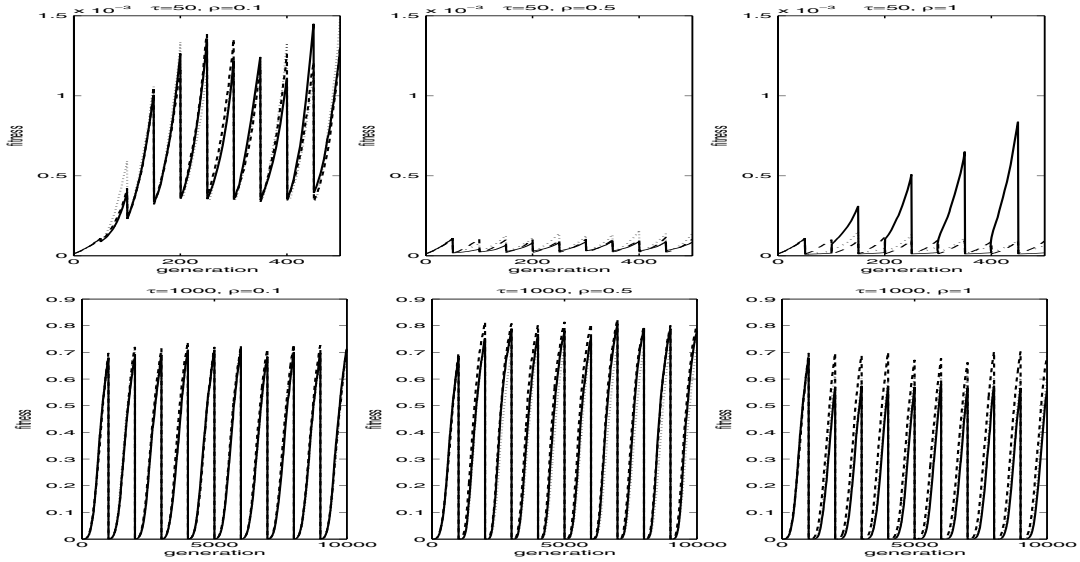

Fig. 5. Mean best-of-generation fitness of GAs on the dynamic Sphere model created by CDOPG2 with $\tau = 50$ and $\tau = 1000$.

Second, comparing the performance of GAs on the dynamic test problems, it can be seen that RIGA outperforms HGA and SGA in those DOPs with severe environmental changes ($\rho = 1.0$). This result agrees with that presented in [4], where the performance of RIGA, HGA, and SGA were compared in DOPs constructed from changing landscapes produced by hills that are shaped using mathematical functions, e.g., sine functions. This result can be explained by the fact that RIGA prepares the population well for possible catastrophic changes due to the increase of the genetic diversity level and more new solutions far from the current best solution are explored by random immigrants. It can be observed that better results for RIGA appear when τ is set to big values instead of 50, i.e., 500 and 1000. This can be explained by the fact that the algorithm had more time to explore good immigrants.

However, the performance of RIGA is worse than that of SGA on some DOPs with slight environmental changes. This fact is explained by the increase in the probability of losing information by replacing individuals that are generally

TABLE III

MEAN BEST-OF-GENERATION FITNESS OF GAS ON THE DYNAMIC GENERALIZED GRIEWANK FUNCTION CREATED BY CDOPG1.

τ	50			500			1000		
ρ	0.1	0.5	1.0	0.1	0.5	1.0	0.1	0.5	1.0
SGA	0.00941	0.00211	0.01306	0.41673	0.19437	0.08652	0.57851	0.34321	0.27136
RIGA	0.00169	0.00541	0.00517	0.44388	0.28559	0.27292	0.61415	0.39427	0.38441
HGA	0.00159	0.00271	0.00257	0.35539	0.21497	0.12334	0.45183	0.34972	0.29606

TABLE IV

MEAN BEST-OF-GENERATION FITNESS OF GAS ON THE DYNAMIC GENERALIZED GRIEWANK FUNCTION CREATED BY CDOPG2.

τ	50			500			1000		
ρ	0.1	0.5	1.0	0.1	0.5	1.0	0.1	0.5	1.0
SGA	0.02464	0.00206	0.01560	0.30329	0.18659	0.07615	0.39800	0.33851	0.26338
RIGA	0.02407	0.00545	0.00524	0.30719	0.28702	0.27153	0.40330	0.39470	0.38308
HGA	0.03060	0.00259	0.00271	0.31895	0.20839	0.10702	0.40655	0.34728	0.28810

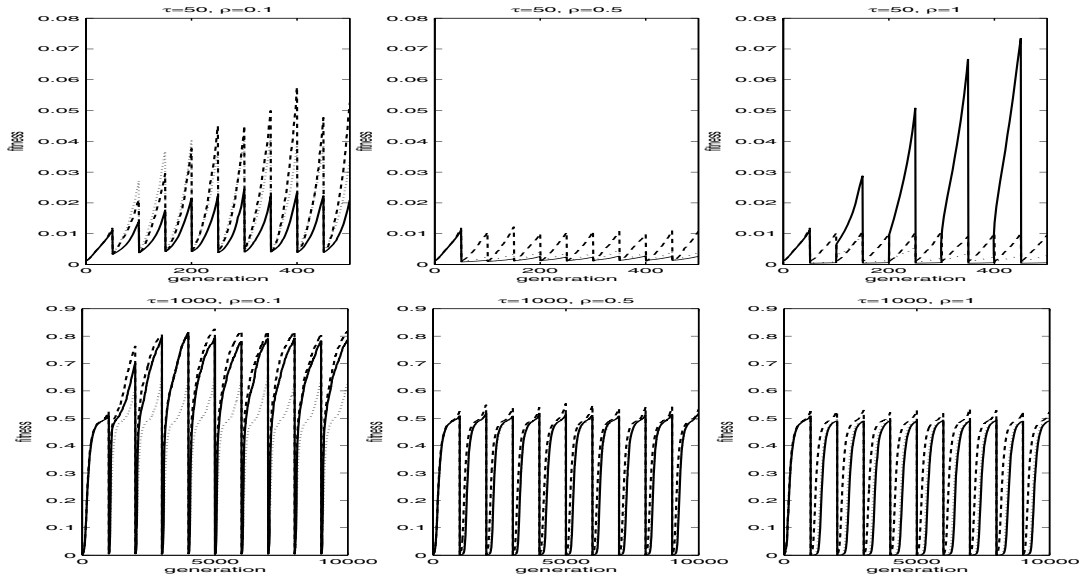

Fig. 6. Mean best-of-generation fitness of GAs on the dynamic Generalized Griewank function created by CDOPG1 with $\tau = 50$ and $\tau = 1000$.

exploring the current best solution by random immigrants. Here, the worse performance of RIGA on DOPs with small environmental changes ($\rho = 0.1$) agrees with those results presented in [4] for DOPs with small changes.

Third, though HGA underperforms RIGA on most dynamic test problems, HGA outperforms SGA on many DOPs. This result validates the benefit of introducing the hypermutation scheme into GAs in dynamic environments. However, when the environment changes slightly, e.g., $\rho = 0.1$, HGA performs worse than SGA on several DOPs. This happens because under slightly changing environments, when a change occurs, the higher mutation rate may destroy good individuals in the previous generation that may also be good in the new environment.

Finally, when comparing the performance of GAs for DOPs created by CDOPG1 and CDOPG2, it can be seen that GAs perform a little better on DOPs created by CDOPG1 than on corresponding DOPs created by CDOPG2.

VI. CONCLUSIONS

Developing benchmark dynamic environments has been an important work in the research of EAs in dynamic environments over the years. A number of DOP generators have been applied to create dynamic test environments to compare the performance of EAs. In this paper, two continuous DOP generators are proposed for evaluating EAs. The first continuous DOP generator is an extension of the XOR generator proposed in [8], [9] for binary codification problems. In this generator, each individual of the current population is moved to a new location in the fitness landscape before being evaluated. In the second continuous DOP generator, the individuals are moved to new locations only when the problem changes.

In both generators, the distances among the individuals in the current population remains unaltered after the linear transformation caused by moving the individuals. In the first generator, the linear transformation is equivalent to change

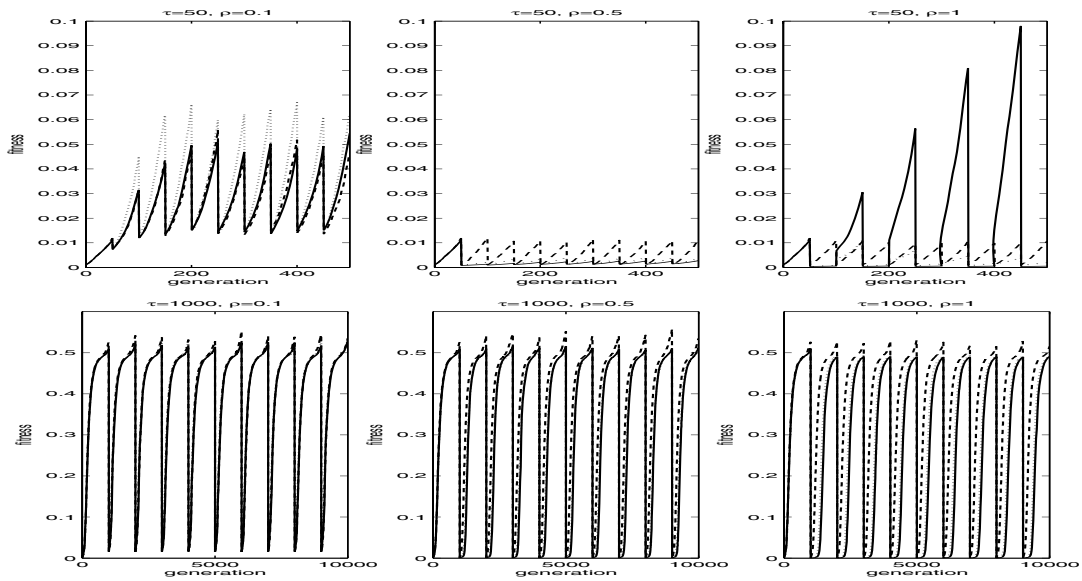

Fig. 7. Mean best-of-generation fitness of GAs on the dynamic Generalized Griewank function created by CDOPG2 with $\tau = 50$ and $\tau = 1000$.

the direction of some axes of the search space while in the second generator, the linear transformation of the individuals is obtained by successive rotations in different planes. In the continuous DOP generators proposed here, it is easy to change the speed of environmental changes by tuning the parameter τ and the degree of environmental changes by tuning the parameter ρ.

In order to investigate the feasibility of the two generators, a set of continuous DOPs were constructed and preliminary experiments were carried out to investigate the performance of several standard GA approaches in dynamic environments. As observed in other works, the experimental results show that both the hypermutation and random immigrants schemes are efficient in improving the performance of GAs in dynamic environments, especially when the environment involves significant changes. However, when the environment changes slightly, both the hypermutation and random immigrants schemes may have negative effect on the performance of GAs.

As the proposed continuous DOP generators do not change the properties of the problem fitness landscape, it can be combined with well studied benchmark optimization problems and with other dynamic problem generators. This will be an interesting future work. The use of only the continuous DOP generators is not interesting when we want to investigate the performance of EAs on DOPs where the fitness landscape changes or the global optimum is located close to the middle of the search space. For DOPs where the global optimum is located close to the middle of the search space, a relevant future work is to investigate new transformations to be applied to the individuals in order to minimize this problem. Another relevant future work is to investigate the use of nonlinear transformations in Eq. 8.

ACKNOWLEDGMENTS

This work was supported by Brazil FAPESP (Proc. 04/04289-6) and UK EPSRC (No. EP/E060722/01).

REFERENCES

[1] J. Branke, *Evolutionary Optimization in Dynamic Environments*. Boston, MA: Kluwer Academic Publishers, 2002.
[2] D. E. Goldberg and R. E. Smith, in *Proc. of the 2nd Int. Conf. on Genetic Algorithms*.
[3] J. Lewis, E. Hart, and G. Ritchie, "A comparison of dominance mechanisms and simple mutation on non-stationary problems," in *Parallel Problem Solving from Nature V*, LNCS 1498, 1998, pp. 139–148.
[4] H. G. Cobb and J. J. Grefenstette, "Genetic algorithms for tracking changing environments," in *Proc. of the 5th Int. Conf. on Genetic Algorithms*, S. Forrest, Ed., 1993, pp. 523–530.
[5] J. Branke, "Memory enhanced evolutionary algorithms for changing optimization problems," in *Proc. of the 1999 Cong. on Evolutionary Computation*, vol. 3, 1999, pp. 1875–1882.
[6] R. W. Morrison and K. A. DeJong, "A test problem generator for non-stationary environments," in *Proc. of the 1999 Congress on Evolutionary Computation*, vol. 3, 1999, pp. 2047–2053.
[7] K. Trojanowski and Z. Michalewicz, "Evolutionary optimization in non-stationary environments," *Journal of Computer Science and Technology*, vol. 1, no. 2, pp. 93–124, 2000.
[8] S. Yang, "Non-stationary problem optimization using the primal-dual genetic algorithm," in *Proc. of the 2003 Congress on Evolutionary Computation*, vol. 3, 2003, pp. 2246–2253.
[9] S. Yang and X. Yao, "Experimental study on population-based incremental learning algorithms for dynamic optimization problems," *Soft Computing*, vol. 9, no. 11, pp. 815–834, 2005.
[10] B. Noble and J. W. Daniel, *Applied linear algebra*. Prentice-Hall, 1977.
[11] K. Weicker and N. Weicker, "Dynamic rotation and partial visibility," in *Proc. of the 2000 Congress on Evolutionary Computation*, 2000, pp. 1125–1131.
[12] X. Yao and Y. Liu, "Fast evolutionary programming," in *Evolutionary Programming V: Proc. 5th Annual Conf. on Evolutionary Programming*, 1996, pp. 451–460.
[13] J. J. Grefenstette, "Genetic algorithms for changing environments," in *Parallel Problem Solving from Nature VII*, 1992, pp. 137–144.