An Analysis of the XOR Dynamic Problem Generator Based on the Dynamical System

Renato Tinós[1] and Shengxiang Yang[2]

[1] Department of Physics and Mathematics, FFCLRP, University of São Paulo (USP)
14040-901, Ribeirão Preto, S.P., Brazil
rtinos@ffclrp.usp.br

[2] Department of Information Systems and Computing, Brunel University
Uxbridge, Middlesex UB8 3PH, U.K.
shengxiang.yang@brunel.ac.uk

Abstract. In this paper, we use the exact model (or dynamical system approach) to describe the standard evolutionary algorithm (EA) as a discrete dynamical system for dynamic optimization problems (DOPs). Based on this dynamical system model, we analyse the properties of the XOR DOP Generator, which has been widely used by researchers to create DOPs from any binary encoded problem. DOPs generated by this generator are described as DOPs with permutation, where the fitness vector is changed according to a permutation matrix. Some properties of DOPs with permutation are analyzed, which allows explaining some behaviors observed in experimental results. The analysis of the properties of problems created by the XOR DOP Generator is important to understand the results obtained in experiments with this generator and to analyze the similarity of such problems to real world DOPs.

1 Introduction

The study of evolutionary algorithms (EAs) for dynamic optimization problems (DOPs) has attracted a rapidly growing interest in recent years due to its importance to real world applications of EAs, where, often, new solutions should be found in short time after a change in the problem [2]. Most researches on EAs for DOPs focus on experimental investigation, and very few investigate the theory behind DOPs [8, 9, 7, 1, 3, 6].

In [7], the standard genetic algorithm (GA) with mutation and selection is investigated in DOPs with regular changes (see Section 3) based on the dynamical system approach (or exact model) of the GA [11]. Despite demanding a large number of equations to track all possible solutions represented by the individuals of the GA, the use of the exact model is attractive as it allows a complete description of the population dynamics [5].

In this paper, we use the dynamical system approach to analyze the XOR DOP Generator [12, 14]. In the XOR DOP Generator, DOPs are created from any binary encoded stationary problem, which allows comparing different algorithms in environments with different properties. This paper investigates the properties of the problems generated by the XOR DOP Generator, which is relevant in order to understand the results obtained in the experiments with this generator and to analyze the similarity of such problems to real world problems.

The rest of this paper is organized as follows. The exact model for the GA in stationary environments (see [5] and [11] for details) is briefly presented in Section 2. Some concepts of DOPs are discussed and formally described based on the dynamical system approach in Section 3. Section 4 presents the analysis of the XOR DOP Generator based on the developed dynamical system model and some experiments are presented in Section 5. Finally, the conclusions of this work with discussions on relevant future work are presented in Section 6.

2 Exact Model of the GA in Stationary Environments

In the exact model proposed by Vose [11], the standard GA is described as a discrete dynamical system [5]. In a GA with binary codification, an individual of a population codifies a possible solution $\mathbf{x} \in \{0,1\}^l$. In the exact model, all possible solutions are represented in an n-dimensional discrete space χ, where each possible solution is enumerated as $\{0, 1, \ldots, n-1\}$ and $n = 2^l$. A population is then defined by an n-dimensional vector $\mathbf{p}$, where each element defines the proportion of each possible solution in the population with size N. As the sum of the elements of $\mathbf{p}$ is equal to 1, population vectors can be described as members of a simplex Λ. This way, the population evolution can be described as a trajectory in the simplex and population vectors can be used to describe the probability distribution of the individuals in the search space. Thus, a generational operator $\mathcal{G} : \Lambda \to \Lambda$ can be defined. The vector $\mathcal{G}(\mathbf{p})$ describes the expected next population [11], i.e., the average over all possible populations of the next generation with variance inversely proportional to the population size N. In the limit $N \to \infty$ (infinite population), the variance goes to zero, and the evolution in the stationary case is deterministically described by the trajectory $\mathbf{p}$, $\mathcal{G}(\mathbf{p})$, $\mathcal{G}^2(\mathbf{p})$, ... In this way, in generation t for the stationary case, the expected population vector for the infinite population model is given by:

$$\mathbf{p}_t = \mathcal{G}^t(\mathbf{p}_0), \tag{1}$$

where $\mathbf{p}_0$ is the initial population vector. When fitness proportional selection and flip mutation are employed, the generational operator can be written as:

$$\mathcal{G}(\mathbf{p}) = \frac{UF\,\mathbf{p}}{\mathbf{f}^{\mathrm{T}}\mathbf{p}}, \tag{2}$$

where $F = \mathrm{diag}(\mathbf{f})$ is a diagonal matrix generated from the fitness vector $\mathbf{f}$ and U is the mutation matrix. The analysis of Eq. (2) can provide insights in understanding the behavior of the GA. The fixed points of $\mathcal{G}$, i.e., points where $\mathcal{G}(\mathbf{p}) = \mathbf{p}$, are given by the eigenvectors of UF. For each eigenvector $\mathbf{p}$, an eigenvalue $\mathbf{f}^{\mathrm{T}}\mathbf{p}$, corresponding to the average fitness of $\mathbf{p}$, can be computed. As UF has only positive values, there is only one eigenvector in Λ, corresponding to the eigenvalue with the largest absolute value [5]. Then, all trajectories in Λ converge to this fixed point, i.e. the system is asymptotically stable [11]. The remaining eigenvectors are not properly fixed points, as, for example, they can lie outside the simplex. However, they play an important role in the evolutionary process as they can change the trajectory in the simplex and can create metastable states that can trap finite populations for several generations [4].

3 Dynamic Optimization Problems

A DOP is an optimization problem where at least one change occurs during the evolutionary process. When a change occurs, the generational operator $\mathcal{G}$ is altered and at least one possible trajectory realized by the population in the simplex Λ is affected. It can be observed that not all modifications in the generational operator can be described as a change. Another important observation is that a change does not necessarily imply a modification in the population or in its current trajectory. For example, if the change does not modify the current trajectory of the population to the fixed point, no effect will be observed in the evolutionary process. The same occurs if the population has converged to the fixed point and this one is not modified by the change.

As the generation operator is modified after a change, Eq. (1) is not valid anymore for every generation t in a DOP. If we consider that changes occur only between the application of two consecutive generational operators, the following equation is valid for the infinite population case:

$$\mathbf{p}_t = \mathcal{G}_t(\mathbf{p}_{t-1}), \tag{3}$$

where $\mathcal{G}_t$ is the generational operator in generation $t \geq 1$.

A series of generational operations between two consecutive changes is called here a change cycle. The first change cycle begins in the first generation of the evolutionary process and ends one generation before the first change, while the last change cycle begins in the generation after the last change and ends until the last generation of the evolutionary process.

The change cycle duration d_e is the number of consecutive generations in change cycle e. If change cycle e begins at generation t_e, then

$$\mathcal{G}_{t_e} = \mathcal{G}_{t_e+1} = \mathcal{G}_{t_e+2} = \ldots = \mathcal{G}_{t_e+d_e-1}, \tag{4}$$

where $d_e > 0$. In abuse of notation, we define now $\mathcal{G}_e$ as the generational operator in change cycle e. In this way, for the infinite population case, the population in generation t is now given by:

$$\mathbf{p}_t = \mathcal{G}_e^{(t-\sum_{i=1}^{e-1} d_i)} \mathcal{G}_{e-1}^{d_{e-1}} \ldots \mathcal{G}_3^{d_3} \mathcal{G}_2^{d_2} \mathcal{G}_1^{d_1}(\mathbf{p}_0), \tag{5}$$

where $e > 0$. It can be observed that a DOP can be viewed as a sequence of stationary processes, where the initial population in the i-th change cycle is the last population generated in the change cycle $i-1$. The minimum value of d_i is one generation, which is the case where the generational operator is modified just one generation after the prior change, while the maximum value of d_i is equal to the index of the current generation, which is the case where the problem is stationary (until the current generation) and Eq. (5) reproduces Eq. (1).

In this paper, we are interested in a class of dynamic problems defined here as DOPs with permutation, which is defined below.

Definition 1. *A **DOP with permutation** is a DOP where the fitness landscape in change cycle $e-1$ is modified according to a permutation matrix, i.e.,*

$$\mathbf{f}_e = \sigma_{\mathbf{k}_e} \mathbf{f}_{e-1},$$

where $\sigma_{\mathbf{k}_e}$ is a permutation matrix mapping the element at position $\mathbf{i}$ of the vector $\mathbf{f}_{e-1}$ to the element at position $\mathbf{i} \oplus \mathbf{k}_e$ of the vector $\mathbf{f}_e$, where $\oplus$ is the bitwise exclusive-or (XOR), or addition modulo 2, operator. The vector $\mathbf{i} \in \{0,1\}^l$ indicates the position of the element in the fitness vector. The vector $\mathbf{k}_e \in \{0,1\}^l$ controls the permutation of the elements of the fitness vector.

In a DOP with permutation, the fitness values are preserved in the search space, i.e., they are only resorted. In [7], DOPs with regular changes, which are a special subset of DOPs with permutation (Definition 1) where the transitional rule is deterministic and belongs to a permutation group where $\sigma_{\mathbf{k}_{e+t}} = (\sigma_{\mathbf{k}_e})^t$ for $t \geq 0$, are defined. As a consequense, in DOPs with regular changes, the fixed points can be computed and the asymptotic states can be then analyzed [7].

4 The XOR DOP Generator

The XOR DOP Generator [14] can generate DOPs from any binary encoded problem. In the XOR DOP Generator, given a stationary problem with fitness function $f(\mathbf{x}_t)$ and the solution $\mathbf{x}_t \in \{0,1\}^l$, the fitness function $f_e(\mathbf{x}_t)$ of an environment, which is periodically changed every τ generations, is computed by:

$$f_e(\mathbf{x}_t) = f\big(\mathbf{x}_t \oplus \mathbf{m}_e\big), \qquad (6)$$

where t is the generation index, $e = \lceil t/\tau \rceil$ is the change cycle index, and $\mathbf{m}_e$ is a binary mask for change cycle e, which is incrementally generated by:

$$\mathbf{m}_e = \mathbf{m}_{e-1} \oplus \mathbf{r}_e, \qquad (7)$$

where $\mathbf{r}_e$ is a binary template randomly created for change cycle e 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 change cycle, $\mathbf{m}_1$ 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 [10]. Instead of evaluating the fitness of the individual at $\mathbf{x}_t$, the fitness is evaluated at $\mathbf{x}_t \oplus \mathbf{m}_e$. It can be observed that the XOR DOP Generator produces DOPs with changes in the fitness landscape. Based on the XOR DOP Generator properties, the following theorem is proposed.

Theorem 1. *The XOR DOP Generator produces DOPs with permutation.*

Proof : It can be observed that only the fitness vector is modified by Eq. (6). The fitness vector in change cycle $e > 1$ for a DOP generated by the XOR DOP Generator from a stationary problem with fitness function $f(\mathbf{x}_t)$ is given by:

$$\mathbf{f}_e = \begin{bmatrix} \mathbf{f}_{e(0)} \\ \mathbf{f}_{e(1)} \\ \vdots \\ \mathbf{f}_{e(n-1)} \end{bmatrix} = \begin{bmatrix} f(\mathbf{x}_{(0)} \oplus \mathbf{m}_e) \\ f(\mathbf{x}_{(1)} \oplus \mathbf{m}_e) \\ \vdots \\ f(\mathbf{x}_{(n-1)} \oplus \mathbf{m}_e) \end{bmatrix} = \begin{bmatrix} f(\mathbf{x}_{(0)} \oplus \mathbf{m}_{e-1} \oplus \mathbf{r}_e) \\ f(\mathbf{x}_{(1)} \oplus \mathbf{m}_{e-1} \oplus \mathbf{r}_e) \\ \vdots \\ f(\mathbf{x}_{(n-1)} \oplus \mathbf{m}_{e-1} \oplus \mathbf{r}_e) \end{bmatrix}, \qquad (8)$$

where $\mathbf{x}_{(i)}$ is the i-th possible solution in the n-dimensional discrete space χ and $\mathbf{f}_{e(i)}$ is its respective fitness.

Defining the i-th solution in change cycle $e-1$ as $\mathbf{x}_{e-1(i)} = \mathbf{x}_{(i)} \oplus \mathbf{m}_{e-1}$, Eq. (8) can be written as:

$$\mathbf{f}_e = \begin{bmatrix} f(\mathbf{x}_{e-1(0)} \oplus \mathbf{r}_e) \\ f(\mathbf{x}_{e-1(1)} \oplus \mathbf{r}_e) \\ \vdots \\ f(\mathbf{x}_{e-1(n-1)} \oplus \mathbf{r}_e) \end{bmatrix} = \sigma_{\mathbf{r}_e} \begin{bmatrix} \mathbf{f}_{e-1(0)} \\ \mathbf{f}_{e-1(1)} \\ \vdots \\ \mathbf{f}_{e-1(n-1)} \end{bmatrix} = \sigma_{\mathbf{r}_e} \mathbf{f}_{e-1}, \tag{9}$$

where $\sigma_{\mathbf{r}_e}$ is a permutation matrix mapping the element at position $\mathbf{j}$ of the vector $\mathbf{f}_{e-1}$ to the element at position $\mathbf{j} \oplus \mathbf{r}_e$ of the vector $\mathbf{f}_e$. Equation (9) indicates that the fitness of the i-th solution in change cycle e is equal to the fitness of the i-th solution in change cycle $e-1$ moved according to the permutation $\mathbf{r}_e$. That is, the XOR DOP Generator produces DOPs with permutation. □

One can still observe that the XOR DOP Generator produces stationary environments for $\rho = 0.0$ and DOPs with regular changes for $\rho = 1.0$. In the latter case, the DOP switches between two environments. For $0.0 < \rho < 1.0$, the changes are not regular because the template $\mathbf{r}_e$ is randomly generated, and, as a consequence, the metastable points for the stationary environments generated for each template $\mathbf{r}_e$ are generally different.

As the DOP is viewed as a sequence of stationary environments, the analysis of how the fixed points and metastable states for each stationary environment are related can provide insights in understanding GA's behavior on the DOP generated by the XOR DOP Generator. Here, the state of the DOP in a change cycle corresponding to the fixed point in the respective stationary environment is called main metastable state. It is important to observe that the main metastable states are not fixed points of the DOP, as the problem changes and the population generally does not converge to a fixed point. However, the metastable states control the trajectory of the population during each change cycle.

In a DOP with permutation, the points of the search space in change cycle $e > 1$ are obtained by the permutation, according to $\sigma_{\mathbf{k}_e}$, of the points of the search space in change cycle $e-1$. As a consequense, the i-th eigenvector $\mathbf{p}_{e(i)}$ of $U_e F_e$ in change cycle e can be obtained by the permutation, according to $\sigma_{\mathbf{k}_e}$, of the respective eigenvector for the environment in change cycle $e-1$, i.e.,

$$\mathbf{p}_{e(i)} = \sigma_{\mathbf{k}_e} \mathbf{p}_{e-1(i)}. \tag{10}$$

Besides, the eigenvalues of $U_e F_e$ for two environments defined by change cycles $e-1$ and e are equal. As the metastable states of $\mathcal{G}_e$ are given by the eigenvectors of $U_e F_e$, the metastable states of $\mathcal{G}_e$ and $\mathcal{G}_{e-1}$ in a DOP with permutation are related by the permutation matrix $\sigma_{\mathbf{k}_e}$ (or $\sigma_{\mathbf{r}_e}$ for environments created by the XOR DOP Generator). Besides, the average fitness (eigenvalue) at the main metastable remains the same.

Theorem 2. *Consider the standard GA with mutation and fitness proportional selection is applied in: i) a DOP with permutation (Definition 1), where the* *duration and permutation matrix of change cycle e are, respectively,* d_e *and* $\sigma_{\mathbf{k}_e}$*; ii) in a stationary environment where the population is permuted according to the permutation matrix* $\sigma_{\mathbf{k}_e}$ *after every cycle* $e = 1, 2, \ldots$ *with duration* d_e*. If both evolutionary processes have the same initial population and parameters, and the fitness function in the first change cycle for the first process is equal to the fitness function in the second process, then the evolution of the population in the two processes is identical, i.e., the two evolutionary processes are equivalent.*

Proof : According to Eqs. (3) and (2), the population for the infinite population case in generation $t > 1$ for a DOP with fitness landscape changes is given by:

$$\mathbf{p}_t = \frac{UF_e \; \mathbf{p}_{t-1}}{\mathbf{f}_e^{\mathrm{T}} \mathbf{p}_{t-1}}. \tag{11}$$

For a change cycle $e > 1$ in a DOP with permutation (Definition 1), $\mathbf{f}_e = \sigma_{\mathbf{k}_e} \mathbf{f}_{e-1}$, and, as a consequense, $F_e = \sigma_{\mathbf{k}_e} F_{e-1} \sigma_{\mathbf{k}_e}$. Then, we can write Eq. 11 as:

$$\mathbf{p}_t = \frac{U\sigma_{\mathbf{k}_e} F_{e-1} \sigma_{\mathbf{k}_e} \; \mathbf{p}_{t-1}}{\mathbf{f}_{e-1}^{\mathrm{T}} \sigma_{\mathbf{k}_e} \mathbf{p}_{t-1}}. \tag{12}$$

Defining $\mathbf{q}_t = \sigma_{\mathbf{k}_e} \mathbf{p}_t$ and considering that U and $\sigma_{\mathbf{k}_e}$ commute, then:

$$\mathbf{q}_t = \frac{UF_{e-1} \; \mathbf{q}_{t-1}}{\mathbf{f}_{e-1}^{\mathrm{T}} \mathbf{q}_{t-1}}. \tag{13}$$

It can be observed that Eqs. (11) and (13) are similar. If, after the change e, we transform the final population at the last generation of change cycle $e - 1$ according to $\mathbf{q}_t = \sigma_{\mathbf{k}_e} \mathbf{p}_t$, then, we can use Eq. (13) with the same fitness landscape of change cycle $e - 1$ to reproduce the dynamics of the population in the infinite population case for change cycle e. □

As a consequence of Theorem 2, the XOR DOP Generator can be simplified. Instead of computing the fitness of each individual $\mathbf{x}_t$ of the population at the position $\mathbf{x}_t \oplus \mathbf{m}_e$ in every generation, each individual of the initial population in change cycle e is moved to $\mathbf{x}_t = \mathbf{x}_t \oplus \mathbf{r}_e$, i.e., the population is moved only one time, and the fitness is computed as $f(\mathbf{x}_t)$, like in the stationary environment. In this way, the complexity of the procedure is reduced from $O(lNd_e)$ to $O(lN)$.

5 Experimental Study

In this section, we present simulations for the evolution of the standard GA with mutation and fitness proportional selection in a DOP created by the XOR DOP Generator from a deceptive fitness function defined by:

$$f(\mathbf{x}) = \begin{cases} l, & \text{if } u(\mathbf{x}) = l \\ (l-1) - u(\mathbf{x}), & \text{otherwise,} \end{cases} \tag{14}$$

where $u(\mathbf{x})$ is the unitation function of a binary vector $\mathbf{x}$ of length l. This function has one global optimum and one local optimum. In the simulations presented in this section, $l = 8$, the mutation rate is 0.01, and the initial population ($\mathbf{p}_0$)

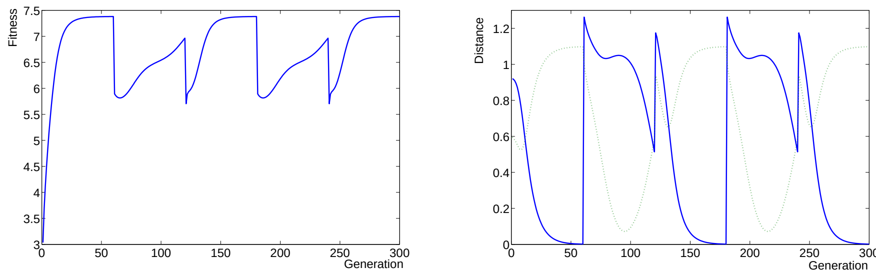
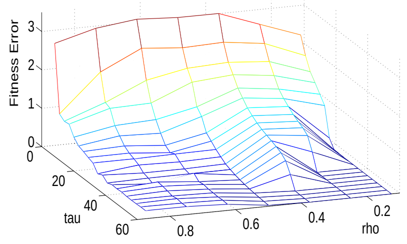


Fig. 1. Mean fitness of the population (left) and distance to the current first (solid) and second metastable states (dotted) for five change cycles with $\tau = 60$ and $\rho = 0.875$

is uniformly distributed. For all simulations, Eq. 11 is employed to generate the population vector $\mathbf{p}(t)$, i.e., the exact model with infinite population is employed in the simulations in order to generate the expected next population during the evolutionary process. In the simulations, the problem changes, according to $\mathbf{f}_e = \sigma_{\mathbf{r}_e}\mathbf{f}_{e-1}$, every τ generations with change degree ρ. Twenty values of τ (from $\tau = 3$ to $\tau = 60$ with a step size 3) and seven values of ρ (from $\rho = 0.125$ to $\rho = 0.875$ with a step size 0.125) are considered. In this way, 140 simulations were executed, one for each pair of τ and ρ. Each evolutionary process is simulated for 30 change cycles of the infinite population model.

Figure 1 shows a simulation with $\rho = 0.875$ and $\tau = 60$, where the mean fitness of the population during the evolution and the Euclidean distance between the population in the current generation and the two eigenvectors with the largest eigenvalues are presented. The first eigenvector corresponds to the current main metastable state (where the number of individuals of the population at the global optimum is larger than the number of individuals at any other place), while the second eigenvector is the metastable state with the second largest eigenvalue (where the number of individuals of the population at the local optimum is larger than the individuals at any other place). It can be observed that, in some change cycles, the population goes to the neighborhood of the second metastable state and, after some generations, goes to the main metastable state. When Eq. (13) is used, i.e., evolution in a stationary environment where the population is permuted according to the permutation matrix $\sigma_{\mathbf{k}_e}$ after every cycle e with duration $d_e = 60$, the same graphics presented in Fig. 1 are obtained if the same parameters are employed (those graphics are not shown here), as stated by Theorem 2.

Figure 2 presents the results for all simulations. The first graph shows the value of $f_{opt} - f(\mathbf{p}_e)$ averaged over all change cycles e, where f_{opt} is the current mean value of fitness in the main metastable state and $f(\mathbf{p}_e)$ is the mean fitness of the population $\mathbf{p}_e$ in the end of change cycle e. The second graph presents the respective mean distance between the current main metastable state and the state $\mathbf{p}_e$ in the end of change cycle e. From Fig. 2, some observations can be made. When τ is close to 60 generations, i.e., in slow changing environments, the



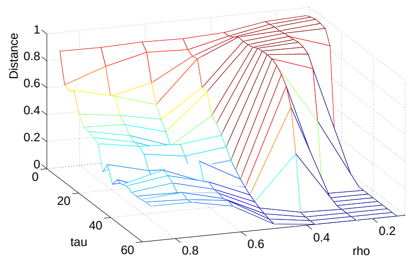


Fig. 2. Fitness error (left) and distance to the current main mestastable (right) in the simulations for different τ and ρ. The values are relative to the average (over 30 change cycles) obtained by the population vector in the generation before the change

population always reaches the main metastable state after changes with small degree of change ρ. When τ is large, there is enough time for the population to go from the neighborhood of the main metastable state (where most of the population is at the global optimum) in change cycle $e-1$ to the neighborhood of the main metastable state in change cycle e. As a consequence, the fitness error in the end of each change cycle is zero when τ is large and ρ is small (see Fig. 1). In the XOR DOP Generator, the parameter ρ controls the degree of change. As ρ controls the percentage of changed bits from template $\mathbf{r}_{e-1}$ to template $\mathbf{r}_e$, the hamming distance between $\mathbf{r}_{e-1}$ and $\mathbf{r}_e$ is $h(\mathbf{r}_e, \mathbf{r}_{e-1}) = \lfloor \rho \times l \rfloor$. In this way, larger ρ imply larger hamming distance between the optima in two consecutive change cycles and in longer trajectories of the population in the simplex, and, thus, more time to reach the neighborhood of the main metastable point.

However, it can be observed that a higher degree of modification in the fitness landscape (larger ρ) does not necessarily imply a worse performance of the GA in the DOP for medium and small τ. One can observe that for medium and small τ, the simulations with $\rho = 0.375$ presented worse performance than those for larger ρ. This behavior can be found in experiments with the XOR DOP Generator for different algorithms (for example, see [13]). The performance of the GA is related to trajectories of the population in the simplex, and the trajectories are related to the fitness vector and the transformation operators. In a medium velocity or fast changing environment, generally, when the population reaches the neighborhood of the main metastable point in change cycle $e-2$, the population after the change is closer to the second metastable state in the next change cycle when ρ is large. In this case, the population does not have enough time to be closer to the new main metastable state neighborhood in change cycle $e-1$ than to the old main metastable neighborhood. However, when the problem changes again, the population is close to the neighborhood of the main metastable state in change cycle e for ρ close to 1. The mean Hamming distance of template $\mathbf{r}_e$ between two change cycles, which is given by $\bar{h}(\mathbf{r}_e, \mathbf{r}_{e-2}) = 2l(\rho - \rho^2)$, explains the behavior of the GA in this case. It can be observed that the mean fitness generally alternates between two different values for larger ρ and medium or small τ (see, for example, Fig. 1). One can observe that the values of distance in Fig. 2 are higher for ρ close to one than for ρ close to zero, as in part of the change cycles, the population remains in the neighborhood of the second metastable state for larger ρ.

Two observations can be made for the previous analysis. First, a higher degree of modification (ρ) in the templates $\mathbf{r}_e$ does not necessarily imply a worse performance of the GA. This result has been observed in several experiments with the XOR DOP Generator (for example, see [13]). The performance of the GA is related to the trajectories of the population in the simplex, which makes more complex the analysis of the performance of the algorithms. Second, the metrics used to compare the algorithms in DOPs cannot be adequate for some problems. For example, in the problem investigated here, an algorithm that keeps the population close to the second metastable neighborhood for a high degree of change in a fast changing environment can have higher mean fitness than an algorithm that allows the population escaping from the local optima, but does not have enough time to reach the main metastable state neighborhood.

6 Conclusion and Future Work

In this paper, DOPs are defined based on the dynamical system (or exact model) [11] and the class of DOPs with permutation is defined. Such definition, and others that can be defined based on the same approach, can be useful to classify real world DOPs and, hence, allow a systematic analysis of such problems based on the properties of each class. Here, the XOR DOP Generator, which allows creating DOPs from any binary encoded stationary problem, is analyzed based on the dynamical system approach and the definition presented in Section 3. In this paper, the optimization process of the GA on the DOP is viewed as a sequence of evolutionary processes, each one described as a stationary optimization problem, where the initial population in a change cycle with duration $d_e \geq 1$ is given by the last population in the previous change cycle. In the problems generated by the XOR DOP Generator, the duration of all change cycles is equal and the fitness vector of the problem in change cycle $e > 1$ is related to the fitness vector in change cycle $e-1$ by a random template $\mathbf{r}_e$. Thus, a problem created by the XOR DOP Generator is identified as a DOP with permutation (Definition 1), where the fitness vector changes according to $\mathbf{f}_e = \sigma_{\mathbf{r}_e}\mathbf{f}_{e-1}$.

When the standard GA with proportional fitness selection and mutation is applied to a DOP with permutation, the eigenvectors of the fixed point equation between two consecutive change cycles are related by the permutation matrix $\sigma_{\mathbf{k}_e}$ (or $\sigma_{\mathbf{r}_e}$ in DOPs created by the XOR DOP Generator). This way, the metastable points in change cycle e can be obtained by the permutation (according to $\sigma_{\mathbf{k}_e}$) of the same points in change cycle $e-1$, and the evolution in a DOP with permutation is equivalent to that in a stationary environment where the population is permuted by the permutation matrix $\sigma_{\mathbf{k}_e}$ after each cycle $e = 1, 2, \ldots$ with duration d_e. Hence, the XOR DOP Generator can be simplified by moving the initial population of a change cycle instead of computing the fitness function of each individual in each generation in a new position. In this paper, the influence of the parameter ρ in the XOR DOP Generator is also analyzed, and the results obtained in experiments related in the literature, where the worst performance for some algorithms are obtained for medium ρ, is explained.

It can be observed that algorithms exploring the properties described on the analysis of the XOR DOP Generator can be proposed. However, it is not clear if such algorithms are useful in real world DOPs. To answer this question, the real world DOPs identified as permutation DOPs should be described, which should allow the use of the XOR DOP Generator to reproduce such problems. In this way, a very relevant future investigation is to analyze real world DOPs, to classify them according to their properties, and to develop DOPs generators based on the identified class of DOPs. Another relevant future work is to analyze algorithms proposed for DOPs, e.g., GA with hypermutation and GA with random immigrants, according to the dynamical system approach.

Acknowledgment

This work was supported by Brazil FAPESP under Grant 04/04289-6 and by UK EPSRC under Grant EP/E060722/2.

References

1. Arnold, D. V., Beyer, H.-G.: Random dynamics optimum tracking with evolution strategies. In: Parallel Problem Solving from Nature VII, pp. 3–12 (2002)
2. Branke, J.: Evolutionary Optimization in Dynamic Environments. Kluwer (2001)
3. Droste, S.: Analysis of the (1+1) EA for a dynamically changing onemax-variant. In: 2002 Congr. on Evol. Comput., pp. 55–60 (2002)
4. Van Nimwegen, E., Crutchfield, J. P., Mitchell, M.: Finite populations induce metastability in evolutionary search. Physics Letters A, 229(3), 144–150 (1997)
5. Reeves, C. R., Rowe, J. E.: Genetic algorithms - principles and perspectives: a guide to GA theory. Kluwer Academic Publishers (2003)
6. Rohlfshagen, P., Lehre, P. K., Yao, X.: Dynamic evolutionary optimisation: an analysis of frequency and magnitude of change. In: 2009 Genetic and Evol. Comp. Conf., pp. 1713–1720 (2009)
7. Ronnewinkel, C., Wilke, C. O., Martinetz, T.: Genetic algorithms in time-dependent environments. In: Kallel, L., Naudts, B., Rogers, A. (eds), Theor. Aspects of Evol. Comp., pp. 263–288, Springer (2001)
8. Rowe, J. E.:.: Finding attractors for periodic fitness functions. In: 1999 Genetic and Evol. Comp. Conf., pp. 557–563 (1999)
9. Rowe, J. E.: Cyclic attractors and quasispecies adaptability. In: Kallel, L., Naudts, B., Rogers, A. (eds), Theor. Aspects of Evol. Comp., pp. 251–259, Springer (2001)
10. Tinós, R., Yang, S.: Continuous dynamic problem generators for evolutionary algorithms. In: 2007 Congr. on Evol. Comput., pp. 236–243 (2007)
11. Vose, M. D.: The Simple Genetic Algorithm: Foundations and Theory. The MIT Press (1999)
12. Yang, S.: Non-stationary problem optimization using the primal-dual genetic algorithm. In: 2003 Congr. on Evol. Comput., pp. 2246–2253 (2003)
13. Yang, S., Tinós, R.: A hybrid immigrants scheme for genetic algorithms in dynamic environments. Int. J. of Autom. and Comput., 4(3), 243–254 (2007)
14. Yang, S., Yao, X.: Experimental study on population-based incremental learning algorithms for dynamic optimization problems. Soft Comput., 9(11), 815–834 (2005)