Genet Program Evolvable Mach (2007) 8:255–286
DOI 10.1007/s10710-007-9024-z

ORIGINAL PAPER

A self-organizing random immigrants genetic algorithm for dynamic optimization problems

Renato Tinós · Shengxiang Yang

Received: 22 July 2005 / Revised: 8 January 2007 / Published online: 15 May 2007

Abstract In this paper a genetic algorithm is proposed where the worst individual and individuals with indices close to its index are replaced in every generation by randomly generated individuals for dynamic optimization problems. In the proposed genetic algorithm, the replacement of an individual can affect other individuals in a chain reaction. The new individuals are preserved in a subpopulation which is defined by the number of individuals created in the current chain reaction. If the values of fitness are similar, as is the case with small diversity, one single replacement can affect a large number of individuals in the population. This simple approach can take the system to a self-organizing behavior, which can be useful to control the diversity level of the population and hence allows the genetic algorithm to escape from local optima once the problem changes due to the dynamics.

Keywords Genetic algorithms · Self-organized criticality · Dynamic optimization problems · Random immigrants

1 Introduction

In recent years evolutionary algorithms (EAs) have been successfully applied to a large number of optimization problems. EAs are a class of meta-heuristic algorithms which are inspired by the principles of natural evolution. While natural evolution deals very well with environmental changes, caused, for example, by natural

R. Tinós
Departamento de Física e Matemática, FFCLRP, Universidade de São Paulo (USP),
Ribeirão Preto, 14040-901, Brazil
e-mail: rtinos@ffclrp.usp.br

S. Yang (✉)
Department of Computer Science, University of Leicester, University Road,
Leicester, LE1 7RH, UK
e-mail: s.yang@mcs.le.ac.uk

cataclysms, geological modifications, and competition for natural resources, they represent a serious challenge for traditional EAs. As a significant part of optimization problems in the real world are dynamic optimization problems (DOPs), there is a growing interest in research of EAs for such problems [4].

In DOPs, the evaluation function, the decision variables, and the constraints of the optimization problem are not fixed [22]. When changes occur in the problem, the solution given by the optimization procedure may be no longer effective, and a new solution should be found [5]. The optimization problem may change for several reasons, like faults, machine degradation, environmental or climatic modifications, or economic factors.

The simplest approach to deal with DOPs is to start a new optimization process whenever a change in the problem is noticed. The optimization process, however, generally requires time and substantial computational effort. If the new solution after the change in the problem is, in some sense, related to the previous solution, knowledge obtained during the search for the old solution can be used to find a new solution [15]. In this case, the search for new solutions based on the old solutions can save substantial processing time. EAs are particularly attractive for such problems. Individuals representing solutions of the problem prior to the changes can be transferred into the new optimization process.

However, in EAs, the population of solutions generally converges in the fitness landscape to points close to the best individual of a population. If the fitness landscape abruptly changes, the actual population can be trapped in local optima close to the old solution. In fact, premature convergence of the solution to a local optimum is not a problem exclusive to DOPs. It can also be a serious problem in stationary optimization problems [19]. In order to avoid premature convergence, several approaches where diversity is re-introduced or maintained throughout the run have been proposed in the literature over recent years (see surveys [5, 15, 22]). Typical examples of such approaches are the random immigrants approach [13], sharing or crowding mechanisms [6], variable local search [23], thermodynamic genetic algorithm [20], and the use of hypermutation [7].

The random immigrants approach is very interesting and simple [13]. In a genetic algorithm (GA) with random immigrants, a fraction of the current population is replaced by randomly generated individuals in each generation of the run. A replacement strategy, like replacing random or worst individuals of the population, defines which individuals are replaced by the immigrants. The random immigrants approach tries to maintain the diversity level of the population, which can be very useful to prepare the GA for possible changes in the fitness landscape [8].

However, if the number of genes in an individual is high and the local optimum of the population has fitness much higher than the mean fitness of all possible solutions of the search space, the survival probability of new random individuals is generally very low. This happens because the selection methods employed in GAs preserve, directly or indirectly, the best individuals of a population, and the probability that the fitness of new random individuals is higher than (or close to) the fitness of current individuals is generally low.

In this paper, instead of substituting the worst or randomly selected individuals with random immigrants in each generation, the worst individual and individuals

with indices close to its index are replaced. The newly introduced immigrants are placed in a subpopulation and are not allowed to be replaced by individuals of the main population. In this way, individuals start to interact among themselves and, if the fitness of the individuals is close, as in the case of low diversity levels, one single replacement of an individual can affect a great number of individuals of the population in a chain reaction. The number of individuals in the subpopulation is not defined by the programmer, but is given by the number of individuals created in the current chain reaction. It is important to observe that this simple approach can take the system to a self-organized behavior useful in DOPs.

The experimental results of our work suggest that the proposed GA shows a kind of self-organizing behavior known as self-organized criticality (SOC) [1], described in Sect. 3 of this paper. Prior to that, we present in Sect. 2 the random immigrants approach. The proposed GA is presented in Sect. 4, and experimental results with DOPs are reported and analyzed in Sect. 5. Section 6 concludes our paper with discussions on relevant future work.

2 The random immigrants genetic algorithm

The *Random Immigrants GA* (RIGA) proposed by Grefenstette [13] is inspired by the flux of immigrants in a biological population. In GAs, the flux of immigrants generally increases the genetic diversity level of a population, allowing to escape from local optima caused by occasional environmental changes. The RIGA can be summarized by Algorithm 1, where i denotes the index of each individual of the population. Algorithm 1 differs from the generational *Simple GA* (SGA) only by the inclusion of line 4.

Algorithm 1 Random Immigrants Genetic Algorithm

Require p_c: crossover rate; p_m: mutation rate; r_r: replacement rate
1: initializePopulation(**P**)
2: evaluatePopulation(**P**)
3: **while** (stop criteria are not satisfied) **do**
4: **P** ← replace FractionPopulation(**P**, r_r)
5: **for** $i \leftarrow 1$ to **P**.*size* **do**
6: $\mathbf{P_{new}}.individual(i) \leftarrow$ selection$(\mathbf{P}, i)$
7: **end for**
8: crossover $(\mathbf{P_{new}}, p_c)$
9: mutation$(\mathbf{P_{new}}, p_m)$
10: evaluatePopulation$(\mathbf{P_{new}})$
11: **P**← $\mathbf{P_{new}}$
12: **end while**
end

In the RIGA, some individuals of the current population **P** are replaced by randomly generated individuals. A replacement rate r_r specifies the number of individuals replaced in each generation. In the standard RIGA, randomly chosen

individuals are replaced in each generation. In another replacement strategy, instead of replacing randomly chosen individuals, the individuals of the current population with the lowest fitness are replaced by random immigrants.

3 Self-organized criticality

Bak et al. suggested in Ref. [2] that systems with several interacting constituents may exhibit a kind of self-organizing behavior, which was named SOC, with interesting properties. It was suggested that several phenomena exhibit SOC, e.g., sand piles, earthquakes, forest fires, electric breakdowns, and growing interfaces.

Systems exhibiting SOC have an interesting characteristic: even without any control action from outside, they self-organize into a critical state [14]. In a system exhibiting non-critical behavior, the distribution of responses to external perturbation is narrow and can be well described by an averaged value. In a system exhibiting critical behavior, no single characteristic response exists, i.e., the system exhibits scale invariance, and a small perturbation in one location of the system may generate a small effect on its neighborhood or a chain reaction that affects all the constituents of the system.

The statistical distributions describing the response of the system exhibiting SOC are given by power laws in the form

$$p(s) \sim s^{-\tau} \tag{1}$$

and

$$p(d) \sim d^{-\alpha}, \tag{2}$$

where s is the number of constituents of the system affected by the perturbation, d is the duration of the chain reaction (lifetime), and τ and α are real constants. As an example, consider the sand pile model described in Ref. [2], where a single grain is added at a random position in every interval of time Δt. In order to characterize the response of the system, one can measure the number of sand grains (s) involved in each avalanche induced by the addition of a single grain and the duration (d) of each avalanche. In the critical state, the statistical distributions describing the response of the sand pile model to the addition of a single grain are given by Eqs. 1 and 2, and the addition of a single grain can affect from only a grain in its neighborhood to the whole sand pile.

Researchers have suggested that SOC occurs in natural evolution too [1]. Evidence of SOC in evolution would be the fact that it does not happen gradually at a slow and constant pace [12]. There are many more small extinction events than large events, such as the Cretaceous extinction of dinosaurs and many other species, and extinction events occur on a large variety of length scales [21]. These facts suggest that extinctions propagate through ecosystems, such as avalanches in a sand pile, and perturbations of the same size can unleash extinction events of a large variety of sizes. This would occur because species coevolve to a critical state [16].

In Ref. [1], a very simple simulation model, known as the Bak–Sneppen Model, was proposed to explore the connection between evolution and SOC. In the one-dimensional version of this model, the individuals (or species in the authors' terminology) are placed in a circle, and a random value of fitness is assigned to each individual. In each generation of the simulation, the individual with the lowest fitness in the current population, one individual located to its right position, and one to its left position will have their fitness values replaced by random values. An analogy of the connection between neighbors in the model is the interaction between species in nature. If, as an example, a prey goes extinct, the fitness of its predators will change. The Bak–Sneppen Model can be summarized by Algorithm 2.

Algorithm 2 Bak–Sneppen Model

1: Find the index j of the individual with the lowest fitness
2: Replace the fitness of individuals with indices j, $j-1$, and $j + 1$ by random values drawn from uniform distribution
end

This simple model can lead to interesting behavior. At the outset of the simulation, mean fitness of the population is low, but, as the number of generation increases, mean fitness increases, too. Eventually, mean fitness ceases to increase, and the critical state is reached. In the Bak–Sneppen Model, a substitution of the fitness of the worst individual causes the substitution of its two next neighbors. In the critical state, the values of fitness of the neighbors are very often replaced by random numbers with smaller values. The new worst individual can be then one of these two neighbors, which are replaced with its two next neighbors, originating a chain reaction, called replacement event in this work, that can affect all the individuals of the population. The replacement events exhibit scale invariance and their statistical distributions are given by power laws in the form of Eqs. 1 and 2. Large replacement events generally occur if almost all individuals of the population have similar high values of fitness.

It is important to observe that SOC avoids the situation where a species gets trapped in a local optimum in the fitness landscape. Because the idea is powerful and simple, researchers proposed the use of SOC in optimization processes. Boettcher and Percus [3] proposed optimization with extremal dynamics, a local-search heuristic to find solutions in problems where constituents of the system are connected, e.g., spin glass optimization problem. Løvbjerg and Krink [18] extended particle swarm optimization with SOC in order to better control the optimization process and to maintain the diversity level.

In GAs, Krink and Thomsen [17] proposed the use of the sand pile model previously discussed to generate power laws to determine which individuals, placed on a grid, should be replaced in each generation. If an individual goes extinct, a mutated version of the best individual of the population is created in its place. It is important to observe that, however, in the algorithm proposed in Ref. [17] SOC appears in the sand pile model used to control the size of the extinctions, and is not the result of a self-organization of constituents of that system (individuals of the GA).

4 Self-organizing random immigrants genetic algorithm

In this paper, we propose the replacement of the individual with the lowest fitness of the current population and other r_r-1 individuals with new randomly generated individuals. The indices of individuals in the population are used to determine which individual will be replaced. In each generation of the algorithm, the individual with the lowest fitness in the current population (index j), $\lceil (r_r - 1)/2 \rceil$ individuals with indices from $j - \lceil (r_r - 1)/2 \rceil$ to $j-1$, and $\lfloor (r_r - 1)/2 \rfloor$ individuals with indices from $j + 1$ to $j + \lfloor (r_r - 1)/2 \rfloor$ are replaced by randomly generated individuals. It can be observed that, as the proposed GA is not spatially distributed, the indices of the individuals inside the population are random.

It must be said that this simple idea alone does not guarantee that the system exhibits SOC since new random immigrants added to the population generally have low fitness and are very often substituted by individuals with high fitness present in the population during the selection phase. As a consequence, the statistical distribution describing the response of the system to a single replacement will not be a power law, but a narrow distribution characterized by a small averaged value.

In the newly proposed variant of RIGA, a second strategy is adopted, by which the new immigrants created during the current chain reaction (called replacement event in this paper), are preserved in a subpopulation. The current size of this subpopulation is not defined by the programmer, but is the number of individuals created in the current replacement event. In other words, the individuals in the current population not belonging to the subpopulation are not allowed to replace individuals of the subpopulation. The subpopulation is allowed to evolve, i.e., is submitted to selection, mutation, and crossover among individuals that belong to the subpopulation.

The hope is that the system can now exhibit SOC in order to increase the diversity level of the population in a self-organized way and, therefore, to avoid a situation where individuals get trapped in local optima of the fitness landscape if the problem changes.

In the proposed GA, called *Self-Organizing Random Immigrants GA* (SORIGA), there are two major modifications from the RIGA. In the first modification, the function ''replaceFractionPopulation($\mathbf{P}$, r_r)'' is modified as presented in Algorithm 3. In this function, the individual with the lowest fitness and other r_r-1 individuals are replaced by randomly generated individuals. The current size (or duration) of each replacement event, i.e., the number of times that the individual with the lowest fitness and other r_r-1 individuals are replaced in the current replacement event, is recorded and denoted by d. Each individual of the population has an index. Each element k of the binary vector **P.replaced** indicates whether the individual with index k has been replaced for at least once in the current replacement event. For example, $\mathbf{P}.replaced[k] = 1$ means individual k has been replaced in the current replacement event at least once and

Algorithm 3 replaceFractionPopulation(**P**,r_r)

```
1: Find the index j of the individual with the lowest fitness
2: if (P.replaced[j] = 0) then
3:    d ← 1
4:    for k ← 1 to P.size do
5:       P.replaced(k) ← 0
6:    end for
7: else
8:    d ← d + 1
9: end if
10: for k ← 1 to P.size do
11:    if (j − ⌈(r_r − 1)/2⌉ ≤ k ≤ j + ⌊(r_r − 1)/2⌋) then
12:       Replace P.individual[k] by a randomly generated individual
13:       P.replaced[k] ← 1
14:    end if
15: end for
end
```

hence it belongs to the current subpopulation. The number of individuals in the current subpopulation equals the number of elements in **P.replaced** that are set to 1. When the chain reaction ceases, i.e., the individual with the lowest fitness does not belong to the subpopulation, the duration of the replacement event d is reset to 1.

The second modification in SORIGA lies in the selection of each individual into the new population, as shown in Algorithm 4. For each new individual, if its index was not involved in the current replacement event, i.e., **P**.*replaced*[k] = 0, the new individual is selected from the main population that consists of individuals with index k satisfying **P**.*replaced*[k] = 0 using a standard selection scheme; otherwise, it is selected from the subpopulation that consists of individuals with index k satisfying **P**.*replaced*[k] = 1 using a standard selection scheme.

Algorithm 4 selection(**P**,i)

```
1: if (P.replaced[i] = 0)
2:    Select an individual from the main population with individuals with index
k satisfying P.replaced[k] = 0
3: else
4:    Select an individual from the subpopulation with individuals with index k
satisfying P.replaced[k] = 1
5: end if
end
```

As an example, SORIGA is applied to a simple problem where the fitness function is defined as

$$f(\mathbf{x}) = \frac{u(\mathbf{x})}{l}, \tag{3}$$

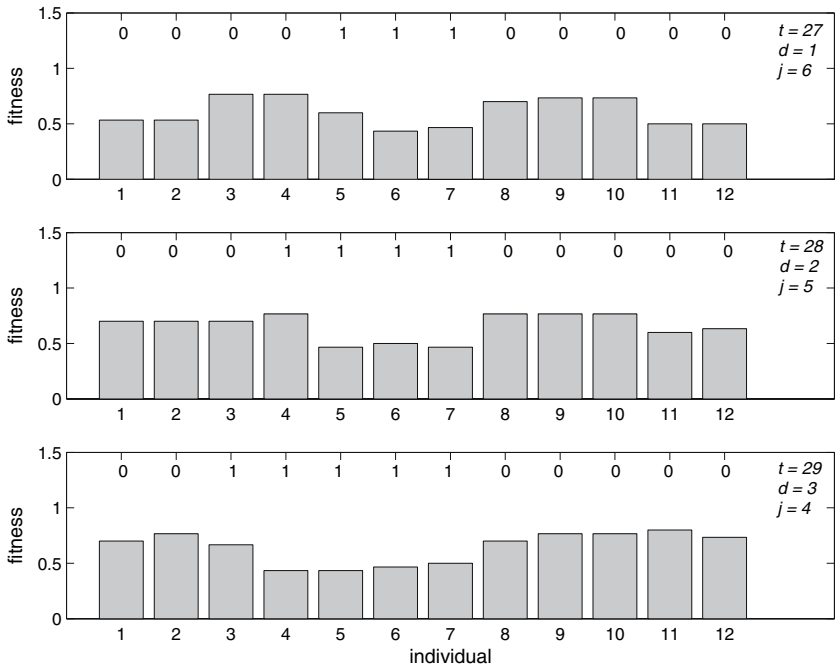

Fig. 1 Fitness of the individuals of the current population at generations 27, 28, and 29 in a run of the SORIGA on the example problem. The 0's and 1's indicates the values of each element of the vector **P.replaced**

where $u(\mathbf{x})$ is the unitation function of a binary vector (individual) $\mathbf{x}$ of length l, returning the number of ones in vector $\mathbf{x}$. In this example, randomly generated individuals of the first generation are selected according to elitism and roulette wheel sampling. Mutation with $p_m = 0.01$ and two-point crossover with $p_c = 0.6$ are employed. The number of individuals in the population is equal to 12 and $l = 30$. Figure 1 presents the first three steps of a replacement event in a run of SORIGA on this example with replacement rate $r_r = 3$. The figure shows the fitness of all individuals in the current population in generations 27, 28, and 29, respectively. In generation 27, the individual with index 6 (index j in Algorithm 3) has the lowest fitness in the population. In this way, this individual and the individuals with indices 5 and 7 are replaced by random individuals. In the next generation, the individual with index $j = 5$ has now the lowest fitness, and together with the individuals with indices 4 and 6 is replaced. In generation 29, the individual with index $j = 4$ has the lowest fitness. It can be observed that the chain reaction was propagated because the remaining individuals have fitness values higher than the individuals in the subpopulation defined by the individuals with indices k, where $\mathbf{P}.replaced[k] = 1$ (see Algorithms 3 and 4).

5 Experimental study

In order to evaluate the performance of SORIGA, five sets of experiments were carried out. In the experiments presented in this work, the dynamic problem generator proposed in [24, 26] is employed to construct DOPs based on five stationary test problems. The dynamic problem generator is presented in Sect. 5.1 while the five stationary test problems used for this purpose are described in Sect. 5.2.

In all experiments presented here, SORIGA is compared to SGA, and two versions of the GA with random immigrants. In the first version, denoted *RIGA1*, r_r individuals randomly chosen are replaced by new random individuals. In the second version, denoted *RIGA2*, the r_r worst individuals, i.e., the individuals with the lowest fitness, are replaced by new random individuals. The experimental design is described in Sect. 5.3, and the results are presented and analyzed, respectively, in Sects. 5.4 and 5.5. The results of including a neighboring scheme to SORIGA are presented in Sect. 5.6.

5.1 Dynamic problem generator

In order to evaluate the performance of different GAs in DOPs, a dynamic problem generator that can generate DOPs from any binary encoded stationary problem was proposed in Refs. [24, 26]. Given a stationary problem where the fitness function is $f(\mathbf{x})$ and $\mathbf{x} \in \{0,1\}^l \in \{0,1\}^l$, the dynamics of an environment that is periodically changed every τ generations is formulated as follows:

$$f(\mathbf{x}, t) = f(\mathbf{x} \oplus \mathbf{M}(k)) \tag{4}$$

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:

$$\mathbf{M}(k) = \mathbf{M}(k-1) \oplus \mathbf{T}(k) \tag{5}$$

where $\mathbf{T}(k)$ 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 dynamic problem generator proposed in Refs. [24, 26] can be used to investigate the performance of different GAs in well studied benchmark optimization problems, e.g., the royal road function (see next section).

5.2 Stationary test problems

Five stationary problems are selected as test suite for the algorithms SGA, RIGA1, RIGA2, and SORIGA. The DOPs are constructed from these stationary problems using the dynamic problem generator described in Sect. 5.1.

5.2.1 Royal road function

Mitchell, Forrest, and Holland proposed a class of fitness landscapes, called royal road functions, to investigate the schema processing [19]. One of these functions, called royal road R1, is defined as:

$$f(\mathbf{x}) = \sum_{s=1}^{q} c_s \delta_s(\mathbf{x}) \tag{6}$$

where q is the number of schemata that are juxtaposed and summed together, $c_s = c$ if the schema s is present in the solution $\mathbf{x}$, and $c_s = 0$ otherwise. In this contribution, the royal road function is defined on a 64-bit string, and each schema is composed of 8 contiguous fixed bits. If all bits of $\mathbf{x}$ corresponding to the fixed bits of the schema s are equal to 1, $c = 8$ is added to the fitness function.

5.2.2 Deceptive functions

Trap functions can be used to create deceptive functions for GAs, i.e., function where there exist low-order schemata that, instead of combining to form high-order schemata, forms schemata resulting in a deceptive solution that is suboptimal [10, 11]. A trap function is defined as follows:

$$f(\mathbf{x}) = \begin{cases} \frac{a}{z}(z - u(\mathbf{x})), & \text{if } u(\mathbf{x}) \leq z \\ \frac{b}{l-z}(u(\mathbf{x}) - z), & \text{otherwise,} \end{cases} \tag{7}$$

where $u(\mathbf{x})$ is the unitation function of a binary vector $\mathbf{x}$ of length l, a is the local and possibly deceptive optimum, b is the global optimum, and z is the slope-change location which separates the attraction basin sizes of the two optima.

A trap function is deceptive on average if the ratio of the fitness of the local optimum to that of the global optimum is constrained by the following relation [9]:

$$\frac{a}{b} \geq \frac{2 - 1/(l-z)}{2 - 1/z} \tag{8}$$

The parameters a, b, and z determine the difficulty for GAs to find the global optimum b as opposed to the local optimum a [25]. Two deceptive functions based on trap functions are considered in this work. In the first deceptive function (deceptive function 1), 10-bit trap functions are employed with a set to 0.82 and z set to 8. In the second deceptive function (deceptive function 2), 50-bit trap functions are employed with a set to 0.80 and z set to 48. In both functions, b is set to 1.0.

5.2.3 Scaling problems

Deception is not the only element that can generate difficulty to a GA. The problem difficulty can also be caused by scaling. The scaling problem arises in functions that

consist of several schemata with different worth to the solution [11]. A scaling problem can be simulated using additively decomposable functions as follows:

$$f(\mathbf{x}) = \sum_{s=1}^{q} c_s f_s(\mathbf{x}_{I_s}) \tag{9}$$

where q is the number of schemata that are juxtaposed and summed together, I_s is the set of the fixed bit positions that form schema s, and c_s is the scaling factor for each subfunction f_s. The royal road function presented in this section is a scaling problem with uniform scaling, i.e., $c_s = c$.

Using Eqs. 7 and 9, it is possible to create different scaling problems based on trap functions by adjusting the parameters c_s. Two scaling problems based on trap functions and with exponential scaling ($c_s = 2^{s-1}$) are considered in this work. In the first scaling problem (scaling problem 1), the fitness function is computed by juxtaposing and summing together four 5-bit trap functions. In the second scaling problem (scaling problem 2), the fitness function is computed by juxtaposing and summing together ten 5-bit trap functions. In both problems, b is set to 1.0, a is set to 0.7, and z is set to 3.

5.3 Experimental design

In the dynamic problem generator described in Sect. 5.1, the fitness function is periodically changed every τ generations according to Eqs. 4 and 5. The algorithm capability of adapting to dynamic environments under different degree of convergence can be investigated by setting τ to different values. Based on our preliminary experiments on stationary problems, τ is set to three different values. The first two values, $\tau = 10$ and $\tau = 200$, imply a change in the fitness function in an early and a medium stage of the optimization process, respectively. The last value, $\tau = 1000$ implies a change in the fitness function at a late or converged stage of the optimization process. Each algorithm was executed for 10 periods of environmental changes. The degree of change in the dynamic problem generator is controlled by setting parameter ρ. Three different values of ρ were used here in the experiments. These values represent different change levels: very light shifting ($\rho = 0.05$), medium variation ($\rho = 0.6$), and very high change ($\rho = 0.95$). In the experiments with deceptive problem 1, as $\lfloor \rho \times l \rfloor = 0$ when $l = 10$ and $\rho = 0.05$, the values of ρ are set to 0.10, 0.60, and 0.90.

In order to compare different GAs, each algorithm was executed 30 times (with 30 random seeds) for each one of the test problems described in the last section and with each one of the nine combinations of the environmental dynamics parameters τ and ρ. For each run of an algorithm for a DOP, the individuals of the initial population were randomly chosen. In each generation, two individuals of the population were selected according to elitism and the remaining individuals were selected according to roulette wheel sampling (Sects. 5.4.1 and 5.4.3) or tournament selection (Sect. 5.4.2). Traditional bit 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 120 individuals. Three replacement rates are considered for the algorithms with random

immigrants in order to compare the performance of the algorithms when the number of replaced individuals changes. The first value of r_r, whose results are discussed in Sects. 5.4.1 and 5.4.2, is set to 3, i.e., 3 individuals (2.5% of the population) are replaced by random immigrants in each generation. The second and third values of r_r with results presented in Sect. 5.4.3 are set to 12 and 24, respectively, i.e., 12 individuals or 24 (10 or 20% of the population) are replaced by random immigrants in each generation.

The comparison of results obtained by different algorithms on DOPs is more complex than the same comparison on stationary problems [22]. For DOPs, it is necessary to evaluate not the final result, but rather the optimization process itself. The mean best-of-generation fitness was used to evaluate the GAs.

5.4 Experimental results

Experiments were carried out on the dynamic version of the test problems described in Sect. 5.2 in order to help analyze the performance of the algorithms on dynamic problems. Results are presented in the following sections for different selection methods and different values of the replacement rates.

5.4.1 Selection with roulette wheel sampling

Selection with roulette wheel sampling is a fitness-proportionate selection, where the expected number of times an individual is selected to reproduce is equal to its fitness divided by the mean population fitness [19]. The experimental results on DOPs averaged over 30 runs with roulette wheel sampling and $r_r = 3$ for the GAs with random immigrants are shown in Tables 1 and 2. Experimental results of mean best-of-generation fitness and some statistical test results over this measure are presented in Table 1. Experimental results of mean population fitness are presented in Table 2. In Table 1, the statistical comparison regarding SORIGA-SGA, SORIGA-RIGA1, and SORIGA-RIGA2 is carried out by t-test with 58 degrees of freedom at a 0.05 level of significance regarding the mean best-of-generation fitness. The t-test results are shown in the parentheses as ''+'', ''−'', or ''∼'' when SORIGA is significantly better than, significantly worse than, or statistically equivalent to other GAs, respectively. Figures 2–11 show the results of mean best-of-generation fitness in experiments where τ is set to 10 or 1,000 and ρ is set to 0.05 or 0.95 (0.10 or 0.90 in the experiments with deceptive function 1), respectively.

5.4.2 Selection with tournament

Tournament selection is a selection procedure computationally cheaper than fitness-proportionate selection. In the simplest version of tournament selection, two individuals of the current population are randomly chosen and the individual with higher fitness is selected to be reproduced if a random number generated with uniform distribution in [0, 1] is smaller than a parameter k_{ts} [19]. Otherwise, the individual with the lower fitness is selected. The parameter k_{ts} is chosen in [0, 1] and can be used to control the selection pressure. Higher values of k_{ts} imply higher

Table 1 Experimental results of the mean best-of-generation fitness (selection with roulette wheel sampling) and relevant statistical comparisons (inside the parentheses)

Problem	Dynamics		Algorithm			
	τ	ρ	SGA	RIGA1	RIGA2	SORIGA
Royal road function	10	0.05	30.60 (∼)	30.03 (∼)	30.80 (∼)	30.94
	10	0.60	6.62 (+)	8.47 (+)	10.94 (∼)	11.47
	10	0.95	12.92 (+)	14.44 (∼)	14.66 (∼)	15.08
	200	0.05	59.97 (∼)	59.69 (∼)	59.91 (∼)	59.79
	200	0.60	29.38 (+)	37.72 (+)	38.94 (+)	41.10
	200	0.95	24.08 (+)	36.10 (+)	38.06 (+)	40.33
	1,000	0.05	63.10 (∼)	63.10 (∼)	63.26 (−)	63.10
	1,000	0.60	53.24 (+)	57.07 (+)	57.43 (+)	57.78
	1,000	0.95	49.70 (+)	57.09 (+)	57.16 (+)	57.75
Deceptive function 1	10	0.10	0.8276 (∼)	0.8328 (∼)	0.8292 (∼)	0.8322
	10	0.60	0.7601 (+)	0.7895 (∼)	0.7829 (+)	0.7966
	10	0.90	0.8418 (+)	0.8711 (∼)	0.8619 (∼)	0.8692
	200	0.10	0.8408 (+)	0.8959 (∼)	0.8912 (+)	0.9127
	200	0.60	0.8193 (+)	0.8681 (+)	0.8708 (+)	0.8841
	200	0.90	0.9092 (+)	0.9314 (∼)	0.9267 (+)	0.9339
	1,000	0.10	0.8422 (+)	0.9685 (+)	0.9669 (+)	0.9838
	1,000	0.60	0.8250 (+)	0.9448 (+)	0.9463 (+)	0.9550
	1,000	0.90	0.9121 (+)	0.9697 (∼)	0.9703 (∼)	0.9743
Deceptive function 2	10	0.05	0.6947 (−)	0.6938 (∼)	0.7000 (−)	0.6894
	10	0.60	0.5565 (+)	0.5676 (∼)	0.5654 (+)	0.5682
	10	0.95	0.5401 (+)	0.5511 (∼)	0.5463 (+)	0.5521
	200	0.05	0.7931 (−)	0.7928 (−)	0.7931 (−)	0.7914
	200	0.60	0.7496 (+)	0.7582 (−)	0.7597 (−)	0.7572
	200	0.95	0.7306 (+)	0.7580 (∼)	0.7597 (−)	0.7572
	1,000	0.05	0.7985 (−)	0.7985 (−)	0.7987 (−)	0.7983
	1,000	0.60	0.7900 (+)	0.7917 (−)	0.7919 (−)	0.7915
	1,000	0.95	0.7867 (+)	0.7916 (∼)	0.7920 (−)	0.7917
Scaling function 1	10	0.05	0.9673 (∼)	0.9722 (−)	0.9635 (∼)	0.9578
	10	0.60	0.7705 (+)	0.8084 (+)	0.8121 (+)	0.8204
	10	0.95	0.8268 (+)	0.8475 (∼)	0.8423 (+)	0.8511
	200	0.05	0.9958 (∼)	0.9985 (∼)	0.9973 (∼)	0.9983
	200	0.60	0.8760 (+)	0.9496 (+)	0.9509 (+)	0.9670
	200	0.95	0.8546 (+)	0.9251 (+)	0.9226 (+)	0.9496
	1,000	0.05	0.9994 (∼)	0.9997 (∼)	0.9995 (∼)	0.9997
	1,000	0.60	0.8917 (+)	0.9870 (+)	0.9857 (+)	0.9932
	1,000	0.95	0.8706 (+)	0.9797 (+)	0.9766 (+)	0.9880

Table 1 continued

Problem	Dynamics		Algorithm			
	τ	ρ	SGA	RIGA1	RIGA2	SORIGA
Scaling function 2	10	0.05	0.9327 (−)	0.9222 (∼)	0.9298 (−)	0.9151
	10	0.60	0.7380 (+)	0.7633 (+)	0.7622 (+)	0.7723
	10	0.95	0.8217 (+)	0.8328 (∼)	0.8364 (∼)	0.8413
	200	0.05	0.9898 (∼)	0.9917 (∼)	0.9912 (∼)	0.9926
	200	0.60	0.8665 (+)	0.9365 (+)	0.9350 (+)	0.9480
	200	0.95	0.8517 (+)	0.9113 (+)	0.9068 (+)	0.9288
	1,000	0.05	0.9963 (∼)	0.9980 (∼)	0.9978 (∼)	0.9979
	1,000	0.60	0.8860 (+)	0.9747 (+)	0.9743 (+)	0.9828
	1,000	0.95	0.8633 (+)	0.9616 (+)	0.9588 (+)	0.9770

levels of selection pressure. Experimental results on DOPs averaged over 30 runs and with two values of k_{ts} are presented in this section in order to compare the performance of the GAs when the selection pressure changes. Results of mean best-of-generation fitness when $k_{ts} = 0.70$ or $k_{ts} = 0.90$ and some statistical test results over this measure are presented in Tables 3 and 4. The statistical comparison regarding SORIGA-SGA, SORIGA-RIGA1, and SORIGA-RIGA2 is carried out by *t*-test with 58 degrees of freedom at a 0.05 level of significance and the *t*-test results are shown in Tables 3 and 4 the same way as in Table 1. In all experiments, r_r is set to 3 for the GAs with random immigrants.

5.4.3 Selection with roulette wheel sampling and with $r_r = 12$ or $r_r = 24$

The results on DOPs averaged over 30 runs for the experiments where roulette wheel sampling selection is employed and r_r is set to 12 or 24 for the GAs with random immigrants are presented in Tables 5 and 6, respectively. Similarly, the statistical comparison is carried out between SORIGA and other GAs and the *t*-test results are shown in Tables 5 and 6 in the same way as in Table 1.

5.5 Analysis of the results

Several results can be observed by analyzing the experimental results. The performance of SORIGA is generally significantly better than the performance of SGA on the DOPs tested in this paper. The performance of SORIGA increases with the period of change τ and with the degree of change ρ for the DOPs. It can be observed that the performance of SORIGA is generally better for $\rho > 0.10$ (medium and high degree of changes). These results can be explained by the fact that SGA had difficulty in escaping from the local optima induced by medium or high changes of the global optima on the DOPs tested. However, random immigrants inserted in every generation provide diversity to the last three GAs and prepare the population for eventual change, which explains the better results of these GAs with random immigrants when compared to SGA. For example, this behavior can be observed in

Table 2 Experimental results of the mean population fitness (selection with roulette wheel sampling)

Problem	Dynamics		Algorithm			
	τ	ρ	SGA	RIGA1	RIGA2	SORIGA
Royal road function	10	0.05	18.41	18.06	18.82	17.87
	10	0.60	4.48	5.55	7.17	7.00
	10	0.95	8.10	9.06	9.12	8.92
	200	0.05	38.91	38.82	39.40	35.88
	200	0.60	20.09	25.68	26.72	25.86
	200	0.95	16.74	24.72	26.15	25.46
	1,000	0.05	40.95	40.99	41.60	37.72
	1,000	0.60	34.97	37.45	38.14	34.92
	1,000	0.95	32.68	37.46	37.99	34.92
Deceptive function 1	10	0.10	0.6606	0.6454	0.6577	0.6305
	10	0.60	0.4845	0.4935	0.4924	0.4947
	10	0.90	0.5709	0.5652	0.5754	0.5639
	200	0.10	0.7630	0.7296	0.7475	0.7228
	200	0.60	0.7342	0.7170	0.7293	0.6952
	200	0.90	0.8168	0.7763	0.7880	0.7391
	1,000	0.10	0.7683	0.7600	0.7673	0.7903
	1,000	0.60	0.7523	0.7075	0.7158	0.6939
	1,000	0.90	0.8261	0.7587	0.7693	0.7308
Deceptive function 2	10	0.05	0.5663	0.5572	0.5710	0.5386
	10	0.60	0.4202	0.4223	0.4272	0.4235
	10	0.95	0.3985	0.4016	0.4032	0.4037
	200	0.05	0.6718	0.6554	0.6730	0.6303
	200	0.60	0.6262	0.6182	0.6351	0.5985
	200	0.95	0.6055	0.6165	0.6332	0.5974
	1,000	0.05	0.6778	0.6607	0.6790	0.6366
	1,000	0.60	0.6689	0.6534	0.6714	0.6300
	1,000	0.95	0.6647	0.6532	0.6709	0.6301
Scaling function 1	10	0.05	0.7329	0.7277	0.7361	0.7008
	10	0.60	0.4777	0.4829	0.4884	0.4831
	10	0.95	0.5880	0.5761	0.5862	0.5596
	200	0.05	0.8569	0.8467	0.8599	0.8167
	200	0.60	0.7436	0.7852	0.8006	0.7734
	200	0.95	0.7357	0.7670	0.7766	0.7596
	1,000	0.05	0.8650	0.8523	0.8671	0.8225
	1,000	0.60	0.7701	0.8376	0.8508	0.8136
	1,000	0.95	0.7545	0.8313	0.8428	0.8091

Table 2 continued

Problem	Dynamics		Algorithm			
	τ	ρ	SGA	RIGA1	RIGA2	SORIGA
Scaling function 2	10	0.05	0.7210	0.7024	0.7185	0.6774
	10	0.60	0.4738	0.4766	0.4826	0.4763
	10	0.95	0.6086	0.5979	0.6017	0.5844
	200	0.05	0.8380	0.8294	0.8413	0.8032
	200	0.60	0.7275	0.7724	0.7816	0.7584
	200	0.95	0.7243	0.7575	0.7646	0.7467
	1,000	0.05	0.8459	0.8373	0.8500	0.8099
	1,000	0.60	0.7534	0.8173	0.8292	0.7973
	1,000	0.95	0.7373	0.8084	0.8183	0.7938

Figs. 3, 5, 7, 9, and 11, where the best-of-generation fitness of all GAs on the five DOPs with τ = 1,000 are plotted.

Experimental results on the tested DOPs with $\rho > 0.10$ agree with those presented in Ref. [8], where the performance of RIGA, Hypermutation GA, and SGA were compared in DOPs constructed from changing landscapes produced by hills that are shaped using mathematical functions. Three types of environmental changes were considered: linear translation of the hills, periodic changes in the location of the maximum hill, and oscillating changes between two landscapes. In the experiments [8], the performance of RIGA was better than the performance of Hypermutation

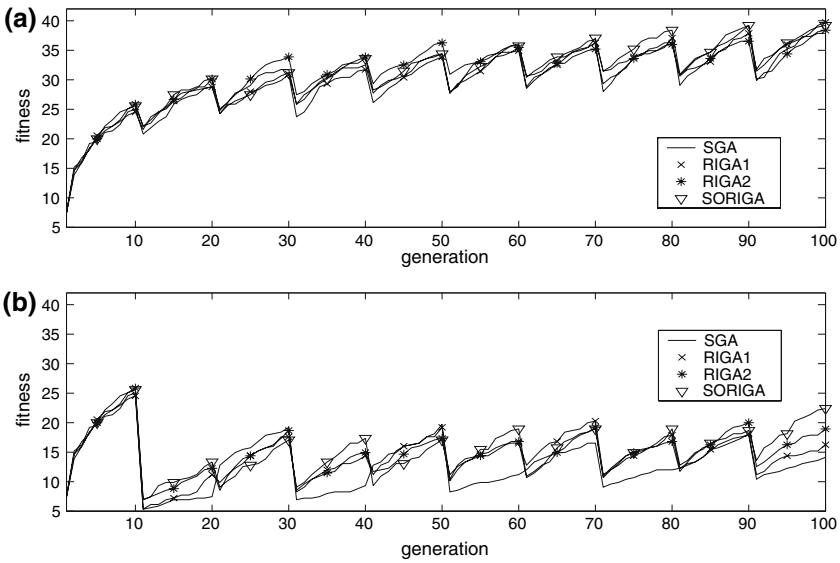

Fig. 2 Best-of-generation fitness of algorithms on dynamic royal road function where the environmental dynamics parameter τ is set to 10 and: (**a**) ρ is set to 0.05, (**b**) ρ is set to 0.95

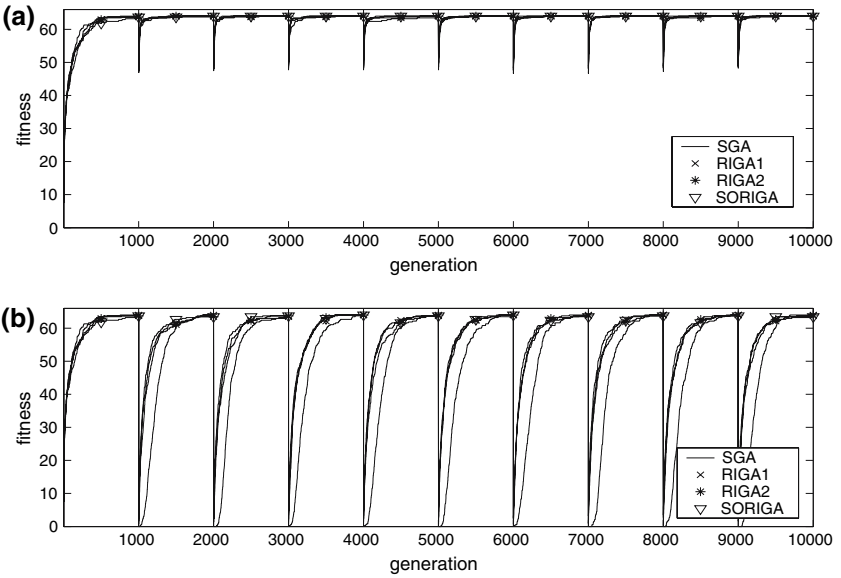

Fig. 3 Best-of-generation fitness of algorithms on dynamic royal road function where the environmental dynamics parameter τ is set to 1,000 and: (**a**) ρ is set to 0.05, (**b**) ρ is set to 0.95

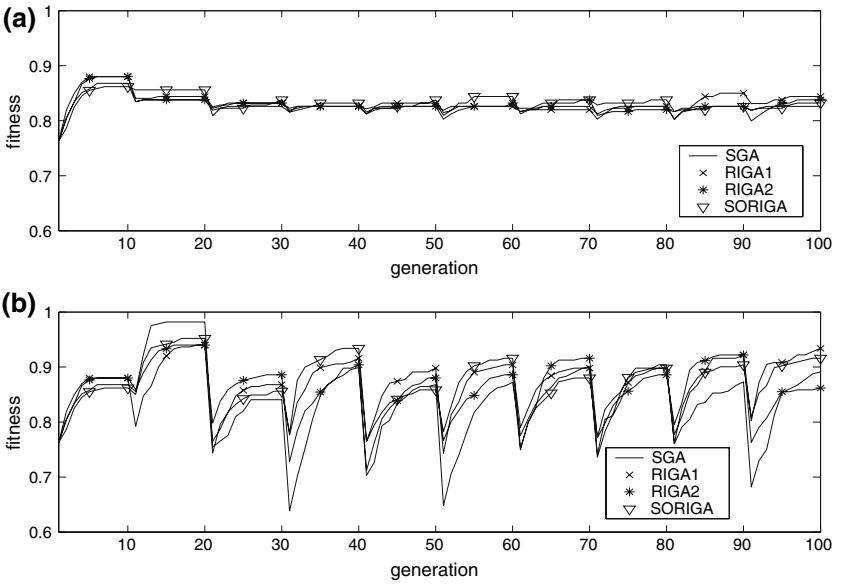

Fig. 4 Best-of-generation fitness of algorithms on dynamic deceptive function 1 where the environmental dynamics parameter τ is set to 10 and: (**a**) ρ is set to 0.10, (**b**) ρ is set to 0.90

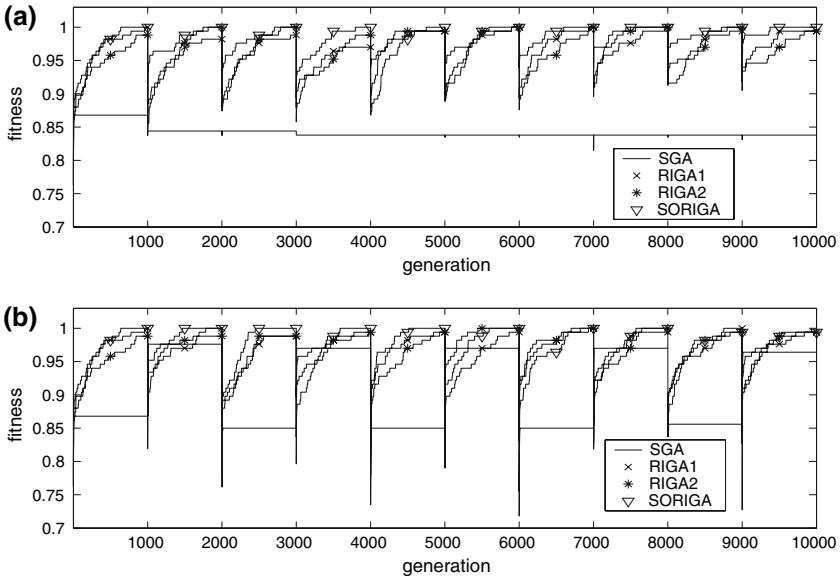

Fig. 5 Best-of-generation fitness of algorithms on dynamic deceptive function 1 where the environmental dynamics parameter τ is set to 1,000 and: (**a**) ρ is set to 0.10, (**b**) ρ is set to 0.90

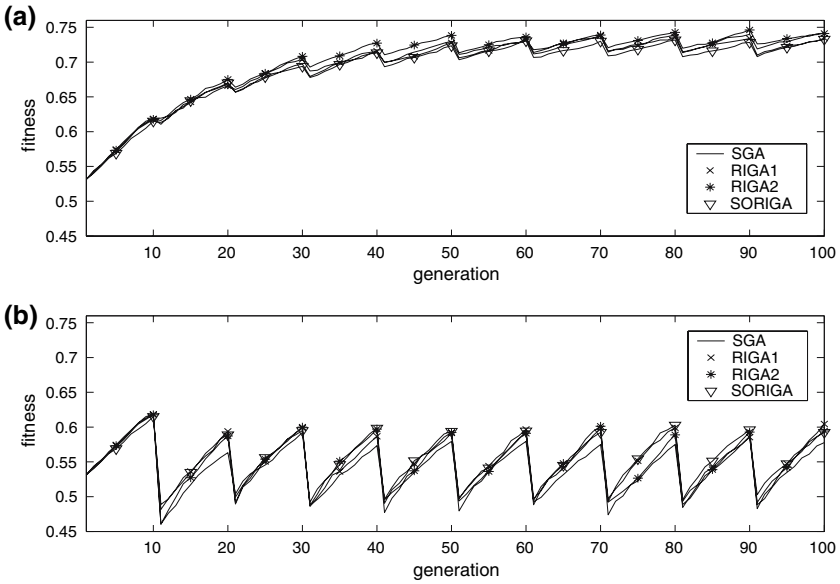

Fig. 6 Best-of-generation fitness of algorithms on dynamic deceptive function 2 where the environmental dynamics parameter τ is set to 10 and: (**a**) ρ is set to 0.05, (**b**) ρ is set to 0.95

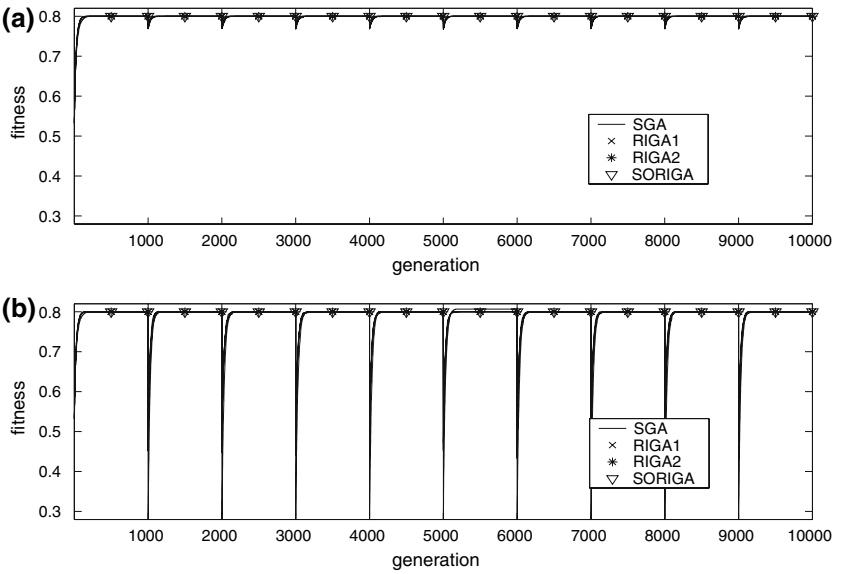

Fig. 7 Best-of-generation fitness of algorithms on dynamic deceptive function 2 where the environmental dynamics parameter τ is set to 1,000 and: (**a**) ρ is set to 0.05, (**b**) ρ is set to 0.95

GA and SGA in those DOPs with severe environmental changes. Such results were explained by the fact that RIGA prepares the population well for possible catastrophic changes through an increase in the genetic diversity. However, the

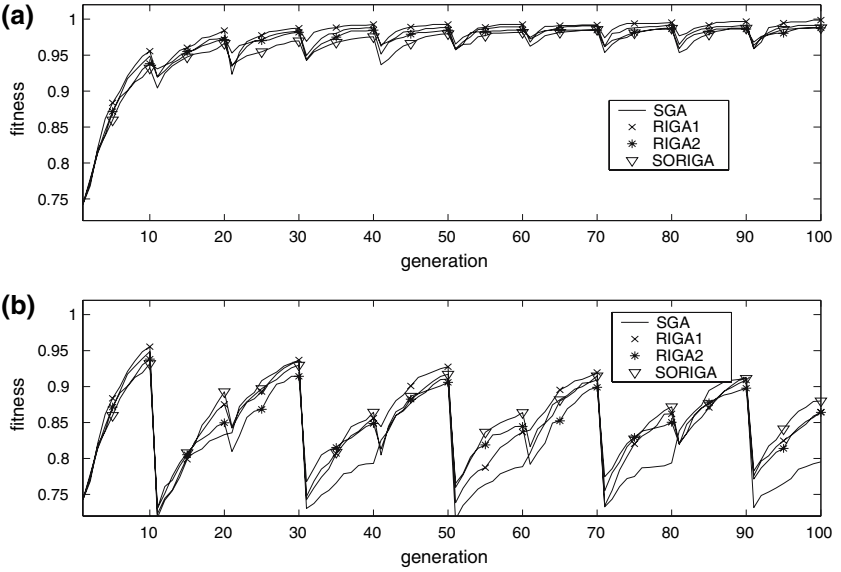

Fig. 8 Best-of-generation fitness of algorithms on dynamic Scaling Problem 1 where the environmental dynamics parameter τ is set to 10 and: (**a**) ρ is set to 0.05, (**b**) ρ is set to 0.95

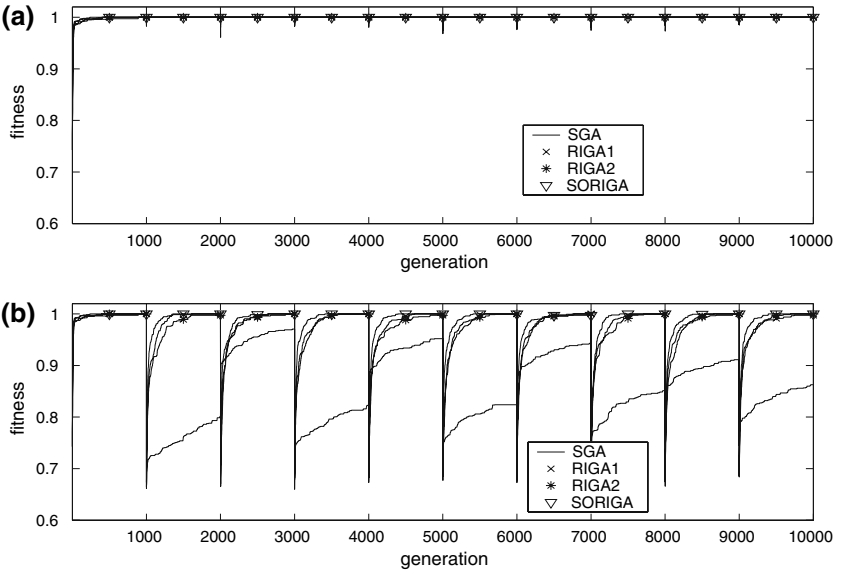

Fig. 9 Best-of-generation fitness of algorithms on dynamic Scaling Problem 1 where the environmental dynamics parameter τ is set to 1,000 and: (**a**) ρ is set to 0.05, (**b**) ρ is set to 0.95

performance of RIGA was worse in DOPs with small environmental changes, explained by an increase in the probability of losing information that may match such changes. Here, the worse performance of GAs with random immigrants occurs again on DOPs with slight environmental changes ($\rho \leq 0.1$).

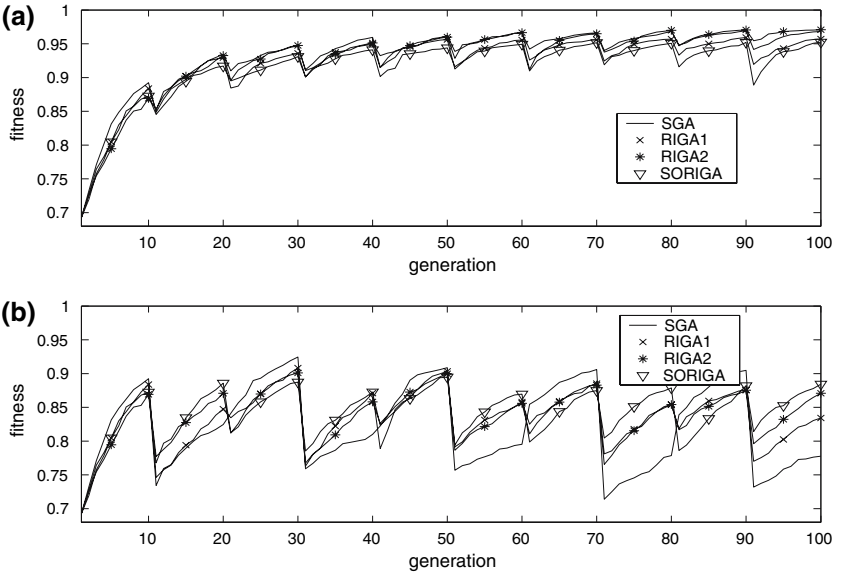

Fig. 10 Best-of-generation fitness of algorithms on dynamic Scaling Problem 2 where the environmental dynamics parameter τ is set to 10 and: (**a**) ρ is set to 0.05, (**b**) ρ is set to 0.95

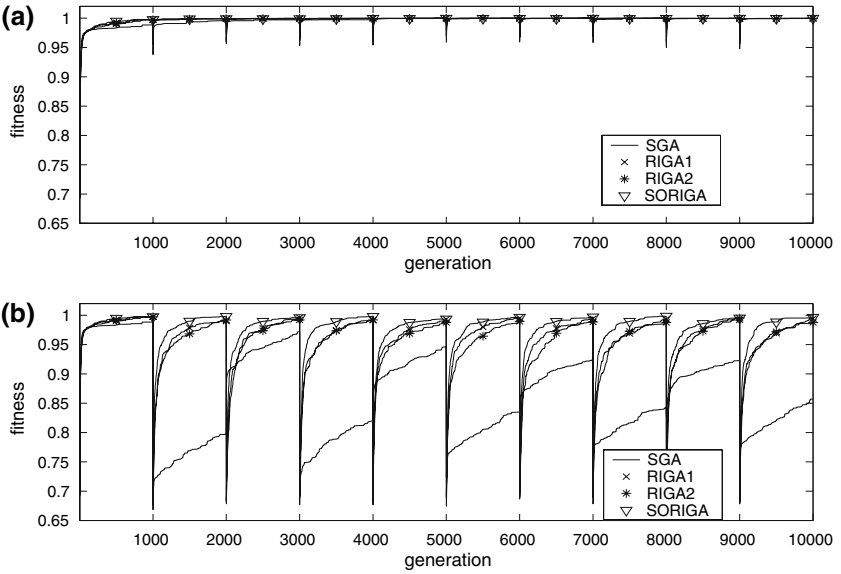

Fig. 11 Best-of-generation fitness of algorithms on dynamic Scaling Problem 2 where the environmental dynamics parameter τ is set to 1,000 and: (**a**) ρ is set to 0.05, (**b**) ρ is set to 0.95

Let us now analyze the results of the three GAs with random immigrants. First, let us investigate how the proposed SORIGA works. In the beginning of the experiments, the individuals of the initial populations and the immigrants generally have low fitness. Since several individuals in the population have low fitness, the probability that one of the replaced individuals becomes the new worst is low. As a consequence, a single replacement of an individual generally does not generate large chain reactions of replacements. As the number of generations increases, the mean fitness increases too, and several individuals of the current population have fitness values higher than the average fitness of the new random individuals. Then, the probability that one of the individuals with indices close to the index of the replaced worst individual becomes the new worst individual increases, and a chain reaction can be developed with a large variety of sizes.

While the performance of SORIGA was generally worse than the performance of other RIGAs on the deceptive function 2, it was generally significantly better on other DOPs for values of $\rho > 0.10$. The better results of SORIGA over other GAs with random immigrants in the experiments presented here can be mainly explained by the higher values of the diversity level of the population in SORIGA. In Fig. 12, the mean Hamming Distance averaged over 30 runs in the initial 2,000 generations of the dynamic Scaling Problem 2 where the environmental dynamics parameter τ is set to 1,000 and ρ is set to 0.95 is presented. The mean Hamming Distance of the population at generation t is computed as

$$h(t) = \frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N}\left(\mathbf{x}_i(t) \oplus \mathbf{x}_j(t)\right) = \frac{2}{N^2}\sum_{i=1}^{N-1}\sum_{j=i+1}^{N}\left(\mathbf{x}_i(t) \oplus \mathbf{x}_j(t)\right) \qquad (10)$$

Table 3 Experimental results of the mean best-of-generation fitness (tournament selection with $k_{ts} = 0.70$) and relevant statistical comparisons (inside the parentheses)

Problem	Dynamics		Algorithm			
	τ	ρ	SGA	RIGA1	RIGA2	SORIGA
Royal road function	10	0.05	24.14 (−)	23.13 (∼)	25.37 (−)	22.58
	10	0.60	10.57 (∼)	10.98 (∼)	11.22 (∼)	10.91
	10	0.95	12.77 (∼)	12.47 (∼)	12.70 (∼)	12.53
	200	0.05	58.24 (−)	55.93 (−)	57.85 (−)	52.83
	200	0.60	33.73 (+)	36.53 (∼)	37.34 (∼)	36.93
	200	0.95	28.50 (+)	34.49 (+)	35.38 (∼)	35.33
	1,000	0.05	62.61 (−)	62.08 (−)	62.54 (−)	60.70
	1,000	0.60	54.57 (∼)	55.03 (−)	56.01 (−)	54.36
	1,000	0.95	51.92 (+)	54.31 (−)	55.46 (−)	53.71
Deceptive function 1	10	0.10	0.8222 (∼)	0.8269 (∼)	0.8280 (∼)	0.8252
	10	0.60	0.7586 (+)	0.7851 (+)	0.7819 (+)	0.7959
	10	0.90	0.8501 (+)	0.8676 (+)	0.8712 (+)	0.8881
	200	0.10	0.8258 (+)	0.8761 (+)	0.8761 (+)	0.8931
	200	0.60	0.8186 (+)	0.8813 (+)	0.8701 (+)	0.8972
	200	0.90	0.9112 (+)	0.9484 (∼)	0.9529 (∼)	0.9546
	1,000	0.10	0.8278 (+)	0.9545 (+)	0.9493 (+)	0.9703
	1,000	0.60	0.8261 (+)	0.9510 (+)	0.9477 (+)	0.9649
	1,000	0.90	0.9236 (+)	0.9802 (∼)	0.9748 (+)	0.9833
Deceptive function 2	10	0.05	0.7331 (−)	0.7244 (∼)	0.7299 (−)	0.7221
	10	0.60	0.5558 (+)	0.5667 (+)	0.5671 (+)	0.5703
	10	0.95	0.5288 (+)	0.5466 (+)	0.5434 (+)	0.5517
	200	0.05	0.7971 (−)	0.7967 (−)	0.7971 (−)	0.7962
	200	0.60	0.7588 (+)	0.7690 (+)	0.7711 (−)	0.7703
	200	0.95	0.7433 (+)	0.7658 (+)	0.7686 (∼)	0.7671
	1,000	0.05	0.7994 (−)	0.7993 (−)	0.7994 (−)	0.7992
	1,000	0.60	0.7919 (+)	0.7938 (+)	0.7942 (−)	0.7941
	1,000	0.95	0.7894 (+)	0.7931 (∼)	0.7948 (∼)	0.7942
Scaling function 1	10	0.05	0.9495 (∼)	0.9593 (∼)	0.9579 (∼)	0.9558
	10	0.60	0.7794 (+)	0.8156 (∼)	0.8055 (+)	0.8158
	10	0.95	0.8333 (+)	0.8525 (∼)	0.8488 (∼)	0.8503
	200	0.05	0.9809 (+)	0.9964 (∼)	0.9963 (∼)	0.9977
	200	0.60	0.8604 (+)	0.9513 (+)	0.9424 (+)	0.9629
	200	0.95	0.8530 (+)	0.9222 (+)	0.9160 (+)	0.9453
	1,000	0.05	0.9874 (+)	0.9993 (∼)	0.9992 (∼)	0.9995
	1,000	0.60	0.8782 (+)	0.9855 (+)	0.9820 (+)	0.9923
	1,000	0.95	0.8554 (+)	0.9759 (+)	0.9689 (+)	0.9873

Table 3 continued

Problem	Dynamics		Algorithm			
	τ	ρ	SGA	RIGA1	RIGA2	SORIGA
Scaling function 2	10	0.05	0.9229 (∼)	0.9256 (∼)	0.9283 (∼)	0.9291
	10	0.60	0.7317 (+)	0.7642 (+)	0.7515 (+)	0.7720
	10	0.95	0.8299 (∼)	0.8381 (∼)	0.8333 (∼)	0.8353
	200	0.05	0.9783 (+)	0.9904 (∼)	0.9940 (∼)	0.9930
	200	0.60	0.8593 (+)	0.9334 (+)	0.9275 (+)	0.9436
	200	0.95	0.8498 (+)	0.9065 (+)	0.9047 (+)	0.9249
	1,000	0.05	0.9860 (+)	0.9982 (∼)	0.9987 (∼)	0.9987
	1,000	0.60	0.8697 (+)	0.9715 (+)	0.9677 (+)	0.9819
	1,000	0.95	0.8527 (+)	0.9601 (+)	0.9500 (+)	0.9712

where $\mathbf{x}_i(t)$ is the chromosome of the i-th individual of the population at generation t and N is the population size. The mean Hamming Distance can be used to estimate the diversity level of the population. Higher values of the mean Hamming Distance imply in higher values of the diversity level of the population. One can observe in Fig. 12 that the mean Hamming Distance decreases after the initial generations and the changes in the problem toward a small mean value. One can still observe that the mean Hamming Distances are higher for the SORIGA, indicating that the diversity level is high for this algorithm when compared to the other GAs. This fact can be observed by analyzing the results presented in Table 2. SORIGA generally presented the lower values of the mean fitness of the population, even though its fitness values of the best-of-generation individuals were higher.

Another factor that explains the better results of SORIGA is that the survival probability of a new random individual, which can be evolved to become a solution of the problem, is generally lower in the standard GAs with random immigrants. This happens because the fitness values for the current individuals, whose locations are generally located in (or close to) local maxima after several generations, are generally much higher than the mean fitness of the search space (i.e., the mean fitness of all possible individuals). SORIGA preserves a new potential solution in a subpopulation and allows it to evolve while the current replacement event is in progress. As it is necessary to develop the potential solution inside the subpopulation, the performance of SORIGA increases with the period of change τ. One can observe in Tables 1–6 that the best results of SORIGA are those with $\tau = 1{,}000$.

Figure 13 shows the results of the mean population and best-of-generation fitness, mean Hamming Distance, and duration of replacement events in the first 2,000 generations of the second trial of the experiments of SORIGA with the roulette wheel sampling selection on dynamic Scaling Problem 2 with $\tau = 1{,}000$ and $\rho = 0.95$. One can observe that larger replacement events generally occurred when the mean population fitness was high and the diversity level of the population was small. Such interesting behavior was reached by self-organization, and not by a rule imposed by the programmer. In the experimental results on the last five DOPs, the

Table 4 Experimental results of the mean best-of-generation fitness (tournament selection with $k_{ts} = 0.9$) and relevant statistical comparisons (inside the parentheses)

Problem	Dynamics		Algorithm			
	τ	ρ	SGA	RIGA1	RIGA2	SORIGA
Royal road function	10	0.05	30.97 (∼)	28.93 (+)	31.39 (∼)	30.91
	10	0.60	10.91 (+)	11.96 (∼)	11.90 (∼)	12.05
	10	0.95	15.49 (∼)	15.26 (∼)	15.19 (∼)	15.44
	200	0.05	60.90 (∼)	60.96 (∼)	61.54 (−)	60.60
	200	0.60	35.89 (+)	40.69 (+)	41.04 (+)	42.72
	200	0.95	29.22 (+)	37.27 (+)	37.37 (+)	39.73
	1,000	0.05	63.35 (∼)	63.39 (∼)	63.54 (−)	63.34
	1,000	0.60	56.81 (+)	58.41 (∼)	58.73 (∼)	58.62
	1,000	0.95	54.10 (+)	57.47 (+)	57.69 (∼)	57.84
Deceptive function 1	10	0.10	0.8224 (∼)	0.8257 (∼)	0.8263 (∼)	0.8247
	10	0.60	0.7219 (+)	0.7621 (∼)	0.7508 (+)	0.7681
	10	0.90	0.8513 (+)	0.8665 (+)	0.8685 (+)	0.8857
	200	0.10	0.8252 (+)	0.8875 (∼)	0.8935 (∼)	0.9011
	200	0.60	0.8144 (+)	0.8703 (+)	0.8685 (+)	0.8808
	200	0.90	0.9083 (+)	0.9406 (∼)	0.9269 (+)	0.9471
	1,000	0.10	0.8314 (+)	0.9745 (∼)	0.9660 (+)	0.9746
	1,000	0.60	0.8234 (+)	0.9496 (+)	0.9436 (+)	0.9587
	1,000	0.90	0.9128 (+)	0.9731 (+)	0.9656 (+)	0.9795
Deceptive function 2	10	0.05	0.7609 (−)	0.7577 (∼)	0.7598 (−)	0.7564
	10	0.60	0.5521 (+)	0.5720 (+)	0.5729 (+)	0.5758
	10	0.95	0.5058 (+)	0.5456 (+)	0.5462 (+)	0.5544
	200	0.05	0.7982 (−)	0.7982 (∼)	0.7983 (−)	0.7981
	200	0.60	0.7682 (+)	0.7786 (+)	0.7791 (+)	0.7798
	200	0.95	0.7482 (+)	0.7779 (∼)	0.7798 (∼)	0.7807
	1,000	0.05	0.7996 (−)	0.7996 (∼)	0.7997 (−)	0.7996
	1,000	0.60	0.7936 (+)	0.7957 (+)	0.7959 (+)	0.7960
	1,000	0.95	0.7943 (∼)	0.7974 (∼)	0.7983 (∼)	0.7970
Scaling function 1	10	0.05	0.9665 (∼)	0.9683 (∼)	0.9685 (∼)	0.9735
	10	0.60	0.7594 (+)	0.7996 (+)	0.7942 (+)	0.8117
	10	0.95	0.8318 (+)	0.8508 (+)	0.8521 (+)	0.8613
	200	0.05	0.9826 (+)	0.9900 (+)	0.9876 (+)	0.9969
	200	0.60	0.8538 (+)	0.9386 (+)	0.9334 (+)	0.9542
	200	0.95	0.8497 (+)	0.9042 (+)	0.9011 (+)	0.9323
	1,000	0.05	0.9832 (+)	0.9982 (+)	0.9970 (+)	0.9994
	1,000	0.60	0.8666 (+)	0.9680 (+)	0.9641 (+)	0.9838
	1,000	0.95	0.8507 (+)	0.9487 (+)	0.9477 (+)	0.9702

Table 4 continued

Problem	Dynamics		Algorithm			
	τ	ρ	SGA	RIGA1	RIGA2	SORIGA
Scaling function 2	10	0.05	0.9402 (∼)	0.9480 (∼)	0.9432 (∼)	0.9515
	10	0.60	0.7240 (+)	0.7541 (+)	0.7520 (+)	0.7699
	10	0.95	0.8273 (+)	0.8351 (∼)	0.8340 (+)	0.8399
	200	0.05	0.9725 (+)	0.9818 (∼)	0.9789 (+)	0.9873
	200	0.60	0.8523 (+)	0.9230 (+)	0.9276 (+)	0.9383
	200	0.95	0.8485 (+)	0.8889 (+)	0.8833 (+)	0.9143
	1,000	0.05	0.9750 (+)	0.9925 (+)	0.9896 (+)	0.9962
	1,000	0.60	0.8652 (+)	0.9507 (+)	0.9476 (+)	0.9695
	1,000	0.95	0.8508 (+)	0.9314 (+)	0.9248 (+)	0.9514

size of the subpopulation self-organized according to the population diversity level. This result can be observed in Tables 3 and 4, where the performance of SORIGA in comparison to other RIGAs increases when the selection pressure increases by changing k_{ts} from 0.70 to 0.90 in the experiments with tournament selection. One can observe that the mean Hamming distances were lower for the experiment with $k_{ts} = 0.90$ (Fig. 12), indicating a low diversity in the population.

It is important to observe that the random immigrants mechanism incurs a cost on the GA as a part of the individuals that could be used to explore the current best solution are replaced by randomly generated individuals. This cost was depreciated by the advantage of increasing the diversity level in problems where there are difficulties in escaping from local optima induced by severe changes using only the traditional GA mechanisms. The balance between the advantage of increasing the diversity of the population by introducing randomly generated individuals and the cost of replacing part of the population that could be used to explore the current best solution should be found by properly setting the parameter r_r in the standard RIGA. In the experiments performed in this work, the performance of the RIGA1 and RIGA2 was generally better when $r_r = 24$ (Table 6), which corresponded to 20% of the population. When r_r is larger than this value, the performance generally depreciated. However, for SORIGA, the better results were generally found for smaller values of r_r. Those results are explained by the fact that, in SORIGA, the size of the subpopulation is generally larger than r_r, which implies that less individuals of the population are employed to explore the current best solution. In this way, SORIGA is interesting, when compared to other RIGAs, for values of r_r smaller than the value which represents the best value for the standard RIGA.

It can also be observed that the performance of SORIGA decreases when the difficulty level introduced by deception increases from dynamic deceptive function 1 to dynamic deceptive function 2 (Tables 1–6). This happens because on dynamic deceptive function 2 the probability to generate a fair random immigrant close to the global optimum and develop it in a subpopulation is very low. It can be seen in Figs. 5 and 7 that, while the best-of-generation individuals of the GAs with random

Table 5 Experimental results of the mean best-of-generation fitness (selection with roulette wheel sampling and $r_r = 12$) and relevant statistical comparisons (inside the parentheses)

Problem	Dynamics		Algorithm			
	τ	ρ	SGA	RIGA1	RIGA2	SORIGA
Royal road function	10	0.05	30.60 (+)	32.73 (∼)	32.51 (∼)	32.95
	10	0.60	6.62 (+)	13.77 (+)	14.63 (+)	15.99
	10	0.95	12.92 (+)	16.39 (+)	16.83 (+)	18.29
	200	0.05	59.97 (∼)	59.90 (∼)	60.83 (−)	59.64
	200	0.60	29.38 (+)	41.38 (+)	42.37 (+)	47.46
	200	0.95	24.08 (+)	40.32 (+)	41.81 (+)	47.22
	1,000	0.05	63.10 (∼)	63.09 (∼)	63.35 (−)	63.07
	1,000	0.60	53.24 (+)	57.48 (+)	58.65 (+)	59.95
	1,000	0.95	49.70 (+)	57.81 (+)	58.81 (+)	59.98
Deceptive function 1	10	0.10	0.8276 (∼)	0.8425 (∼)	0.8421 (∼)	0.8394
	10	0.60	0.7601 (+)	0.8020 (+)	0.8034 (+)	0.8321
	10	0.90	0.8418 (+)	0.8880 (∼)	0.8922 (∼)	0.8945
	200	0.10	0.8408 (+)	0.9817 (∼)	0.9423 (+)	0.9863
	200	0.60	0.8193 (+)	0.9372 (+)	0.9334 (+)	0.9541
	200	0.90	0.9092 (+)	0.9562 (+)	0.9690 (−)	0.9624
	1,000	0.10	0.8422 (+)	0.9986 (∼)	0.9886 (+)	0.9992
	1,000	0.60	0.8250 (+)	0.9862 (+)	0.9854 (+)	0.9902
	1,000	0.90	0.9121 (+)	0.9874 (+)	0.9933 (−)	0.9901
Deceptive function 2	10	0.05	0.6947 (−)	0.6883 (−)	0.7048 (−)	0.6618
	10	0.60	0.5565 (+)	0.5760 (∼)	0.5761 (∼)	0.5736
	10	0.95	0.5401 (+)	0.5647 (∼)	0.5610 (+)	0.5670
	200	0.05	0.7931 (−)	0.7913 (−)	0.7943 (−)	0.7868
	200	0.60	0.7496 (−)	0.7582 (−)	0.7675 (−)	0.7454
	200	0.95	0.7306 (+)	0.7583 (−)	0.7679 (−)	0.7453
	1,000	0.05	0.7985 (−)	0.7981 (−)	0.7988 (−)	0.7975
	1,000	0.60	0.7900 (−)	0.7917 (−)	0.7934 (−)	0.7890
	1,000	0.95	0.7867 (+)	0.7916 (−)	0.7936 (−)	0.7890
Scaling function 1	10	0.05	0.9673 (∼)	0.9652 (∼)	0.9691 (−)	0.9616
	10	0.60	0.7705 (+)	0.8395 (+)	0.8450 (+)	0.8510
	10	0.95	0.8268 (+)	0.8664 (+)	0.8680 (∼)	0.8731
	200	0.05	0.9958 (∼)	0.9986 (∼)	0.9988 (−)	0.9983
	200	0.60	0.8760 (+)	0.9784 (+)	0.9772 (+)	0.9855
	200	0.95	0.8546 (+)	0.9676 (+)	0.9592 (+)	0.9834
	1,000	0.05	0.9994 (∼)	0.9997 (∼)	0.9998 (−)	0.9997
	1,000	0.60	0.8917 (+)	0.9958 (+)	0.9952 (+)	0.9971
	1,000	0.95	0.8706 (+)	0.9933 (+)	0.9913 (+)	0.9964

Table 5 continued

Problem	Dynamics		Algorithm			
	τ	ρ	SGA	RIGA1	RIGA2	SORIGA
Scaling function 2	10	0.05	0.9327 (−)	0.9359 (−)	0.9292 (∼)	0.9241
	10	0.60	0.7380 (+)	0.7992 (∼)	0.7897 (+)	0.8019
	10	0.95	0.8217 (+)	0.8421 (∼)	0.8435 (∼)	0.8402
	200	0.05	0.9898 (∼)	0.9929 (∼)	0.9918 (∼)	0.9922
	200	0.60	0.8665 (+)	0.9583 (+)	0.9582 (+)	0.9670
	200	0.95	0.8517 (+)	0.9449 (+)	0.9371 (+)	0.9587
	1,000	0.05	0.9963 (∼)	0.9982 (−)	0.9982 (∼)	0.9978
	1,000	0.60	0.8860 (+)	0.9872 (+)	0.9860 (+)	0.9907
	1,000	0.95	0.8633 (+)	0.9825 (+)	0.9794 (+)	0.9887

The results of SGA are equal to those presented in Table 1

immigrants converge to the global optimum after the changes on deceptive function 1, they converge to the local optimum on deceptive function 2.

In the experiments on DOPs presented here, just like that the fossil records data for the extinction events in nature [21], there are more small replacement events than large ones, and the replacement events occur on a large variety of length scales. When the distribution of the number of replacement events against each size is plotted in a log–log scale, the results exhibit power laws (see Sect. 3) even without any apparent tuning, suggesting the presence of SOC.

5.6 Including a neighboring scheme

In the SORIGA presented in Sect. 4, the relations among the replaced individuals in each generation are determined by the indices of the individuals, i.e., the relations are random. In each generation, the individuals with indices from $j - \lceil (r_r - 1)/2 \rceil$ to $j + \lfloor (r_r - 1)/2 \rfloor$, where j is the index of the individual with the lowest fitness in the current population, are replaced by randomly generated individuals.

In this subsection, another replacement scheme, called *neighborhood reserving selection scheme*, is investigated. In this scheme, the individuals in the population are ordered according to the index of their first parent after the reproduction step. As an example, suppose a population with only 5 individuals cyclically indexed as [1–2–3–4–5–1]. After reproduction, suppose that the individuals with the following first parents [3,4,2,3,1] are present in the new population. In the neighborhood reserving selection scheme, the new population is rearranged to [1,2,3,3,4]. In this way, a neighborhood is spatially reserved to the maximal degree.

Table 7 presents the results of the SORIGA with this neighboring scheme when selection with roulette wheel sampling was employed and 3 individuals were replaced in each generation, denoted *SORIGA2*. Comparing the results presented in Table 7 to those presented in Table 1 for the SORIGA, it can be observed that the performance of the two SORIGAs are similar for most problems while both show good performance in comparison to other GAs.

Table 6 Experimental results of the mean best-of-generation fitness (selection with roulette wheel sampling and $r_r = 24$) and relevant statistical comparisons (inside the parentheses)

Problem	Dynamics		Algorithm			
	τ	ρ	SGA	RIGA1	RIGA2	SORIGA
Royal road function	10	0.05	30.60 (∼)	33.29 (∼)	33.98 (−)	32.09
	10	0.60	6.62 (+)	16.73 (+)	17.55 (+)	15.99
	10	0.95	12.92 (+)	18.49 (∼)	18.76 (∼)	18.20
	200	0.05	59.97 (−)	59.82 (−)	61.48 (−)	55.11
	200	0.60	29.38 (+)	42.84 (+)	45.57 (+)	47.43
	200	0.95	24.08 (+)	42.53 (+)	45.24 (+)	47.97
	1,000	0.05	63.10 (−)	63.08 (−)	63.51 (−)	57.16
	1,000	0.60	53.24 (+)	58.20 (−)	59.59 (−)	55.52
	1,000	0.95	49.70 (+)	58.16 (−)	59.81 (−)	55.63
Deceptive function 1	10	0.10	0.8276 (+)	0.8645 (∼)	0.8468 (+)	0.8675
	10	0.60	0.7601 (+)	0.8408 (∼)	0.8251 (+)	0.8459
	10	0.90	0.8418 (+)	0.8999 (∼)	0.9138 (−)	0.8950
	200	0.10	0.8408 (+)	0.9958 (−)	0.9682 (+)	0.9801
	200	0.60	0.8193 (+)	0.9619 (+)	0.9659 (+)	0.9768
	200	0.90	0.9092 (+)	0.9702 (+)	0.9824 (∼)	0.9833
	1,000	0.10	0.8422 (+)	0.9986 (∼)	0.9886 (+)	0.9992
	1,000	0.60	0.8250 (+)	0.9862 (+)	0.9854 (+)	0.9902
	1,000	0.90	0.9121 (+)	0.9874 (+)	0.9933 (−)	0.9901
Deceptive function 2	10	0.05	0.6947 (−)	0.6736 (−)	0.7094 (−)	0.6330
	10	0.60	0.5565 (+)	0.5803 (−)	0.5868 (−)	0.5755
	10	0.95	0.5401 (+)	0.5744 (−)	0.5747 (−)	0.5710
	200	0.05	0.7931 (−)	0.7890 (−)	0.7946 (−)	0.7765
	200	0.60	0.7496 (−)	0.7535 (−)	0.7710 (−)	0.7151
	200	0.95	0.7306 (+)	0.7583 (−)	0.7716 (−)	0.7173
	1,000	0.05	0.7985 (−)	0.7979 (−)	0.7989 (−)	0.7922
	1,000	0.60	0.7900 (−)	0.7904 (−)	0.7942 (−)	0.7782
	1,000	0.95	0.7867 (−)	0.7906 (−)	0.7944 (−)	0.7786
Scaling function 1	10	0.05	0.9673 (−)	0.9637 (−)	0.9694 (−)	0.9366
	10	0.60	0.7705 (+)	0.8586 (∼)	0.8568 (∼)	0.8587
	10	0.95	0.8268 (+)	0.8739 (∼)	0.8765 (−)	0.8693
	200	0.05	0.9958 (∼)	0.9986 (−)	0.9986 (−)	0.9959
	200	0.60	0.8760 (+)	0.9862 (∼)	0.9847 (∼)	0.9852
	200	0.95	0.8546 (+)	0.9809 (+)	0.9722 (+)	0.9859
	1,000	0.05	0.9994 (−)	0.9997 (−)	0.9997 (−)	0.9987
	1,000	0.60	0.8917 (+)	0.9971 (−)	0.9970 (−)	0.9966
	1,000	0.95	0.8706 (+)	0.9959 (+)	0.9941 (+)	0.9966

Table 6 continued

Problem	Dynamics		Algorithm			
	τ	ρ	SGA	RIGA1	RIGA2	SORIGA
Scaling function 2	10	0.05	0.9327 (−)	0.9315 (−)	0.9386 (−)	0.9033
	10	0.60	0.7380 (+)	0.8123 (∼)	0.8114 (∼)	0.8116
	10	0.95	0.8217 (∼)	0.8507 (−)	0.8501 (−)	0.8271
	200	0.05	0.9898 (−)	0.9936 (−)	0.9956 (−)	0.9855
	200	0.60	0.8665 (+)	0.9681 (−)	0.9694 (−)	0.9648
	200	0.95	0.8517 (+)	0.9571 (+)	0.9493 (+)	0.9645
	1,000	0.05	0.9963 (−)	0.9982 (−)	0.9999 (−)	0.9936
	1,000	0.60	0.8860 (+)	0.9912 (−)	0.9910 (−)	0.9877
	1,000	0.95	0.8633 (+)	0.9883 (−)	0.9855 (+)	0.9871

The results of SGA are equal to those presented in Table 1

6 Conclusions and future work

In recent years, several approaches have been developed into EAs to deal with dynamic optimization problems. The random immigrants scheme is a simple yet interesting approach, which aims to improve the diversity of the population by

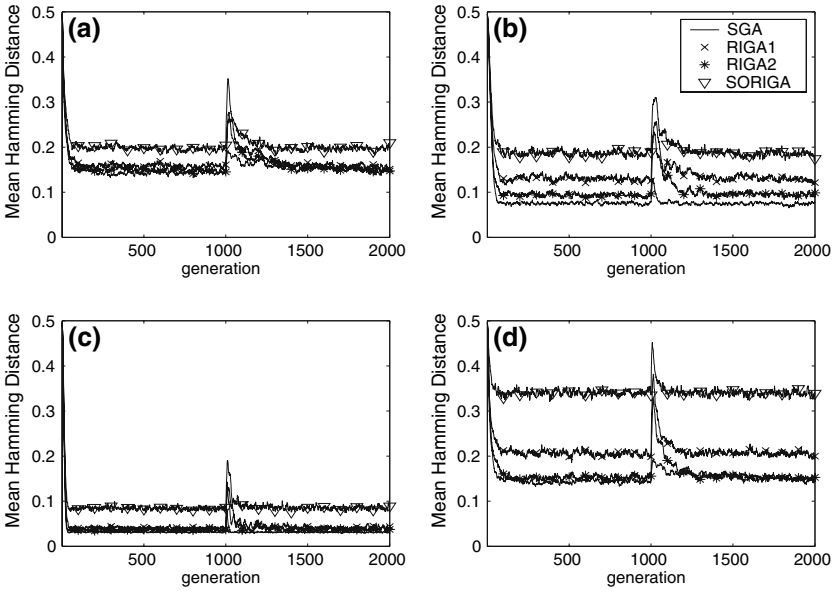

Fig. 12 Mean Hamming Distance averaged over 30 runs in the initial 2,000 generations of the dynamic Scaling Problem 2 with τ = 1,000 and ρ = 0.95: (**a**) roulette wheel sampling selection with r_r = 3, (**b**) tournament selection with k_{ts} = 0.70, (**c**) tournament selection with k_{ts} = 0.90, (**d**) roulette wheel sampling selection with r_r = 12

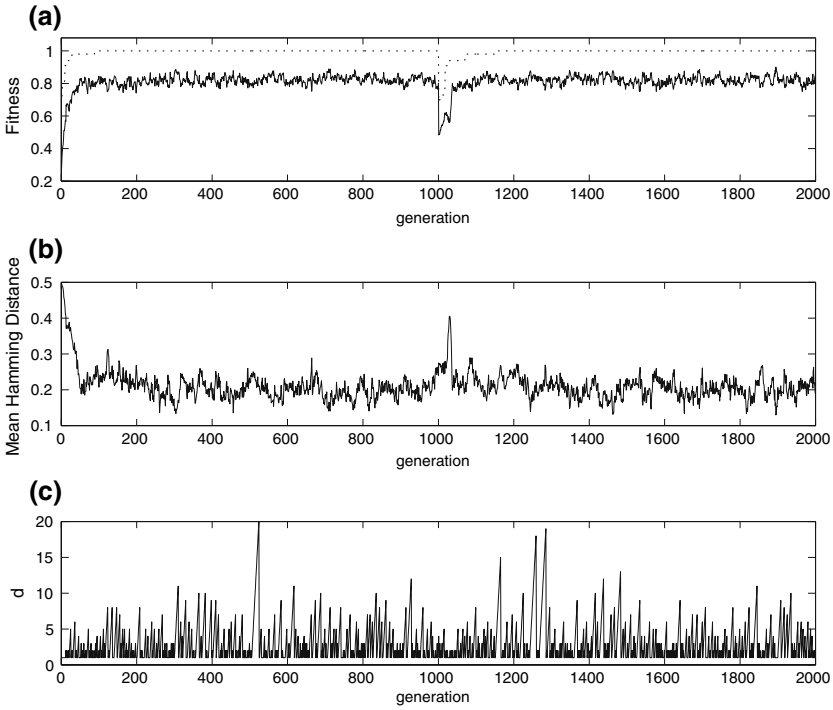

Fig. 13 Initial 2,000 generations of SORIGA with roulette wheel sampling selection on the second trial of the dynamic Scaling Problem 2 with $\tau = 1{,}000$ and $\rho = 0.95$: (**a**) mean population (solid line) and best-of-generation fitness (dotted line), (**b**) mean Hamming distance, (**c**) duration of replacement events (in generations)

replacing random individuals into the population. However, traditional random immigrants scheme has some shortcomings. The newly inserted random immigrants may not survive the selection phase due to their low fitness. In this paper, a new self-organizing random immigrants scheme is proposed for GAs for DOPs, where the individual with the lowest fitness and those with indices close to its index are replaced by random immigrants in every generation. In order to protect these random immigrants from being removed immediately by the selection phase, a subpopulation is used to preserve them. The subpopulation evolves in parallel with the main population and may enlarge or shrink until going extinct, which is called a replacement event.

The individual replacement and subpopulation strategies in the proposed immigrants scheme are dependent: they only have positive effect when applied together. Without the subpopulation strategy, the proposed replacement strategy makes no sense. On the other hand, the use of subpopulation without the proposed replacement strategy implies the use of subpopulations whose sizes are not self-organized. For example, if random individuals are replaced in every generation, as in traditional RIGAs, and a subpopulation is used to keep new individuals, the subpopulation should have a fixed size.

Table 7 Experimental results of the mean best-of-generation fitness when the neighborhood reserving selection scheme is employed (SORIGA2) and the statistical comparison regarding SORIGA2–SORIGA (inside the parentheses)

Dynamics		Royal road function	Deceptive function 1	Deceptive function 2	Scaling function 1	Scaling function 2
τ	ρ					
10	0.05	31.36 (∼)	0.8291 (∼)	0.6925 (∼)	0.9627 (∼)	0.9202 (∼)
10	0.60	11.63 (∼)	0.7947 (∼)	0.5701 (∼)	0.8213 (∼)	0.7765 (∼)
10	0.95	15.08 (∼)	0.8766 (∼)	0.5508 (∼)	0.8509 (∼)	0.8384 (∼)
200	0.05	58.48 (−)	0.8906 (−)	0.7910 (∼)	0.9981 (∼)	0.9925 (∼)
200	0.60	40.82 (∼)	0.8781 (∼)	0.7569 (∼)	0.9670 (∼)	0.9489 (∼)
200	0.95	40.12 (∼)	0.9391 (∼)	0.7569 (∼)	0.9499 (∼)	0.9322 (+)
1,000	0.05	62.91 (−)	0.9588 (−)	0.7983 (∼)	0.9996 (∼)	0.9981 (∼)
1,000	0.60	56.88 (−)	0.9594 (∼)	0.7914 (∼)	0.9930 (∼)	0.9842 (∼)
1,000	0.95	56.78 (−)	0.9758 (∼)	0.7911 (−)	0.9881 (∼)	0.9761 (∼)

The results of SORIGA are presented in Table 1

In SORIGA where the proposed immigrants scheme is applied, individuals start to interact between themselves and, when the fitness of the individuals are close, as in the case of low diversity, one single replacement can affect a great number of individuals in a replacement event. It is important to observe that the proposed simple approach can take the system to a self-organizing behavior, which can be useful for DOPs to maintain diversity of solutions and to allow the GA to escape from local optima if the problem changes. The experimental results presented in this paper indicate that SORIGA produces a higher diversity level in the population than SGA and traditional RIGAs.

Implementing self-organizing behavior, such as the self-organized criticality studied here, and including it in GAs has shown to be beneficial for their performance under dynamic environments. Future work will compare the self-organizing property with other properties, such as speciation schemes, and investigate its effect on constrained DOPs, e.g., the dynamic Knapsack problem. Further work has to investigate other kinds of interaction among individuals that might produce self-organization.

Acknowledgments The authors would like to thank Prof. Wolfgang Bazhaf and the anonymous reviewers for their thoughtful suggestions and helpful comments, which greatly contributed to the improvement of the paper. This work was supported by FAPESP (Proc. 04/04289-6).

References

1. Bak, P.: How Nature Works: The Science of Self-organized Criticality. Oxford University Press (1997)
2. Bak, P., Tang, C., Wiesenfeld, K.: Self-organized criticality. An explanation of 1/f noise. Phys. Rev. Lett. **59**(4), 381–384 (1987)
3. Boettcher, S., Percus, A.G.: Optimization with extremal dynamics. Complexity **8**(2), 57–62 (2003)
4. Branke, J.: Evolutionary Optimization in Dynamic Environments. Kluwer (2001)

5. Branke, J.: Evolutionary approaches to dynamic optimization problems – introduction and recent trends. In: Branke, J. (ed.) Proceedings of the GECCO Workshop on Evolutionary Algorithms for Dynamic Optimization Problems, pp. 2–4 (2003)
6. Cedeno, W., Vemuri, V.R.: On the use of niching for dynamic landscapes. In: Proceedings of the 1997 IEEE International Conference on Evolutionary Computation, pp. 361–366 (1997)
7. Cobb, H.G.: An investigation into the use of hypermutation as an adaptive operator in genetic algorithms having continuouis, time-dependent nonstationary environments. Technical Report AIC-90-001, Naval Research Laboratory, Washington, USA (1990)
8. Cobb, H.G., Grefenstette, J.J.: Genetic algorithms for tracking changing environments. In: Forrest, S. (ed.) Proceedings of the 5th International Conference on Genetic Algorithms, pp. 523–530 (1993)
9. Deb, K., Goldberg, D.E.: Analyzing deception in trap functions. In: Whitley, L.D. (ed.) Foundation of Genetic Algorithms 2, pp. 93–108 (1993)
10. Goldberg, D.A.: Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley Publishing Company Inc. (1989)
11. Goldberg, D.A.: The Design of Innovation: Lessons from and for Competent Genetic Algorithms. Kluwer Academic Publishers, Boston, MA (2002)
12. Gould, S.J.: Wonderful Life: The Burgess Shale and the Nature of History. W. W. Norton and Company (1989)
13. Grefenstette, J.J.: Genetic algorithms for changing environments. In: Maenner, R., Manderick, B. (eds.) Parallel Problem Solving from Nature 2, pp. 137–144. North Holland (1992)
14. Jensen, H.J.: Self-organized Criticality: Emergent Complex Behavior in Physical and Biological Systems. Cambridge University Press (1998)
15. Jin, Y., Branke, J.: Evolutionary optimization in uncertain environments – a survey. IEEE Trans. Evol. Comput. **9**(3), 303–317 (2005)
16. Kauffman, S.A.: The Origins of Order: Self-organization and Selection in Evolution. Oxford University Press (1993)
17. Krink, T., Thomsen, R.: Self-organized criticality and mass extinction in evolutionary algorithms. In: Proceedings of the 2001 Congress on Evolutionary Computation, vol. 2, pp. 1155–1161 (2001)
18. Løvbjerg, M., Krink, T.: Extending particle swarm optimisers with self-organized criticality. In: Proceedings of the 2002 IEEE Congress on Evolutionary Computation, vol. 2, pp. 1588–1593 (2002)
19. Mitchell, M.: An Introduction to Genetic Algorithms. MIT Press (1996)
20. Mori, N., Kita, H., Nishikawa, Y.: Adaptation to a changing environment by means of the feedback thermodynamical genetic algorithm. In: Eiben, A.E., Bäck, T., Schoenauer, M., Schwefel, H.-P. (eds.) Parallel Problem Solving from Nature, number 1498 in LNCS, pp. 149–158. Springer (1998)
21. Raup, D.M.: Biological extinction in earth history. Science **231**, 1528–1533 (1986)
22. Trojanowski, K., Michalewicz, Z.: Evolutionary optimization in non-stationary environments. J. Comput. Sci. Technol. **1**(2), 93–124 (2000)
23. Vavak, F., Fogarty, T.C., Jukes, K.: A genetic algorithm with variable range of local search for tracking changing environments. In: Voigt, H.-M. (ed.) Parallel Problem Solving from Nature, number 1141 in LNCS. Springer Verlag, Berlin (1996)
24. Yang, S.: Non-stationary problem optimization using the primal-dual genetic algorithm. In: Sarker, R., Reynolds, R., Abbass, H., Tan, K.-C., McKay, R., Essam, D., Gedeon, T. (eds.) Proceedings of the 2003 IEEE Congress on Evolutionary Computation, vol. 3, pp. 2246–2253 (2003)
25. Yang, S.: Constructing dynamic test environments for genetic algorithms based on problem difficulty. In: Proceedings of the 2004 IEEE Congress on Evolutionary Computation, vol. 2, pp. 1262–1269 (2004)
26. Yang, S., Yao, X.: Experimental study on population-based incremental learning algorithms for dynamic optimization problems. Soft Comput. **9**(11), 815–834 (2005)