5

Genetic Algorithms with Self-Organizing Behaviour in Dynamic Environments

Renato Tinós[1] and Shengxiang Yang[2]

[1] Departamento de Física e Matemática, FFCLRP, Universidade de São Paulo (USP), Av. Bandeirantes 3900, Ribeirão Preto, SP, 14040-901, Brazil
rtinos@ffclrp.usp.br

[2] Department of Computer Science, University of Leicester
University Road, Leicester LE1 7RH, United Kingdom
s.yang@mcs.le.ac.uk

Summary. In recent years, researchers from the genetic algorithm (GA) community have developed several approaches to enhance the performance of traditional GAs for dynamic optimization problems (DOPs). Among these approaches, one technique is to maintain the diversity of the population by inserting random immigrants into the population. This chapter investigates a self-organizing random immigrants scheme for GAs to address DOPs, where the worst individual and its next neighbours are replaced by random immigrants. In order to protect the newly introduced immigrants from being replaced by fitter individuals, they are placed in a subpopulation. In this way, individuals start to interact between themselves and, when the fitness of the individuals are close, one single replacement of an individual can affect a large number of individuals of the population in a chain reaction. The individuals in a subpopulation are not allowed to be replaced by individuals of the main population during the current chain reaction. The number of individuals in the subpopulation 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-organization behaviour, which can be useful for GAs in dynamic environments.

5.1 Introduction

A significant part of optimization problems in real world is dynamic optimization problems (DOPs), where the evaluation function and the constraints of the problem are not fixed [24]. 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 [4]. The optimization problem may change for several factors, like faults, machine degradation, environmental or climatic modifications, and economic factors. In fact, the natural evolution, which is the inspiration for genetic algorithms (GAs) , is always dynamic. The occurrence of natural cataclysms, geological modifications, competition for natural

resources, coevolution between species, and climatic modifications are only a few examples of changes related to natural evolution.

The simplest approach to deal with DOPs is to start a new optimization process whenever a change in the problem is noticed. However, the optimization process 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, the knowledge obtained during the search for the old solution can be utilized to find the new solution [16]. In this case, the search for new solutions based on the old solutions can save substantial processing time. Evolutionary algorithms are particularly attractive to such problems as individuals representing solutions of the problem before the changes can be transferred into the new optimization process.

However, in GAs, the population of solutions generally converges in the fitness landscape to points close to the best individual of the population. If the fitness landscape abruptly changes, the actual population can be trapped in local optima located close to the old solution. In fact, the premature convergence of the solution to a local optima is not a problem exclusive to DOPs, but it can be a serious problem in stationary optimization problems too [20]. In order to avoid the premature convergence, several approaches where the diversity level is re-introduced or maintained throughout the run have appeared in literature over the past years (see the surveys [4, 16, 24]). Typical examples of such approaches are the *random immigrants GA* (RIGA) [14], the sharing or crowding mechanisms [5], the variable local search [25], the thermodynamical genetic algorithm [21], and the use of hypermutation [6].

RIGA, which is inspired by the flux of immigrants that wander in and out of a population between two generations in nature, is very interesting and simple [7, 14]. In RIGA, some individuals of the current population are 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 RIGA tries to maintain the diversity level of the population, which can be very useful to prepare the population for possible fitness landscape changes [7].

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

In this work, instead of substituting the worst individuals or the random individuals in each generation like in the standard RIGA, the worst individual and its next neighbours are replaced. In order to protect the newly introduced immigrants from being replaced by fitter individuals, they are placed in a subpopulation. In this way, individuals start to interact between themselves

and, when the fitness of the individuals are close, as in the case where the diversity level is low, one single replacement of an individual can affect a large number of individuals of the population in a chain reaction. The individuals in the subpopulation are not allowed to be replaced by individuals of the main population during the current 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-organization behaviour, which can be useful in DOPs.

The experimental results suggest that the proposed GA presents a kind of self-organizing behaviour, known as self-organized criticality (SOC) [1], which is described in Section 5.2. The proposed GA is presented in Section 5.3, and the experimental results are presented in Section 5.4. In Section 5.4.3, the proposed GA and the experimental results are analyzed. Finally, Section 5.5 concludes the work with discussions on relevant future work.

5.2 Self-Organized Criticality

Systems consisting of several interacting constituents may present an interesting kind of self-organizing behaviour known as SOC [2], [15]. Researchers have suggested that several phenomena exhibit SOC, like sand piles, earthquakes, forest fires, electric breakdowns, and growing interfaces [1].

An interesting behaviour appears in systems exhibiting SOC: they self-organize into a particular critical state without the need of any significant tuning action from outside. The critical state is described by the response of a system to external perturbation. In a system exhibiting noncritical behaviour, the distribution of responses to perturbation at different positions and at different times is narrow and well described by an averaged value. In a system exhibiting critical behaviour, no single characteristic response exists, i.e., the system exhibits scale invariance. A small perturbation in one given location of the system may generate a small effect on its neighbourhood 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

$$P(s) \sim s^{-\tau} \tag{5.1}$$

and

$$P(d) \sim d^{-\alpha}, \tag{5.2}$$

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 constants. The sand pile model described in [2], where a single grain is added at a random position in every interval of time Δt is an example of a system exhibiting SOC. In order to characterize the response of the sand pile model,

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 system to the addition of a single grain are given by Eqs. 5.1 and 5.2, and the addition of a single grain can affect only a grain in its neighbourhood or can affect the whole sand pile.

Bak has suggested that SOC occurs in natural evolution too [1]. An evidence of SOC in evolution would be the fact that it does take place through bursts of activity intercalated by calm periods, instead of gradually at a slow and constant pace [13]. 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 [23]. Bak has suggested 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 [1]. In such hypothesis, species coevolve to a critical state [17].

A very simple simulation model to study the connection between evolution and SOC was proposed by Bak and Sneppen [1]. In the one-dimensional version of the model, the individuals (or species in the authors' terminology) are placed in a circle, and a random value of fitness is assigned to each one of them. In each generation of the simulation, the values of fitness of the individual with the lowest fitness in the current population, one individual located in its right position, and one located in its left position are replaced by new random values. An analogy of the connection between neighbours in this simple model is the interaction between species in nature: if a prey is extinct, the fitness of its predators will change. The Bak-Sneppen Model is summarized in Fig. 5.1.

---

begin
  Find the index j of the individual with the lowest fitness
  Replace the fitness of the individuals with index j, $j-1$, and $j+1$ by random values drawn with uniform density
end

Fig. 5.1. The Bak-Sneppen model

The Bak-Sneppen model presents an interesting behaviour. In the beginning of the simulation, the mean fitness of the population is low, but, as the number of generation increases, the mean fitness increases too. Eventually, the mean fitness ceases to increase, and the critical state is reached. In the Bak-Sneppen Model, a replacement of the fitness of the worst individual causes the replacement of its two next neighbours. In the critical state, the values of fitness of the neighbours are very often replaced by random numbers with

smaller values. The new worst individual can be then one of these two neighbours, which are replaced with its two next neighbours, 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. 5.1 and 5.2. Large replacement events generally occur when almost all individuals of the population have similar high values of fitness.

It is important to observe that SOC avoids the situation where the species get trapped in local optima in the fitness landscape in the Bak-Sneppen evolution model. The idea is interesting and relatively simple, and soon researchers proposed the use of SOC in optimization processes. Boettcher and Percus [3] proposed the optimization with extremal dynamics, a local-search heuristic for finding solutions in problems where constituents of the system are connected, e.g., the spin glass optimization problem. Løvbjerg and Krink [19] extended Particle Swarm Optimization with SOC in order to improve the optimization process and to maintain the diversity level.

In GAs, Krink and Thomsen [18] proposed the use of the sand pile model previously discussed to generate power laws to be utilized to control the size of spatial replacement zones in a diffusion model. When an individual is extinct, a mutated version of the best individual of the population is created in its place. It is important to observe that, in the algorithm proposed in [18], SOC appears in the sand pile model utilized to control the size of the replacements, and not as a result of the self-organization of the constituents of the system (individuals of the GA).

5.3 Random Immigrants Genetic Algorithm with Self-Organizing Behaviour

In the standard RIGA, randomly chosen individuals of the current population $\mathbf{P_t}$ are replaced by randomly generated individuals. A replacement rate specifies the number of individuals replaced in each generation. The standard RIGA can be summarized in Fig. 5.2, which differs from the generational *standard GA* (SGA) only by the inclusion of the procedure "replace($\mathbf{P_t}$)", where randomly chosen individuals of the current population are replaced by randomly generated individuals.

In this work, we propose the replacement of the individual with the lowest fitness of the current population and its two next neighbours for new randomly generated individuals in RIGA. The indices of the individuals are used to determine the neighboring relations. In each generation of the algorithm, the individual with the lowest fitness in the current population (index j), one individual located in its right position (index $j + 1$), and one located in its left position (index $j - 1$) are replaced by new random individuals. One can observe that, as the proposed GA is not spatially distributed, the neighbouring relations are random.

```
Require: N: population size; p_c: crossover rate; p_m: mutation rate
begin
  t ← 1
  initialize(P_t,N)
  evaluate(P_t)
  while (stop criteria are not satisfied) do
    P_t ← replace(P_t)
    for i = 1 to N do
      P_{t+1}(i) ← selection(P_t,i)
    end for
    crossover(P_{t+1},p_c)
    mutation(P_{t+1},p_m)
    evaluate(P_{t+1})
    t ← t + 1
  end while
end
```

Fig. 5.2. The random immigrants genetic algorithm (RIGA)

A second strategy is still adopted, where the new immigrants created during the current chain reaction (called replacement event in this work), which occurs along the generations, are preserved in a subpopulation. The size of this subpopulation is not defined by the programmer, but is given by the number of individuals created in the current replacement event. The individuals in the current population that do not belong to the subpopulation are not allowed to replace individuals present in the subpopulation. The individuals that belong to the subpopulation are allowed to evolve, i.e., they are submitted to selection, crossover, and mutation. It is important to observe that selection and crossover are allowed only among individuals that belong to the subpopulation.

We hope that, with such strategies, the system can exhibit SOC in order to increase the diversity level of the population in a self-organized way and, then, to avoid the situation where the individuals get trapped in local optima in the fitness landscape when the problem changes.

In the proposed *self-organizing random immigrants GA* (SORIGA), there are two major modifications from the standard RIGA. In the first modification, the procedure "replace($\mathbf{P_t}$)" is modified as presented in Fig. 5.3. The current size (or duration) of each replacement event, i.e., the number of times that we replace the worst individual and its neighbours in the current replacement event, is recorded and denoted by d, and the minimum and maximum index values of the replaced individuals ($i_{min}-1$ and $i_{max}+1$) are employed to compute the number of individuals affected by the current replacement event. The number of individuals in the subpopulation, i.e., the size of subpopulation, equals to $(i_{max}+1)-(i_{min}-1) = i_{max}-i_{min}+2$. When the chain

reaction ceases, i.e., the individual with the lowest fitness does not belong to the subpopulation, the size of the replacement is set to 1.

```
Procedure replace(P_t)
begin
  Find the index j of the individual with the lowest fitness
  Replace the individuals of P_t with indices j, j − 1, and j + 1 by randomly generated individuals
  if (i_min − 1 ≤ j ≤ i_max + 1) then
    d ← d + 1
    if (j = i_min − 1) then
      i_min ← j
    end if
    if (j = i_max + 1) then
      i_max ← j
    end if
    else
      d ← 1
      i_min ← j
      i_max ← j
  end if
end
```

Fig. 5.3. The replace approach

```
Procedure selection(P_t, i)
begin
  if (i < i_min − 1) or (i > i_max + 1) then
    Select an individual from P_t
  else
    Select an individual from the subset of individuals of P_t with index in [i_min − 1, i_max + 1]
  end if
end
```

Fig. 5.4. The selection approach

The second modification, which is presented in Fig. 5.4, lies in the selection approach for each individual in the population. Two cases can occur. If the index of the new individual was not affected by the current replacement event, the new individual is selected according to the standard approach. Otherwise, i.e., if the index was affected by the current replacement, the new individual

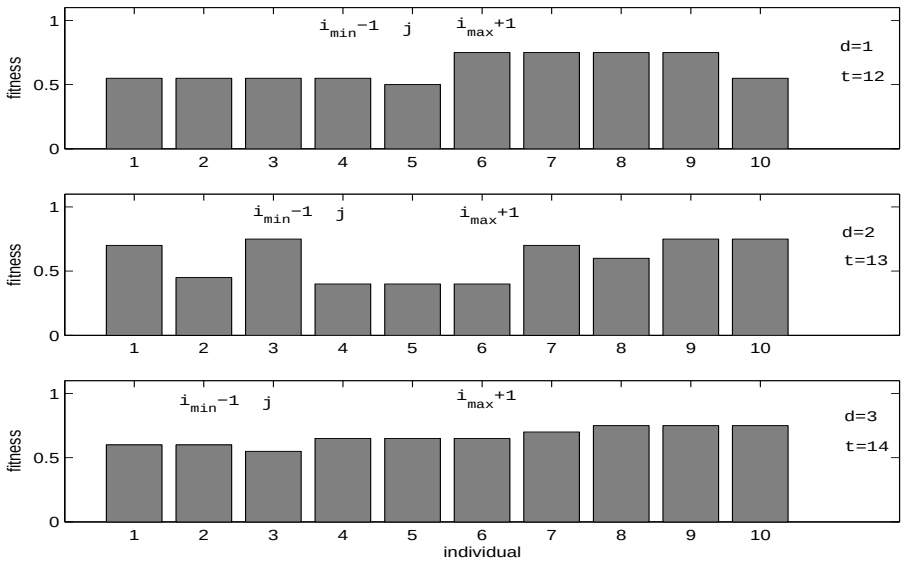

Fig. 5.5. Fitness of the individuals of the current population at generations 12, 13, and 14 in a run of the SORIGA on the example problem

is selected from the subpopulation that consists of the individuals replaced in the current replacement event (individuals with index values from $i_{min} - 1$ to $i_{max} + 1$).

In order to illustrate its working, SORIGA is applied to a simple problem where the fitness function is defined as

$$f(\mathbf{x}) = \frac{u(\mathbf{x})}{l}, \tag{5.3}$$

where $u(\mathbf{x})$ is the unitation function of a binary vector (individual) $\mathbf{x}$ of length l, which returns the number of ones in vector $\mathbf{x}$. The individuals, which are randomly generated in the first generation, are selected according to elitism and the roulette wheel method. Mutation with rate $p_m = 0.01$ and two-point crossover with rate $p_c = 0.7$ are employed. The number of individuals in the population is equal to 10 and $l = 20$. Fig. 5.5 presents the first three steps of an replacement event in a run of SORIGA on this example. The figure shows the fitness of all individuals in the current population in generations 12, 13, and 14 respectively. In generation 12, the individual with index 5 (index j in Fig. 5.3) has the lowest fitness in the population. In this way, individual with index 5 and its two next neighbours (individuals with indices 4 and 6) are replaced by randomly generated individuals. In the next generation, the individual with index $j = 4$ has now the lowest fitness, and it together with its two next

neighbours (individuals with indices 3 and 5) are then replaced. In generation 14, the individual with index $j = 3$ has the lowest fitness. It can be observed that the chain reaction is propagated because the remaining individuals have fitness values higher than the individuals in the subpopulation defined by the limits $i_{min} - 1$ and $i_{max} + 1$ (see Fig. 5.3 and Fig. 5.4). The individuals that do not belong to this subpopulation are not allowed to replace an individual of this subpopulation.

5.4 Experimental Study

In order to evaluate the performance of proposed SORIGA, two sets of experiments are carried out. In the first set of experiments, the dynamic test environment for GAs proposed by Yang [26] is employed (Subsection 5.4.1). In the second set of experiments, evolutionary robots are simulated in dynamic environments (Subsection 5.4.2). In the experiments, SORIGA is compared to SGA, and two versions of RIGA. In the first version, denoted *RIGA1*, three individuals randomly chosen from the current population are replaced by randomly generated individuals in each generation. In the second version, denoted *RIGA2*, the three worst individuals, i.e., the individuals with the lowest fitness, are replaced by randomly generated individuals. The analysis of the results is presented in Subsection 5.4.3.

5.4.1 Dynamic Test Environment

In order to evaluate the performance of different GAs in DOPs, Yang [26] proposed a dynamic environment generator based on unitation and trap functions. A trap function is defined as follows

$$f(\mathbf{x}) = \begin{cases} \frac{a}{z}(z - u(\mathbf{x})), & \text{if } u(\mathbf{x}) \leq z \\ \frac{b}{l-z}(u(\mathbf{x}) - z), & \text{otherwise,} \end{cases} \tag{5.4}$$

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 can be a deceptive function for GAs, i.e., a 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 sub-optimal [11]. 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]

$$r = \frac{a}{b} \geq \frac{2 - 1/(l - z)}{2 - 1/z} \tag{5.5}$$

Deception is not the only element that can generate difficulty to a GA. The problem difficulty can also be caused by exogenous noise and scaling. The scaling problem arises in functions that consist of several schemata with different worth to the solution [12]. A scaling problem can be simulated using additively decomposable functions as follows

$$f(\mathbf{x}) = \sum_{i=1}^{m} c_i \ f_i(\mathbf{x}_{I_i}), \tag{5.6}$$

where m is the number of schemata that are juxtaposed and summed together, I_i is the set of the fixed bit positions that form schema i, and c_i is the scaling factor for each sub-function f_i.

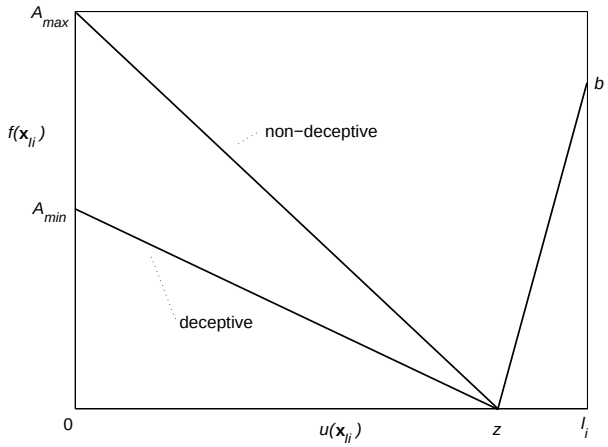

Fig. 5.6. *Illustration of the trap function $f(\mathbf{x})$. The global optimum changes between b and A_{max} in every δ_t generations.*

Employing Eqs. 5.4 and 5.6, it is possible to create different dynamic environments where the problem difficulty can be adjusted. In this work, dynamic environments where the deception difficulty is modified by changing the peak heights of optima are employed [26]. In these dynamic environments, the fitness of an individual $\mathbf{x}$ is given by additively decomposable trap functions defined as follows

$$f(\mathbf{x}) = \sum_{i=1}^{m} c_i \ f_i(\mathbf{x}_{I_i}, t) \tag{5.7}$$

$$f(\mathbf{x}_{I_i}, t) = \begin{cases} \frac{a_i(t)}{z_i}(z_i - u(\mathbf{x}_{I_i})), \text{ if } u(\mathbf{x}_{I_i}) \leq z_i \\ \frac{b_i}{l_i - z_i}(u(\mathbf{x}_{I_i}) - z_i), \text{ otherwise,} \end{cases} \tag{5.8}$$

where $i = 1, \ldots, m$, $\mathbf{x}^{\mathrm{T}} = [\mathbf{x}_{I_1}^{\mathrm{T}} \ \ldots \ \mathbf{x}_{I_m}^{\mathrm{T}}]$, $\mathbf{x}_{I_i} = [x_{(i-1)l_i+1} \cdots x_{(i-1)l_i+l_i}]^{\mathrm{T}}$, $b_i = b$=1.0, $z_i = z$, the scaling is given by $c_i = 2^{i-1}$, and a_i switches between $A_{min} > 0$ and $A_{max} > b_i$ in every δ_t generations. The parameter A_{min} is constrained by Eq. 5.5. That is, it is chosen in order that the trap functions are deceptive on average. In this way, in every δ_t generations, the global optimum changes between b and $a_i = A_{max}$, and the problem changes between deceptive and non-deceptive (Fig. 5.6).

Three dynamic environments are generated as the test bed for all GAs in this work. In the first (Environment 1), $l = 36$, $z = 5$, $m = 6$, l_i=6, $A_{min} = 0.6$, and $A_{max} = 1.4$. In the second (Environment 2), $l = 36$, $z = 4$, $m = 6$, l_i=6, $A_{min} = 0.9$, and $A_{max} = 1.9$. In the third (Environment 3), $l = 45$, $z = 4$, $m = 9$, l_i=5, $A_{min} = 0.6$, and $A_{max} = 1.4$.

Experimental Design

For each run of an algorithm in a dynamic environment, the individuals of the initial population are randomly chosen. The individuals are selected in each generation according to elitism and the roulette wheel method. Mutation with rate p_m and two-point crossover with rate p_c are utilized. Nine experiments with different parameters are presented in this section. Table 5.1 presents the parameters utilized in each experiment.

Table 5.1. Experimental Setttings - Dynamic Test Environments

Experiment	Environment	Population Size	p_m	p_c	δ_t
1a	1	100	0.01	0.7	5000
1b	1	20	0.01	0.7	5000
1c	1	300	0.01	0.7	5000
1d	1	100	0.001	0.7	5000
1e	1	100	0.05	0.7	5000
1f	1	100	0.01	0.2	5000
1g	1	300	0.01	0.7	10000
2	2	100	0.01	0.7	5000
3	3	100	0.01	0.7	5000

The comparison of the results obtained by different algorithms on DOPs is more complex than the same comparison on stationary problems [24]. For DOPs, it is necessary to evaluate not the final result, but rather the optimization process itself. Here, the measure *adaptability*, proposed in [24] and based on a measure proposed by De Jong [8], is utilized to evaluate the GAs. Adaptability is computed as the difference, averaged over the entire run, between the fitness of the current best individual of each generation and the corresponding optimum value. The best results for the adaptability measure are those with the smallest values.

Table 5.2. Adaptability - Dynamic Test Environments

Experiment	SGA	RIGA1	RIGA2	SORIGA
1a	0.1882 (15.68%)	0.0177 (1.47%)	0.0217 (1.81%)	0.0086 (0.72%)
1b	0.1996 (16.63%)	0.0267 (2.22%)	0.0325 (2.71%)	0.0173 (1.44%)
1c	0.1622 (13.52%)	0.0137 (1.15%)	0.0132 (1.10%)	0.0068 (0.57%)
1d	0.2001 (16.67%)	0.0198 (1.65%)	0.0263 (2.19%)	0.0106 (0.88%)
1e	0.0368 (3.07%)	0.0071 (0.59%)	0.0076 (0.63%)	0.0064 (0.54%)
1f	0.1924 (16.03%)	0.0370 (3.08%)	0.0414 (3.46%)	0.0200 (1.67%)
1g	0.1491 (12.42%)	0.0071 (0.59%)	0.0073 (0.61%)	0.0036 (0.30%)
2	0.1956 (13.49%)	0.0143 (0.98%)	0.0166 (1.15%)	0.0084 (0.58%)
3	0.1655 (13.79%)	0.0090 (0.75%)	0.0104 (0.87%)	0.0052 (0.43%)

Table 5.3. Mean Fitness of the Population - Dynamic Test Environments

Experiment	SGA	RIGA1	RIGA2	SORIGA
1a	0.9286	1.0559	1.0755	1.0174
1b	0.9610	1.0340	1.0796	0.8730
1c	0.9198	1.0480	1.0569	1.0368
1d	0.9891	1.1356	1.1534	1.0865
1e	0.8590	0.8614	0.8729	0.8342
1f	0.9263	1.0566	1.0694	1.0330
1g	0.9323	1.0544	1.0631	1.0405
2	1.1519	1.2914	1.3199	1.2360
3	0.9503	1.0726	1.0930	1.0321

Experimental Results

Tables 5.2 and 5.3 respectively present the experimental results regarding the adaptability and the mean fitness of all individuals of the population averaged over 20 trials, each one with a different random seed. In Table 5.2, the percentage inside the parentheses indicates the adaptability over the optimum fitness values.

Hypothesis tests, considering the Student's t-distribution, indicate that the measure adaptability is smaller for SORIGA with a level of significance equal to 0.01 in all experiments except Experiment 1e, where the level of significance equals 0.095.

Fig. 5.7 shows the fitness of the best individuals averaged over the 20 trials for SGA and SORIGA in Experiment 1a.

5.4.2 Evolutionary Robotics

Robots in which artificial evolution is used as a fundamental form of adaptation or design are known as evolutionary robots [22]. In the experiments presented in this section, mobile robots are simulated in DOPs using a modified version of the Evorobot simulator developed by S. Nolfi [22]. In the simulator

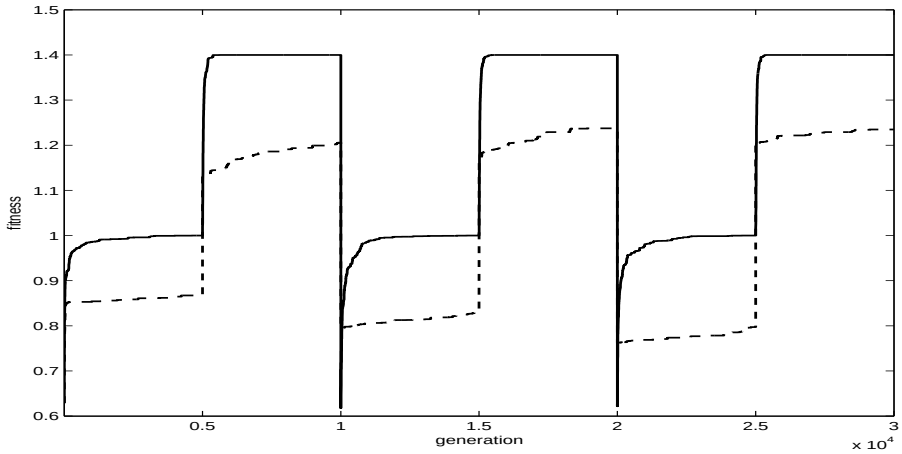

Fig. 5.7. *Averaged fitness of the best individual in Experiment 1a (dynamic test environment). The solid and dashed lines represents the results for SORIGA and SGA respectively.*

utilized in the experiments presented in this section, the robots are controlled by a recurrent artificial neural network (Elman Network). GAs have been employed to perform several tasks in artificial neural networks (ANNs), such as architecture design, synaptic weight adjustment, learning rule adaptation, and synaptic weight initialization [27]. In the experiments presented here, the synaptic weights of the ANN used to control the robot are adjusted by GAs.

Two evolutionary robot experiments are presented in this section. The two experiments are inspired by the experiment proposed by Floreano and Mondada [10], where a Khepera robot with eight infrared distance sensors (six sensors in one side and two in another side of the robot), two ambient light sensors, and one floor brightness sensor navigates in an arena. The robot has a measurable limited energy, which is recharged every time the robot crosses a battery recharge area. The battery recharge area is indicated by a different color of the floor and by a light source mounted in a tower inside the area.

Experimental Design

In the experiments presented in this section, the fitness function is given by the accumulated averaged rotation speed of the two wheels of the robot during its life time, i.e., while the battery has energy and while the robot does not crash into a wall or an obstacle, considering a maximum limit of 60 seconds. A fully charged battery allows the robot to move for 20 seconds. The fitness is not computed while the robot remains in the battery recharge area. Although the fitness function does not specify that the robot should

return to the battery recharge area, the individuals that develop the ability to find it and periodically return to it while exploring the arena without hitting the obstacles accumulate more fitness. The neural network utilized to control the robots has 17 inputs (8 infrared sensors, 2 ambient light sensors, 1 floor brightness sensor, 1 sensor for the battery energy, and 5 recurrent units), 5 hidden neurons, and 2 outputs (2 motors).

Two experiments with 2000 generations each are presented in this section. In the first experiment (Experiment 4), the environment where the robot is evolving is changed after 1000 generations. Environment changes frequently occur in real problems, where some aspects of the environment are frequently modified. Besides, robots are frequently evolved in simulations to avoid damage, and, when a satisfactory behaviour is reached, the neural networks employed to control the simulated robot are transferred to the real ones. In the experiments, the robot evolves in an arena of 40cm $\times$ 45cm free of obstacles during the first 1000 generations, and in an arena of 60cm $\times$ 35cm with four cylindrical obstacles during the last 1000 generations.

In the second experiment (Experiment 5), the robot is affected by a failure after 1000 generations. The responses of the six infrared sensors located in one side of the robot are set to zero when it is affected by this failure. We are interested in investigating the reconfiguration of the robot after an abrupt failure. The robot should evolve in an arena of 60cm $\times$ 35 cm with three cylindrical obstacles.

In the runs, the individuals of the initial population are randomly chosen. The evolving robot always starts in a fixed position on the environment, but with a random initial orientation. The individuals are represented by a vector of real values corresponding to the synaptic weights of the ANN. In each generation of the GAs, the 20 best individuals are selected and each one generates 5 children ($N = 100$). In both experiments, $p_m = 0.01$ and crossover is not utilized.

Experimental Results

Table 5.4 presents the experimental results with respect to the adaptability (supposing a maximum fitness equal to 1.0), the mean fitness of all individuals of the population, and the fitness of the best individual after 2000 generations averaged over 20 trials, each one with a diffente random seed, which indicates the performance of the robot after the change in the problem. For most of the times, a new solution is found, which allows the robot to navigate in the environment and to return to the battery recharge area only when the battery level is low. When the problem changes, the fitness values of the robot become small, and a new solution is searched.

Hypothesis tests, considering the Student's t-distribution, indicate that the fitness of the best individual after 2000 generations is higher for SORIGA in Experiment 4 with the level of significance equal to 0.04, 0.1, and 0.28 when compared to SGA, RIGA1, and RIGA2 respectively. In Experiment 5,

Table 5.4. Experimental Results - Evolutionary Robotics

Measure	Algorithm	Experiment 4 (environment changing)	Experiment 5 (failure reconfiguration)
Adaptability	SGA	0.3614 (36.14%)	0.6002 (60.02%)
	RIGA1	0.3508 (35.08%)	0.5201 (52.01%)
	RIGA2	0.3129 (31.29%)	0.6268 (62.68%)
	SORIGA	0.3022 (30.22%)	0.5191 (51.91%)
Mean Fitness	SGA	0.2991	0.1861
	RIGA1	0.3406	0.2317
	RIGA2	0.3540	0.1837
	SORIGA	0.3484	0.2301
Fitness of the best individual (end of the simulation)	SGA	0.6380	0.2840
	RIGA1	0.6690	0.3880
	RIGA2	0.7270	0.3590
	SORIGA	0.7550	0.4410

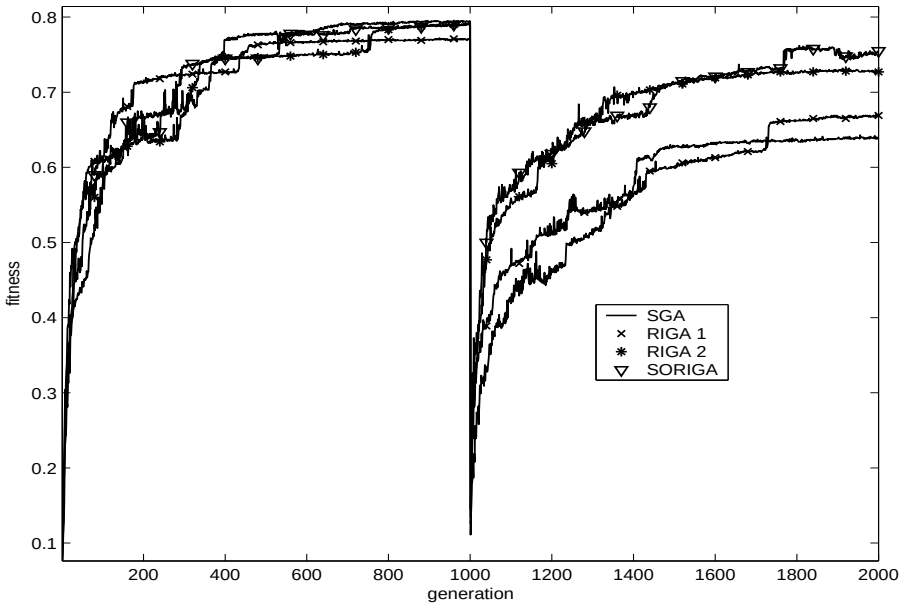

Fig. 5.8. *Averaged fitness of the best individual in Experiment 4 (evolutionary robots)*

the fitness of the best individual after 2000 generations is higher for SORIGA with the level of significance equal to 0.1, 0.34, and 0.25 when respectively compared to SGA, RIGA1, and RIGA2.

Figure 5.8 and Figure 5.9 show the averaged fitness for all GAs in Experiment 4 and Experiment 5 respectively.

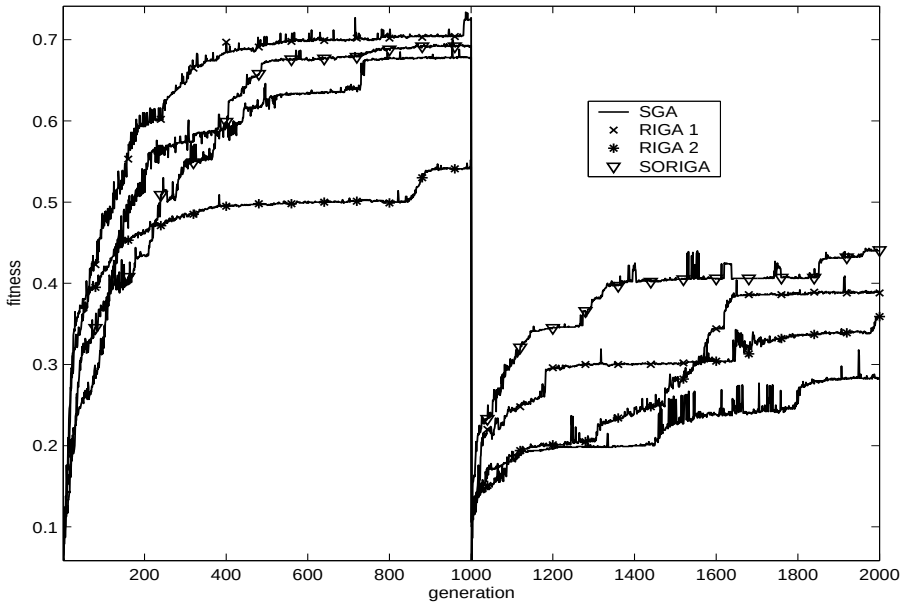

Fig. 5.9. *Averaged fitness of the best individual in Experiment 5 (evolutionary robots)*

5.4.3 Analysis of the Results

In the experiments presented in Section 5.4.1, the values of adaptability for the three GAs with random immigrants are smaller than the values for SGA, i.e., the averaged fitness of the best individuals is higher for the GAs with random immigrants. These results can be explained by the fact that it is difficult for the SGA to escape from the local optima induced by the deceptive problem and by changing the global optima. However, random immigrants inserted in every generation provide diversity to the populations of the last three GAs, which explains their better results.

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 generally have low fitness. In SORIGA, the new individuals that replace the individual with the lowest fitness and its neighbours generally have low fitness too. Since several individuals in the population have low fitness, the probability that one of the neighbours of the current worst individual becomes the new worst is small. As a consequence, a single replacement of an individual generally does not generate large chain reactions of replacements. That is, the distribution of the duration of replacement events is narrow and well described by a small average value. As the number of generations increases, the mean fitness increases too. In this situation, 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 two neighbours of the old worst individual, which were replaced in the last generation, becomes the new worst individual increases. When this new worst individual is replaced with its two next neighbours, a chain reaction can be developed and the replacement events can have, then, a large variety of sizes. In this case, the replacement events can not be characterized by a narrow distribution.

The better results of SORIGA over other RIGA in the experiments presented here can be explained by two major factors. First, the number of different individuals that are replaced in a fixed period of generations is generally large for SORIGA. In the RIGA where the worst individuals are replaced, it is common that new individuals replace individuals with the same index in next generation, because the new individuals usually have small fitness values. In this way, the number of different individuals that are replaced in a fixed period of generations is usually smaller in comparison with SORIGA, and hence the diversity becomes smaller too. This fact can be observed by analyzing the results presented in Table 5.2. SORIGA presents the smallest values of the mean fitness of the population, even though its fitness values of the best individuals are the highest (i.e., its adaptability values are the smallest).

The second major fact 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 smaller 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). This occurs because the selection methods employed in GAs preserves, directly or indirectly, the best individuals of the population. An immigrant usually survives during the evolution only if its fitness is close to the mean fitness of the population. This is a rare event when the number of parameters in the solution is high or when the local optimum where the population is found has fitness values much higher than the mean fitness of the search space. On the other hand, SORIGA preserves a new potential solution in a subpopulation and allows it to evolve while the current replacement event is in progress. When the replacement event ends, evolved versions of possible new solutions given by fair immigrants are generally present in the current population and can be combined with the individuals of the main population to generate new solutions.

Figures 5.10 and 5.11 show the mean fitness and the duration of the replacement events (d) in the first trial of Experiments 1a and 1e respectively. It can be seen that when the global optimum changes from a smaller to a higher value, the mean fitness of the population increases. This leads to higher mean duration values of the replacement events and hence increases the diversity of the population. Such interesting behaviour is reached by self-organization, and not by a rule imposed by the programmer. The size of the subpopula-

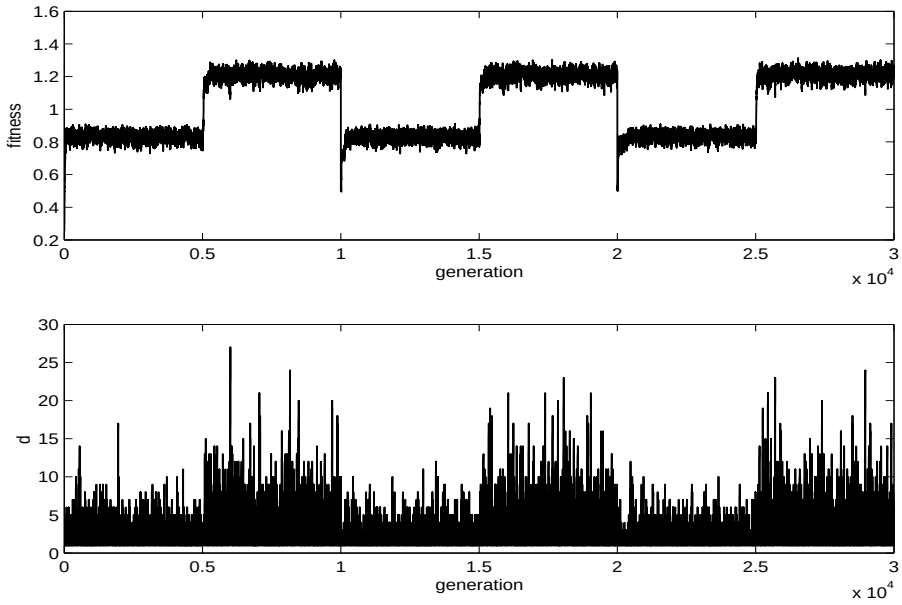

Fig. 5.10. *Mean fitness and duration of replacement events (in generations) in the first trial of Experiment 1a*

tion self-organizes according to the population diversity level. This fact can be seen by comparing Figures 5.10 and 5.11. In Experiment 1e the mutation rate is higher than in Experiment 1a (see Table 5.1), which results in a higher diversity level. In this way, it is not necessary to generate large replacement events to increase the diversity level. It can be observed that the duration of the larger replacement events is higher in Experiment 1a. This fact explains the worse results of SORIGA when compared to other GAs in Experiment 1e (Table 5.2).

The same analysis can be done to the evolutionary robot experiments (Table 5.4). In experiments 4 and 5, the changes in the problems are so strong that new solutions completely different from the old ones should be found. One can consider, as an example, the experiment where a failure is introduced in the robots (Experiment 5). In the experiments with evolutionary robots, navigation strategies where the robot always moves in the same direction are initially developed. Most of the times, the developed direction of moving is that one that provides more sensing capabilities to the robot, i.e. that one where the front of the robot is the side with more infrared sensors (6). When a failure in the 6 infrared sensors is introduced, the current navigation strategy is no longer interesting. Moving the robot in the developed former direction generally causes collision since it can not detect walls and obstacles in its front side. In this case, a new navigation strategy should be developed, where

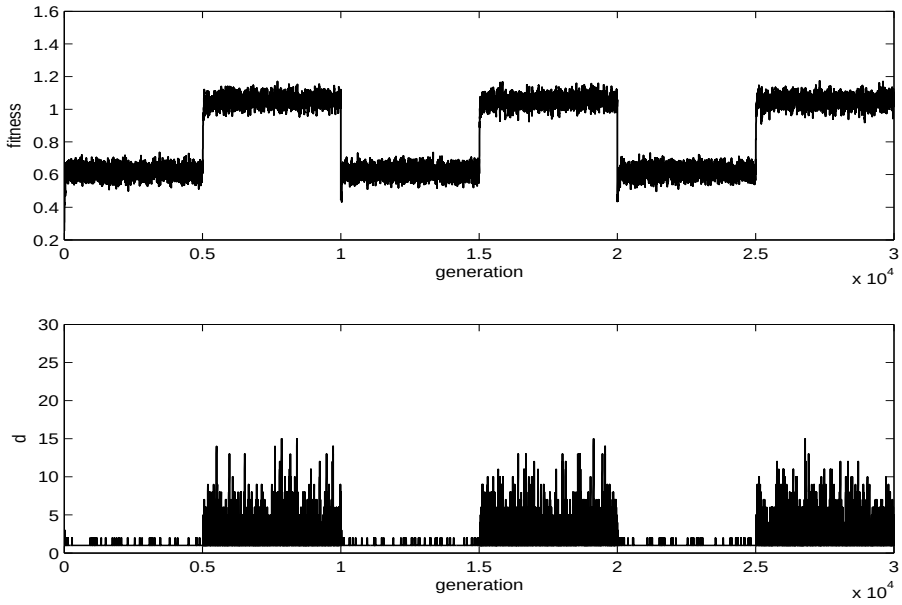

Fig. 5.11. *Mean fitness and duration of replacement events (in generations) in the first trial of Experiment 1e*

the side originally chosen as the rear of the robot, i.e., the side with the two infrared sensors that are working, is now in the front of the robot. Such changes causes a drastic modification to the ANN responsible for the robot control. In the SGA, the probalibity to find a new solution with such characteristics after the introduction of the failure is generally smaller, as the SGA can became trapped in local optima given by the old solution. The old solution is generally better than most new solutions, where the robot generally does not know how to navigate in a straight way.

However, SORIGA can eventually generate new solutions far from the local optima and develop them in the subpopulation. These facts can explain the better results of SORIGA regarding the final fitness of the best individuals presented in Table 5.4. Fig. 5.12 presents the fitness of the best individual and the duration of the replacement events for SORIGA in the tenth trial of this experiment. One can observe that, after the introduction of the failure at generation 1000, the fitness of the best individual becomes low. After some replacement events, a new solution is found (after generation 1800). This solution allows the robot to navigate in the environment and return to the battery recharge area only when the battery level is low, even with the presence of 6 faulty infrared sensors. One can still observe that, like in the results shown in Fig. 5.10, higher mean values for the duration of the replacement events occur for higher fitness values.

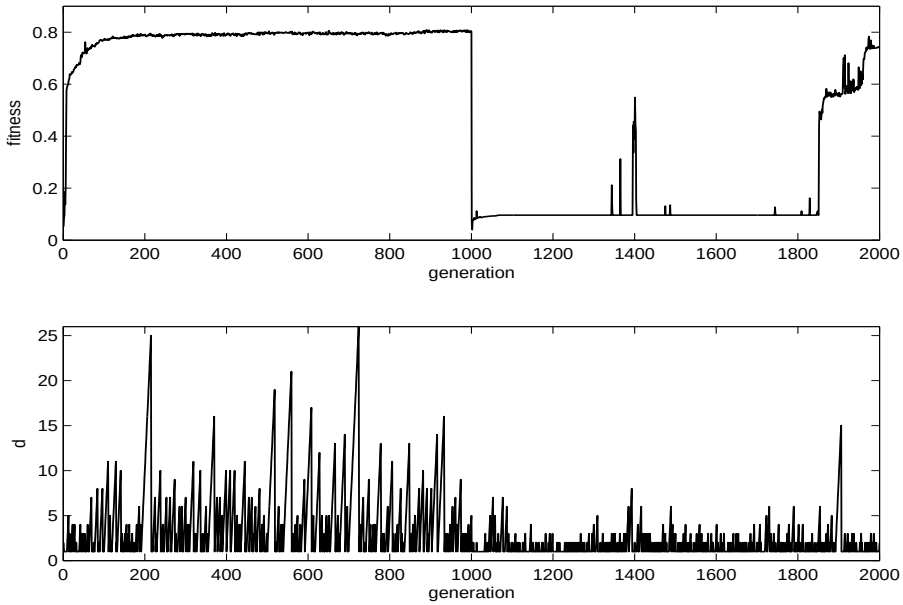

Fig. 5.12. *Fitness of the best individual and duration of replacement events in the tenth trial of the evolutionary robots Experiment 5 (failure reconfiguration)*

In the experiments presented in the last section, like in the fossil recorded data for the replacement events in nature [23], there are more small than large replacement events, and the replacement events occur on a large variety of length scales. In Figures 5.13 and 5.14, the distribution of the number of replacement events against each size is plotted in a log-log scale for trials of the dynamic test environment Experiment 1g and evolutionary robots Experiment 5 respectively. One can observe that the results exhibit power laws (see Section 5.2) even without any apparent tuning, indicating the presence of SOC. This kind of self-organization behaviour arises in systems where many degrees of freedom are interacting and the dynamics of the system is dominated by the interaction between these degrees of freedom, rather than by the intrinsic dynamics of the individual degrees of freedom [15]. In SORIGA, the population self-organizes in order to allow the occurrence of replacement events with a large variety of length scales. Large replacement events generally occur when the mean fitness of the population is high and the diversity level of the population is low. The population diversity is controlled by self-organization, allowing the GA to escape from local optima when the problem changes.

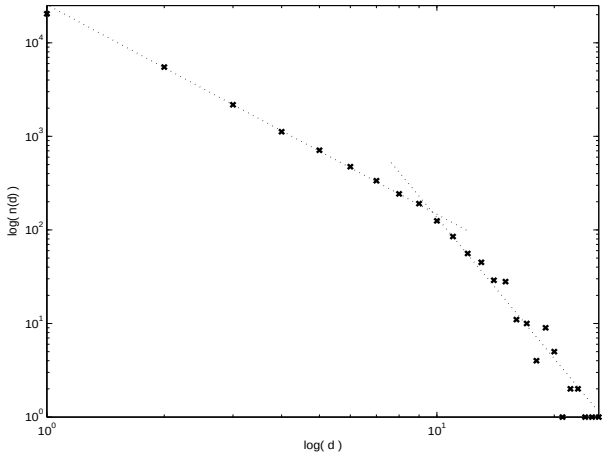

Fig. 5.13. *Number of occurrences for each size of the replacement events in the first trial of Experiment 1g*

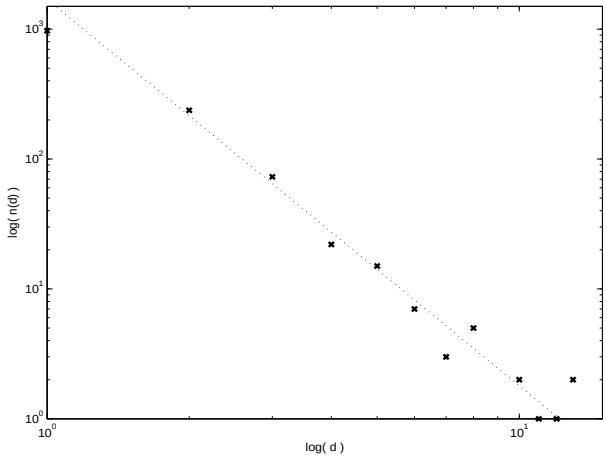

Fig. 5.14. *Number of occurrences for each size of the replacement events in the fourth trial of Experiment 5*

5.5 Conclusions

In this work, a GA with random immigrants where the worst individual and its next neighbors are replaced in every generation is proposed. The new individuals are preserved in a subpopulation, which size is not defined by the programmer, but is given by the number of individuals created in the current replacement event. In SORIGA, the individual starts to interact between themselves and, when the fitness of the individuals are close, as in the case

where the diversity level is low, one single replacement can affect a large number of individuals in an replacement event. It is important to observe that this simple approach can take the system to a self-organization behaviour, which can be useful for DOPs to maintain the diversity of the solutions and, then, to allow the GA to escape from local optima when the problem changes. In this way, the proposed GA is interesting for DOPs where the new solution is located in a peak that is hardly reached from the location of the old solution by traditional GA operators.

Studying and combining self-organizing behaviours, such as the self-organized criticality studied in this work, into GAs have shown to be beneficial for their performance under dynamic environments. Further work can be done in this area. A relevant future work is to compare the self-organizing property with other properties, such as the speciation schemes, for GAs under more comprehensive dynamic environments. Another future work is to investigate the use of other neighbouring relations in the proposed algorithm.

Acknowledgments

The authors would like to thank FAPESP (Proc. 04/04289-6) for the financial support.

References

1. P. Bak. *How nature works: the science of self-organized criticality.* Oxford University Press, 1997.
2. P. Bak, C. Tang, and K. Wiesenfeld. Self-organized criticality. an explanation of 1/f noise. *Physical Review Letters*, 59(4):381–384, 1987.
3. S. Boettcher and A. G. Percus. Optimization with extremal dynamics. *Complexity*, 8(2):57–62, 2003.
4. J. Branke. Evolutionary approaches to dynamic optimization problems - introduction and recent trends. In J. Branke, editor, *GECCO Workshop on Evol. Alg. for Dynamic Optimization Problems*, pages 2–4, 2003.
5. W. Cedeno and V. R. Vemuri. On the use of niching for dynamic landscapes. In *Proc. of the 1997 IEEE Int. Conf. on Evolutionary Computation*, pages 361–366, 1997.
6. H. G. Cobb. 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.
7. H. G. Cobb and J. J. Grefenstette. Genetic algorithms for tracking changing environments. In S. Forrest, editor, *Proc. of the 5th Int. Conf. on Genetic Algorithms*, pages 523–530, 1993.
8. K. A. De Jong. *An analysis of the behavior of a class of genetic adaptive systems.* PhD Dissertation, University of Michigan, 1975.

9. K. Deb and D. E. Goldberg. Analyzing deception in trap functions. In *Foundation of Genetic Algorithms 2*, pages 93–108, 1993.
10. D. Floreano and F. Mondada. Evolution of homing navigation in a real mobile robot. *IEEE Trans. on Systems, Man, and Cybernetics - Part B: Cybernetics*, 26(3):396–407, 1996.
11. D. A. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley Publishing Company, Inc., 1989.
12. D. A. Goldberg. *The Design of Innovation: Lessons from and for Competent Genetic Algorithms*. Boston, MA: Kluwer Academic Publishers., 2002.
13. S. J. Gould. *Wonderful Life: The Burgess Shale and the Nature of History*. W. W. Norton and Company, 1989.
14. J. J. Grefenstette. Genetic algorithms for changing environments. In R. Maenner and B. Manderick, editors, *Parallel Problem Solving from Nature 2*, pages 137–144. North Holland, 1992.
15. H. J. Jensen. *Self-organized criticality: emergent complex behavior in physical and biological systems*. Cambridge University Press, 1998.
16. Y. Jin and J. Branke. Evolutionary optimization in uncertain environments - a survey. *IEEE Trans. on Evol. Computation*, 9(3):303–317, 2005.
17. S. A. Kauffman. *The origins of order: self-organization and selection in evolution*. Oxford University Press, 1993.
18. T. Krink and R. Thomsen. Self-organized criticality and mass extinction in evolutionary algorithms. In *Proc. of the 2001 Congress on Evolutionary Computation*, volume 2, pages 1155–1161, 2001.
19. M. Løvbjerg and T. Krink. Extending particle swarm optimisers with self-organized criticality. In *Proc. of the 2002 Congress on Evolutionary Computation*, volume 2, pages 1588–1593, 2002.
20. M. Mitchell. *An Introduction to Genetic Algorithms*. MIT Press, 1996.
21. N. Mori, H. Kita, and Y. Nishikawa. Adaptation to a changing environment by means of the feedback thermodynamical genetic algorithm. In A. E. Eiben, T. Bäck, M. Schoenauer, and H.-P. Schwefel, editors, *Parallel Problem Solving from Nature*, number 1498 in LNCS, pages 149–158. Springer, 1998.
22. S. Nolfi and D. Floreano. *Evolutionary robotics: the biology, intelligence, and technology of self-organizing machines*. MIT Press/Bradford Books: Cambridge, USA, 2000.
23. D. M. Raup. Biological extinction in earth history. *Science*, 231:1528–1533, 1986.
24. K. Trojanowski and Z. Michalewicz. Evolutionary algorithms for non-stationary environments. In M. A. Klopotek and M. Michalewicz, editors, *Intelligent Inf. Systems, Proc. of the 8th Int. Workshop on Intelligent Inf. Syst.*, pages 229–240, 1999.
25. F. Vavak, T. C. Fogarty, and K. Jukes. A genetic algorithm with variable range of local search for tracking changing environments. In H.-M. Voigt, editor, *Parallel Problem Solving from Nature*, number 1141 in LNCS. Springer Verlag Berlin, 1996.
26. S. Yang. Constructing dynamic test environments for genetic algorithms based on problem difficulty. In *Proc. of the 2004 Congress on Evolutionary Computation*, volume 2, pages 1262–1269, 2004.
27. X. Yao. Evolving artificial neural networks. *Proc. of the IEEE*, 87(9):1423–1447, 1999.