Available online at www.sciencedirect.com

ScienceDirect

ELSEVIER

Neurocomputing 70 (2006) 44–54

NEUROCOMPUTING

www.elsevier.com/locate/neucom

Use of gene dependent mutation probability in evolutionary neural networks for non-stationary problems

Renato Tinós[a,*], André C.P.L.F. de Carvalho[b]

[a]Departamento de Física e Matemática, FFCLRP, Universidade de São Paulo (USP), 14040-901, Ribeirão Preto, SP, Brazil
[b]Departamento de Ciência da Computação, ICMC, Universidade de São Paulo (USP), 13560-970, São Carlos, SP, Brazil

Available online 22 August 2006

Abstract

In this article, the authors investigate the application of genetic algorithms (GAs) with gene dependent mutation probability to the training of artificial neural networks (ANNs) in non-stationary problems (NSPs). In the problems studied, the function mapped by an ANN changes during the search carried out by the GA. In the GA proposed, each gene is associated with an independent mutation probability. The knowledge obtained during the evolution is used to update the mutation probabilities. If the modification of a set of genes is useful when the problem changes its profile, the mutation probabilities of these genes are increased. As a result, the search is concentrated into regions associated with genes presenting higher mutation probabilities.
© 2006 Elsevier B.V. All rights reserved.

Keywords: Evolutionary neural networks; Non-stationary problems; Genetic algorithms

1. Introduction

Associated with the use of artificial neural networks (ANNs) in practical problems is the necessity, for the designer, to optimize the values of the network free parameters [23]. The performance obtained by ANNs is determined by the choice of these values. In recent years, evolutionary algorithms (EAs) have been successfully employed for the design of ANNs [37,2,23,35,17,1,18]. ANNs using EAs as a fundamental form of adaptation are known as evolutionary artificial neural networks (EANNs) [37].

One of the free parameters whose value is frequently optimized in the EANNs is the set of synaptic weights. The synaptic weight adjustment by EAs is particularly attractive, because it can lead the network to a global minimum, without getting stuck into a local minimum in vast, complex, multimodal, and non-differentiable surfaces [37,23,30,25]. Besides, like in reinforcement learning, it is not necessary to know the exact desired output values during the training phase.

However, a solution found by an optimization procedure may no longer be effective when changes occur in the problem. In a real process, the optimization problem can be modified by several causes, like component degradation, faults, changes in the bias, operational point modifications, and environmental changes. In such non-stationary problems (NSPs), a new solution for the optimization problem, e.g. the set of synaptic weights, should be found whenever a change occurs.

The simplest approach to deal with NSPs using ANNs is to start a new optimization process whenever a change in the problem is noticed. However, the optimization of an ANN can require a high computational effort. If the new solution after the change in the problem is, in some sense, related to the previous solution, as when the changes are relatively small, the knowledge obtained during the search for the previous solution can be used to find the new solution [5,34]. The search for new solutions based on previous solutions can save substantial processing time.

EAs, known by their remarkable adaptability to changes, are particularly attractive to NSPs using ANNs. Individuals representing solutions before changes in the

*Corresponding author. Tel.: +55 16 3602 3773.
E-mail addresses: rtinos@ffclrp.usp.br (R. Tinós), andre@icmc.usp.br (A.C.P.L.F. de Carvalho).

0925-2312/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.neucom.2006.07.005

problem can be transferred to the new optimization process. In [27,28], genetic algorithms (GAs) are used to evolve ANNs capable of learning in artificial agents under changing environments in order to investigate the relationship between learning and evolution.

However, in spite of a significant part of real world optimization problems being NSPs, research in EAs has been mainly focused on stationary problems. In fact, the natural evolution, which is the inspiration for such algorithms, is always non-stationary.

In [33], a GA with gene dependent mutation probabilities (GAGDM) was proposed for NSPs. In the GAGDM, the knowledge obtained during the non-stationary process is employed to modify the independent mutation probabilities of the genes in a GA.

The use of genes with independent mutation probabilities in GAs for the weight adjustment of ANNs applied to NSPs is investigated in this paper. This approach can be interesting in NSPs where only some weights of the ANN should be updated after changes in the problem. It must be noticed that, the performance of the GAGDM can be poor in cases where a large number of weights should be updated after a change in the problem.

This article is organized as follows: EANNs are introduced in Section 2; in Section 3, the use of EAs for NSPs are briefly discussed; the GAGDM is described in Section 4; a method to set the parameters of the GAGDM is presented in Section 5; the experiments and their results are presented in Section 6; finally, this article is concluded in Section 7.

2. Evolutionary artificial neural networks

Several works have investigated the employment of EAs for the design of ANNs in recent years. For a good review on EANNs, see [37]. Most of the works on EANNs can be grouped into one or more of the following categories:

- *Architecture design*: The choice of the topological parameters of an ANN, such as the number of hidden layers and the number of neurons in each layer, has a significant impact in the inductive bias and training efficiency. Several articles can be found in the literature reporting the use of EAs to select the topological parameters of ANNs [37]. Although most of these works involve the use of GAs, other EA approaches have also been investigated. In [17], a general method for adjusting the probabilities of applying variation operators in structure optimization of ANNs was proposed.
- *Learning rule adaptation*: The design of the optimal learning rule is a very complex task when there is either no or little prior knowledge regarding the best network architectures. A few works have proposed the use of EAs for the selection of learning rules and algorithmic parameters for the training of ANNs [37,27,28].
- *Synaptic weight adjustment*: Most algorithms for training ANNs, such as the Backpropagation algorithm, are based on gradient descent. In spite of a large number of successful applications, methods based on gradient descent often get trapped in local optima in the error landscape. Besides, such methods require the use of desired outputs during the training phase and differentiable error surfaces. One way to overcome the use of the gradient descent is to employ EAs for the synaptic weight adjustment [30]. EAs are potentially useful in complex, non-differentiable, and multimodal spaces. Several papers describe the use of EAs for the adjustment of weights in ANNs [37,18]. The performance presented by GAs and Evolution Strategies (ESs) in the optimization of ANNs have been compared to backpropagation in [30,25], respectively.

In the EANNs employed in this paper, each individual of the GA population specifies one EANN, whose architecture is defined by the designer (e.g. feed-forward or recurrent architecture). The designer also decides the representation of the weights (binary or real number coded), and the genetic operators. Individuals are selected in each generation according to their fitness, which is based on the performance of the EANN during the training phase. As it is not necessary to differentiate the activation function, the threshold function can be used as the activation function of the neurons. In the real-valued representation, the chromosome of each individual is a vector of real numbers, each one representing a synaptic weight. In the binary-valued representation, each synaptic weight is represented by a number of bits with a fixed length.

3. EAs for NSP

Most of the EA approaches to deal with NSPs that have appeared in literature over the past years can be grouped into one of the following categories [5,20]:

- *Maintenance or re-introduction of the diversity level throughout the run*: The diversity can be useful when the GA needs to search for a solution in regions of the solution space hardly reached by applying the traditional genetic operators in a population that has already converged. Typical examples of this approach are the hypermutation and the random immigrants mechanism [7].
- *Use of multiple subpopulations*: Some subpopulations can be used to track the current best solution while others can be used to search for a new solution when the problem changes [6].
- *Use of the knowledge obtained during the evolution*: Past experiences are used to guide the search for the new solution. The experiences can be memorized by numeric [31], symbolic [29], or exact memory [19].

In such approaches to solve NSPs, all the genes of each individual of the GA have the same mutation probability.

However, in nature, the codifying sequences of nucleotides for the genes can have different mutation probabilities. Possible reasons may be the size and type of the sequence, the use of different enzymes during the replication process, and the use of synonymous codon [36]. How the mutation probability of the gene influences the evolution process still needs to be explained.

The idea that the genetic codes have evolved to their current configurations has been repeatedly suggested [11]. According to this theory, in the beginning, the nucleotides were randomly used in the genetic codes to encode genes. During the evolutionary process, sequences of nucleotides that encode the genes seems to have been "chosen" in order to increase or decrease the stability of the controlled characteristics [24,36]. For example, if a sequence codifies for a protein that is vital for the cell, independently of environmental changes, a low mutation probability should be more natural than a high mutation probability. In the opposite case, a high mutation probability would be more natural in genes that control characteristics that should be modified after an environmental change, like those controlling the resistance of bacteria to certain chemical substances. Moreover, researchers have related cases where the probability of an advantageous mutation suddenly increases when environmental changes take place [13]. This interesting behavior could explain the fast adaptation of some organisms, e.g. bacteria, to environmental changes.

Several researchers proposed the use of changing mutation parameters in order to improve the performance of EAs. In a survey [9], several strategies for parameter control in EAs are reviewed and classified in three major classes: deterministic, where the parameters are changed by deterministic rules; adaptive, where feedback from the optimization process is employed for parameter control; and self-adaptive, where parameters of the EA are encoded in the chromosome and allowed to evolve. In most of the cases, only one mutation control parameter can be updated for all individuals, like in the $\frac{1}{5}$ success rule in ESs [4]. In other cases, one mutation control parameter for each individual [9] can be changed.

Alternative strategies use mutation control parameters in the gene level. In [21], to search the maximum of multiple-peak variable-separable stationary functions, stochastic automata are used to learn the locus of effective genes on the individual where mutation yields a higher fitness. In ESs, the basic idea of self-adaptation is to use evolvable strategy parameters, which controls the mutation strength of each gene, encoded in the genome of each object [3]. In the covariance matrix adaptation ES (CMA-ES), the covariance matrix of the Gaussian mutation distribution is deterministically adapted in order to increase the probability to reproduce steps in the search space that have led to the current population [16]. In the CMA-ES, the search path of the population over past generations is used to adapt the covariance matrix. In [18], CMA-ES was successfully used for the evolutionary optimization of ANNs in stationary problems.

In [33], GAs with GAGDM were proposed for NSPs. In the GAGDM, each gene of each individual is associated with an independent mutation probability. The knowledge obtained during the non-stationary process is employed to modify the independent mutation probabilities of the genes. If the modification of a set of genes improves the adaptation to the changes taking place in the NSP, the mutation probabilities of this set are increased. Thus, the search in the solution space is concentrated into regions associated with higher mutation probabilities. This search, however, is not restricted to such regions, as all mutation probabilities are always higher than zero. In the next section, the main aspects of GAGDM are described.

4. Genetic algorithm with gene dependent mutation probability

In the standard GA, a population of individuals representing solutions of the problem is evolved according to mechanisms inspired by natural selection and genetics [12]. The solution $\mathbf{x}_i$, represented by the ith individual, $i = 1, \ldots, N$, is evaluated by a fitness function $f(\mathbf{x}_i)$. The standard GA uses two main genetic operators, crossover and mutation, to evolve the solutions. In the crossover operator, genes of two individuals are exchanged with a crossover probability rate, known as crossover rate. After crossover, the genes of the individuals are mutated with a mutation probability rate, known as mutation rate. In the standard GA, the genes of all individuals have the same mutation rate p_m.

If the fitness function $f(\mathbf{x}_i)$ changes over time, the problem can be described as an NSP. In these problems, the change of the fitness function $f(\mathbf{x}_i)$ can be discrete or continuous. The discrete changes, which appear from time to time after stagnation periods, are investigated here. Several are the examples of this type of NSPs, like those that involve intermittent faults, seasonal changes, and consumption variation. For simplicity, in this paper, the stagnation period, here named era, is fixed in ξ generations.

In the GAGDM, the mutation probability of each gene is independent and is allowed to change. Each gene g ($g = 1, \ldots, n$) of each individual i is associated with a variable υ_{i_g}, which controls the genes mutation probability. The mutation probability of the gene i_g (gth gene of the ith individual) in the GAGDM is given by

$$\gamma_{i_g} = n p_m \frac{\upsilon_{i_g}}{\sum_{j=1}^{n} \upsilon_{i_j}}, \tag{1}$$

where $\{\upsilon_{i_g} \in \mathbb{R} \mid 0 < \upsilon_{\min} \leqslant \upsilon_{i_g} \leqslant \upsilon_{\max}\}$, and $\upsilon_{\min}$ and $\upsilon_{\max}$ are constants. If υ_{i_g} has the same value for all genes of the individual, then $\gamma_{i_g} = p_m$ for $g = 1, \ldots, n$ and the GAGDM is converted into the standard GA.

In the beginning of the algorithm execution, every υ_{i_g} is equal to $\upsilon_{\min}$. Therefore, all genes have the same mutation probability p_m. In nature, the variation of the gene stability is controlled by natural selection. When it is known which genes should have higher mutation probabilities, υ_{i_g} can be chosen a priori in a GA. In this work, it is considered that the set of genes that should have higher mutation probabilities is not known. In the GAGDM, the knowledge obtained across the evolutionary process is used to change υ_{i_g}. The method employed to update υ_{i_g} in one generation is presented next.

The offspring generated by crossover inherit the values of υ_{i_g} from their parents. In the offspring generated by mutations, three cases can be considered. When the fitness of the offspring is higher than the fitness of its parent, the value of υ_{i_g} for all gene g mutated in this individual in the current generation is increased in this offspring and in its parent. In this situation, the genes that were not mutated do not have their mutation probabilities updated. Successive advantageous mutations across the eras result in higher mutation probabilities. When the fitness of the offspring is smaller than the fitness of its parent, υ_{i_g} has its value decreased in the parent and increased in the offspring. In this way, the mutation of the gene g is stimulated in the offspring and unstimulated in the parent. When the fitness of the offspring is equal to the fitness of its parent, υ_{i_g} is not updated.

Thus, when a gene is mutated, the value of υ_{i_g} is updated according to

$$\Delta\upsilon_{i_g} = \begin{cases} +\beta\alpha & \text{if } f(\mathbf{x}_o) > f(\mathbf{x}_i), \\ -\alpha & \text{if } f(\mathbf{x}_o) < f(\mathbf{x}_i), \\ 0 & \text{otherwise,} \end{cases} \qquad (2)$$

where g is the gene inherited from individual i and mutated in individual o, and the parameters $\beta > 0$ and $\alpha > 0$ control the size of the update. The variable υ_{o_g} in the offspring is updated according to

$$\Delta\upsilon_{o_g} = \begin{cases} +\beta\alpha & \text{if } f(\mathbf{x}_o) > f(\mathbf{x}_i), \\ +\alpha & \text{if } f(\mathbf{x}_o) < f(\mathbf{x}_i), \\ 0 & \text{otherwise.} \end{cases} \qquad (3)$$

The GAGDM based in the generational GA is summarized by Algorithm 1. Compared to the standard GA, the GAGDM requires more computational memory to store the n-dimensional vectors of the parameters υ_{i_g} for each individual of the population, and requires more computational time in order to update such parameters in each generation.

Algorithm 1.

```
define the initial population
do $\upsilon_{i_g} = \upsilon_{\min}$ for genes $g = 1,\dots,n$ and individuals
$i = 1,\dots,N$
compute the fitness $f(\mathbf{x}_i)$ for $i = 1,\dots,N$
while (stop criteria are not satisfied) do
    apply elitism
    while (the new population is not complete) do
        select two individuals from the current population
        according to their fitness
        apply crossover with probability $p_c$
        copy the parameter $\upsilon_{i_g}$ from the parents to each
        offspring for all gene $g = 1,\dots,n$
        apply mutation with probability $\gamma_{i_g}$ (Eq. 1) in each
        gene $g = 1,\dots,n$ of each offspring
        compute the fitness of each offspring
        compute the new values of $\upsilon_{i_g}$ for each offspring and
        each parent (eqs. 2-3)
    end while
    update generation
end while
```

The GAGDM can be slower than the standard GA in the beginning, because the search produced by the mutation operator will be concentrated in those regions related to the mutated genes that produced offspring with higher fitness (and, sometimes, have already converged to the desired value). To avoid this problem, the parameter β is initialized with a value equal to zero and converges to a value $\beta_v > 0$ across the eras. Thus, the mutation probabilities of all genes are equal in the initial era, and are gradually modified later. Here, β is updated according to

$$\beta = \beta_v(1 - e^{\hat{k}/\sigma}), \qquad (4)$$

where $\hat{k} = 0, 1, \dots$ is an estimate of the index of the era and σ controls the update of β. The index of the era (k) is not directly available for the algorithm. However, it can be estimated by monitoring the fitness of the best individual in the population (e.g., an estimate of the index of the era can be increased every time that the fitness of the best individual is significantly decreased).

The GAGDM can be more effective than the standard GA for a class of NSPs where only some genes should be modified after a change in the problem, because the search in the solution space by the mutation operator is concentrated into regions associated with the genes with larger mutation probabilities [33]. As a result, the expected adaptation speed is, in general, higher for the GAGDM for the considered class of NSPs. The search, however, is not restricted to such regions, as all mutation probabilities are always larger than zero. Thus, the GAGDM can modify the mutation probabilities in order to adapt the individuals to changes that require a modification of the current configuration of mutation probabilities. As all the mutation probabilities are larger than zero, the mutation probabilities can be adapted to this new scenario. However, in this case, the adaptation will be slower for the GAGDM, when compared to the standard GA. Besides, the performance of GAGDM is generally worse in cases where the mutation probabilities of several genes are high, since it can lead to instability.

5. Parameter settings

The maximum and minimum mutation probabilities for each gene can be computed by Eq. (1). The minimum probability is reached if $\upsilon_{i_g} = \upsilon_{\min}$ and $\upsilon_{i_j} = \upsilon_{\max}$ for $j = 1, \ldots, n$; $j \neq g$. In this way, the minimum probability is given by

$$\gamma_{\min} = np_m \frac{\upsilon_{\min}}{\upsilon_{\min} + (n-1)\upsilon_{\max}}, \tag{5}$$

where $0 < \gamma_{\min} < p_m < 1$, $n > 1$, and $\upsilon_{\min} > 0$. From Eq. (5), $\upsilon_{\max}$ can be computed as a function of $\gamma_{\min}$ and $\upsilon_{\min}$, and the maximum probability, which happens when $\upsilon_{i_g} = \upsilon_{\max}$ and $\upsilon_{i_j} = \upsilon_{\min}$ for $j = 1, \ldots, n$, $j \neq g$, can be computed by

$$\gamma_{\max} = np_m \frac{\upsilon_{\max}}{\upsilon_{\max} + (n-1)\upsilon_{\min}} \tag{6}$$

which results in $\gamma_{\max} > p_m$. In this way, Eqs. (5) and (6) can be employed to define the parameters $\upsilon_{\max}$ and $\gamma_{\max}$ given the parameters $\upsilon_{\min}$ and $\gamma_{\min}$.

In this paper, the parameters α and β_υ are chosen in order to make the average of $\Delta\upsilon_{i_g}$ equal or larger than 0 when at least one advantageous mutation occurs in the gene i_g across one era. The maximum probability of mutation is taken into account. When only one advantageous mutation of gene i_g occurs across one era $k \geqslant 0$, then, by Eq. (2), the sum of updates caused by advantageous mutations is given by

$$\Delta\upsilon_{i_g k+} = +\beta_\upsilon \alpha 1 = +\beta_\upsilon \alpha. \tag{7}$$

When deleterious mutations occur in the gene i_g with maximum probability for $(\xi - 1)$ generations (the number of generations in one era minus 1), the expected sum of updates caused by deleterious mutations (2) can be computed as

$$\Delta\upsilon_{i_g k-} = -\alpha\gamma_{\max}(\hat{\xi} - 1), \tag{8}$$

where $\hat{\xi}$ is an estimate of the number of generations in one era, which can be estimated in the same way as $\hat{k}$. As the average of the sum of updates should be equal or greater than zero across one era, then, from Eqs. (7) and (8), the following equation can be derived

$$\beta_\upsilon \geqslant \gamma_{\max}(\hat{\xi} - 1). \tag{9}$$

The parameter α is chosen in order to make $\upsilon_{i_g} = \upsilon_{\min}$ after k_p eras when the gene i_g did not have any advantageous mutation and the mutation probability is maximum. The expected sum of negative variation (Eq. (8)) in k_p eras is given by

$$\Delta\upsilon_{i_g k_p -} = -k_p \gamma_{\max} \hat{\xi} \alpha = \upsilon_{i_g} - \upsilon_{\min} \tag{10}$$

and, then, α can be computed, assuming $\upsilon_{i_g} = \upsilon_{\max}$, by

$$\alpha = \frac{\upsilon_{\max} - \upsilon_{\min}}{k_p \gamma_{\max} \hat{\xi}}. \tag{11}$$

There are situations where the value of $\hat{\xi}$ is too high. In these cases, $\hat{\xi}$ can be reduced to a small value and, when this number of generations is reached, υ_{i_g} is kept at its current value. When a degradation is observed in the fitness of the best individual, the update of υ_{i_g} is started again and last for further $\hat{\xi}$ generations.

6. Experimental results

Experiments with two NSPs are presented in this section. First, a fault classification problem, where supervised learning and real-valued representation are employed, is addressed. Next, a simulated evolutionary robots problem with binary representation and where desired outputs are not known is addressed.

It is important to observe that GAGDM can be used with other strategies for NSPs, like the maintenance of the diversity level throughout the run and the adaptation mechanisms. In the experiments carried out, the GAGDM is compared to the standard GA, and to the standard GA with random immigrants [7]. Besides, results of the GAGDM combined with the random immigrants mechanism are presented. In the GAs with random immigrants, individuals randomly chosen in the current population are replaced by random individuals in each generation of the run. The random immigrants approach tries to maintain the diversity level of the population, which can be very useful to prepare the GA for possible fitness landscape changes. In the fault classification problem, the GAGDM is compared to the CMA-ES too.

In the CMA-ES [14], the individuals of the current population are updated according to

$$\mathbf{x}_i^{(\text{gen}+1)} \sim \mathbf{m}^{(\text{gen})} + \sigma^{(\text{gen})} \mathcal{N}(\mathbf{0}, \mathbf{C}^{(\text{gen})}) \quad \text{for } i = 1, \ldots, N, \tag{12}$$

where $\mathbf{m}^{(\text{gen})} \in \mathbb{R}^n$ is the mean value of the search distribution at generation gen, $\sim$ denotes equality in distribution, $\sigma^{(\text{gen})} \in \mathbb{R}_+$ is the step size, $\mathcal{N}(\mathbf{0}, \mathbf{C}^{(\text{gen})})$ is a multivariate normal distribution with mean $\mathbf{0} \in \mathbb{R}^n$, and covariance matrix $\mathbf{C}^{(\text{gen})} \in \mathbb{R}^{n \times n}$. In the CMA-ES, the mean $\mathbf{m}^{(\text{gen}+1)}$, or recombination point, is a weighted average of the μ selected parents with highest fitness at generation *gen*. The covariance matrix is deterministically adapted in order to increase the probability to reproduce the evolution path in the search space that have led to the current population. For a tutorial on CMA-ES, see [14], and for references on the use of CMA-ES in optimization of ANNs, see [17,18]. In this paper, the Matlab implementation of the CMA-ES developed by Hansen [15] was employed in the fault classification experiments.

6.1. Fault classification

The classification problem investigated in this section is a simplified version of a class of real process diagnosis problems [22] where ANNs have been successfully employed. In such problems, it is assumed that a process has a normal operation (class 1) condition $\mathbf{y}_0$, and n_p possible faults (classes from 2 to $n_p + 1$), represented by changes in the process parameters $\mathbf{p}$. Here, the EANN should detect and isolate the faults based on the classification of the process measurements $\mathbf{y}$ (in some cases, the residual vector is classified instead of the process measurements [32]). The effect of the variation of $\mathbf{p}$ on the process measurements $\mathbf{y}$ is given by

$$y_j = y_{0j} + \eta_j + \sum_{k=1}^{n_p} a_{jk} p_k \quad \text{for } j = 1, \ldots, n_y, \tag{13}$$

where the matrix $\mathbf{a}$ relates the effect of $\mathbf{p}$ on $\mathbf{y}$, and η_j is the Gaussian noise corresponding to the measurement y_j.

6.1.1. *Experimental design*

In order to visualize the results in a plane, the number of process measurements (n_y) is set to two. The normal operation condition is $\mathbf{y}_0 = \mathbf{0}$, the directions of the faults are given by $\mathbf{a} = [1\ 1\ ;\ 1\ -1]$, and the measurement noise η_j has normal distribution with mean zero and standard deviation 0.01. The classes are defined as:

C_1 (normal): $p_{1l} < p_1 < p_{1u}$ and $p_{2l} < p_2 < p_{2u}$;
C_2 (fault 1): $p_1 \leqslant p_{1l}$ or $p_1 \geqslant p_{1u}$;
C_3 (fault 2): $p_2 \leqslant p_{2l}$ or $p_2 \geqslant p_{2u}$.

In era 0, the constraints in the process parameters $\mathbf{p}$ are $p_{1l} = p_{2l} = -0.1$ and $p_{1u} = p_{2u} = +0.1$. The NSP is characterized by changing one of these constraints to a random value between 0.05 and 0.5 in the beginning of each era. The constraints in the process parameters can change due to several factors in a real process, like component degradation, modeling errors (in the case of the residual vector), bias, changes in the measurement noise distribution, and environmental changes (e.g. variation in the temperature or pressure).

The EANN considered in this section is a feed-forward ANN using threshold activation function in the hidden neurons and linear activation function in the output neurons. Each individual of the EA specifies one EANN with two input units, four neurons in a single hidden layer, and three output neurons. Each individual of the EA specifies one EANN and each gene determines the value of one synaptic weight. When a gene is mutated in the GAs, a random number obtained from a normal distribution with mean 0 and standard deviation 0.25 is added to its previous value. Each EANN classifies the vectors $\mathbf{y}$ from the training set, and generates the classification of the patterns. The fitness function for an individual $\mathbf{x}_i$ is given by

$$f(\mathbf{x}_i) = 1 - N_\mathrm{e}/N_\mathrm{t}, \tag{14}$$

where N_e is the number of patterns of the training set incorrectly classified and N_t is the total number of patterns.

In the GAs, the individuals are selected in each generation according to elitism and to tournament selection (with 2 individuals, where the individual with the highest fitness is select with probability 0.75). The two-point crossover operator is used, $N = 50$ (population size), $p_c = 0.3$ (crossover rate), and $p_m = 0.06$ (mutation rate). In the GAGDM, $\upsilon_{\min} = 1$, $\upsilon_{\max} = 103.8077$, $\alpha = 0.3052$, and $\beta_\upsilon = 32.388$. In the GAs with random immigrants (RIs), three individuals randomly chosen are replaced by randomly generated individuals in every generation. In the CMA-ES, $\sigma^{(0)} = 0.03$ and the default strategy parameters [14] are used (the population size is set to 13).

Ten eras with 1300 function evaluation each are executed for each experiment. The number of function evaluations is adopted to define an era because the population sizes of the CMA-ES and GAs are different. In this way, even with a different number of generations is each era ($\xi = 26$ generations for the GAs and $\xi = 100$ generations for the CMA-ES), the number of function evaluations in an era is equal in all algorithms. In the beginning of each era, 1000 patterns for each class are randomly chosen from a set of 100 000 patterns generated with normal distribution (mean zero and standard deviation 0.25).

Four experiments are considered, one for each changing process parameter (p_{1l}, p_{2l}, p_{1u}, and p_{2u}). For each experiment, 10-fold cross validation is used (810 training patterns and 90 test patterns) and a generalization set with 100 patterns is used. If the fitness over the generalization set significantly decreases, the training procedure in the current era is stopped. While the fitness values over the training patterns are employed to evaluate the individuals during the evolutionary process, the fitness values over the test patterns are recorded to evaluate the quality of the classification (the fitness over the test set is not employed during the evolutionary process). The algorithms are executed ten times for each data partition. In this way, each one of the four experiments is executed 100 times for each EA. It is important to observe that the training, generalization, and test sets are generated after each change in the process parameters. In order to compare the results produced by all algorithms, for each experiment, the individuals obtained by the standard GA in the end of era 0 compose the initial population of era 1 for all GAs. For the CMA-ES, the best individual produced by the standard GA in the end of era 0 is employed as the initial search point for the era 1.

6.1.2. *Results*

The comparison of the results obtained from two algorithms in NSPs is more complex than the same comparison in stationary problems [34]. In NSPs, it is necessary to evaluate not only the final result, but also the optimization process itself. The adaptability measure, which was proposed in [34] based on a previous measure from [8], is used here to evaluate the obtained results. Adaptability is computed as the difference, averaged over the entire run, between the fitness of the current best individual of each generation and the current optimum value. The lowest adaptability values indicate the best results.

Table 1 presents the adaptability values averaged over all trials of the fault classification experiments. Table 1 also illustrates the mean fitness for all individuals in the GAs and the mean fitness of the best individual for the test sets generated in the end of the corresponding eras averaged over all trials. The fitness over the test sets is computed applying Eq. (14) to the best individual in the end of each era, where the errors are taken over the test patterns. The best results for the fitness are those with the largest values. The results

Table 1
Results—experiments for each changing process parameter p_{1u}, p_{1l}, p_{2u}, and p_{2l}

	EA	p_{1u}	p_{1l}	p_{2u}	p_{2l}
Adaptability (best individual)	Standard	0.05825	0.06620	0.06043	0.05116
	GAGDM	0.05287	0.05810	0.05489	0.04242
	Standard with RIs	0.06008	0.06800	0.06013	0.05216
	GAGDM with RIs	0.05690	0.06096	0.05704	0.04646
	CMA-ES	0.03614	0.03883	0.03717	0.03735
Fitness for the test sets in the end of the era (best individual)	Standard	0.94436	0.93964	0.94243	0.95682
	GAGDM	0.95004	0.94927	0.95066	0.96267
	Standard with RIs	0.94346	0.93828	0.94401	0.95599
	GAGDM with RIs	0.94524	0.94642	0.94748	0.95829
	CMA-ES	0.97263	0.97120	0.96877	0.96902
Mean fitness (all individuals)	Standard	0.65855	0.66037	0.65393	0.65496
	GAGDM	0.70425	0.70636	0.69264	0.71210
	Standard with RIs	0.63830	0.63876	0.67097	0.63746
	GAGDM with RIs	0.67550	0.68141	0.80753	0.68586

obtained by running a hypothesis tests (z-test) indicate that the adaptability values are smaller for GAGDM when compared to the standard GA and to the standard GA with RIs with significance level equal to 0.98. The comparison of the optimization results achieved by the investigated techniques show that, for this data set, CMA-ES presented the best results for the adaptability and fitness for the test sets.

In the EANNs evolved in these experiments, decision boundaries generated by the four hyperplanes created by the hidden neurons are used to separate the classes. Fig. 1 respectively shows the hyperplanes created by the hidden neurons of the EANNs generated by the best individual in the end of two different eras (for GAGDM) in the first trial of the experiment with the changing process parameter p_{1u}. One can observe that the changes in the location of the hyperplanes 1, 2, and 4 are small, while the change in the location of hyperplane 3 are large.

In all experiments, one hyperplane presented, during the evolutionary process, larger displacements in its location due to the changes in the problem as only one process parameter changes during the optimization process. As a result, the GAGDM increased more the mutation probabilities of those genes linked to the synaptic weights associated with this hyperplane. For example, the third hyperplane of the EANN associated with the best individual of the experiment with the changing process parameter p_{1u} generated the decision boundary with the larger displacements in its location. The occurrence of several advantageous mutations in the genes associated with the third hyperplane resulted in an increase in the mutation probabilities of the corresponding genes (Fig. 2). In this way, the number of mutations in genes corresponding to the synaptic weights associated with the third hyperplane became higher than those for the other genes. It is important to observe that the mutation probabilities of other genes were increased with a smaller intensity because of the adjustment of the hyperplane locations due to the measurement noise. The results are similar for the other

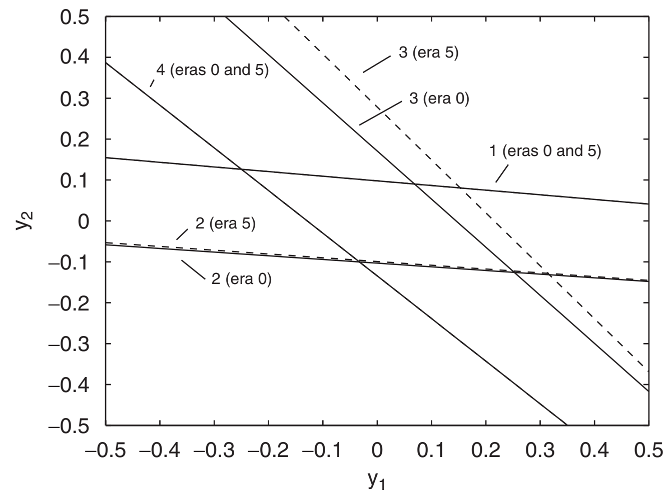

Fig. 1. Hyperplanes of the EANNs generated by the best individual in the end of eras 0 and 5 (EANN for the changing process parameter p_{1u}) generated in the first partition of the first trial). The hyperplane i is created by the hidden neuron "i".

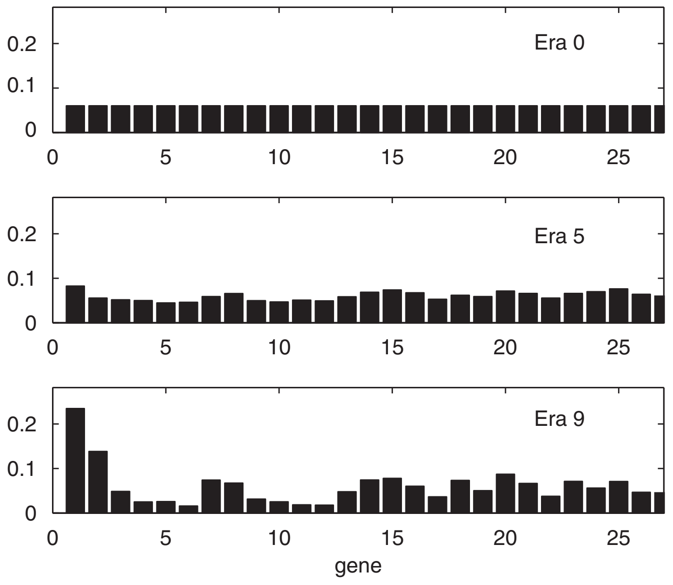

Fig. 2. Mean mutation probabilities of the best individual in the end of eras 0, 5, and 9 (the values were averaged over all trials of the experiment with the changing process parameter p_{1u}). The genes from 1 to 3 are associated with the third hyperplane.

experiments, but with other hyperplanes presenting larger displacements across the eras.

In order to compare the effects of the mutations in the components of $\mathbf{x}_i$ in the GAGDM to those in the CMA-ES, the multivariate normal distribution presented in Eq. (12) [14] is re-written as

$$\mathcal{N}(\mathbf{0}, \mathbf{C}) \sim \mathbf{C}^{1/2} \mathcal{N}(\mathbf{0}, \mathbf{I}), \tag{15}$$

where, for simplicity, the generation index is not shown and $\mathbf{I}$ denotes the identity matrix. Using the Schwarz Inequality, the following equation can be written

$$|\mathbf{c}_g^{1/2} \mathcal{N}(\mathbf{0}, \mathbf{I})| \leqslant \|\mathbf{c}_g^{1/2}\| \, \|\mathcal{N}(\mathbf{0}, \mathbf{I})\|, \tag{16}$$

where $\|.\|$ denotes the Euclidean norm and $\mathbf{c}_g^{1/2}$ is the gth line of $\mathbf{C}^{1/2}$. By analyzing the matrix $\mathbf{C}$ and $\|\mathbf{c}_g^{1/2}\|$ (see Fig. 3), it is possible to observe how the vector $\mathcal{N}(\mathbf{0},\mathbf{I})$, which produces an isotropic distribution, influences each parameter of $\mathbf{x}_i$.

One can observe in Fig. 3, that the elements with the largest values are located in the main diagonal of the mean final covariance matrix $\mathbf{C}$, indicating a small dependency among the components of the update vector. One can still observe that the components $\|\mathbf{c}_g^{1/2}\|$ related to the synaptic weights associated with the third hyperplane present larger values, indicating a orientation of the search distribution in the search space. During the search process, the changes in the components related to the synaptic weights associated with the third hyperplane improved the fitness. In this way, the matrix $\mathbf{C}$ was updated in order to modify the search distribution in a way that this direction of change presents larger variance. This result is similar to those of the GAGDM (Fig. 2). However, while the GAGDM increases the number of mutations in the genes whose mutations have improved the adaptability in the NSP, the CMA-ES increases the variance of the change sizes in those genes. One can remember that the variance of the step sizes does not change during the evolutionary process in the GAGDM. Therefore, more mutations are needed in the GAGDM, which explains the better results of the CMA-ES.

6.2. Evolutionary robotics

Robots in which artificial evolution is used as a fundamental form of adaptation or design are known as evolutionary robots [26]. In the experiments presented in this section, mobile robots are simulated in NSPs using a modified version of the Evorobot simulator developed by Nolfi [26]. In the simulator employed in the experiments presented in this section, the robots are controlled by a recurrent ANN (Elman network) with synaptic weights adjusted by GAs.

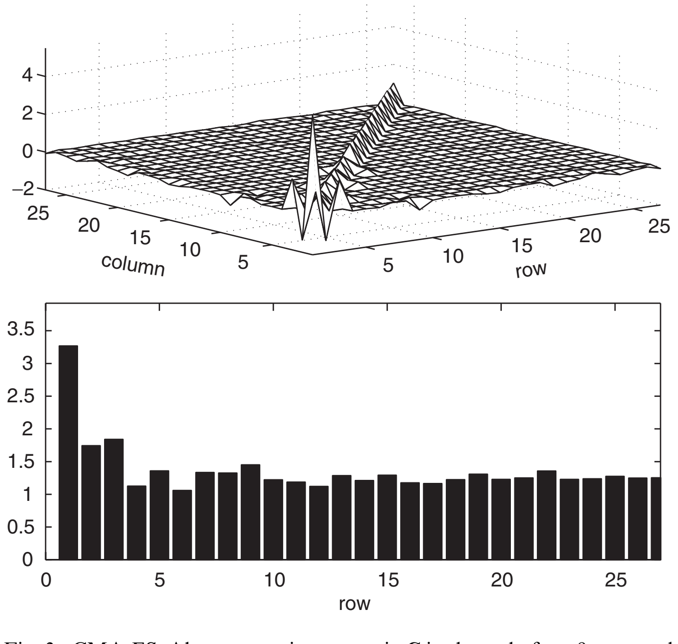

Fig. 3. CMA-ES. Above: covariance matrix $\mathbf{C}$ in the end of era 9 averaged over all trials in the experiment with the changing process parameter p_{1u}. Below: Euclidean norm of the rows of $\mathbf{C}^{1/2}$.

The experiments presented here are inspired in the experiment proposed by Floreano and Mondada [10], where a Khepera robot with eight infrared distance sensors (six in one side and two in another side of the robot), two ambient light sensors, and one floor brightness sensor should navigate in an arena of 40×45 cm. 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.

6.2.1. Experimental design

In the experiments presented in this section, the fitness function is given by the accumulated average rotation speed of the two wheels of the robot during its life time. While the battery has energy and while it does not crash into a wall or an obstacle, a maximum limit of 60 s is assumed (a fully charged battery allows the robot to move for 20 s). 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 and periodically return to the battery recharge area, while exploring the arena without hitting the obstacles, accumulate more fitness. The ANN used 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).

Experiments with 2000 generations each are presented in this section. In these experiments, the environment where the robot is evolving is modified after a fixed number of generations. Environment changing frequently occurs in real problems, where some aspects of the environment are frequently modified. Besides, ANNs are frequently evolved in simulations to avoid damage to the robot and, when a satisfactory behavior is reached, they are transferred to real robots. In the experiments presented here, the environments switches between an arena of 40×45 cm free of obstacles and an arena of 60×35 cm with five cylindrical obstacles in each $\xi = 100$ generations. In every function evaluation, the initial orientation of the robot is randomly chosen.

At each run, the individuals of the initial population are randomly selected. The individuals are represented by a vector of bits where each byte encodes one synaptic weights of the ANN. Three experiments are considered, where traditional bit mutation is used and crossover is not employed. In each generation of the first two experiments (Experiments 1 and 2), the five best individuals are selected and each one generates four offspring (population size $N = 20$). In Experiment 1, the mutation rate (p_m) is equal to 0.01, while $p_m = 0.02$ in Experiment 2. In Experiment 3, $p_m = 0.01$, and the 20 best individuals are selected and each one generates five offspring ($N = 100$).

In the GAGDM, $\upsilon_{\min} = 1$, $\upsilon_{\max} = 20.02$, $\alpha = 0.0485$, and $\beta = 38.8631$ in Experiment 1, $\upsilon_{\min} = 1$, $\upsilon_{\max} = 20.02$, $\alpha = 0.0969$, and $\beta = 19.431$ in Experiment 2, and $\upsilon_{\min} = 1$, $\upsilon_{\max} = 2.0011$, $\alpha = 0.0501$, and $\beta = 1.979$ in Experiment 3. In the GAs with RIs, 2 individuals randomly chosen are replaced by randomly generated individuals in the beginning of each generation.

6.2.2. Results

Table 2 presents the adaptability (supposing a maximum fitness equal to 1.0) and the mean fitness of all individuals of the population averaged over 50 trials for Experiments 1–3. In Fig. 4, the averaged fitness for the standard GA and for the GAGDM are shown for Experiment 2. Most of the times, a new solution is found after some generations, allowing the robot to navigate in the environment and to return to the battery recharge area only when the battery level is low. When the environment changes, the fitness values of the robots become low (Fig. 4), and a new solution is searched. This behavior is repeated in every 100 generations.

Table 2
Results—evolutionary robotics

	GA	Experiment 1	Experiment 2	Experiment 3
Adaptability	Standard	0.67455	0.75408	0.47280
(best individual)	GAGDM	0.65483	0.73905	0.43897
	Standard with RIs	0.67751	0.81240	0.39326
	GAGDM with RIs	0.73256	0.79401	0.45598
Mean fitness	Standard	0.18256	0.12284	0.25242
(all individuals)	GAGDM	0.18207	0.12669	0.26064
	Standard with RIs	0.18506	0.10603	0.30752
	GAGDM with RIs	0.14937	0.10825	0.27145

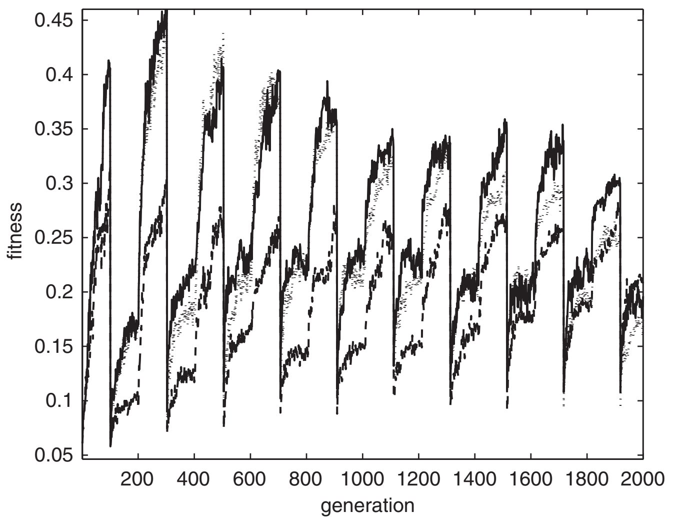

Fig. 4. Averaged fitness of the best individual in Experiment 2. The results for the standard GA (dotted lines), standard GA with RIs (dashed lines), and for the GAGDM (solid lines) are shown.

Hypothesis tests (z-test) indicate that the values of adaptability are smaller for GAGDM when compared to the standard GA with significance equal to 0.68 for Experiment 1, 0.66 for Experiment 2, and 0.70 for Experiment 3. When compared to the GA with RIs, the GAGDM averaged adaptability is smaller for Experiments 1 and 2 with, respectively, significance level equal to 0.80 and 0.94 for Experiment 2. However, it is higher for Experiment 3 with significance level equal to 0.88, indicating a better performance of the GA with RIs in this experiment.

6.3. Analysis of the results

In the GAGDM, the knowledge obtained during the evolutionary process is used to modify the mutation probabilities of the genes. In the experiments carried out, the mutation probabilities of the set of genes whose advantageous mutations have improved the adaptability in the NSP became higher. In this way, the search in the solution space has been concentrated into regions associated with higher mutation probabilities. As a result, the mean fitness of the best individuals and their fitness for the test sets were generally higher for the GAGDM, as can be observed in the results.

In the fault classification experiments, the mutation probabilities of almost all the genes were modified because small adjustments of the hyperplane locations were necessary due to the measurement noise. However, the changes in the classification space required larger adjustments of only one hyperplane in each experiment. In this way, the mutation probabilities of the genes associated with this hyperplane were higher. In the evolutionary robots experiments, a similar effect occurred. Some genes presented higher mutation probabilities than others, indicating that the modification of the synaptic weights of the ANN related to these genes improved the performance of the robots after the environmental changes.

A drawback of the GAGDM is the use of mutations with spherical (isotropic) distribution in the experiments with real-valued NSPs. In this class of optimization problems, which includes the fault classification problem presented in this paper, the CMA-ES is a very interesting option. The reason is that CMA-ES changes the search distribution in order to increase the variance of the changes in the direction where advantageous mutations have occurred after modifications in the problem.

Another negative effect in the GAGDM was the decrease in diversity, which generally resulted in mean fitness with higher values for the individuals of the GAGDM in the experiments (see Tables 1 and 2). The diversity decreased in the GAGDM because the mutations frequently occurred in genes with higher mutation probabilities. The RIs mechanism presented the smaller mean fitness for all individuals because the diversity of the populations was increased.

The diversity of the solutions, which is the main advantage of the RIs mechanism, can be occasionally used by the crossover operator to find the new optimum.

The higher diversity level explains the best results for the GAs with RIs in the third evolutionary robots experiment. In this experiment, new solutions produced by the RIs mechanism resulted in better solutions for the problem when the environments changed. In such experiments, the best approach was to find new solutions different from the previous ones, instead of finding solutions related to them. In this case, even presenting better results when compared to the standard GA, the GAGDM was not the best technique. However, the results found by the GAs with RIs were worse than the solutions found by the GAGDM in the first two evolutionary robots experiments, where the size of the population was smaller (20 in Experiments 1 and 2, and 100 in Experiment 3). In Experiments 1 and 2, the selective pressure was too high, and, then, the new individuals generated by the RIs approach, which generally presents small values of fitness, could not evolve. In spite of the better results for the experiment with the population of 100 individuals, very often, real experiments with large populations are not feasible. In a real experiment, the computation of the fitness of a population of 100 individuals would require 100 min, if one considers that the evaluation of each robot takes 1 min, while 20 min would be required for a population of 20 individuals. Moreover, compared to the standard GA, the GA with RIs requires more computational time to compute the fitness of the random individuals inserted in every generation.

7. Conclusions

In this work, a GA with gene dependent mutation probabilities (GAGDM) applied to the weight adjustment of EANNs in NSPs is investigated. In the GAGDM, each gene of each individual is associated with an independent mutation probability. The knowledge obtained during the evolutionary process is employed to modify the mutation probabilities of the genes. If the modification of a set of genes improves the adaptation of the individual to the modifications of the NSP, the mutation probabilities of the genes in this particular set are increased.

It is important to observe that the GAGDM generally improves adaptation in a class of NSPs where only some genes should be modified after changes in the problem. In EANNs, GAGDM can improve adaptation when few synaptic weights should be modified after a change in the problem. In these cases, some genes have their mutation probabilities increased, and the search in the solution space by the mutation operator is concentrated into regions associated with such genes. As a result, the expected adaptation speed is, in general, higher for the GAGDM, when compared to the standard GA, for the class of NSPs considered.

In the fault classification experiments performed, the authors compared GAGDM with another evolutionary approach, CMA-ES, which presented better classification results. The better results of the CMA-ES can be explained by how the algorithms implement the mutation. While the GAGDM increases the number of mutations in the genes whose mutations have improved the adaptability in the class of NSPs considered, the CMA-ES increases the variance of the change sizes in those genes in real-valued optimization. The variance of the change sizes in the genes does not change in the GAGDM. In this way, the GAGDM needs more mutations in the same gene to produce the required changes in the individual. These results suggest that the CMA-ES algorithm is very efficient in the class of NSPs considered, and further work should be performed in order to test the use of CMA-ES in other NSPs with real-valued representation. It is important to mention that this superiority was obtained using a real-valued representation for the chromosomes. Different from CMA-ES, GAGDM is indicated for cases where the binary representation and traditional bit mutation are adopted. While some researchers have argued that the binary representation might not be the best in EANNs [37], the advantage of the use of real-valued or binary representation (and traditional bit mutation) should still be investigated in NSPs.

The performance of the GAGDM can be poor in cases where several genes should be modified, like the cases where the values of several synaptic weights should be adjusted. It is important to point out that the presence of noise with high variance and the use of training sets that do not represent well the problem can imply in the adjustment of several synaptic weights when a change occurs in the NSP. When several genes should be modified after a change in the NSP, diversity plays an important role and the mutation probabilities should be equal. It is important to observe that the number of genes modified after a change is not only a property of the NSP, but it is also a property of the representation employed by the GA and the architecture of the EANN.

The NSPs where the changes of the fitness function are continuous and the use of the GAGDM for EANNs used in real evolutionary robots should be studied in future works.

Acknowledgments

The authors would like to thank FAPESP (Proc. 04/04289-6) for the financial support to this project, and the Guest Editor Aluízio Araújo and the anonymous reviewers for their helpful comments.

References

[1] A. Abraham, Meta learning evolutionary artificial neural networks, Neurocomputing 56 (2004) 1–38.
[2] P.J. Angeline, G.M. Saunders, J.B. Pollack, An evolutionary algorithm that constructs recurrent neural networks, IEEE Trans. Neural Networks 5 (1) (1994) 54–65.
[3] H.-G. Beyer, Toward a theory of evolution strategies: self-adaptation, Evol. Comput. 3 (3) (1996) 311–347.

[4] H.-G. Beyer, H.S. Schwefel, Evolution strategies: a comprehensive introduction, Natural Comput. 1 (2002) 3–52.
[5] J. Branke, Evolutionary approaches to dynamic optimization problems—introduction and recent trends, in: J. Branke (Ed.), GECCO Workshop on Evolutionary Algorithm for Dynamic Optimization Problems, 2003, pp. 2–4.
[6] J. Branke, T. Kaußler, C. Schmidt, H. Schmeck, A multi-population approach to dynamic optimization problems, in: Adaptive Computing in Design and Manufacturing 2000, Springer, Berlin, 2000.
[7] H.G. Cobb, J.J. Grefenstette, Genetic algorithms for tracking changing environments, in: S. Forrest (Ed.), Proceedings of the Fifth International Conference on Genetic Algorithm, 1993, pp. 523–530.
[8] K.A. De Jong, An analysis of the behavior of a class of genetic adaptive systems, Ph.D. Dissertation, University of Michigan, 1975.
[9] A.E. Eiben, R. Hinterding, Z. Michalewicz, Parameter control in evolutionary algorithms, IEEE Trans. on Evol. Comp. 3 (2) (1999) 124–141.
[10] D. Floreano, F. Mondada, Evolution of homing navigation in a real mobile robot, IEEE Trans. on Systems, Man, and Cybernetics—Part B: Cybernet. 26 (3) (1996) 396–407.
[11] D. Gilis, S. Massar, N.J. Cerf, M. Rooman, Optimality of the genetic code with respect to protein stability and amino-acid frequencies, Genome Biol. 2 (11) (2001).
[12] D.A. Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley, Reading, MA, 1989.
[13] B.G. Hall, Increased rates of advantageous mutations in response to environmental challenges, ASM News 57 (2) (1991) 82–86.
[14] N. Hansen, The CMA evolution strategy: a tutorial, ⟨http://www.bionik.tu-berlin.de/user/niko/⟩, November 2005.
[15] N. Hansen, The CMA evolution strategy for noisy and global optimization: implementations in Matlab, ⟨http://www.bionik.tu-berlin.de/user/niko/cmaes-inmatlab.html⟩, November 2005.
[16] N. Hansen, A. Ostermeier, Completely derandomized self-adaptation in evolution strategies, Evol. Comput. 9 (2) (2001) 159–195.
[17] C. Igel, M. Kreutz, Operator adaptation in evolutionary computation and its application to structure optimization of neural networks, Neurocomputing 55 (2003) 347–361.
[18] C. Igel, S. Wiegand, F. Friedrichs, Evolutionary optimization of neural systems: the use of self-adaptation, in: M.G. de Bruin, D.H. Mache, J. Szabados (Eds.), Trends and Applications in Constructive Approximation, International Series of Numerical Mathematics, Birkhäuser, Basel, 2005, pp. 103–123.
[19] T. Isokawa, N. Matsui, H. Nishimura, F. Peper, Coping with nonstationary environments: a genetic algorithm using neutral variation, IEEE Trans. on Systems, Man, and Cybernetics—Part A: Syst. and Humans 32 (4) (2002) 497–504.
[20] Y. Jin, J. Branke, Evolutionary optimization in uncertain environments—a survey, IEEE Trans. on Evol. Comput. 9 (3) (2005) 303–317.
[21] S. Kitamura, M. Hiroyasu, Genetic algorithm with stochastic automata-controlled, relevant gene-specific mutation probabilities, in: Proceedings of the IEEE Conference on Evolutionary Computation, 1995, pp. 352–355.
[22] J.A. Leonard, M.A. Kramer, Radial basis function networks for classifying process faults, IEEE Control Syst. 11 (3) (1991) 31–38.
[23] C.G. Looney, Pattern Recognition Using Neural Networks, Oxford University Press, Oxford, 1997.
[24] T. Maeshiro, The importance of the robustness and changeability in evolutionary systems, in: Proceedings of the IEEE Conference on Systems, Man, and Cybernetics, 1998, pp. 2342–2347.
[25] M. Mandischer, A comparison of evolution strategies and backpropagation for neural network training, Neurocomputing 42 (1–4) (2002) 87–117.
[26] S. Nolfi, D. Floreano, Evolutionary Robotics: The Biology, Intelligence, and Technology of Self-organizing Machines, MIT Press/Bradford Books, Cambridge, USA, 2000.
[27] S. Nolfi, D. Parisi, Learning to adapt to changing environments in evolving neural networks, Adaptive Behavior 5 (1) (1997) 75–98.
[28] T. Sasaki, M. Tokoro, Evolving learnable neural networks under changing environments with various rates of inheritance of acquired characters: comparison between darwinian and lamarckian evolution, Artificial Life 5 (3) (1999) 203–223.
[29] M. Sebag, M. Schoenauer, C. Ravis, Inductive learning of mutation step-size in evolutionary parameter optimizations, in: P.J. Angeline, R.G. Reynolds, J.R. McDonell, R. Eberhart (Eds.), Evolutionary Programming, vol. VI, 1997, pp. 247–261.
[30] R.S. Sexton, J.N.D. Gupta, Comparative evaluation of genetic algorithm and backpropagation for training neural networks, Inform. Sci. 129 (2000) 45–59.
[31] P.D. Stroud, Kalman-extended genetic algorithm for search in nonstationary environments with noisy fitness evaluations, IEEE Trans. on Evol. Comput. 5 (1) (2001) 66–77.
[32] M.H. Terra, R. Tinós, Fault detection and isolation in robotic manipulators via neural networks—a comparison among three architectures for residual analysis, J. Robotic Syst. 18 (7) (2001) 357–374.
[33] R. Tinós, A.C.P.L.F. Carvalho, A genetic algorithm with gene dependent mutation probability for non-stationary optimization problems, in: Proceedings of the 2004 Congress on Evolutionary Composition (CEC'2004), vol. 2, 2004, pp. 1278–1285.
[34] K. Trojanowski, Z. Michalewicz, Evolutionary algorithms for non-stationary environments, in: M.A. Klopotek, M. Michalewicz (Eds.), Intelligent Information Systems, Proceedings of the Eighth International Workshop on Intelligent Information Systems (IIS'99), 1999, pp. 229–240.
[35] D. Wicker, M.M. Rizki, L.A. Tamburino, E-net: Evolutionary neural network synthesis, Neurocomputing 42 (1–4) (2003) 171–196.
[36] C.O. Wilke, C. Adami, Evolution of mutational robustness, Mut. Res. 522 (2003) 3–11.
[37] X. Yao, Evolving artificial neural networks, Proc. IEEE 87 (9) (1999) 1423–1447.

Renato Tinós received the B.S. degree in electrical engineering from State University of São Paulo (UNESP), Brazil, in 1994, and the M.S. and Ph.D. degree in electrical engineering from the University of São Paulo (USP) at São Carlos, Brazil, in 1999 and 2003, respectively. He then joined the Department of Computer Science of USP at São Carlos as a Research Scientist. Since 2004 he is Assistant Professor in the Department of Physics and Mathematics of USP at Ribeirão Preto. His research interests include evolutionary algorithms, dynamic optimization, robotics, fault tolerance, and neural networks.

A.C.P.L.F. de Carvalho received his B.Sc. degree in Computer Science and M.Sc. degree in Informatics both from the Federal University of Pernambuco, Brazil. He received his Ph.D. degree in Electronic Engineering from the University of Kent at Canterbury, UK. Dr. de Carvalho is an Associate Professor at the Department of Computer Science, University of São Paulo, Brazil. In 2000, Prof. André de Carvalho was Associate Professor in the University of Guelph, Canada. He has published two books and over 200 papers in refereed Conferences and Journals. Dr. de Carvalho is a Founding Co-Editor of the International Journal of Computational Intelligence and Applications, IJCIA, published by Imperial College Press and World Scientific. His main interests are Machine Learning, Neural Networks, Genetic Algorithms, Hybrid Intelligent Systems, Data Mining and Bioinformatics.