CORE
Metadata, citation and similar papers at core.ac.uk
Provided by Brunel University Research Archive

Self-Adaptation of Mutation Distribution in Evolution Strategies for Dynamic Optimization Problems

Renato Tinós*,Shengxiang Yang**

*Department of Computing and Mathematics, FFCLRP, University of São Paulo (USP)

14040-901, Ribeirão Preto, SP, Brazil, E-mail: rtinos@ffclrp.usp.br

**Department of Information Systems and Computing, Brunel University

Uxbridge, Middlesex UB8 3PH, U.K., E-mail: shengxiang.yang@brunel.ac.uk

Abstract

Evolution strategies with q-Gaussian mutation, which allows the self-adaptation of the mutation distribution shape, is proposed for dynamic optimization problems in this paper. In the proposed method, a real parameter q, which allows to smoothly control the shape of the mutation distribution, is encoded in the chromosome of the individuals and is allowed to evolve. In the experimental study, the q-Gaussian mutation is compared to Gaussian and Cauchy mutation on experiments generated from the simulation of evolutionary robots and on dynamic optimization problems generated by the Moving Peaks generator.

1 Introduction

Dynamic optimization problems (DOPs) have attracted increasing attention from the evolutionary computation community in recent years. In DOPs, the evaluation function, the representation of the solution, the dimension of the search space, and/or the constraints of the problem may change during the optimization process [3], which represents a great challenge for traditional *evolutionary algorithms* (EAs). Hence, new approaches have to be developed to enhance the performance of EAs to deal with DOPs.

The simplest approach to deal with a problem change is to restart the optimization process, a procedure used in many DOPs (e.g., see [8]). However, for many DOPs, a new solution must be found in a short period after a change, which is generally impossible when the optimization procedure requires a substantial computational effort. When the new solution after the change in the problem is related to the past solutions, the search procedure based on previous solutions can save substantial processing time.

Another common approach is to use adaptation mechanisms to deal with the changes in DOPs. The self-adaptation mechanism used in *evolution strategies* (ESs), which allows to modify the parameters of the mutation operator during the run, represents an intrinsic mechanism for adaptation to eventual changes, which makes the use of ESs interesting for DOPs [1, 9, 18]. ESs were proposed in the 1960's to optimize candidate solutions composed of real-valued parameters [2], and represent a good choice when gradient based methods present bad performance due to rugged fitness landscapes in continuous optimization problems. Similar to other EAs, new candidate solutions are generated in ESs by using a stochastic mutation.

Traditionally in ESs, the Gaussian distribution is used to generate new candidate solutions by mutation [2]. However, in recent years, researchers have argued that the use of mutation distributions with longer tails and infinite second moment in ESs can be useful in allowing the population escape from local optima in multimodal problems. For example, in [19], the Cauchy distribution was employed to generate new candidate solutions. The use of mutation taken from heavy tail distributions implies jumps of scale-free sizes, which eventually allows to reach distant regions of the search space faster. This property is interesting for DOPs too, as it can enable the population to escape faster from local optima located close to the best solution before the change and to explore new promising regions of the search space. However, when mutation distributions with longer tails are employed, less local candidate solutions are generated, and the convergence to the new optima may be slower. Thus, the use of one or other mutation distribution may result in very different performance on a DOP.

In this paper, the use of the q-Gaussian mutation in ESs to address DOPs is proposed. In the q-Gaussian mutation, which was previously used in evolutionary programming for stationary optimization problems [16] and DOPs [15], self-adaptation is employed, not only to control the mutation strength parameter, but also to control the mutation distribution. The q-Gaussian distribution allows to smoothly control the shape of the distribution by setting a real parameter q and can reproduce either finite second moment distributions, like the Gaussian distribution, or infinite second moment distributions, like the Cauchy distribution. In EAs with q-Gaussian mutation, the real parameter q is encoded in the chromosome of the individuals and is allowed to evolve. Thus, in the q-Gaussian mutation, the decision on which mutation distribution shape should be used (and when) is made by self-adaptation. The main contributions of this paper are as follows: (1) the use of the q-Gaussian mutation in ESs is proposed for DOPs; (2) ESs with Gaussian, Cauchy, and q-Gaussian mutations are compared in experiments based on the Moving Peaks Benchmark (MPB) DOP generator [3] and on simulations of evolutionary robots.

The rest of this paper is organized as follows. The q-Gaussian mutation is discussed in Section 2. The ES with q-Gaussian mutation is presented in Section 3. The experimental study with DOPs generated by the MPB generator and from the simulation of evolutionary robots is presented in Section 4. Finally, the conclusions of the paper are presented in Section 5.

2 The q-Gaussian Mutation

When mutation is applied in ESs, the i-th candidate solution $\vec{\tilde{x}}_i$ is generated from an m-dimensional solution $\vec{x}_i$ according to:

$$\vec{\tilde{x}}_i = \vec{x}_i + \mathbf{C}\ \vec{z}, \tag{1}$$

where $\vec{z}$ is an m-dimensional vector generated from a random distribution with zero mean and the matrix $\mathbf{C}$, in the standard ES, is a diagonal matrix, composed of the elements of vector $\vec{\sigma} = [\sigma(1)\ \sigma(2) \ldots \sigma(m)]^{\mathrm{T}}$, which defines the mutation strength in each coordinate of the search space. For the q-Gaussian mutation, the vector $\vec{z}$ is generated by sampling m independent q-Gaussian random variables, i.e., the q-Gaussian distribution is employed instead of the Gaussian distribution (Gaussian mutation) [2] or the Cauchy distribution (Cauchy mutation) [19].

When $-\infty < q < 3$, the q-Gaussian distribution density [11] is given by:

$$p_{q(\bar{\mu}_q, \bar{\sigma}_q)}(x) = \frac{\sqrt{B_q}}{A_q}\ e_q^{-B_q (x-\bar{\mu}_q)^2}, \tag{2}$$

where $\bar{\mu}_q$ and $\bar{\sigma}_q$ are the q-mean and the q-variance, respectively, A_q is the normalization factor, B_q controls the width of the q-Gaussian distribution, and $e_q^{-u^2}$ (q-Gaussian function of u) is defined as follows:

$$e_q^{-u^2} \equiv \begin{cases} \left(1 + (q-1)u^2\right)^{-\frac{1}{q-1}}, & \text{if } 1 + (q-1)u^2 \geq 0 \\ 0, & \text{otherwise} \end{cases} . \tag{3}$$

In Eq. (2), the q-mean $\bar{\mu}_q$ and the q-variance $\bar{\sigma}_q$ [11] are respectively defined as follows:

$$\bar{\mu}_q \equiv \frac{\int x p(x)^q dx}{\int p(x)^q dx}, \tag{4}$$

$$\bar{\sigma}_q^2 \equiv \frac{\int (x - \bar{\mu}_q)^2 p(x)^q dx}{\int p(x)^q dx}, \tag{5}$$

and respectively reduce to the usual mean and variance when $q \to 1$. In Eq. (2), the normalization factor A_q is given by $A_q = \int_{-\infty}^{+\infty} e_q^{-(x-\bar{\mu}_q)^2} dx$ [17] and B_q is given by

$$B_q = \left((3-q)\bar{\sigma}_q^2\right)^{-1}. \tag{6}$$

Figure 1 illustrates the q-Gaussian function for different values of q. In the q-Gaussian function, the real parameter q controls the shape of the function, which allows to smoothly and continuously change the shape of the q-Gaussian distribution. For $q < 5/3$, the second order moment is finite and for $q \to 1$, the q-Gaussian distribution reproduces the usual Gaussian distribution (see Figure 1). For $q < 1$, the q-Gaussian distribution has a compact form, and decays asymptotically according to a power law for $1 < q < 3$. When $q = 2$, the q-Gaussian distribution reproduces the Cauchy distribution [10]. In this paper, the generalized Box-Müller method proposed in [11], which is very simple (see its pseudo-code in [11]) and allows to generate samples from q-Gaussian distributions for $-\infty < q < 3$, is employed to generate q-Gaussian random variables.

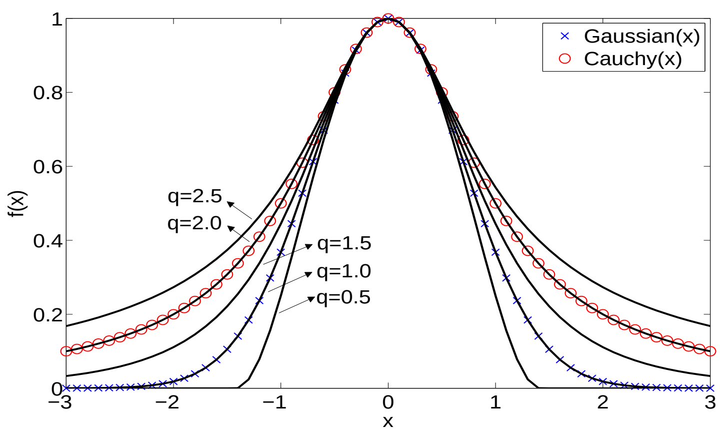

Figure 1: The q-Gaussian function for different values of q. The Gaussian and Cauchy functions are also illustrated.

3 ES with q-Gaussian Mutation

In ESs, two main selection procedures are usually employed. In the (μ, λ)-ES, a population of μ parents creates $\lambda > \mu$ offspring. The best μ offspring are then selected to compose the next population. In the $(\mu + \lambda)$-ES, the new population is composed of the best μ individuals obtained from the union of the μ parents and λ offspring. It can be observed that while the $(\mu + \lambda)$-ES is elitism-based, i.e., it always preserves the best individuals from one generation to the next one, the (μ, λ)-ES is not elitism-based. In this way, for DOPs, a procedure to detect the changes in the problem should be used when the $(\mu + \lambda)$-ES is employed. Thus, the (μ, λ)-ES is frequently used in DOPs.

Generally in ESs, the mutation strength parameter of each element $j = 1, \dots, m$ of the vector $\vec{\sigma}_i$ is updated as follows:

$$\tilde{\sigma}_i(j) = \sigma_i(j) e^{\tau_b \mathcal{N}(0,1)_i + \tau_c \mathcal{N}(0,1)}, \tag{7}$$

where τ_b denotes the standard deviation of the Gaussian distribution used to generate the random deviation $\mathcal{N}(0,1)_i$, which is common for all elements of the vector $\vec{\sigma}_i$, and τ_c is the standard deviation of the Gaussian distribution used to generate the separated random deviation $\mathcal{N}(0,1)$ for each element $j = 1, \dots, m$. The parameters τ_b and τ_c, as suggested by theoretical and empirical works [2], are defined by $\tau_b = \frac{k_b}{\sqrt{2m}}$ and $\tau_c = \frac{k_c}{\sqrt{2\sqrt{m}}}$, where k_b and k_c are positive real numbers.

In this paper, the use of the q-Gaussian mutation, as described in the previous section, in (μ, λ)-ES is proposed for DOPs. Based on the mutation strength self-adaptation [2], the parameter q_i of each individual i is added to its chromosome and is multiplicatively updated. The parameter q_i of each individual i is modified as follows:

$$\tilde{q}_i = q_i e^{\tau_q \mathcal{N}(0,1)}, \tag{8}$$

where τ_q denotes the standard deviation of the Gaussian distribution and $\mathcal{N}(0,1)$ denotes a sample vari-

Algorithm 1 (μ,λ)-ES with q-Gaussian mutation (qGES)

```
1: Initialize the population of individuals $(\vec{x}_k, \vec{\sigma}_k, q_k)$ for $k = 1, \ldots, \mu$
2: Evaluate the individuals $(\vec{x}_k, \vec{\sigma}_k, q_k)$ for $k = 1, \ldots, \mu$
3: while (stop criteria are not satisfied) do
4:     Use recombination to generate the individuals $(\vec{\tilde{x}}_i, \vec{\tilde{\sigma}}_i, \tilde{q}_i)$ for $i = 1, \ldots, \lambda$ from the individuals $(\vec{x}_k, \vec{\sigma}_k, q_k)$ for $k = 1, \ldots, \mu$
5:     for $i \leftarrow 1$ to $\lambda$ do
6:         if $rand(0, 1) \geq r_q$ then
7:             Update the mutation strength vector $\vec{\tilde{\sigma}}_i$ according to Eq. (7).
8:         else
9:             Update the parameter $\tilde{q}_i$ according to Eq. (8)
10:        end if
11:        $\vec{\tilde{x}}_i \leftarrow \vec{\tilde{x}}_i + \mathbf{C}_i \; \vec{z}$, where $\vec{z}$ is a vector generated from $q$-Gaussian distribution with parameter $\tilde{q}_i$ and $\mathbf{C}_i = \mathrm{diag}(\vec{\tilde{\sigma}}_i^{\mathrm{T}})$
12:    end for
13:    Evaluate the offspring $(\vec{\tilde{x}}_i, \vec{\tilde{\sigma}}_i, \tilde{q}_i)$ for $i = 1, \ldots, \lambda$
14:    Select the best $\mu$ individuals from the offspring $(\vec{\tilde{x}}_i, \vec{\tilde{\sigma}}_i, \tilde{q}_i)$, $i = 1, \ldots, \lambda$, to compose the new population with individuals
       $(\vec{x}_k, \vec{\sigma}_k, q_k)$, $k = 1, \ldots, \mu$
15: end while
```

able taken from the Gaussian distribution with zero mean and standard deviation one. This way, different distributions can be reproduced during different stages of the evolutionary process.

Here, τ_q in Eq. (8) is given by:

$$\tau_q = \frac{k_q}{\sqrt{2m}}, \tag{9}$$

where k_q is a positive real number. As it is not possible to identify the separated influence of a change in $\vec{\sigma}_i$ or q_i in the fitness of individual i if both mutation strength vector and parameter q are mutated together for individual i (e.g., a beneficial mutation in $\vec{\sigma}_i$ can be masked in the fitness of individual i if the parameter q_i is mutated to a bad value in the same generation), only $\vec{\sigma}_i$ or q_i are allowed to mutate in each generation. Thus, the mutation strength vector $\vec{\sigma}_i$ is updated for individual i in each generation if a uniform random number in the range $[0, 1]$ is equal to or larger than a real parameter $r_q \in [0, 1]$; otherwise, the value of q_i is updated.

The (μ,λ)-ES with q-Gaussian mutation, called *qGES* in this paper, is presented in Algorithm 1. The main difference of the ES presented in Algorithm 1 from the standard ES and the fast ES [19] lies in that, in qGES, the q-Gaussian mutation is employed (step 11) instead of the Gaussian mutation (in the standard ES) or Cauchy mutation (in the fast ES), and a procedure to update the parameter q is adopted (steps 6, 8, 9 and 10).

4 Experimental Study

In order to evaluate the performance of the q-Gaussian mutation in ESs for DOPs, two sets of experiments are carried out. In the first set of experiments, the MPB generator [3] is employed to create DOPs (see

Section 4.1). In the second set of experiments, evolutionary robots are simulated in dynamic environments (see Section 4.2). In each experiment, three ESs were executed 50 times (with different random seeds). In the first algorithm, qGES, the parameter q of the q-Gaussian mutation is allowed to evolve during the optimization process (see Algorithm 1). In the other two algorithms, the parameter q is fixed, i.e., it starts with a given value and is not modified during the evolutionary process. In algorithm GES, $q = 1$, i.e., the q-Gaussian distribution reproduces the Gaussian distribution. In algorithm CES, $q = 2$, i.e., the Cauchy distribution is reproduced by the q-Gaussian distribution. Thus, the three types of mutation, q-Gaussian, Gaussian, and Cauchy are compared in the experiments.

For each run, the individuals of the initial population were randomly chosen. In all experiments, the initial q-Gaussian parameter q in qGES was set to 1.0 (a value where the Gaussian distribution is reproduced), $r_q = 0.8$, and the minimum and maximum values of the q-Gaussian parameter q were respectively set to 0.8 and 2.2, i.e., values respectively smaller and higher than the values of q where the Gaussian and Cauchy mutations are reproduced. The parameters k_b and k_c, respectively used to compute τ_b and τ_c, were set to 1 (as suggested by theoretical and empirical works), while in Eq. (9), $k_q = 1.5$ in the evolutionary robots experiments and $k_q = 1.0$ in the MPB generator experiments (a higher value of k_q was chosen for the evolutionary robots experiments because the individuals in those experiments are longer). Experiments, not presented here, with different parameter settings showed similar results to those presented in the following sections.

4.1 The Moving Peaks Benchmark (MPB) Generator

The MPB generator was suggested by Branke [3] in order to generate continuous dynamic optimization problems where the search space is composed by peaks with changeable parameters. A similar DOP generator was proposed by Morrison [6]. In the MPB generator, n_p peaks defined in the m-dimensional space are used to compute the fitness function according to:

$$f(\mathbf{x}, e) = \max\left(B(\mathbf{x}), \max_{i=1,\ldots,n_p} P(\mathbf{x}, h_i(e), w_i(e), \mathbf{p}_i(e)) \right), \tag{10}$$

where $e = \lceil t/\tau \rceil$ is the index of the change cycle (period where the fitness function is stationary between two changes), τ is the number of generations in a change cycle (τ remains constant during the evolutionary process), $B(\mathbf{x})$ is a stationary function (base) and $P(\mathbf{x}, h_i(e), w_i(e), \mathbf{p}_i(e))$ is a function that defines the shape of peak i with its height, width, and position in change cycle e given by $h_i(e)$, $w_i(e)$, and $\mathbf{p}_i(e)$, respectively.

Here, the parameters $h_i(e)$ and $w_i(e)$, for $e > 1$, are respectively changed according to:

$$h_i(e+1) = h_i(e) + \rho\, \mathcal{N}(0,1), \tag{11}$$

and

$$w_i(e+1) = w_i(e) + \rho\, \mathcal{N}(0,1), \tag{12}$$

while the vector defining the position of the center of each peak is changed by rotation, multiplying the position vector in each change cycle e by a rotation matrix composed of successive simple rotations in random planes, as in the continuous generator proposed in [13]. The angles of the rotation matrix, in degrees, are given by 180ρ. While the parameter ρ controls the degree of changes, the parameter τ controls the velocity of changes. For example, Figure 2 shows a 2-dimensional landscape that was changed by the MPB generator with $\rho = 0.2$. It can be observed that the height, width, and position of the peaks were changed in this illustration.

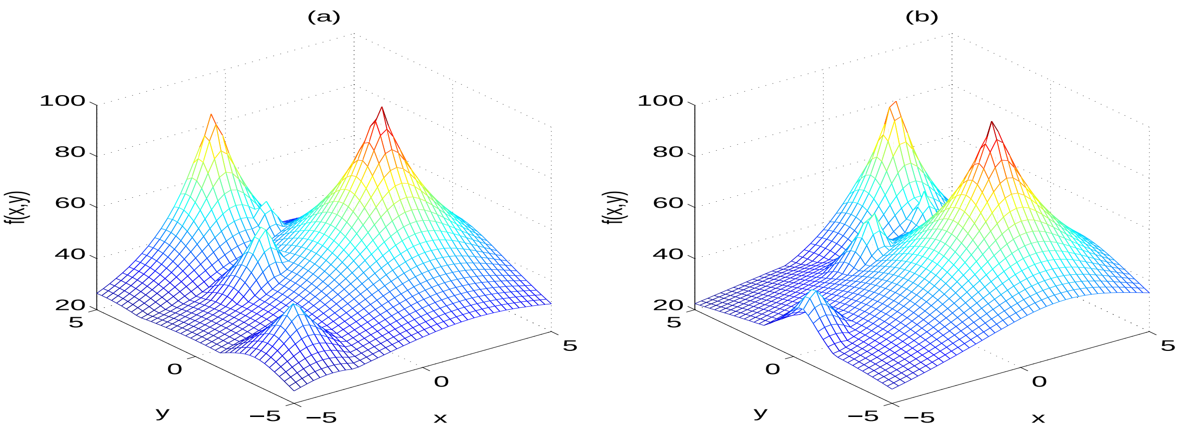

Figure 2: Two landscapes generated by the Moving Peaks Benchmark generator. The first landscape (a) is changed with $\rho = 0.2$ into the second landscape (b).

In this section, experiments with 5 fixed values of ρ, i.e., where ρ does not change during the run, are presented. Experiments where the value of ρ changes during the run according to a chaotic dynamics (using a discrete logistic function with parameter 3.67) are also presented (such experiments are identified as $\rho = c$ in the tables and figures). Experiments with $\tau = 10$, 50, and 300 are presented. This way, 18 experiments with different values of ρ and τ are generated, i.e., the algorithms are tested with different combinations of degree and velocity of the changes. For all experiments, the number of changes (n_c) during the evolutionary process was set to 50, $m = 10$, and the number of peaks is $n_p = 10$. For the algorithms, the number of parents (μ) was set to 15, while the number of offspring (λ) was 100. Intermediate recombination with two parents for each offspring, where the variable values of the offspring are computed as the mean of the variable values of the parents, was used.

In order to evaluate the performance of the algorithms, the mean best fitness error found in each change cycle is computed. The mean best fitness error reached in each change cycle for run j is given by:

$$\bar{e}_j = \frac{1}{n_c} \sum_{i=1}^{n_c} e^*_{ij} \qquad (13)$$

where n_c is the number of changes in run j, $e^*_{ij} = f^o_{ij} - f^*_{ij}$ is the best fitness error found in change cycle i for run j, and f^*_{ij} and f^o_{ij} are, respectively, the fitness of the best individual and the fitness at the global optimum

Table 1: Experimental results of the mean error of the best fitness in each change cycle (Eq. (13)) obtained in the Moving Peaks experiment (Experiment 1) over 50 runs (the best values for the median are in bold).

		GES			CES			qGES		
τ	ρ	Median	Mean	STD	Median	Mean	STD	Median	Mean	STD
10	0.1	**4.00E+00**	4.37E+00	1.99E+00	4.42E+00	4.46E+00	1.81E+00	4.05E+00	4.10E+00	1.80E+00
10	0.2	**4.47E+00**	4.68E+00	1.64E+00	4.62E+00	4.66E+00	1.40E+00	4.56E+00	4.69E+00	1.58E+00
10	0.3	4.49E+00	4.75E+00	1.32E+00	4.73E+00	4.93E+00	1.10E+00	**4.33E+00**	4.67E+00	1.17E+00
10	0.4	5.41E+00	5.62E+00	1.30E+00	5.52E+00	5.82E+00	1.28E+00	**5.35E+00**	5.62E+00	1.27E+00
10	0.5	7.14E+00	7.63E+00	2.12E+00	**7.05E+00**	7.54E+00	2.24E+00	7.07E+00	7.71E+00	2.27E+00
10	c	8.36E+00	8.64E+00	2.41E+00	**7.84E+00**	8.31E+00	2.45E+00	8.44E+00	8.61E+00	2.37E+00
50	0.1	4.11E+00	3.52E+00	1.72E+00	**3.51E+00**	3.34E+00	1.61E+00	3.69E+00	3.35E+00	1.73E+00
50	0.2	2.95E+00	3.23E+00	1.70E+00	2.77E+00	2.81E+00	1.48E+00	**2.70E+00**	2.88E+00	1.58E+00
50	0.3	1.74E+00	1.86E+00	1.17E+00	1.66E+00	1.74E+00	9.65E-01	**1.26E+00**	1.42E+00	7.96E-01
50	0.4	1.46E+00	1.77E+00	1.23E+00	1.44E+00	1.66E+00	1.03E+00	**1.40E+00**	1.62E+00	9.45E-01
50	0.5	2.46E+00	2.85E+00	1.84E+00	2.44E+00	2.85E+00	1.85E+00	**2.35E+00**	2.70E+00	1.69E+00
50	c	3.22E+00	3.86E+00	2.30E+00	3.10E+00	3.74E+00	2.28E+00	**2.86E+00**	3.70E+00	2.33E+00
300	0.1	3.69E+00	3.46E+00	1.95E+00	3.59E+00	3.15E+00	1.80E+00	**3.13E+00**	3.07E+00	1.91E+00
300	0.2	3.01E+00	2.93E+00	1.63E+00	2.51E+00	2.58E+00	1.45E+00	**2.45E+00**	2.65E+00	1.61E+00
300	0.3	1.36E+00	1.49E+00	9.68E-01	1.12E+00	1.33E+00	9.04E-01	**9.86E-01**	1.34E+00	8.72E-01
300	0.4	1.22E+00	1.57E+00	1.17E+00	1.22E+00	1.42E+00	1.08E+00	**1.20E+00**	1.47E+00	1.11E+00
300	0.5	1.91E+00	2.43E+00	1.71E+00	1.93E+00	2.48E+00	1.66E+00	**1.69E+00**	2.42E+00	1.71E+00
300	c	2.51E+00	3.17E+00	2.19E+00	2.57E+00	3.21E+00	2.17E+00	**2.42E+00**	3.15E+00	2.23E+00

in change cycle i for run j. The objective of the optimization process is to minimize the error $\bar{e}_j$.

4.1.1 Experimental Results

Table 1 presents the results regarding the mean best fitness error found in each change cycle for the experiment with DOPs created by the MPB generator (Experiment 1), where better results mean smaller fitness errors. The median, mean, and standard deviation for the results over 50 runs for each combination of τ and ρ (including the case for changing ρ, which is identified by $\rho = c$) and for each algorithm are presented in Table 1. In Figure 3, the median of the mean best fitness error found in each change cycle for each algorithm for different values of τ and ρ are presented. The averaged Euclidean norm of the mutation strength parameter vector and distribution parameter q of the best individual in the last 10 change cycles of the MPB experiment for $\tau = 300$ and $\rho = 0.1$ or $\rho = 0.2$ are shown in Figure 4.

In Table 2, the statistic comparison of the algorithms regarding the mean best fitness error found in each change cycle (Eq. (13)) is carried out by the Wilcoxon Signed Rank Test [5]. Table 2 shows the p-value of the Wilcoxon Signed Rank Test for each experiment, which indicates the significance for testing the null hypothesis that the difference between the matched samples of the results regarding Alg. X and Alg. Y comes from a distribution with median equal to zero. For each experiment, the result regarding the comparison Alg. X - Alg. Y is shown, in parentheses, as "=" when the values of the median of Alg. X and Alg. Y are

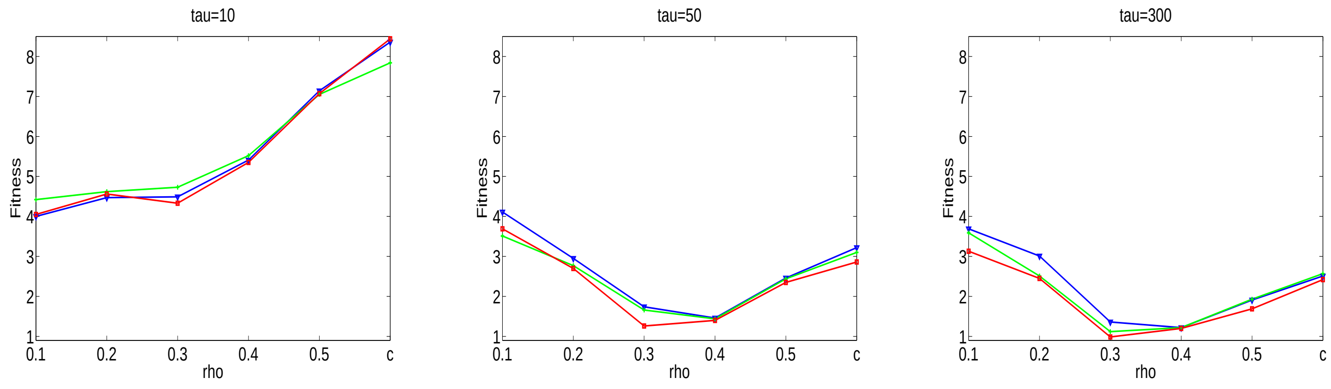

Figure 3: Median of the mean error of the best fitness in each change cycle (Eq. (13)) in the Moving Peaks experiment (Experiment 1) for different values of ρ (GES: blue, triangle; CES: green, plus; qGES: red, square).

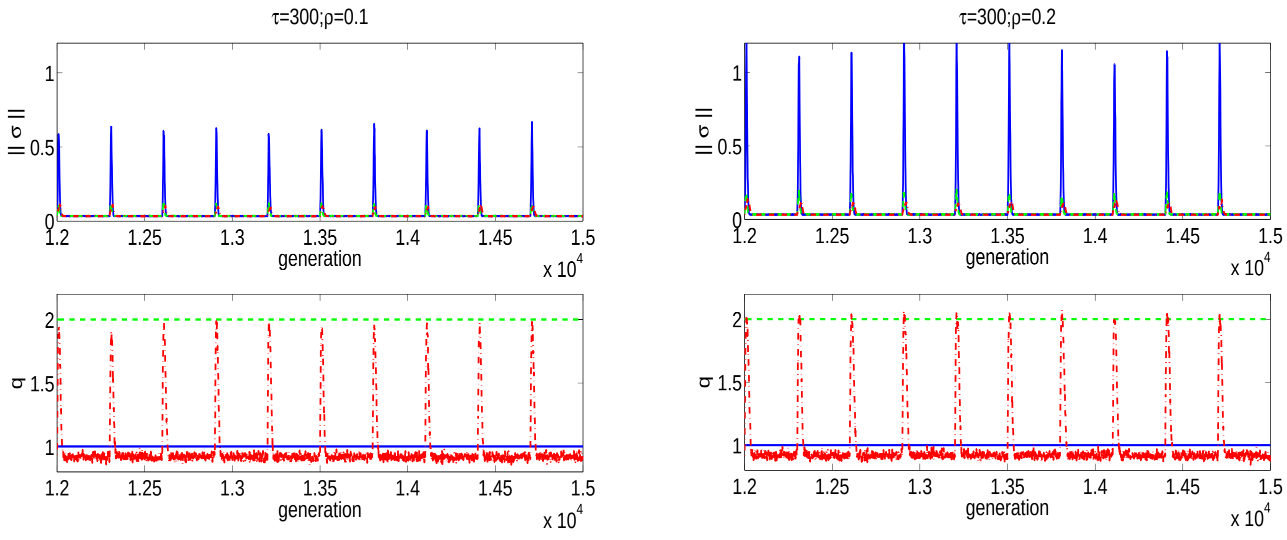

Figure 4: Average Euclidean norm of the mutation strength parameter vector and distribution parameter q of the best individual in the last 10 change cycles of the Moving Peaks experiment (Experiment 1) for $\tau = 300$ and $\rho = 0.1$ or $\rho = 0.2$ (GES: blue, solid; CES: green, dashed; qGES: red, dashdot).

equal. When the values of the median are different but the p-value is higher than 0.05, i.e., the test indicates that the hypothesis that the median of the difference between the results is zero cannot be rejected at the 5% level, the result is respectively shown as "+" when the median of Alg. X is smaller than the median of Alg. Y and "−" when the median of Alg. X is higher than the median of Alg. Y. Otherwise, when the result is statistically significant, the result is shown as "$s+$" or "$s-$" when the median of Alg. X is smaller or higher than the median of Alg. Y, respectively.

4.1.2 Analysis of the Results

It can be observed from Figure 4 that for the three approaches (with Gaussian, Cauchy, and q-Gaussian mutations), the mutation strength parameter increases when the environment changes and then converges to small values after the change. When the change is severer (for $\rho = 0.2$), the maximum value reached by the

Table 2: Statistical comparisons regarding the error of the best fitness in each change cycle for the algorithms in the Moving Peaks experiment (Experiment 1).

τ	ρ	qGES - GES	qGES - CES
10	0.1	5.02E-01 $(-)$	2.18E-03 $(s+)$
10	0.2	3.04E-01 $(-)$	6.40E-01 $(+)$
10	0.3	6.40E-01 $(+)$	2.31E-04 $(s+)$
10	0.4	8.36E-01 $(+)$	4.43E-06 $(s+)$
10	0.5	1.81E-01 $(+)$	1.60E-02 $(s-)$
10	c	6.89E-01 $(-)$	1.64E-04 $(s-)$
50	0.1	1.97E-01 $(+)$	2.74E-02 $(s-)$
50	0.2	6.59E-02 $(+)$	9.04E-01 $(+)$
50	0.3	1.44E-02 $(s+)$	1.28E-03 $(s+)$
50	0.4	9.59E-02 $(+)$	7.83E-01 $(+)$
50	0.5	2.48E-02 $(s+)$	2.36E-02 $(s+)$
50	c	9.02E-03 $(s+)$	3.04E-01 $(+)$
300	0.1	8.31E-07 $(s+)$	2.19E-02 $(s+)$
300	0.2	1.05E-03 $(s+)$	7.98E-01 $(+)$
300	0.3	2.24E-02 $(s+)$	9.88E-01 $(+)$
300	0.4	2.36E-02 $(s+)$	3.27E-01 $(+)$
300	0.5	6.12E-01 $(+)$	1.38E-01 $(+)$
300	c	8.36E-01 $(+)$	5.78E-02 $(+)$

norm of the mutation strength parameter is higher. Such a behavior occurs in other stages of the evolutionary process and for other runs, too. Increasing the mutation strength parameter's norm implies that mutation distributions have a larger second order moment (note that the mutation strength parameters define the standard deviation in the Gaussian mutation distribution). When the mutation strength parameters increase after an environmental change, larger jumps occur more often, which can help the population to reach new optima faster. Then, smaller mutation strength parameters are selected in order to increase the search close to the local neighborhood of the current best solution.

Additionally, in the ES with q-Gaussian mutation, the parameter q increases after an environmental change, which results in distributions with heavier tails. Then, smaller values of q are selected, resulting in more compact distributions, and increasing the search close to the local neighborhood of the current best solution. It can be observed that, similar to the mutation strength parameters, the maximum value reached by q for qGES in Figure 4 is higher when the change is severer (for $\rho = 0.2$). Such results confirm the hypothesis that self-adaptation in ES is useful for DOPs as it provides an intrinsic mechanism of adaptation to the changes of the problem.

Let us now analyze the results of the algorithms with Gaussian and Cauchy mutations. It can be observed from Figure 3 that the Gaussian mutation presents better results than the Cauchy mutation for $\tau = 10$ (fast changing environment) when $\rho < 0.5$. When $\tau = 10$, the algorithms do not have enough time to explore new peaks after a change, and most of the advantageous mutations occur due to small jumps in the search space (hill-climbing process). As the Gaussian distribution produces more small jumps than the Cauchy mutation, the Gaussian mutation generates better performance when compared to the Cauchy mutation.

For larger τ, the algorithms have more time to explore other peaks and, then, larger jumps can eventually allow the candidate solutions to jump from the current peak to a new promising peak. As the Cauchy distribution generates more larger jumps than the Gaussian distribution, it can allow the population to escape from local optima faster than the Gaussian mutation, mainly in the later stages of the evolution where the mutation strength parameters have converged to small values. It can be observed in Figure 3 and Table 1 that CES presents better performance than GES for $\tau > 10$, mainly for $\tau = 300$.

In Figure 3, it can also be observed that for $\tau = 50$ and $\tau = 300$, different from the results for $\tau = 10$, the performance of the algorithms are better for ρ in the range $0 < \rho < 0.4$, and worse for larger ρ ($\rho \geq 0.4$), i.e., the performance curve is U-shaped for $\tau = 50$ and $\tau = 300$. In the MPB generator employed here, larger ρ implies larger mean difference between the fitness of the current best solution after the change and the height of the highest peaks (Eq. (11)) and, as a consequence, a larger volume of the region of the search space with fitness larger than the fitness of the current best solution. Then, the probability of jumping from the current peak to a promising peak increases with ρ due to the larger changes in the height of the peaks. This explains the decrease of the fitness error in the range $0 < \rho < 0.4$ for $\tau = 50$ and $\tau = 300$, when the algorithms have more time to explore new regions of the search space by larger jumps (one can observe that the results for $\tau = 300$ are better than the results for $\tau = 50$). However, larger ρ implies also larger changes in the positions of the centers of peaks, and more time to reach the center of the current peak. This explains the shape of the curve for $\tau = 10$, when the algorithms do not have enough time to explore the current peak, and for the range $\rho \geq 0.4$, where the changes in the positions of the peaks are very large.

In the experiments with the q-Gaussian mutation, distributions with heavier tails (i.e., higher values of q) are used by the algorithm just after the changes and compact distributions are used in the later stages after the changes. It can be observed that the mean value of q in Figure 4 reaches values close to or higher than 2.0 (Cauchy distribution) and decreases to values smaller than 1.0 (Gaussian distribution) after the changes. This way, larger steps occur more often in the generations just after the changes, which helps the population to escape from local optima or to converge faster. In the later generations, the distribution with small values of q enhances local search, which helps the algorithm to reach the best local optima. In the proposed algorithm (qGES), higher values of the q parameter are employed on some occasions to allow longer jumps between two peaks.

The ES with q-Gaussian mutation presents a performance better than or similar to the GES algorithm when the Gaussian mutation is better than Cauchy mutation, and better or similar to the CES when the Cauchy mutation is better than the Gaussian mutation. This behaviour is reached by the self-adaptation of the parameter q of the q-Gaussian mutation distribution

4.2 Evolutionary Robots

In the experiments presented in this section, DOPs are generated through the simulation of evolutionary mobile robots navigating in dynamic environments with or without faults. In evolutionary robotics, artificial evolution is the fundamental force in the adaptation and design of robots and their control laws [7]. Particularly here, ESs are employed to adjust the synaptic weights in an Elman *artificial neural network* (ANN) used to control simulated mobile robots [12]. In the experiments, the robots are simulated in dynamic environments using a modified version of the Evorobot simulator developed by Nolfi [7].

The four experiments presented in this section are generated from the experiment proposed in [4], 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 circular and is indicated by a different color of the floor. The arena is rectangular and has a light source mounted in a tower in one of its walls or corners.

In the experiments, 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 ANN used to control the robot has 17 inputs (8 infrared sensors, 2 light sensors, 1 floor brightness sensor, 1 sensor for the battery energy, and 5 recurrent units), 5 hidden neurons, and 2 outputs (2 motors coupled to the wheels of the robot). The architecture of the ANN is illustrated in Figure 5.

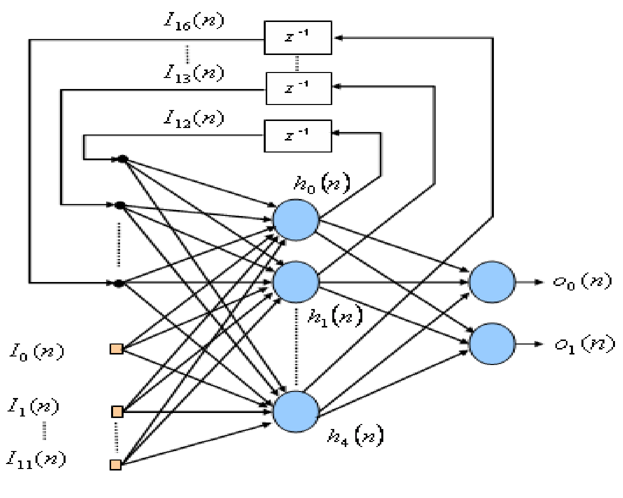

Figure 5: Artificial neural network used to control the robot.

In the first three experiments (Experiment 2, Experiment 3, and Experiment 4), we are interested in investigating the reconfiguration of the robot after faults [12]. In these experiments, the environment where the robot evolves is switched every $\tau = 10$ (fast changing problem), 40, or 70 (slow changing problem) generations between two configurations. In both configurations, the size of the arena is 40cm×45cm. The first configuration (default arena) is free of obstacles. In the second configuration, the position of the light source and the recharge area are changed, and a cylindrical obstacle is added.

In Experiment 2 (faults in the light sensors), the responses of the light sensors are reduced by a factor changed every τ generations. In Experiment 3 (faults in motor 2), the power of the second motor of the robot is reduced by a factor changed in each τ generations. The factor applied in the response of the light sensors (Experiment 2) and in the power of the second motor (Experiment 3) in each one of the 10 change cycles (including the first τ generations) is given by $\vec{v}_f = [1.0\ 0.5\ 0.2\ 0.9\ 0.7\ 0.4\ 0.8\ 0.6\ 0.1\ 0.7]^{\mathrm{T}}$, e.g., only 50% of the power computed by the respective ANN's output is applied to the second motor in the second change cycle (between generations $\tau + 1$ and 2τ) of Experiment 3.

In Experiment 4 (faults in the infrared sensors), we are interested in investigating the reconfiguration after intermittent faults in the infrared sensors. During the evolutionary process, the responses of the infrared sensors of the robots are affected by two faults, which are switched every $\tau = 10$, 40, or 70 generations. In the first fault, the responses of the six infrared sensors located in one side of the robot are set to zero when it is affected by the fault. In the second fault, the responses of the remaining two sensors (located in the other side) are set to zero. This way, the robot should learn how to navigate using different sets of sensors in each change cycle.

The last experiment (Experiment 5) is carried out to investigate how a changing environment affects the learning process (faults are not simulated in the experiment). Environmental changes frequently occur in real world 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 ANN employed to control the simulated robot is transferred to the real one. In Experiment 5 (changing environment), the environment where the robot is evolving is changed after $\tau = 10$, 40, or 70 generations. The robot evolves for the first τ generations in the default arena, which is changed in its dimensions, configuration and in the number of cylindrical obstacles present in the environment every τ generations.

For all experiments, the number of changes during the evolutionary process is ten. The evolving robot always starts in a fixed position on the arena, but the initial orientation is randomly varied in a range of 10 degrees. A white noise with a range equal to 0.05 is added to the measures generated by the infrared and light sensors. The individuals are represented by a vector of integer values corresponding to the synaptic weights of the ANN. Following [4], in each generation, the 20 best individuals (μ) are selected and each one generates 5 children ($\lambda = 100$) by mutation. Recombination is not used in the evolutionary robot experiments because

Table 3: Mean best fitness found in each change cycle obtained in the Evolutionary Robots experiments for algorithms GES, CES, and qGES over 50 runs (the best values for the median are in bold).

		GES			CES			qGES		
Experiment	τ	Median	Mean	STD	Median	Mean	STD	Median	Mean	STD
2	10	5.80E-01	5.26E-01	1.89E-01	3.83E-01	4.09E-01	2.11E-01	**5.89E-01**	5.40E-01	1.75E-01
2	40	6.31E-01	5.89E-01	1.92E-01	6.99E-01	5.76E-01	2.33E-01	**7.22E-01**	6.66E-01	1.50E-01
2	70	7.13E-01	6.04E-01	1.97E-01	**7.47E-01**	6.23E-01	2.01E-01	7.00E-01	6.47E-01	1.61E-01
3	10	4.17E-01	3.69E-01	1.38E-01	3.63E-01	3.20E-01	1.20E-01	**4.39E-01**	3.92E-01	1.23E-01
3	40	4.75E-01	4.36E-01	1.35E-01	4.85E-01	4.06E-01	1.69E-01	**4.92E-01**	4.79E-01	1.11E-01
3	70	4.21E-01	3.87E-01	1.40E-01	4.95E-01	4.30E-01	1.48E-01	**5.02E-01**	4.59E-01	1.15E-01
4	10	2.35E-01	2.74E-01	1.34E-01	2.08E-01	2.57E-01	1.32E-01	**2.76E-0**1	2.87E-01	1.18E-01
4	40	3.85E-01	3.61E-01	1.43E-01	3.50E-01	3.99E-01	1.57E-01	**3.99E-01**	4.00E-01	1.61E-01
4	70	3.49E-01	3.54E-01	1.45E-01	**4.67E-01**	4.67E-01	1.33E-01	3.82E-01	3.85E-01	1.58E-01
5	10	**5.15E-01**	4.79E-01	1.36E-01	3.83E-01	4.05E-01	1.17E-01	5.09E-01	4.87E-01	1.34E-01
5	40	5.72E-01	5.35E-01	1.26E-01	5.85E-01	5.56E-01	1.35E-01	**6.08E-01**	5.83E-01	1.07E-01
5	70	5.42E-01	5.14E-01	1.42E-01	**6.35E-01**	5.79E-01	1.31E-01	5.85E-01	5.75E-01	1.04E-01

crossing two ANNs in general does not produce good results, as observed in [4].

As the global optimum is not known in the experiments presented in this section, the mean best fitness found in each change cycle is computed in order to evaluate the performance of the algorithms. The mean best fitness reached in each change cycle for run j is given by:

$$\bar{f}_j = \frac{1}{n_c} \sum_{i=1}^{n_c} f^*_{ij} \tag{14}$$

where n_c is the number of changes in run j, and f^*_{ij} is the fitness of the best individual found in change cycle i for run j. The objective of the evolutionary process is to maximize the fitness $\bar{f}_j$ in each run j.

4.2.1 Experimental Results

Table 3 presents the results regarding the mean best fitness found in each change cycle for the experiment with the simulation of the evolutionary robot (Experiments 2, 3, 4, and 5). Better results mean higher fitness. The median, mean, and standard deviation for the results over 50 runs for each τ and for each algorithm are presented in Table 3. Figures 6, 7, and 8 show the mean best fitness in each generation averaged over 50 runs for the four experiments with τ = 10, 40, and 70 generations, respectively.

In Table 4, the statistic comparison of the algorithms regarding the mean best fitness found in each change cycle (see Eq. (14)) is carried out by the Wilcoxon Signed Rank Test. The result is respectively shown as "+" when the median of Alg. Y is smaller than the median of Alg. X and "−" when the median of Alg. Y is higher than the median of Alg. X. Otherwise, when the result is statistically significant, the result is respectively shown as "$s+$" or "$s-$" when the median of Alg. Y is smaller or higher than the median of Alg. X.

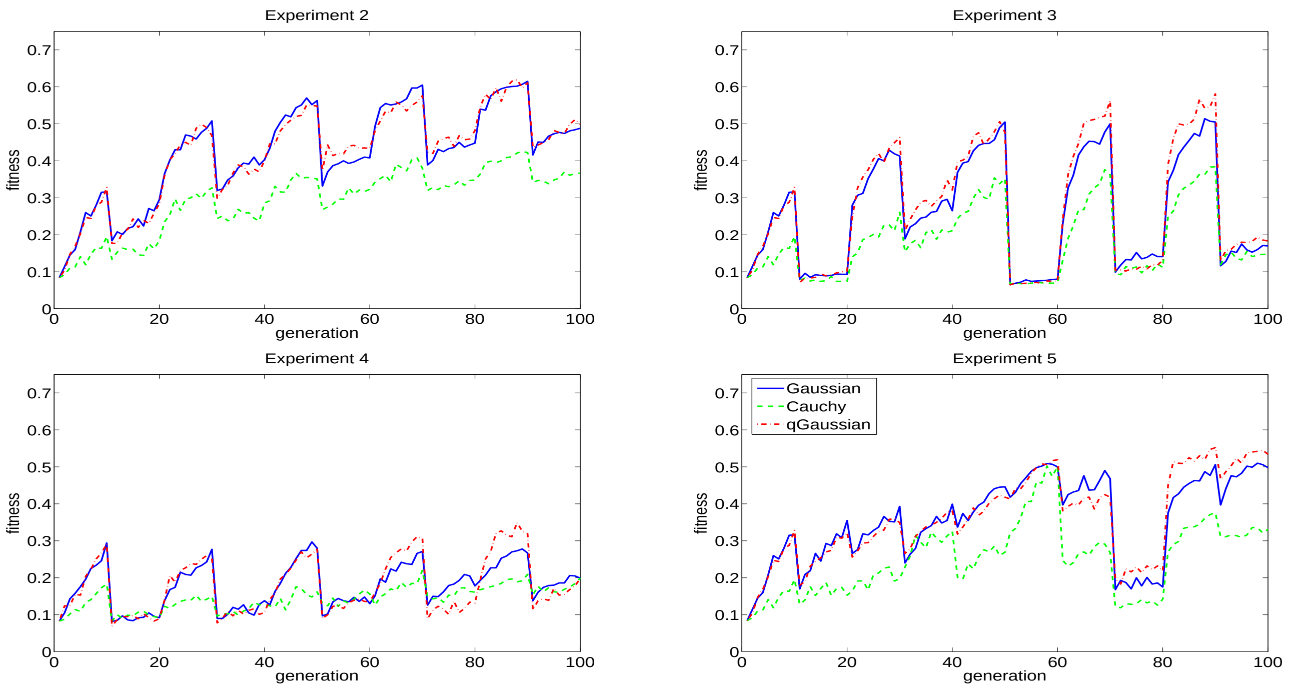

Figure 6: Average fitness of the best individual in Evolutionary Robots experiments for $\tau = 10$.

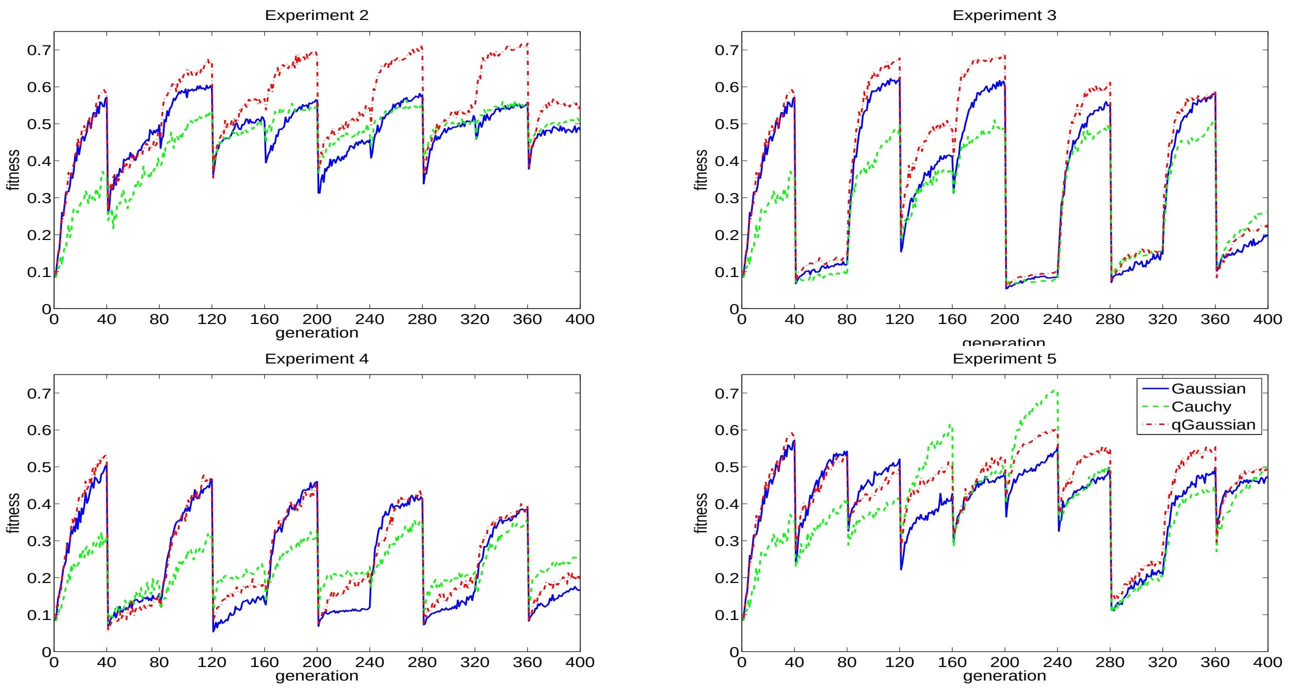

Figure 7: Average fitness of the best individual in Evolutionary Robots experiments for $\tau = 40$.

4.2.2 Analysis of the Results

In the experiments, ESs find, in the first change cycle (before the first change) of most runs, weights of the ANNs (individuals) that allow the simulated robots navigate in the environment, and periodically return to the battery recharge area when the battery charge is low. After a change, the strategy adopted by the robot is evolved in order to allow the navigation in the new environment. This behaviour can be observed in Figure 9, where two simulations of the evolutionary robot for 200 iterations of the ANN (or until the collision with

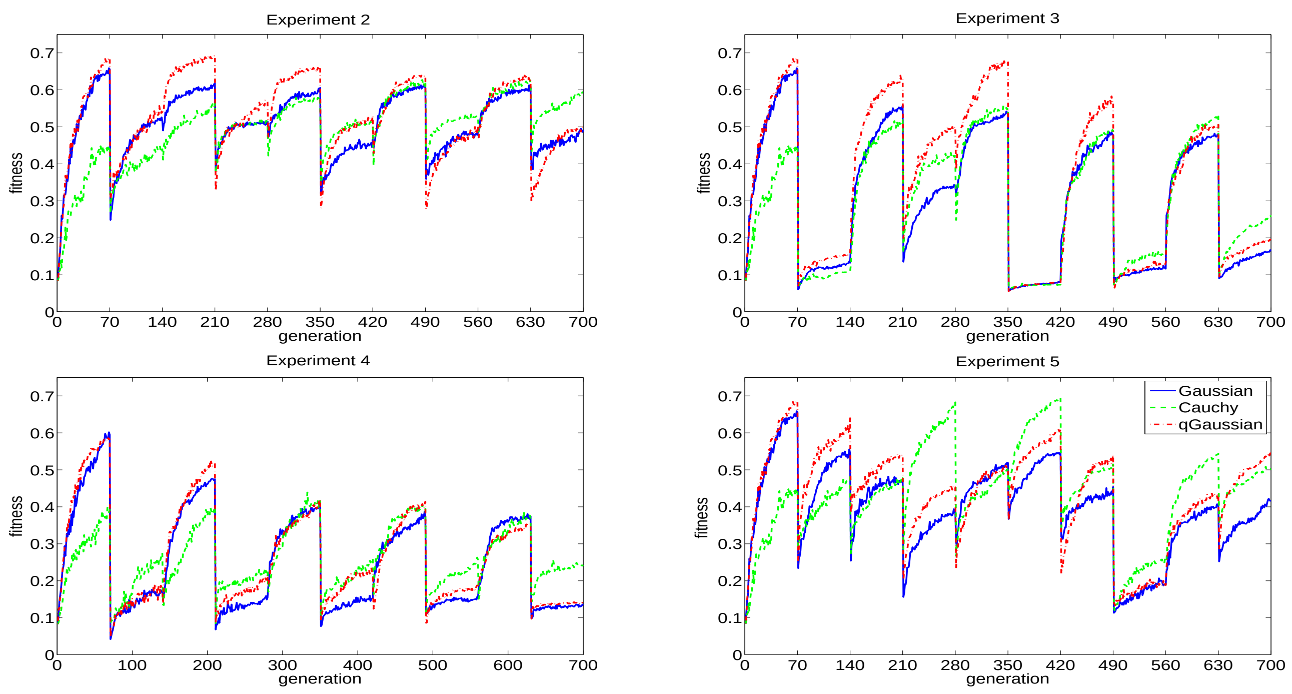

Figure 8: Average fitness of the best individual in Evolutionary Robots experiments for $\tau = 70$.

Table 4: Statistical comparisons for the mean best fitness found in each change cycle in the Evolutionary Robots experiments.

Experiment	τ	qGES - GES	qGES - CES
2	10	4.60E-01 $(+)$	2.90E-03 $(s+)$
2	40	7.83E-03 $(s+)$	9.02E-02 $(+)$
2	70	1.26E-01 $(-)$	7.76E-01 $(-)$
3	10	4.20E-01 $(+)$	5.35E-03 $(s+)$
3	40	2.74E-02 $(s+)$	2.81E-02 $(s+)$
3	70	2.15E-03 $(s+)$	4.90E-01 $(+)$
4	10	8.73E-01 $(+)$	2.26E-01 $(+)$
4	40	3.66E-02 $(s+)$	9.65E-01 $(+)$
4	70	4.04E-01 $(+)$	3.84E-03 $(s-)$
5	10	7.03E-01 $(-)$	1.33E-03 $(s+)$
5	40	1.19E-02 $(s+)$	3.57E-01 $(+)$
5	70	3.72E-03 $(s+)$	6.75E-01 $(-)$

a wall or obstacle) are presented. One can observe in Figure 9 that when the robot controlled by the weights defined by the best individual in the first change cycle, which were obtained while the robot moves in the default arena, is transferred to a new environment, the robot crashes to an obstacle before the end of the simulation. However, it is possible to observe that an individual capable of navigating in this environment can be found by the evolutionary process. One can observe that the strategy adopted by both individuals are different.

It can be observed in Figures 6-8 that GES (where the Gaussian mutation is reproduced) presents better results than CES (where the Cauchy mutation is reproduced) in the first change cycle. In the simulated

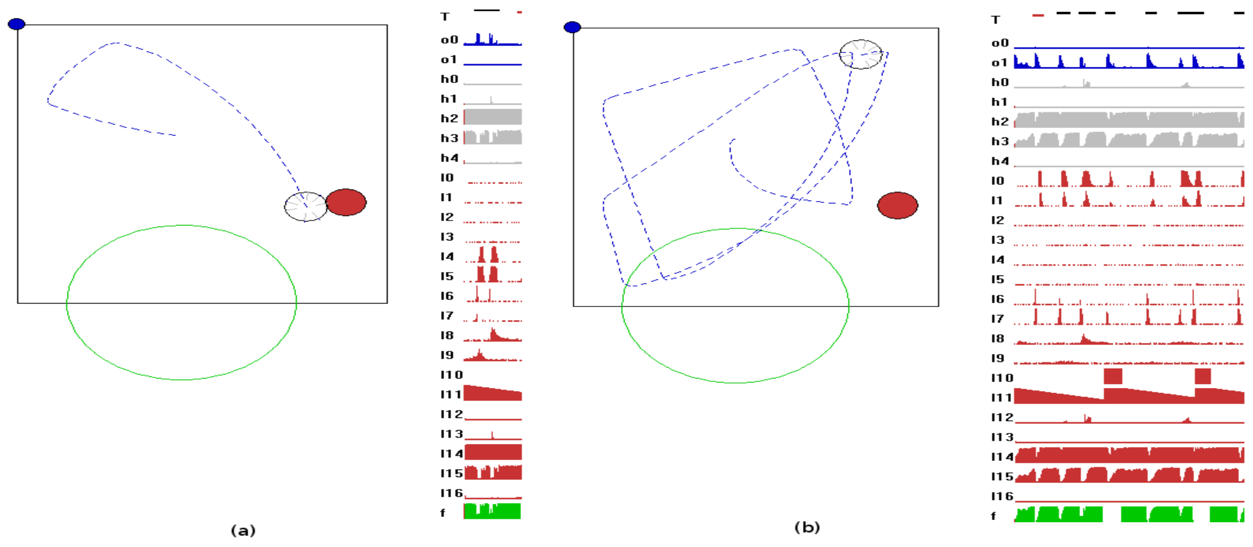

Figure 9: Two simulations of the evolutionary robot in the last environment of Experiment 5. The robot is represented by a white circle and its trajectory by a dashed line. The recharge area and light tower are, respectively, represented by a large circle with green line and by a small blue circle. The cylindrical obstacles are represented by red circles. The output signals $o0$ and $o1$ (see Figure 5), the input signals $I0$ to $I16$, and the outputs of the hidden nodes $h0$ to $h4$ during the simulations are shown in the graphics on the right of the environment representation. The first simulation is for the best individual found before the first change (a). The second simulation is for the best individual in the last generation (b). The individuals are obtained in the same run of the algorithm qGES for Experiment 5 with $\tau = 70$. The figures were generated by the Evorobot simulator.

evolutionary robot, large modifications in the vector of synaptic weights cause a large change in the current navigation strategy found by the evolutionary process. This way, compact mutation distributions produce better results in the first change cycle as more solutions are generated close to the current best solution. In qGES, smaller values of q are selected by self-adaptation during the first change cycle. Hence, the performance of qGES is close to the performance of GES in the first change cycle.

When changes occur in the environment, new navigation strategies should be found. In Experiment 2, the changes in the problem are, in general, small, as the changes in the light of the environment cause small modifications in the weights of the ANNs. As a result, it can be observed that GES presents better results than CES in most change cycles. However, it is possible to observe that CES presents better results than GES in some change cycles, where the changes are severer, and in the last change cycles for $\tau = 70$. For slower changing environments, the algorithms have more time to reach the optima and it is more difficult to escape from it using small jumps. It can be observed that qGES presents the best result in this experiment for $\tau = 10$ and $\tau = 70$. As the value of q is modified according to the problem in qGES, small values are generally selected when the changes in the problem are small (and GES presents better results than CES), while larger values of q are selected when the changes are severer (and CES presents better results than GES).

Similar results can be found in other experiments. In Experiments 3 and 4, the changes in the problems are severer, which explains the smaller values of mean final fitness found in each change cycle. Particularly in Experiment 4, the reconfiguration of the navigation strategy after the change in the problem is very difficult. The robot learns to navigate in the direction that it has more sensors (called here front of the robot). When the problem changes, all the sensors in the front of the robot are affected by the fault. This way, the robot should learn how to navigate in the new environment after losing all infrared sensors in its front. It can be observed that it is difficult for the ESs to find new navigation strategies for change cycles 2, 4, 6, 8, and 10, as the robot tries to keep the navigation strategy learnt in the first change cycle. In the figures, it is possible to observe that CES presents better results than GES in change cycles 2, 4, 6, 8, and 10 for larger τ. For larger τ, it is harder to find a new strategy for the even change cycles as the fitness value found in change cycle 1 is high and the diversity of the population is reduced due to the longer change cycle duration. This way, longer jumps occasionally occurred in order to allow escaping from the best solution before the change (local optimum). This explains the better results of CES. It can be observed that qGES presents the best results in Experiments 3 and 4 because the shape of the distribution is self-adapted during the evolutionary process. Over the runs, the values of q are higher as mutation distributions with longer tails are eventually selected by the population in order to allow the individuals to escape from the local optima (solutions before the change) generated by changing the problem.

In Experiment 5, small and large environmental changes occur in the problem, as the degree of adaptation of the control laws varies according to the different environments. As a consequence, GES presents better results than CES in some change cycles where the required adaptation of the ANN is small, while CES presents better results in others change cycles where the ANN is required to change in a higher degree. In Experiment 5, qGES presents good performance too as, like in the other experiments, the mutation distribution is self-adapted in order to allow less or more longer jumps according to the environment.

5 Conclusions

In this paper, self-adaptation of the mutation distribution in ESs is proposed for DOPs. The shape of the mutation distribution is controlled by changing the parameter q of the q-Gaussian distribution, which can reproduce either finite or infinite second moment distributions. In the proposed algorithm, the real parameter q is encoded in the chromosome of the individual and is allowed to evolve, what allows to smoothly change the shape of the mutation distribution.

In the proposed method, the decision of choosing which distribution is more indicated for a given problem and at a given moment of the evolutionary process is minimized by letting the algorithm to decide which mutation distribution should be used. The experimental results indicate that this property can be useful for ES in DOPs as, after the environmental changes, the parameter q can be increased resulting in a higher number of long jumps (like the Cauchy mutation), which can help the population to escape from local optima generated by the changes in the problem. In later stages, after the environmental changes, the parameter q reaches small values, which improves the local search (like the Gaussian mutation).

In the future, other control methods for the q parameter should be investigated, including self-organization [14]. Also, the proposed algorithm will be investigated on other DOPs.

Acknowledgment: This work was supported by FAPESP of Brazil under Grant 2010/09273-1, and by the Engineering and Physical Sciences Research Council of UK under Grant EP/E060722/1 and Grant EP/E060722/2.

References

[1] D. V. Arnold and H.-G. Beyer. Random Dynamics Optimum Tracking with Evolution Strategies. In J.J. Merelo, P. Adamidis, H.-G. Beyer, J.L. Fernández-Villacañas, and H.-P. Schwefel, editors, *Parallel Problem Solving from Nature*, pages 3–12, Heidelberg, 2002. Springer.

[2] H.-G. Beyer and H. S. Schwefel. Evolution strategies: a comprehensive introduction. *Natural Computing*, 1:3–52, 2002.

[3] J. Branke. *Evolutionary Optimization in Dynamic Environments.* Kluwer, 2001.

[4] 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.

[5] S. García, D. Molina, M. Lozano, and F. Herrera. A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: a case study on the CEC'2005 Special Session on Real Parameter Optimization. *Journal of Heuristics*, 15:617–644, 2009.

[6] R. W. Morrison. *Designing evolutionary algorithms for dynamic environments.* Springer-Verlag New York Inc, 2004.

[7] S. Nolfi and D. Floreano. *Evolutionary robotics: the biology, intelligence, and technology of self-organizing machines.* MIT Press/Bradford Books: Cambridge, USA, 2000.

[8] F. Raman and F. B. Talbot. The job shop tardiness problem: A decomposition approach. *European Journal of Operational Research*, 69(2):187–199, 1993.

[9] L. Schönemann. Evolution Strategies in Dynamic Environments. In S. Yang, Y.-S. Ong, and Y. Jin, editors, *Evolutionary Computation in Dynamic and Uncertain Environments, Studies in Computational Intelligence, Vol. 51*, pages 51–77. Springer, 2007.

[10] A. M. C. Souza and C. Tsallis. Student's t- and r-distributions: Unified derivation from an entropic variational principle. *Physica A: Statistical Mechanics and its Applications*, 236(1-2):52 – 57, 1997.

[11] W. Thistleton, J. A. Marsh, K. Nelson, and C. Tsallis. Generalized Box-Muller method for generating q-Gaussian random deviates. *IEEE Transactions on Information Theory*, 53(12):4805–4810, 2007.

[12] R. Tinós and A. C. P. L. F. Carvalho. Use of gene dependent mutation probability in evolutionary neural networks for non-stationary problems. *Neurocomputing*, 70(1-3):44–54, 2006.

[13] R. Tinós and S. Yang. Continuous dynamic problem generators for evolutionary algorithms. In *Proc. of the 2007 IEEE Cong. on Evolutionary Computation*, pages 236–243, 2007.

[14] R. Tinós and S. Yang. Self-organizing random immigrants genetic algorithm for dynamic optimization problems. *Genetic Programming and Evolvable Machines*, 8(3):255–286, 2007.

[15] R. Tinós and S. Yang. Evolutionary Programming with q-Gaussian Mutation for Dynamic Optimization Problems. In *Proc. of the 2008 IEEE Cong. on Evolutionary Computation*, pages 1823–1830, 2008.

[16] R. Tinós and S. Yang. Use of the q-Gaussian mutation in evolutionary algorithms. *Soft Computing*, published online: 31 december, 2010.

[17] S. Umarov, C. Tsallis, and S. Steinberg. On a q-central limit theorem consistent with nonextensive statistical mechanics. *Milan Journal of Mathematic*, 76(1): 307-328, 2008.

[18] K. Weicker and N. Weicker. On evolution strategy optimization in dynamic environments. In *Proc. of the 1999 Cong. on Evolutionary Computation*, pages 2039–2046, 2000.

[19] X. Yao and Y. Liu. Fast evolution strategies. *Control and Cybernetics*, 26(3):467–496, 1997.