Chilean Journal of Statistics
Vol. 9, No. 2, September 2018, 19-32

Convergence analysis of an elitist non-homogeneous genetic algorithm with crossover/mutation probabilities adjusted by a fuzzy controller

André Pereira[1,*], Viviane Campos[1], José Roveda[2], Fágner Santana[1] and Francisco de Medeiros[3],

[1]Department of Mathematics, Federal University of Rio Grande do Norte, Natal, Brazil,
[2]Department of Environmental Engineering, State University Julio de Mesquita Filho, Sorocaba, Brazil,
[3]Department of Statistics, Federal University of Rio Grande do Norte, Natal, Brazil

(Received: March, 2018 · Accepted in final form: November, 2018)

Abstract

In recent years, several attempts to improve the efficiency of the canonical genetic algorithm have been presented. The advantage of the elitist non-homogeneous genetic algorithm is that, variations of the mutation probabilities permit the algorithm to broaden its search space at the start and restrict it later on, however the way in which the mutation probabilities vary is defined before the algorithm is initiated. To solve this problem various types of controllers can be used to adjust such changes. This work presents an elitist non-homogeneous genetic algorithm where the mutation probability is adjusted by a fuzzy controller. Many simulation studies have used fuzzy controllers to adjust the parameters in order to improve the performance of the genetic algorithm. However, no previous investigation has discussed the conditions that must be met by the controller in order to ensure convergence of the genetic algorithm. A generalized example will be used to illustrate how sufficient conditions for the algorithm convergence can be readily achieved. And finally, numerical simulations are used to compare the proposed algorithm with the canonical genetic algorithm.

Keywords: Convergence · Fuzzy controller · Genetic algorithms · Global optimization · Markov chain.

Mathematics Subject Classification: 60J20 · 65C40.

1. Introduction

The Canonical Genetic Algorithm (CGA), presented in Holland (1975), is a computational tool to describe the natural genetic evolutionary process of a population, involving three stages: selection, crossover (mating) and mutation. The CGA considers a population of N individuals or chromosomes, $(u_1, u_2, \ldots, u_N)$. An evaluation function $f : E \to (0, \infty)$ assigns a fitness value $0 < f(u_i) < \infty$ to each individual, u_i. In the selection stage, the current population is re-sampled; individuals with higher fitness are more likely to be selected, while those with low fitness will tend to be eliminated (elitist selection). Following the natural evolutionary process, biological reproduction (crossover) and occasional muta-

[*]Corresponding author. Email: andre.gustavo.campos.pereira@gmail.com

ISSN: 0718-7912 (print)/ISSN: 0718-7920 (online)

tion occur. In the crossover stage, individuals are independently chosen for crossover, with a prescribed probability p_c. Mutation also operates independently on each individual, with a prescribed probability p_m. For simpler implementation, each individual is represented by a binary vector of length l, where l depends on the desired precision. Further details, as well as implementation procedures, see Campos et al. (2012a), Andrade and Pereira (2015) and Goldberg (1989).

In the optimization context, CGAs are used to solve problems of the type $\max\{f(x), x \in E\}$ with the objective function satisfying $0 < f(x) < \infty$. The individuals represent the feasible solutions, and in the selection stage, the best fitted/searched points are preserved with higher probability. In the crossover stage, neighboring points are searched, allowing a refined comparison in the surroundings. In the mutation stage, random points, possibly distant from the preserved ones, are visited as a strategy to avoid being trapped in local optimum points. In Rudolph (1994) it is shown that this algorithm does not converge almost surely to the set of populations that contains the optimum point as one of its points. Moreover, in Rudolph (1994), was proposed a modification called the Elitist Genetic Algorithm (EGA) and its convergence was proved.

In Grefensttete (1986) is presented a series of parameters to the GA and simulations were developed to illustrate that variations on those parameters interfere the output of the algorithm. In Lee and Takagi (1993) is related that, based on Grefensttete (1986), one attempt of defining the way of varying the parameters in order to improve the performance of the algorithm was tried but unsuccessfully. For this reason, a fuzzy controller was proposed in that work as a tool to be used to vary the parameters. However, nothing beyond simulations was presented to show that this proposal could be interesting.

The non-homogeneous Genetic Algorithm (NHGA) in Campos et al. (2012b) was introduced as an attempt to improve the efficiency of the CGA, by allowing the mutation and crossover probabilities to vary according to certain hypotheses. A non-homogeneous version of the EGA, called the elitist non-homogeneous genetic algorithm (ENHGA), see Cruz and Pereira (2012), was introduced in order to improve the efficiency of the EGA. Other attempts to improve the efficiency of the CGA, without changing the mutation and crossover probabilities, can be found in Dorea et al. (2010). Numerical comparisons between ENHGA and EGA can be found in Campos et al. (2012a), and a proper way of running the ENHGA is described in Andrade and Pereira (2015).

The advantage of the ENHGA over the EGA is that variations of the mutation probabilities (starting high and decreasing) permit the algorithm to broaden its search space at the start and restrict it later on. The problem in using the ENHGA is that the way in which the mutation probabilities vary is defined before the algorithm is initiated; in other words, it is already previously defined when the algorithm starts. The ideal would be for the parameters to vary, rather than only diminish, depending on a certain measure of dispersion of the elements of the current population, as well as the number of iterations of the algorithm. To this end, controllers are introduced in the intermediate stages of the algorithm in order to adjust such changes. Various types of controllers can be used for this task, ranging from deterministic methods to those that employ fuzzy logic. Many simulation studies have used fuzzy controllers to adjust the parameters in order to improve the performance of the genetic algorithm. However, no previous investigation has discussed the conditions that must be met by the controller in order to ensure convergence of the genetic algorithm, see Yun and Gen (2003). Without a convergence proof for the algorithm we do not have guarantee that the algorithm behaves as it supposed to do.

In many others papers GA and fuzzy controllers are used to obtain a rule basis and membership functions in a dynamic way, so that the performance of the fuzzy controller is improved. An example of that use can be seen in Lam et al (2004) where a GA, called improved GA, is used to adjust the rules and functions mentioned above. However, no additional piece of information is given to explain why that GA is better.

It is shown in Section 4 that when you are looking for a pair of parameters (p_m, p_c) that improves the convergence rate of the algorithm, it is worth to concentrate your efforts on p_m. The contribution of this work is to show how analyze the membership functions and the rule basis of the fuzzy controller, which is adjusting just the mutation probability, so as to guarantee the convergence of the GA.

A very simple fuzzy controller with one input variable and one output variable was constructed, using underlying membership functions as well as rule basis in such process. The goal of this paper, besides the convergence results, is to illustrate how to use the membership functions and the rule basis in order to guarantee the convergence of the GA. In the example that was developed, the input variable is the number of iterations (N_g) and the output variable is the mutation probability (p_m).

This paper is divided into 4 parts. In Section 2, definitions and results concerning the non-homogeneous Markov chains that will be used in the rest of the paper are presented. In Section 3, the fuzzy controller is introduced, the way the controller is used in the evolution of the genetic algorithm is explained, and convergence results are obtained in Theorem 3.1 and Theorem 3.2. Theorem 3.2 could be easily extended to the case where more than one input variable and more than one output variable, e.g. p_m and p_c, are used by the fuzzy controller. In Section 4, numerical comparisons between the canonical genetic algorithm and the algorithm presented in Section 3 are developed.

2. Preliminaries

Let $f : E \to (0, \infty)$ be a function. An elitist non-homogeneous genetic algorithm was built in order to find the point

$$x^* = \arg\max_x \{f(x), x \in A\},$$

where A is a discretization of E, the domain of the function f. To proceed the following steps of the algorithm, such points are represented as binary vectors of length l, where l depends on the desired precision. A population of size N is considered and let $Z = \{(u_1, u_2, \ldots, u_N); u_i \in A, i = 1, 2, \ldots, N\}$ be the set of all populations of size N. Z is the state space of the Markov chain that is used to prove the convergence of the algorithm (see Campos et al. (2012b), Dorea et al. (2010), Cruz and Pereira (2012) and Rudolph (1994)).

The evolution of the ENHGA is different from the evolution of the EGA just by the update of the values of parameters p_m and p_c. Thus, the elitist algorithm can be summarized in the following sketch:

a) Choose randomly an initial population having N elements, each one being represented by a binary vector of length l, and create one more position, the $(N+1)$-th entry of the population vector, which will keep the best element from the N previous elements.
b) Repeat
    (1) Perform selection with the first N elements.
    (2) Perform crossover with the first N elements.
    (3) Perform mutation with the first N elements.
    (4) If the best element from this new population is better than that of the $(N+1)$-th position, change the $(N+1)$-th position by this better element, otherwise, keep the $(N+1)$-th position unchanged.
    (5) Change the values of p_c and p_m as previously planned.
c) Until some stopping criterion applies.

Note that one new position was added to the population vector, now the set of all populations is different from Z, so denote this new state space by $\tilde{Z}$. In Cruz and Pereira (2012), it is shown that the ENHGA is a non-homogeneous Markov chain, with a finite state space $\tilde{Z}$, whose transition matrices are given by $P_n = SC_nM_n, \forall n \in I\!N$, where S, C_n, M_n are transition matrices which represent the selection, crossover and mutation stages respectively. Here, M_n is composed by the third and fourth steps described in sketch just presented. In the same paper it is shown that there is a sequence $\{\alpha_n\}_{n\in I\!N}$ satisfying a Doeblin type condition

$$\inf_{i\in\tilde{Z}, j\in\tilde{Z}^*} P_n(i,j) \geq \alpha_n,$$

where $\tilde{Z}^* \subset \tilde{Z}$, which contains all populations that have the optimum point as one of its points. The following results were obtained.

Lemma 2.1 Let $\{X_n\}_{n\in I\!N}$ be the Markov chain which models the elitist non-homogeneous genetic algorithm. If the sequence above is such that $\sum_{k\geq 1} \alpha_k = \infty$, then

$$P(\lim_{n\to\infty} X_n \in \tilde{Z}^*) = 1, \tag{1}$$

that is, the chain finds the optimum point almost surely.

A more simple condition, to run in simulations, that guarantee the previous result is:

Lemma 2.2 Let $\{X_n\}_{n\in I\!N}$ be the Markov chain which models the elitist non-homogeneous genetic algorithm. If the mutation probabilities $\{p_m(n)\}_{n\in I\!N} \subset (0,1)$ are such that $p_m(n) > \gamma > 0$ for all $n \in I\!N$, then (1) holds.

So far we have the hypothesis the algorithm has to satisfies in order to enter into the set where at least one coordinate is the searched point. However, one could be interested in the equilibrium distribution of the algorithm. The algorithm we are dealing with is a non-homogeneous Markov chain and to find the equilibrium distribution we need to remember the weak and strong convergence definitions.

So, let $\{X_n\}$ be a non-homogeneous Markov chain with finite state space $\tilde{Z}$ and with transition matrices given by $\{P_m\}_{m\geq 0}$, where

$$P_m(i,j) = P(X_{m+1} = j | X_m = i) = P^{(m,m+1)}(i,j),\ \ i,j \in \tilde{Z}.$$

By the properties of non-homogeneous Markov chains, the k step transition is given by the product of the transitions matrices $P_mP_{m+1}\cdots P_{m+k-1}$, for all $m \geq 0$. Thus,

$$P^{(m,m+k)}(i,j) = \sum_{i_1\in\tilde{Z},\ldots,i_{k-1}\in\tilde{Z}} P_m(i,i_1)P_{m+1}(i_1,i_2)\cdots P_{m+k-1}(i_{k-1},j)$$

and we can write the Chapman-Kolmogorov equation as

$$P^{(m,m+k)} = P^{(m,m+r)} \cdot P^{(m+r,m+k)}\ ,\ 1 \leq r < k.$$

Definition 2.3 A non-homogeneous Markov chain $\{P_n\}_{n\in I\!N}$ is said to be weakly ergodic if it satisfies

$$\lim_{k\to\infty} \|\mu_0 P^{(m,k)} - \mu_1 P^{(m,k)}\| = 0\ ,\ \forall m \geq 0 \tag{2}$$

where μ_0 and μ_1 are any probability distributions, $\|P\| = \sup_i \sum_{j\in\tilde{Z}} |P_{ij}|$ and $\|\mu\| = \sum_{j\in\tilde{Z}} |\mu_j|$.

In Issacson and Madsen (1976), it is proved that (2) is equivalent to:

$$\lim_{k\to\infty} \delta(P^{(m,k)}) = 0,\ \forall m \geq 0.$$

Where, for a stochastic matrix $Q = (q_{ij})_{i,j\in\tilde{Z}}$, the Dobrushin's delta coefficient is defined by

$$\delta(Q) = \sup_{i,k\in\tilde{Z}} \sum_{j\in\tilde{Z}} [q_{ij} - q_{kj}]^+ \quad \text{with} \ \ [q_{ij} - q_{kj}]^+ = \max\{0, q_{ij} - q_{kj}\},$$

or equivalently,

$$\delta(Q) = \frac{1}{2} \sup_{i,k\in\tilde{Z}} \sum_{j\in\tilde{Z}} |q_{ij} - q_{kj}|.$$

We see easily that $\delta(P) \leq 1$, for any stochastic matrix P.

Definition 2.4 A non-homogeneous Markov chain $\{P_n\}_{n\in I\!N}$ is said to be strongly ergodic if there exists a constant matrix P_∞ such that

$$\lim_{k\to\infty} \|P^{(m,m+k)} - P_\infty\| = 0\ ,\ \forall m \geq 0. \tag{3}$$

In Cruz (1998) is proved the following theorem:

Theorem 2.5 Let $\{P_n\}_{n\in I\!N}$ be a non-homogeneous Markov chain and P a transition kernel which is weakly ergodic. If

$$\lim_{n\to\infty} \|P_n - P\| = 0 \tag{4}$$

then $\{P_n\}_{n\in I\!N}$ is strogly ergodic.

So, when a Markov chain is strongly ergodic, that is (3) holds, then there exists a equilibrium distribution π which is one of the rows of P_∞.

In the next section it is shown how to use the membership functions and the rule basis in a very simple case so as to guarantee that the hypotheses of Lemma 2.2 are satisfied. The procedure used in such example are easily extended to the cases where more input/output variables are used. Furthermore, with aditional hypothesis on p_m and p_c and using Theorem 2.5 it is shown that the algorithm is strongly ergodic.

3. The Fuzzy Controller

A fuzzy controller has the ability to associate an output value with an input value. Its functioning involves four essential components: fuzzification, the inference method, the rule basis, and defuzzification, as shown in Figure 1, for more details see Pedrycz (1993).

To illustrate how to use the fuzzy controller to adjust the values of the mutation probability p_m, let the input variable N_g be the number of iterations of the GA and the output variable is the new value of p_m that will be used by the algorithm in the next iteration.

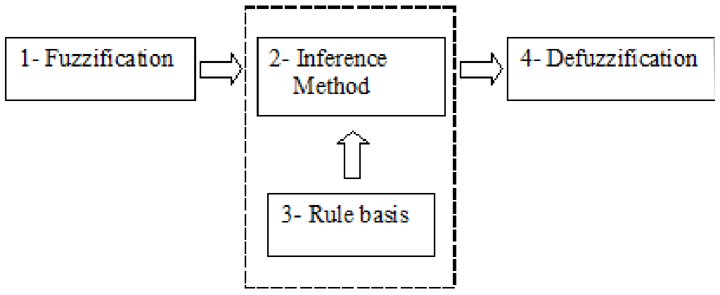

Figure 1. Fuzzy controler scheme

Fuzzification is the first part of the process, and consists of converting the numerical input into fuzzy sets. For such task we need the membership functions for the input and output variables, as presented in Figure 2 and 3.

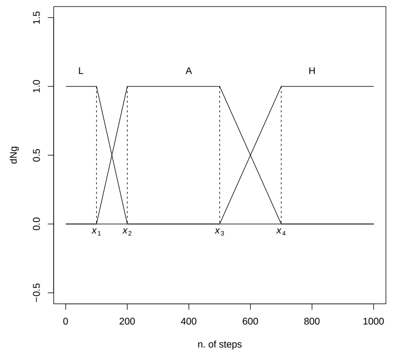

Figure 2. Membership function for N_g

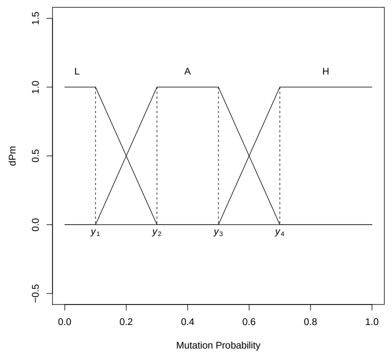

Figure 3. Membership function for p_m

The second part involves the definition of an inference method and a rule basis in order to get from an input value a fuzzy set as output. Considering the following rule basis

(1) If N_g is low (L), then p_m is high (H);
(2) If N_g is average (A), then p_m is average (A);
(3) If N_g is high (H), then p_m is low (L).

and the Mamdani inference method, based on max-min operators, we obtain different fuzzy sets (the shadowed areas) as presented in Figures 4 and 5.

Finally, defuzzification is an operation that converts a fuzzy set to a numerical value, which can be achieved using a technique such as the Center of Gravity method, described by the formula

$$C = \frac{\int u\varphi(u)du}{\int \varphi(u)du} \tag{5}$$

where $\varphi(u)$ is the function whose area below it and above the x-axis is the shadowed area in Figures 4 and 5.

Theorem 3.1 Consider a fuzzy controller with some input variables and with the mutation probability p_m and crossover probability p_c as its output variables. In order to use this controller to ensure the convergence of an EHNGA, it is sufficient that there exists a positive integer L and a positive real number γ, such that, from the L-th iteration of the

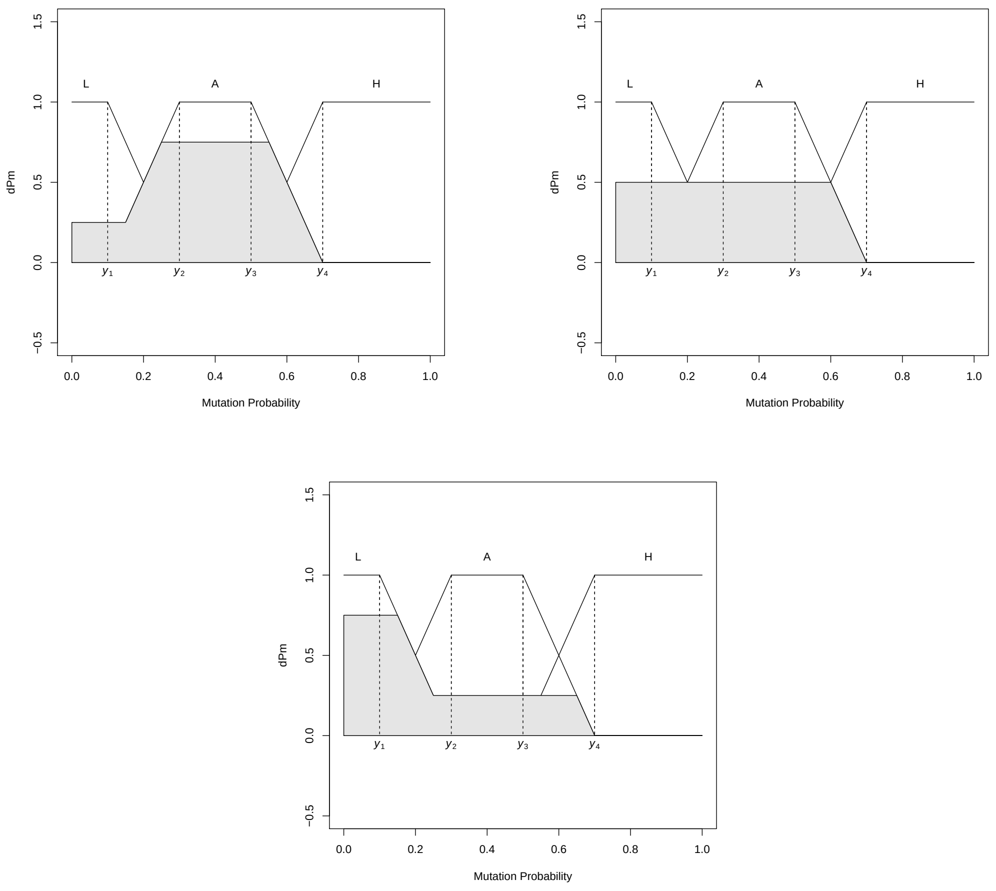

Figure 4. Possible outcomes when $x \in (x_3, x_4)$

algorithm on, the p_m, calculated as in the Equation (5), satisfies the following condition:

$$p_m \geq \gamma > 0. \tag{6}$$

Proof If the output of the controller satisfies Equation (7) then it satisfies the hypotheses of Lemma 2.2, so the algorithm converges. ■

For a special class of fuzzy controllers we can state

Theorem 3.2 An ENHGA that has its crossover/mutation probabilities adjusted by a fuzzy controller, that use input and output variables with membership functions like those presented in Figures 2 and 3 and the Mamdani inference method, based on max-min operators converges.

Proof We want to prove that the hypothesis of the Theorem 3.1 is satisfied, that is, p_m satisfies (6). The only difference of one or more input variables is the number of cases to be analysed. For this reason, the one input variable case is explained in details and the case with more than one variable is easily obtained by the finiteness of the number of cases to be analysed.

Considering the input variable membership function as that of Figure 2 and the output variable membership function as that of Figure 4, we have that just one rule is triggered when $x \in (0, x_1) \cup (x_2, x_3) \cup (x_4, 1000)$. By the membership functions for p_m we have:

If N_g triggers just the L ($x \in (0, x_1)$), then p_m is H and since $dN_g(L)(x) = 1$ this implies

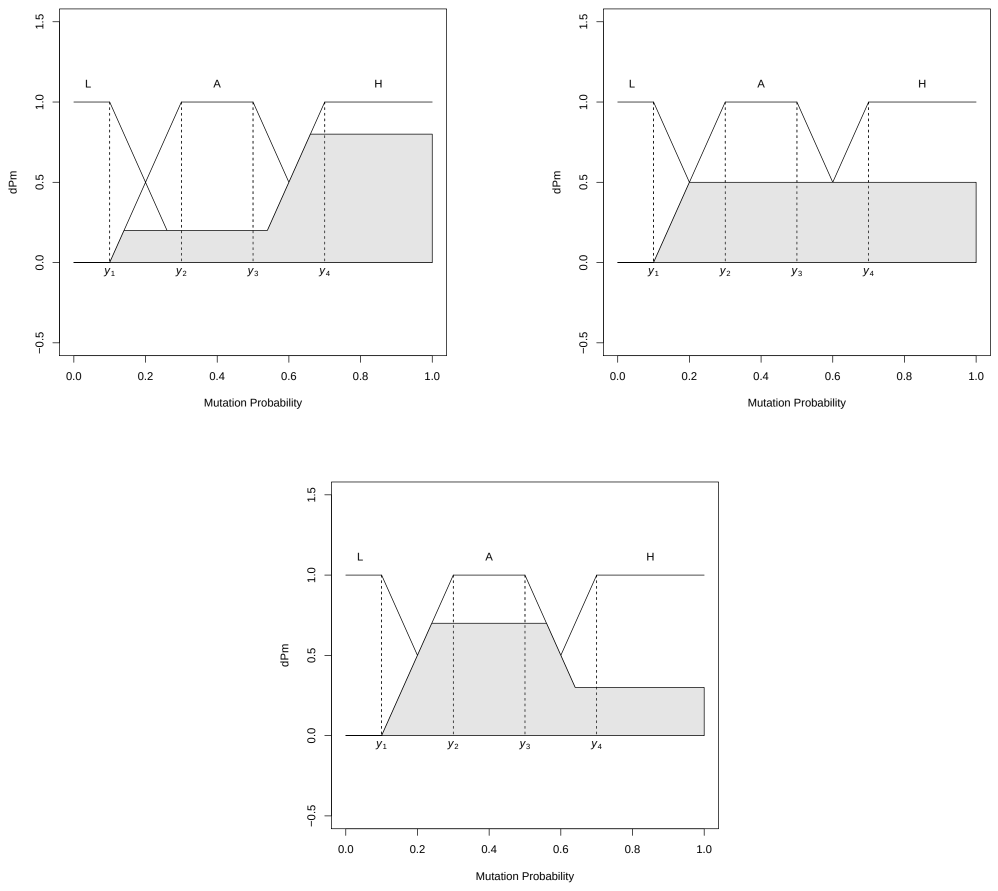

Figure 5. Possible outcomes when $x \in (x_1, x_2)$

that $\varphi(u) = dp_m(H)(u)$, and

$$\frac{\int u\varphi(u)du}{\int \varphi(u)du} \geq \frac{\int y_3\varphi(u)du}{\int \varphi(u)du} = \frac{y_3 \int \varphi(u)du}{\int \varphi(u)du} \geq y_3.$$

If N_g triggers just A ($x \in (x_2, x_3)$), then p_m is A and since $dN_g(A)(x) = 1$ this implies that $\varphi(u) = dp_m(A)(u)$, and

$$\frac{\int u\varphi(u)du}{\int \varphi(u)du} \geq \frac{\int y_1\varphi(u)du}{\int \varphi(u)du} = \frac{y_1 \int \varphi(u)du}{\int \varphi(u)du} \geq y_1.$$

When N_g triggers just H then $dp_m(L)$ is activated so $\varphi(u) = dp_m(L)(u)$ and

$$\frac{\int u\varphi(u)du}{\int \varphi(u)du} \geq \frac{\int_0^{y_2} udp_m(L)(u)du}{\int_0^{y_2} dp_m(L)(u)du} \geq \frac{\int_0^{y_1} udp_m(L)(u)du}{\int_0^{y_2} 1du} \geq \frac{y_1^2}{2y_2}.$$

If two rules are triggered. This happens when $x \in (x_1, x_2) \cup (x_3, x_4)$.

If $x \in (x_1, x_2)$ then $dN_g(L)$ and $dN_g(A)$ are trigged, so we have one of the possible outputs illustrated in Figure 5.

In any of the above situations we have

$$\frac{\int u\varphi(u)du}{\int \varphi(u)du} = \frac{\int_{y_1}^{1} u\varphi(u)du}{\int_{y_1}^{1} \varphi(u)du} \geq \frac{\int_{y_1}^{1} y_1\varphi(u)du}{\int_{y_1}^{1} \varphi(u)du} = y_1$$

If $x \in (x_3, x_4)$, then the rules $N_g(H)$ and $N_g(A)$ are triggered and we have as output $dp_m(L)$ and $dp_m(A)$ respectively, and one of the possible situations showed in Figure 4 will happen.

Observe that when two rules are triggered, this implies $N_g(H)(x), N_g(A)(x) < 1$ and when that happens the maximum value of the output which is $\max\{N_g(A)(x), N_g(H)(x)\}$ occurs in a interval whose length is bigger than $\min\{(y_3 - y_2), y_1\}$. Let denote such interval by I. So,

$$\frac{\int u\varphi(u)du}{\int \varphi(u)du} = \frac{\int_0^{y_4} u\varphi(u)du}{\int_0^{y_4} \varphi(u)du} \geq \frac{\int_I u\varphi(u)du}{y_4 \max\{N_g(A)(x), N_g(H)(x)\}}$$
$$= \frac{\max\{N_g(A)(x), N_g(M)(x)\} \int_I udu}{y_4 \max\{N_g(A)(x), N_g(M)(x)\}} \geq \frac{\beta}{y_4}$$

where $\beta = \min\{\frac{y_1^2}{2}, \frac{y_3^2 - y_2^2}{2}\}$.

In all cases, we have found lower bounds for p_m ($\frac{\beta}{y_4}, \frac{y_1^2}{2y_2}, y_1$ and y_3). All of these lower bounds are fixed and positive, so, defining $\gamma = \min\{\frac{\beta}{y_4}, \frac{y_1^2}{2y_2}, y_1, y_3\}$, we have $p_m \geq \gamma > 0$ and the condition required in Theorem 3.1 is satisfied. ■

So far we just guaranteed that in a finite time the chain will have a searched point as one of its coordinates (the last one) almost surely. Under these same hypotheses we are not able to guarantee if there exists a equilibrium distribution for such algorithm, but with aditional hypotheses on $\{p_m(n)\}$ and $\{p_c(n)\}$ the algorithm is shown to be strongly ergodic.

Theorem 3.3 If the rule basis are chosen in such a manner that as the times goes by the outputs $p_m(n)$ and $p_c(n)$ converges to positive values $p_m > 0$ and $p_c > 0$ respectively, then the algorithm is strongly ergodic and has a equilibrium distribution.

Before proving this theorem note that the example we have been developing satisfies the hypotheses of this theorem, because we have $p_c(n) = p_c > 0$ for all n, and

$$p_m(n) \to \frac{\int u\varphi(u)du}{\int \varphi(u)du} = \frac{\int_0^{y_2} udp_m(L)(u)du}{\int_0^{y_2} dp_m(L)(u)du} > 0.$$

Proof It is known that the inputs of the transition matrix P_n of the elitist non-homegeneous genetic algorithm is a polynomial expression whose variables are $p_c(n)$ and $p_m(n)$. Since $p_m(n)$ and $p_c(n)$ converge to positive constants, then $P_n \to P$ and even though P is not positive we have that $\delta(P) < 1$ and hence the chain is weakly ergocidic. So, all hypotheses of Theorem 2.5 are satisfied and we have that the algorithm is strongly ergodic. ■

4. Numerical comparisons

In this section we develop some tools to measure the mean time that the algorithm takes to find the optimum point of an objective function. In the following examples, we analyze the importance of the values of the mutation and crossover probabilities on the convergence rate.

In the next examples, 100 repetitions of each algorithm are run and in each repetition, 1000 iterations are performed. After that, we observe the number of repetitions where the algorithm finds the optimum point (called here success) in x iterations, $x = 1, \dots, 1000$. The summarized results are shown in Table 1, where a_0 is the number of success observed when the algorithms starts, a_1 is the number of success after one iteration, $\dots$ and a_{1000} is the number of success after thousand iterations. Based on the results of the simulations

Iteration	Number of successes
0	a_0
1	a_1
$\vdots$	$\vdots$
1000	a_{1000}

Table 1. Number of successes in each iteration

we are able to develop a measure to evaluate the mean time the algorithm takes to find the optimum point of the function f, namely:

$$mtexp(f) = \sum_{k=0}^{1000} kp_k$$

where $a_0, (a_1 - a_0), \dots, (a_{1000} - a_{999})$ represent the number of repetitions that the algorithm has found the optimum point at the step $0, 1, 2, \dots, 1000$ respectively. Moreover, $p_0 = \frac{a_0}{100}$ and $p_i = \frac{a_i - a_{i-1}}{100}$, $i = 1, 2, \dots, 1000$ are the corresponding proportions of realizations where the optimum was found at step $0, 1, \dots, 1000$, respectively.

We observe that after 1000 iterations, in some realizations (mostly when the population is small) the algorithm does not find the optimum point. So, a penalization function is introduced to increase the mean time on those realizations that the maximum is not found after 1000 iterations. We define the penalization function as

$$p(f) = \frac{100 - a_{1000}}{100} \times penalty,$$

where the factor *penalty* represents the estimated number of iterations to find the optimum point. Such penalty is applied to all cases, and the larger the number of trajectories that does not find the maximum in 1000 iterations is; the larger the value of its penalization function is. So, the mean time function is given by

$$mt(f) = mtexp(f) + p(f).$$

The evaluation functions considered in the examples are widely used in the literature to test algorithms' performance:

(1) $f(x, y) = 6 + x^2 - 3cos(2\pi x) + y^2 - 3cos(2\pi y)$ whose domain is $D = [-2, 1] \times [-2, 1]$.
(2) $f(x, y) = 0.5 - (sin(\sqrt{x^2 + y^2})^2 - 0.5)/(1 + .001 * (x^2 + y^2))^2$ whose domain is $D = [\frac{-1280}{63}, \frac{1240}{63}] \times [\frac{-1280}{63}, \frac{1240}{63}]$.
(3) $f(x, y) = 1/(0.3 + x^2 + y^2)$ whose domain is $D = [-4, 2] \times [-4, 2]$.
(4) $f(x, y) = (1 + (19 - 14x + 3x^2 - 14y + 6xy + 3y^2)(x + y + 1)^2)(30 + (18 - 32x + 12x^2 + 48y - 36xy + 27y^2)(2x - 3y)^2)$ whose domain is $D = [-2, 2] \times [-2, 2]$.

(5) $f(x, y) = (y - (5.1x^2)/(4\pi^2) + (5x)/\pi - 6)^2 + 10(1 - 1/(8\pi)) * cos(x) + 10$ whose domain is $D = [-5, 10] \times [0, 15]$.

In the following simulations, the domain of the functions are discretized into a net of 2^{12} points and the factor penalty is 1200. Then, the EGA is used to search the optimum point of each function. For each function the algorithm was run for the following values of p_m and p_c:

- Mutation probabilities $= \{0.01, 0.03, 0.05, 0.07, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 0.93, 0.95, 0.97, 0.99\}$.
- Crossover probabilities $= \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$.

Moreover, we repeat the above scenarios considering different population sizes (10, 20, 30, 40 and 50).

In the following figures we show a pair of boxplots for each function. The boxplot on the left was obtained from the simulations for a population of size 20, for some values of p_c while the mutation probability is fixed. The boxplot on the right is obtained under the same conditions but the p_c is ketp fixed while p_m varies. It can be seen that when p_m is

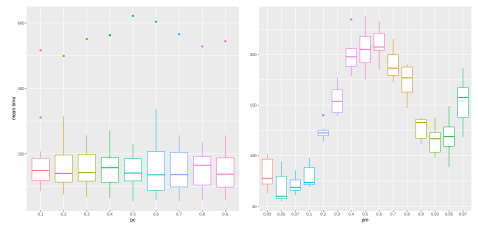

Figure 6. Function 1

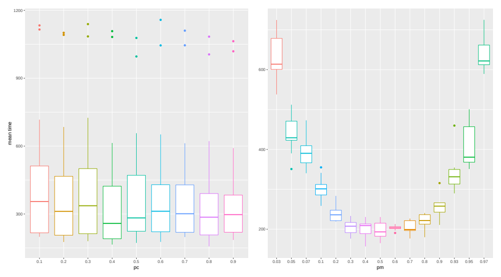

Figure 7. Function 2

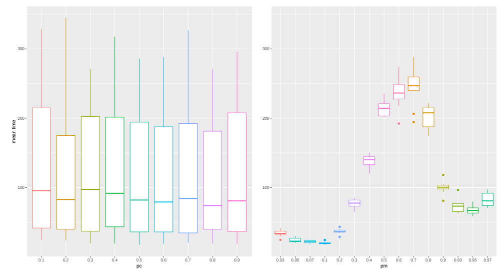

Figure 8. Function 3

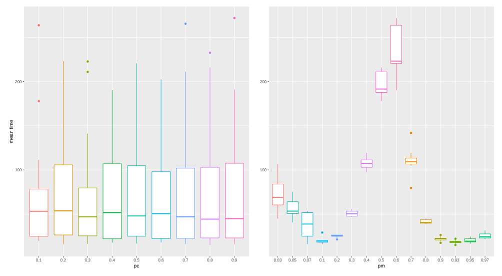

Figure 9. Function 4

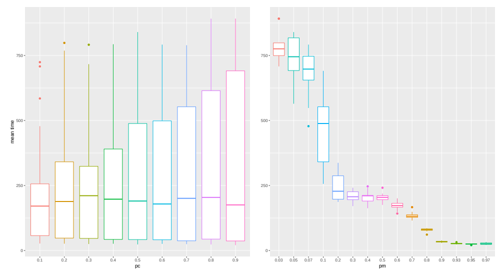

Figure 10. Function 5

fixed, the median of the mean time presents a low variability as a function of p_c and its distribution is, in general, right skewed independently of the function we are dealing with.

On the other hand, when p_c is fixed, the median of the mean time presents a high variability as a function of p_m and its distribution presents different behaviors (simetric, right skewed, left skewed) depending on the function we are dealing with. Thus, we can say that there is not a pair (p_m, p_c) that improves the convergence rate of the algorithm independetly of the objective function and that for a given objective function if you are looking for a pair of parameters (p_m, p_c) that improves the convergence rate of the algorithm, it is worth to concentrate your efforts on p_m, once it is found, if p_c varies then the mean time for such pair of parameters will not vary widely.

The following figures show the graphics of the mean time for the previous functions in three different scenarios, namely:

(1) For a fixed $p_m = 0.01$
(2) For a decreasing p_m, it varies from 0.99 to 0.01 linearly.
(3) Using a fuzzy controller, which has input variable N_g and outuput variable p_m, having membership functions as those presented in Figures 2 and 3, where $x_1 = 200, x_2 = 300, x_3 = 400, x_4 = 500$ and $y_1 = 0.05, y_2 = 0.1, y_3 = 0.85, y_4 = 0.9$. Moreover the rule basis, is given by: If $N_g = L$ then $p_m = H$, if $N_g = A$ then $p_m = L$ and if $N_g = H$ then $p_m = L$.

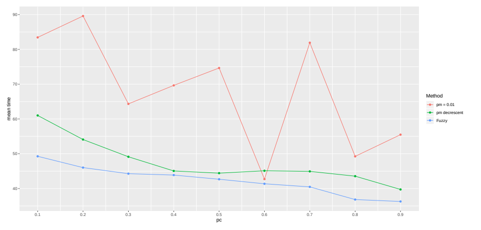

Figure 11. Function 1

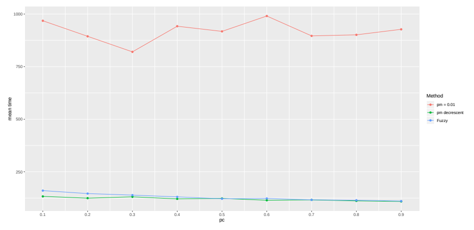

Figure 12. Function 2

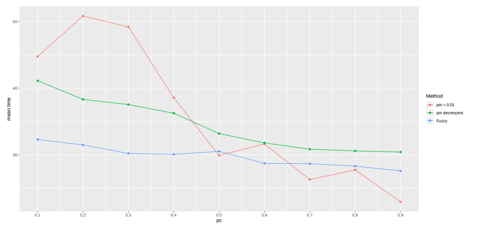

Figure 13. Function 3

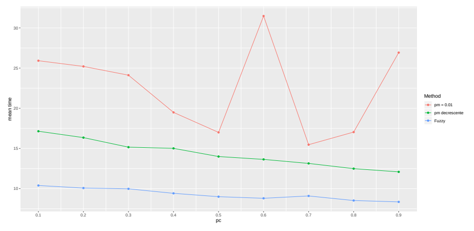

Figure 14. Function 4

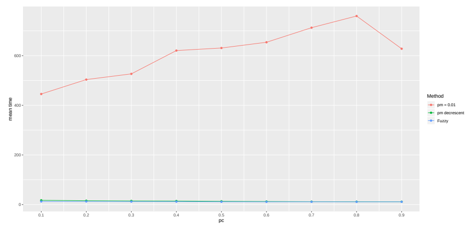

Figure 15. Function 5

5. Conclusion

Although the literature contains many studies in which fuzzy controllers have been used to adjust the parameters of a genetic algorithm, all of them are exclusively concerned about simulations involving different rule basis and membership functions for the controller. Unfortunately, without a mathematical convergence proof, such experiments performed in those papers can not be used in practice with other functions because no convergence can be guaranteed. The present work states sufficient conditions that a fuzzy controller has to satisfy in order to guarantee the convergence of an ENHGA whose parameters (mutation/crossover probabilities) are adjusted by such controller. Theorem 3.1 shows a safety way to set a class of fuzzy controllers to adjust the parameters of an ENHGA in order to find the optimum point of an objective function and Theorem 3.3 shows that with additional hypotheses there exists an equilibrium distribution for the algorithm. Moreover, experiment shows that even naive controllers as the one used in this paper (based on the number of iterations of the algorithm), can improve the convergence rate of the algorithm. Finally, it was shown that there is not a pair of parameters (p_m, p_c) that improves the performance of the algorithm for all objective functions and if one decides to search the best parameters for each situation, it is worth to focus first in finding the mutation probability.

Acknowledgements

The authors are grateful to FAPESP, PROCAD/CAPES and CNPq for financial support.

References

Andrade, B.B., and Pereira, A.G.C., 2015. On the Genetic algorithm with adaptive mutation rate and selected statistical applications. Computational Statistics 30, 131-150.

Campos, V.S.M., Pereira, A.G.C., Carlos, L.A. and Assis, I.A.S., 2012. Algoritmo genético por cadena de Markov homogénea versus no-homogénea: Un estudio comparativo. Journal of the Chilean Institute of Operations Research 2, 30-35.

Campos, V.S.M., Pereira, A.G.C. and Cruz, J.A.R., 2012. Modeling the genetic algorithm by a non-homogeneous markov chain: Weak and strong ergodicity. Theory of Probability and its Applications 1, 185-192.

Cruz, J.A.R, 1998. Non-homogeneous Markov chains convergence: strong and weak ergodicities. Unpublished Ph.D. Thesis, Department of Mathematics. University of Brasilia, Brazil.

Cruz, J.A.R, and Pereira, A.G.C., 2012. The elitist non-homogeneous genetic algorithm: Almost sure convergence. Statistics and Probability Letters 83, 2179-2185.

Dorea, C.C.Y., Junior, R.M.G. and Pereira, A.G.C., 2010. Multistage markov chain modeling of the genetic algorithm and convergence results. Numerical Functional Analysis and Optimization 31, 164-171.

Goldberg, D.E., 1989. Genetic Algorithms in Search, Optimizations and Machine Learning. Addison Wesley, MA.

Grefensttete, J.J., 1986. Optimization of control parameters for genetic algorithm. IEEE Transactions on Fuzzy Systems 16, 122-128.

Holland, J.H., 1975. Adaptation in natural and artificial systems. The University of Michigan Press, Ann Arbor.

Isaacson, D.L. and Madsen, R.W., 1976. Markov Chains Theory and Applications. Wiley, New York.

Lam, H.K., Ling, S.H., Leung, F.H.F. and Tam, P.K.S., 2004. Function estimation using a fuzzy neural-fuzzy network and an improved genetic algorithm. International Journal of Approximate Reasoning 36, 243-260.

Lee, M.A. and Takagi, H., 1993. Dynamic control of genetic algorithms using fuzzy logic techniques. Procceding of 5th International conference on Genetic Algorithms (ICGA'93), Urbana-Champaign, 76-83.

Pedrycz, W., 1993. Fuzzy Sets Engineering. CRC Press, London.

Rudolph, G., 1994. Convergence Analysis of Canonical Genetic Algorithms. IEEE Transactions on Neural Networks 5, 96-101.

Yun, Y. and Gen, M., 2003. Performance Analysis of Adaptative Genetic Algorithms with Fuzzy Logic and Heuristics. Fuzzy Optimization and Decision Making 2, 161-175.