Adaptive-Surrogate based on a Neuro-Fuzzy Network and Granular Computing

Israel Cruz-Vega, Mauricio Garcia, Hugo Jair Escalante
Instituto Nacional de Astrofísica, Óptica y Electrónica.
Computer Science Department
Puebla, 72840, Mexico
isrcruz@ieee.org

ABSTRACT

Surrogate-based methods aim at reducing the evaluation of expensive fitness functions in optimization processes. Several surrogate-based methods for evolutionary optimization have been proposed so far, including those based on granular computing / clustering. Granular computing provides granules as an assemblage of entities arranged together by their similarity, functional or physical adjacency, indistinguishability, coherency, or the like. Techniques like this avoid multiple and unnecessary evaluations of individuals repeatedly. In this paper, with the aim of granular computing as a method of grouping data, such information is exploited to obtain knowledge of the structure and parameters of individuals and then, design a Neuro-Fuzzy network that adapts granules' parameters, providing convergence to acceptable solutions with a reduced number of evaluations of the fitness function. We implement this adaptive surrogate in a genetic algorithm and show its performance using benchmark functions.

Categories and Subject Descriptors

I.2.8 [**Artificial Intelligence**]: Problem Solving—*Heuristic methods*

Keywords

Surrogate Modeling, Neuro-Fuzzy Networks.

1. INTRODUCTION

Traditional optimization is computationally expensive and normally highly susceptible to some issues such as high dimensionality, non-differentiability, non-linearity, highly expensive function calculation, among others. Evolutionary algorithms (EAs) are bio-inspired meta-heuristics that have shown flexibility, adaptability and good performance in an important variety of difficult optimization problems, alleviating some of the above mentioned issues. One of the main drawbacks of EAs is the requirement of the repeated evaluation of the objective function. Clearly, for problems where estimating the objective function is computationally expensive, using an EA may become intractable. Surrogate modeling deals with the construction of approximation models to estimate the performance of a system, and to develop relationships between specific system inputs and outputs.

This paper introduces a surrogate modeling approach called GA - FGNFN: *Genetic-Algorithm based on Fuzzy Granulation and Neuro-Fuzzy Networks*. Under the proposed approach, reduction of fitness evaluations is performed via the construction of fuzzy granules, avoiding the use of multiple fitness evaluations on individuals of similar characteristics. The process of granules' construction provides information to be used by a Neuro-Fuzzy Network, such information includes the number of granules and their positions and widths; which is considered the basic information for training of the network. The network will update the above mentioned parameters every certain number of generations, n, so that, the genetic algorithm will converge to the optimum solution faster, and will reduce the number of fitness evaluations significantly. We evaluate the performance of the proposed method in a suite of benchmark functions. Experimental results show that the proposed approach reduces considerably the number of fitness evaluations without significantly compromising the quality of solutions.

The rest of the paper is organized as follows. Next section reviews related work. Section 3 describes a surrogate model based on fuzzy granules [1] that forms the basis for our proposal. Section 4 introduces the adaptive part of surrogate fuzzy-granules parameters. Section reports numerical results on traditional benchmark functions. Finally, Section 5, outlines conclusions and future work directions.

2. RELATED WORK

Several approaches based on surrogate modeling have been proposed so far. A popular formulation is fitness inheritance or imitation [8, 20, 22, 26], where the fitness of the individual is defined by either their parents or other individuals previously encountered along the search process (fitness is transmitted or inherited). Salami et al. proposed the "fast evolutionary strategy", in which the fitness of a child-individual is the weighted sum of the fitness values from its parents, reliability values are assigned to each new individual, and then the actual fitness function is only evaluated when the reliability value is below a certain threshold [22]. In [20] Reyes-Sierra and Coello incorporated the concept of fitness inheritance into the multi-objective particle swarm optimizer to reduce the number of fitness evaluations. How-

GECCO'14, July 12–16, 2014, Vancouver, BC, Canada.
Copyright 2014 ACM 978-1-4503-2662-9/14/07 ...$15.00.
http://dx.doi.org/10.1145/2576768.2598376.

ever, as shown in [6], the performance of parents may not be a good predictor of their children for sufficiently complex and multi-objective problems, rendering fitness inheritance inappropriate under such circumstances.

Techniques based on universal approximators, whose quality depends on the training data, are based on Artificial Neural Networks (ANN) [9, 11, 14, 17, 24, 25]. In [15], selection of centers in the RBF surrogate model, is done in an unsupervised manner with Learning Vector Quantization (LVQ) and Self-Organizing Maps, this formulation tackles the problem of good generalization, that represents an estimation of objective functions for new individuals.

Adaptive surrogates have been developed in a variety of works, for example, in [34], a hybrid surrogate modeling paradigm is developed where different surrogate models are combined (Kriging, RBF and Extended RBF), the adaptive part is done by contribution of surrogates (working as local models) providing a local measure of accuracy and defining a zone of accuracy determined on the called "crowding distance". In [13] is presented a selection-based criterion with different metrics for measuring the quality of meta-models in EAs, where the evaluation of surrogates is done with Neural Networks, an adaptive scheme is suggested for adapting the number of individuals to be evaluated using the surrogate.

In order to reduce fitness evaluations, a variety of clustering algorithms have been proposed for grouping similar patterns of data [16]. Nevertheless, in practice there are many situations in which the data could be classified as belonging to one cluster almost as well as to another one. The sense of belonging to certain clusters is used via the theory of fuzzy sets, taking advantage of the concept of membership function [4]. Some related works are presented in [2, 10]. In [10] fitness estimation models are based on fuzzy clustering, the objective is to reduce direct evaluations and to keep the diversity of the population and quality of solutions satisfactory. They use Fuzzy c-Means an the adaptive technique is Partitionary Learning Fuzzy Clustering Algorithm, which is unsupervised and able to group data to find an appropriate number of clusters. In [1], Akbarzadeh-T et al. proposed using fitness granulation, via an adaptive fuzzy similarity analysis, to reduce the number of fitness evaluations. A pool of fuzzy granules with Gaussian measures of similarity is built up and the enlargement and shrinkage of granules are adaptive each generation depending on the population fitness, but the position of granules does not change during the search process, we refer this method as GA-AFFG [1].

This paper describes a surrogate modeling approach, called GA-FGNFN, that extends GA-AFFG by incorporating a Neuro-Fuzzy Network that aims at making the granules to better adapt during the search process. As in [1], in our proposed approach reduction of fitness evaluations is performed via the construction of fuzzy granules, avoiding the use of multiple fitness evaluations on individuals of similar characteristics. Opposed to [1], we learn a model that determines how to modify the granules' parameters. In this way, we are able to further reduce the number of evaluations.

3. GA-AFFG SURROGATE MODEL

In order to reduce fitness evaluations, surrogate modeling used in this paper is based on the property of fuzzy granulation, which involves decomposition of the whole into parts [1]. An Information granule (fuzzy sets) formalizes the concept of finite precision representation of objects in real life situation, and reduces the core of an information system (both in terms of objects and features) in a granule universe drawn together by indistinguishability, similarity, proximity and functionality [33]. Granular computing concerns situations in which computation and operations are performed on information granules (clump of similar objects or points). Consequently, it leads to have both data compression and a gain in computation time, since in granular computing computations/operations are performed on granules, rather than on the individual points, so that, the computational effort is greatly reduced. The rest of this section describes a granulation-based technique for surrogate modeling that forms the base of our proposal [1].

3.1 Fuzzy Granules

In the part of granules construction, we initially have a random parent population, $P_i = \left\{x_1^i, x_2^i, ..., x_t^i\right\}$, where x_j^i is the j^{th} individual in the i^{th} generation. Conventional fuzzy clustering algorithms initially establish a partition of the universe of discourse with a fixed number of clusters (modifying latter the centers of granules according to the adaptive laws, and satisfying an objective function J) [4]. In this paper, following the main purpose of reducing fitness evaluations, it is desirable to explore the fitness landscape looking for individuals with similar characteristics. Hence, the base algorithm starts by finding a first cluster, and then goes to find the second one, and so on, see Figure 3. This process was introduced in [1]. For comparing new solutions to existing clusters a density function representing a data density measure is constructed; the density measure also describes a Gaussian similarity neighborhood between individuals. The density function is described by the equation,

$$\mu_k\left(x_j^i\right) = \exp\left(-\left(x_j^i - C_k\right)^2 / \left(\sigma_k\right)^2\right) \tag{1}$$

for $k = 1, 2, ..., l$ number of fuzzy granules, C_k is the vector of centers values, initially the first individual is chosen as the center of the granule. The fitness of individuals x_j^i is either calculated by computing the exact fitness function or associated with one of the existing granules, this is done according to the threshold measure of closeness[1],

$$\theta^i = \alpha \frac{\min\left\{f\left(x_1^{i-1}\right), f\left(x_2^{i-1}\right), ..., f\left(x_t^{i-1}\right)\right\}}{\bar{f}^{i-1}} \tag{2}$$

where $\bar{f}^i = \sum_{j=1}^{t} \frac{f\left(x_j^i\right)}{t}$, and $\alpha > 0$ is a constant of proportionality, $f(\cdot)$ is the real fitness evaluation, the minimal value is due to that the benchmark functions are subjected to be minimized. Then, according to θ, fitness evaluation of individuals is performed as follows,

$$f\left(x_j^i\right) = \begin{cases} f\left(C_k\right) & \text{if } \max\left\{\mu_k\right\} > \theta^i \\ f\left(x_j^i\right) & \text{computed by fitness function otherwise} \end{cases} \tag{3}$$

Radius σ_k of each granule is used to control the degree of similarity between individuals, this also determines the decision boundary of fitness evaluation of individuals, so that, the granules can shrink or enlarge in reverse proportion to their fitness,

$$\sigma_k = \gamma \frac{1}{\left(e^{f(C_k)}\right)^{\beta}} \tag{4}$$

[1]One should note that we have modified the expression for θ^i described in [1] in order to obtain better results.

where $\beta > 0$ is an emphasis operator, and γ is a constant of proportionality that is usually set at 1. Initially, σ_k is larger and then, as the algorithm evolves, fitness evaluations of best individuals will be evaluated in granules of reduced radius in a zone near the optimal solutions. Therefore, equation (4) adapts the width of the granules in order to have more exact fitness computed around individuals who perform very well, and fewer fitness computations around individuals who have poor performance. computational cost.

In order to prevent uncontrolled growth of fuzzy granules, only a pre-specified number of granules, with higher fitness values, will remain during each generation of the algorithm. Fig. 1 shows construction of fuzzy If-Then rules and group in granules for the case of two inputs, one output data pairs.

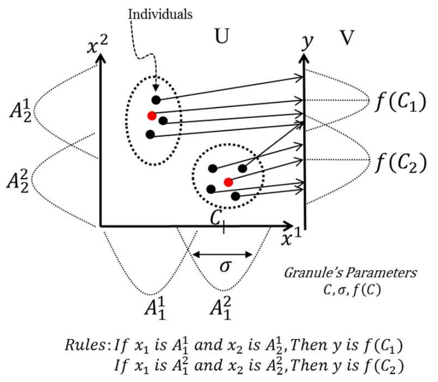
Individuals
U
V
x^2
y
A_2^1
A_2^2
$f(C_1)$
$f(C_2)$
C
x^1
σ
Granule's Parameters
$C, \sigma, f(C)$
A_1^1
A_1^2
Rules: If x_1 is A_1^1 and x_2 is A_2^1, Then y is $f(C_1)$
If x_1 is A_1^2 and x_2 is A_2^2, Then y is $f(C_2)$

Figure 1: Construction of fuzzy granules with the respective fuzzy rules for two input, one output case

4. NEURO-FUZZY: ADAPTIVE PART

In the first part of the algorithm, using fuzzy granulation property, the number of fitness function evaluations is reduced, this step is realized by the adaptive part of the threshold of similarity between individuals and modifying the radius of each granule by an adaptive law, see Equation (2). This process suggests a reduction of computational cost changing radius of granules, but these do not change in position during the search process. Therefore, properties of mutation and crossover, which let individuals in each generation jump to optimal values, will be enclosed in granules established from the very first generations. In this paper, we propose the use of a Neuro-Fuzzy Network that will adapt free parameters of granules, these parameters are: centers, radius and fitness values of granules. The adaptive part of the granules parameters will be performed every $n-$generations, and not every generation. This is because this could cause an increase of computational time of calculus and an increase in the number of fitness evaluations.

From the last section, we have values of the granules $G = \{C_k, \sigma_k, f(C_k)\}$, where C_k is the $m-$dimensional vector of centers, σ_k is the width of membership functions of the kth fuzzy granule and $f(C_k)$ is the fitness value of granules. The fuzzy system that we are going to design has the following form,

$$f(x) = \frac{\sum_{k=1}^{M} \bar{y}^k \left[\prod_{i=1}^{n} \exp\left(-\left(x_j^i - C_k\right)^2 / (\sigma_k)^2\right)\right]}{\sum_{l=1}^{M} \left[\prod_{i=1}^{n} \exp\left(-\left(x_j^i - C_k\right)^2 / (\sigma_k)^2\right)\right]} \quad (5)$$

where $\mu_k\left(x_j^i\right) = \exp\left(-\left(x_j^i - C_k\right)^2 / (\sigma_k)^2\right)$, represents the fuzzy sets or fuzzy granules, $\bar{y}^k$ is the fitness value of the respective granule $(f(C_k))$, M is fixed and indicates the number of fuzzy rules (fuzzy granules), $\bar{y}^k, C_k$ and σ_k are free parameters. To determine these parameters in some optimal fashion, it is helpful to represent the fuzzy system of granules (5) as a feedforward network. Specifically, the mapping from the input of individuals $x \in U \subset R^n$ to the output of their desired fitness values $f(x)$ can be implemented according to the following operations: first, the input x (individual) is passed through a product Gaussian operator (granule's structure construction) to become $z^k = \prod_{i=1}^{n} \exp\left(-\left(x_j^i - C_k\right)^2 / (\sigma_k)^2\right)$; then, the z^k are passed through a summation operator and a weighted summation operator (current fitness value of the granule) to obtain $b = \sum_{k=1}^{M} z^k$ and $a = \sum_{k=1}^{M} \bar{y}^k z^k$; finally, the output of the fuzzy system is computed as $f(x) = a/b$. This three-stage operation is shown in Fig. 2 as a three-layer feedforward network.

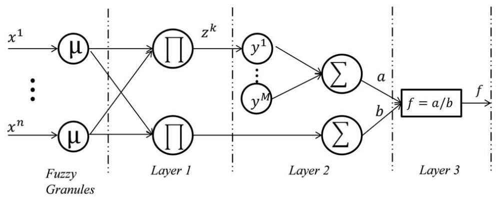

Figure 2: Three-layer feedforward network as a processing unit of fuzzy granules

4.1 Parameter update

Now, the task is updating parameters of the fuzzy granule's system (5), such that the matching error

$$e^p = \frac{1}{2}\left[f(x)^p - y^p\right]^2 \quad (6)$$

is minimized. That is, the task is to determine the parameters $\bar{y}^k, C_k$ and σ_k such that e^p of (6) is minimized. In the sequel, e, f and y will be used to denote $e^p, f(x^p), y^p$, respectively.

In this part, using a gradient descent algorithm to determine the parameters. Specifically, to determine $\bar{y}^k$, is using the algorithm

$$\bar{y}^k(q+1) = \bar{y}^k(q) - \alpha \frac{\partial e}{\partial \bar{y}^k}\big|_q \quad (7)$$

where $k = 1, ..., M, q = 0, 1, 2, ...,$ and α is a constant step size. If $\bar{y}^k(q)$ converges as q goes to infinity, then from (7) we have $\frac{\partial e}{\partial \bar{y}^k} = 0$ at the converged $\bar{y}^k$, which means that the converged $\bar{y}^k$ is a local minimum of e. From Fig. it is seen that f (and hence e) depend on $\bar{y}^k$ only through a, where $f = a/b, a = \sum_{k=1}^{M} \bar{y}^k z^k, b = \sum_{k=1}^{M} z^k$, and $z =$

$\prod_{i=1}^{n} \exp\left(-\left(x_j^i - C_k\right)^2 / (\sigma_k)^2\right)$; hence, using the Chain Rule, we have

$$\frac{\partial e}{\partial \bar{y}^k} = (f-y)\frac{\partial f}{\partial a}\frac{\partial a}{\partial \bar{y}^k} = (f-y)\frac{1}{b}z^k \tag{8}$$

Substituting (8) into (7), we obtain the training algorithm for $\bar{y}^k$:

$$\bar{y}^k(q+1) = \bar{y}^k(q) - \alpha\frac{f-y}{b}z^k \tag{9}$$

where $k = 1, 2, ..., M$, and $q = 0, 1, 2, ...$

To determine C_k, we use

$$C_k(q+1) = C_k(q) - \alpha\frac{\partial e}{\partial C_k}\mid_q \tag{10}$$

we see from Fig. 2 that f (and hence e) depend on C_k only through z^k; hence, using the Chain Rule, we have

$$\frac{\partial e}{\partial c^k} = (f-y)\frac{\partial f}{\partial z^k}\frac{\partial z^k}{\partial c^k} = (f-y)\frac{\bar{y}^k - f}{b}z^k\frac{2\left(x_i^p - c^k\right)}{\sigma_k^2} \tag{11}$$

Substituting (11) into (10), we obtain the training algorithm for C_k :

$$C_k(q+1) = C_k(q) - \alpha\frac{(f-y)}{b}\left(\bar{y}^k(q) - f\right)z^k\frac{2\left(x_i^p - c^k(q)\right)}{\sigma_k^2(q)} \tag{12}$$

Using the same procedure, we obtain the training algorithm for σ_k :

$$\begin{aligned}\sigma_k(q+1) &= \sigma_k(q) - \alpha\frac{\partial e}{\partial \sigma_k}\mid_q &(13)\\ &= \sigma_k(q) - \alpha\frac{f-y}{b} &(14)\\ &\left(\bar{y}^k(q) - f\right)z^k\frac{2\left(x_i^p - c^k(q)\right)}{\sigma_k(q)}\end{aligned}$$

The training algorithm (9), (12) and (14) performs an error back-propagation procedure, that will update parameters of granules like centers, radius and fitness values, providing the genetic algorithm process fast convergence rate as well as lower computational cost.

4.2 Setting-up the training data set

The data set used in the training phase of the Neuro-Fuzzy Network, is obtained from the granules' information of the last generations. The values of centers, width and fitness values of each granule, $G = \{C_k, \sigma_k, f(C_k)\}$ (Figure 1), at generation n-1, are the initial conditions of the network. These parameters will be updated as the weights of the network with the training algorithm. Now, after n-1 generations have passed, and that genetic operations like selection, mutation and reproduction have been performed in each generation, the proposed algorithm selects automatically the best-fitted values of a certain percentage of the last generations of granules, and then, the training data set is set up. The selection of a certain percentage of the best-fitness values is above a 50% or less of the total fitness values of the granules, and is used as the target of the network. This selection of a certain percentage of best-fitness values is related with the "natural selection process" of individuals in the GA, and is realized trying to avoid falling into a local optimum value.

4.3 Algorithm

Now, general steps of the algorithm GA-FGNFN are as follows,

1. An initial random population is created:
   $P_0 = \left\{x_1^1, x_2^1, ..., x_j^1\right\}$, where x_j^i is the jth individual in the ith population

2. Structural information data are obtained from the construction of the fuzzy set of granules (fuzzy partition). The number of fuzzy granules implies the number of rules of the model.

3. Individuals within a pre-specified threshold are evaluated with the center value of the granule, otherwise, construction of a new granule is done with the respective fitness evaluation of the new center.

4. After n number of generations has passed, and genetic operations are performed each generation. Now, parameters of the fuzzy granule structure will be updated with the Neuro-Fuzzy Network and the gradient descent parameter update law. The training data of the Neuro-Fuzzy Network are the centers of granules, width of granules and fitness values of these granules. In this step, fitness function values of granules will be replaced by a certain n_i percent of the total fitness value of granules, replacing by lowest fitness function values or highest values (depending on the maximization or minimization case, respectively). This is the data generated by the granule's process, that will be used as the training data to update the parameters of the Neuro-Fuzzy system.

5. Tuning fuzzy parameters, back-propagating the error through the network and updating each one of the main parameters of granules, that is, centers, radius and fitness values.

6. Perform genetic operations and continue the genetic algorithm process for the next generations until again, update of granule's parameters is required after the pre-specified n number of generations, and until certain stop criteria or total number of generations is achieved.

We can see a flowchart of the GA-FGNFN in Figure 3, the next section reports results of experiments that aim at assessing the effectiveness of the proposed approach.

5. NUMERICAL RESULTS

This section reports experimental results that aim at showing the effectiveness of the GA-FGNFN approach. For the experimental study we considered a set of benchmark functions that comprise a variety of problems with the following characteristics: nonlinearity, unimodality, multimodality, low and high dimensionality [3, 5]. The set of benchmark functions we considered is described in Table 1.

The genetic algorithm in this paper was run with random initial populations, decimal-coded chromosomes, single-point crossover, mutation, fitness scaling, and an elitist stochastic universal sampling selection strategy. Parameters of simulations, percentage of crossover: 1, percentage of mutation: 0.01, population size: 20, and generation size: 100. In these

Function	Formulation
F_1 : De Jong	$\sum_{i=1}^{m} x_i^2$
Dimensions, range	$m = 3, -5.12 \leq x_i \leq 5.12$
F_2 :	$\sum_{i=1}^{m} x_i^4 + Gauss\left(0,1\right)$
Dimensions, range	$m = 30, -1.28 \leq x_i \leq 1.28$
F_3 : Michalewicz	$-\sum_{i=1}^{m}\left(\sin\left(x_i\right)\cdot\left(\sin\left(\frac{i\cdot x_i^2}{\pi}\right)\right)^{(2\cdot n)}\right)$
Dimensions, range	$n = 10, m = 10, 0 \leq x_i \leq \pi$
F_4 : Rastrigin	$10\cdot m + \sum_{i=1}^{m}\left(x_i^2 - 10\cdot\cos\left(2\cdot\pi\cdot x_i\right)\right)$
Dimensions, range	$m = 20, -5.12 \leq x_i \leq 5.12$
F_5 : Schwefel	$\sum_{i=1}^{m}\left(-x_i\cdot\sin\left(\sqrt{\lvert x_i\rvert}\right)\right)$
Dimensions, range	$m = 20, -500 \leq x_i \leq 500$
F_6 : Griewangk	$1 + \sum_{i=1}^{m}\frac{x_i^2}{4000} - \prod_{i=1}^{m}\cos\left(\frac{x_i}{\sqrt{i}}\right)$
Dimensions, range	$m = 20, -600 \leq x_i \leq 600$

Table 1: The set of benchmark functions used for the performance evaluation of the GA-FGNFN Algorithm

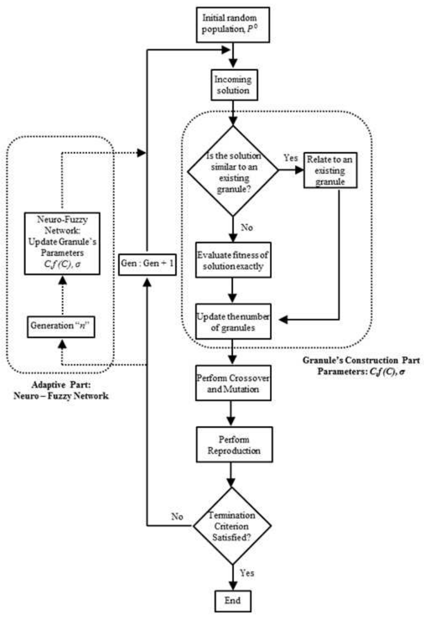
Initial random population, P^0
Incoming solution
Is the solution similar to an existing granule?
Yes
Relate to an existing granule
No
Evaluate fitness of solution exactly
Update the number of granules
Granule's Construction Part Parameters: $C, f(C), \sigma$
Neuro-Fuzzy Network: Update Granule's Parameters $C, f(C), \sigma$
Generation "n"
Adaptive Part: Neuro – Fuzzy Network
Gen : Gen + 1
Perform Crossover and Mutation
Perform Reproduction
Termination Criterion Satisfied?
No
Yes
End

Figure 3: Flowchart of GA-FGNFN Algorithm

simulations, update of parameters of the granules is required every n generations.

We report in each of our experiments the average performance obtained by the different methods over five independent runs. Simulations are compared among the ordinary genetic algorithm (GA) of Matlab, using the same parameters above mentioned; the Genetic Algorithm with Adaptive Fuzzy Fitness Granulation (GA-AFFG) proposed in [1]; and the proposed GA-FGNFN algorithm. It is important to mention that due to the proved efficiency of the Matlab algorithm in GA, GA-AFFG and our proposed algorithm GA-FGNFN, were implemented over the Matlab-GA base algorithm. We report the average of: number of fitness function evaluations, percentage of optimal solutions reached and best fitness value. Also we report the $p-value$ obtained with a paired statistical $t-test$, comparing the performance of the standard GA algorithm and the other variants. Unless otherwise stated the learning rate parameter of the Network was fixed to: 0.1.

Average results of simulations are presented in Table 2. From this table we can see that in terms of reduction of real function evaluations (No. FFE) the proposed method obtains the best performance for all of the considered functions. The decrease of number of evaluations is of about 80% less than simulations with the GA-Matlab algorithm, and about 40% less compared with GA-AFFG. These results are obtained without degrading significantly the quality of the solutions found. The $p-$values of each one of the benchmark functions, shows that there was not a statistically significant difference.

Another point to be analyzed is the question of when to apply the Neuro-Fuzzy Network? Therefore, a comparison table of number of generations in which, when the algorithm should be applied, and reduction in fitness values is presented for the benchmark functions in the Table 3. As before, the main interest is to keep a trade-off between reduction of fitness function evaluations and quality of solutions. The results from Table 3, were obtained by running the GA-FGNFN algorithm every $n = \{5, 10, 20, 30, 50\}$ generations. In the rest of this section we analyze the performance of our method for each of the considered functions making reference to results from both Tables 2 and 3.

Function	Values	GA	GA-AFFG	GA-FGNFN
F_1	No. FFE	2000	268	220
	% Opt. Sol.	80%	80%	80%
	$p -$ value		0.35617	0.45752
	Learning Par.		$\beta = 0.1, \gamma = 1$	$\beta = 0.1, \gamma = 1$ $n = 20, n_i = 100, \mu = 0.1$
	Best-Fitness Value	0.10908	0.1092	0.1144
F_2	No. FFE	2000	730.6	509.8
	% Opt. Sol.	80%	80%	80%
	$p -$ value		0.2505	0.12681
	Learning Par.		$\beta = 0.04, \gamma = 1.7$	$\beta = 0.001, \gamma = 2$ $n = 50, n_i = 100, \mu = 0.1$
	Best-Fitness Value	0.8873	1.4667	1.0795
F_3	No. FFE	2000	750.5	209
	% Opt. Sol.	80%	80%	80%
	$p -$ value		0.17	0.123
	Learning Par.		$\beta = 0.4, \gamma = 1.85$	$\beta = 0.01, \gamma = 1.5$ $n = 50, n_i = 5, \mu = 0.1$
	Best-Fitness Value	-4.3514	-2.08	-2.967
F_4	No. FFE	2000	842	332
	% Opt. Sol.	60%	60%	60%
	$p -$ value		0.4568	0.3654
	Learning Par.		$\beta = 0.004, \gamma = 0.15$	$\beta = 0.004, \gamma = 120$ $n = 50, n_i = 100, \mu = 0.9$
	Best-Fitness Value	103.8504	126.710	109.9558
F_5	No. FFE	2000	945.33	253
	% Opt. Sol.	60%	60%	60%
	$p -$ value		0.2610	0.1456
	Learning Par.		$\beta = 0.0008, \gamma = 300$	$\beta = 0.004, \gamma = 200$ $n = 50, n_i = 100, \mu = 0.1$
	Best-Fitness Value	-3435.37	-3265.49	-3394.5
F_6	No. FFE	2000	863.53	427.6
	% Opt. Sol.	80%	80%	80%
	$p -$ value		0.69206	0.23058
	Learning Par.		$\beta = 0.00012, \gamma = 190$	$\beta = 0.009, \gamma = 100$ $n = 20, n_i = 100, \mu = 0.01$
	Best-Fitness Value	80.8125	105.1937	80.4752

Table 2: Comparison of benchmark functions with a conventional Matlab-GA algorithm, GA-AFFG and GA-FGNFN Algorithms

In the case of F_1 De Jong's and F_2 functions, which are unimodal, good results are obtained in terms of solutions' quality; F_1 is low-dimensional and the neural network can be implemented every 10 generations without modifying the computational time and the quality of solutions; however, in the case of F_2 it is not recommended to constantly apply the neural network due to its high-dimensionality, in this case, the number of evaluations in the learning process of the neural network increase considerably; therefore, we apply the learning process for F_2 each 50 generations.

The F_4 Rastrigin's function is based on De Jong's function with the addition of cosine modulation to produce many local minima, thus, it is highly multi-modal, however, the locations of the minima are regularly distributed. In this case, GA-FGNFN algorithm, presents reduction of the fitness evaluations in about 80% compared with the traditional genetic algorithm, it is recommended to apply the neural network each 50 generations due to the high-nonlinearity and high-dimensionality of the function.

The F_6 Griewangk's function is similar to Rastrigin's function, it has many widespread local minima, however, the locations of the minima are regularly distributed. In this case, the total number of optimal solutions is better in comparison with the other functions, reducing the number of fitness evaluations in a 88%, better results were obtained when the neural network is applied each 20 generations.

For F_3 Michalewicz function, if we apply the neural network each 50 generations, the best results are obtained in terms of percentage of optimal solutions and with a good number of fitness evaluations.

Summarizing, we can say that the proposed approach effectively reduces the number of fitness function evaluations without degrading the quality of solutions. The proposed method outperforms the base approach [1] in terms of reduction of evaluation of the fitness function. We would like to emphasize that in preliminary experimentation we optimized the parameters of the reference approach to have a competitive baseline for our method. Regarding our method, we have to be careful when the dimensionality of data increases, the total number of computing operations also increase and the processes of granule's construction together with the adaptive part, increase in the number of operations

Function	No. of Generations Parameters	$n = 5$	$n = 10$	$n = 20$	$n = 30$	$n = 50$	GA - Matlab
F_1 :De Jong	No. FFE	1740	259	260	220	295	2000
	% Opt. Sol.	60%	80%	80%	80%	80%	80%
	Best-Fitness Value	1.654	0.1092	0.1144	1.1130	1.6506	0.10908
F_2	No. FFE	2000	460.8	569.2	455.8	440	2000
	% Opt. Sol.	80%	40%	40%	80%	80%	80%
	Best-Fitness Value	1.1706	1.0795	2.2572	1.9965	1.0182	0.8873
F_3 :Michalewicz	No. FFE	119.2	147.2	173	282	381.6	2000
	% Opt. Sol.	60%	60%	60%	40%	80%	80%
	Best-Fitness Value	-2.8510	-2.77	-2.49	-2.6287	-2.967	-4.3514
F_4 :Rastrigin	No. FFE	457	412	370	365	332	2000
	% Opt. Sol.	40%	60%	60%	60%	60%	80%
	Best-Fitness Value	128.3852	116.9746	157.8378	169.5834	109.9558	103.8504
F_5 :Schwefel	No. FFE	288	221	321	367	252.8	2000
	% Opt. Sol.	40%	40%	40%	40%	80%	80%
	Best-Fitness Value	-4786.24	-4293.34	-3987.31	-3897.3	-3394.5	-3435.37
F_6 :Griewangk	No. FFE	414	412	884	1055	964	2000
	% Opt. Sol.	60%	80%	60%	60%	60%	80%
	Best-Fitness Value	111.3852	80.4752	86.3752	105.2752	143.9852	80.8125

Table 3: Different number of generations where the GA-FGNFN was applied, percentage of optimal solutions, and best-fitness value. The last column has the values of GA-Matlab implementation

per generation. In general, we must consider, to modify parameters such as γ which modifies the shrinkage of granules and it is related to having more or less fitness evaluations on granules; the parameter μ, of the neural networks, allows to have the best approximation to the training data but also, it can cause an increase of evaluations when it is lower than the recommended value of 0.1; the last consideration is how many times do we have to apply the neural network, in general, according to the results, it is recommended to apply it 2 times per total number of generations, this is, each 50 generations in our case.

6. CONCLUSIONS

Genetic algorithms produce in each generation individuals with similar characteristics, where the objective is to try to identify those who present better conditions of adaptability and improved performance; this condition is measured by the fitness function. For some problems, estimating the fitness function can be a very time consuming process, slowing the optimization process. Surrogate modeling has been adopted for substituting the real fitness function by approximations obtained by a model.

This paper introduces a surrogate modeling approach based on granular computing and a Neuro Fuzzy Network. Granular computing leads to have data compression, gain in computation time and knowledge extraction of this individual with respective similarities between neighbors. The main idea of this paper is to take advantage of this information and use it to elaborate the structure of an adaptive network, that, according to genetic algorithms objective, will produce a reduction in fitness evaluations and, therefore, a reduction of the computational burden of the process. Our proposal extends a previous model by allowing the adaptation of granules' parameters.

Results of experiments over a suite of classical benchmark functions show the efficiency of the algorithm GA-FGNFN in reduction of fitness evaluations with at least 40 % compared with a standard genetic algorithm without compromising significantly the quality of solutions. The proposed approach also outperformed a state-of-the-art surrogate model based on granular computing. Comparative studies reveal the main importance in keeping a trade-off between reduction of fitness function evaluations and also in percentage of optimal solutions.

Future work directions include implementing a weight-decay approach in order to further improve the adaptation capabilities of our method. Also we are working on the application of the proposed surrogate for highly-complex problems from engineering applications.

7. REFERENCES

[1] M.R. Akbarzadeh-T, M. Davarynejad, and N. Pariz. Adaptive fuzzy fitness granulation for evolutionary optimization. International Journal of Approximate Reasoning, 49(3):523–538, 2008.

[2] J. C. Bezdeck, "Pattern Recognition with Fuzzy Objective Function Algorithms", Kluwer, 1981.

[3] K.A. De Jong, An analysis of the behavior of a class of genetic adaptive systems, Ph.D. Thesis, University of Michigan, 1975.

[4] J. V. De Oliveira, W. Pedrycz, Advances in Fuzzy Clustering and its Applications. John Wiley and Sons, 2007.

[5] J. G. Digalakis, K. G. Margaritis, An experimental study of benchmarking functions for Genetic Algorithms. IEEE International Conference on Systems, Man, and Cybernetics, Vol. 5, pp. 3810–3815, 2002.

[6] E. Ducheyne, B. De Baets, R. deWulf, Is fitness inheritance useful for real-world applications? Evolutionary Multi-Criterion Optimization, LNCS 2631, pp. 31–42 Springer, 2003,.

[7] Chen, S. Billings, P. Grant, Recursive hybrid algorithm for nonlinear system identification using

radial basis function networks, Int. J. Control 55 (1992) 1051–1070.

[8] J.-H. Chen, D. Goldberg, S.-Y. Ho, K. Sastry, Fitness inheritance in multiobjective optimization, in: Proceedings of the 2002 International Conference on Genetic and Evolutionary Computation Conference, pp. 319–326, 2002.

[9] M. Farina. A neural network based generalized response surface multiobjective evolutionary algorithms. IEEE Congress on Evolutionary Computation, pp. 956–961, 2002.

[10] F. M. Filho, F. Gomide, Fuzzy Clustering in Fitness Estimation Models for Genetic Algorithms and Applications, IEEE International Conference on Fuzzy Systems, pp. 1388–1395, 2006.

[11] L. Graening, Y. Jin, and B. Sendhoff. Efficient evolutionary optimization using individual-based evolution control and neural networks: A comparative study. In European Symposium on Artificial Neural Networks, pp. 273-278, 2005.

[12] A.K. Jain, M.N. Murty , and P .J. Flynn. Data clustering: A review. ACM Computing Surveys, 31(3):264-323, 1999.

[13] Jin Y., Hüsken M., Sendhoff B., Quality measures for approximate models in evolutionary computation, Proceedings of GECCO Workshops: Workshop on Adaptation, Learning and Approximation in Evolutionary Computation, pp. 170–174, 2003.

[14] I. C Kampolis, D. I. Papadimitriou, and K. C. Giannakoglou. Evolutionary optimization using a new radial basis function network and the adjoint formulation. Inverse Problems in Science and Engineering, 14:397–410, 2006.

[15] M. K. Karakasis, K. C. Giannakoglou, On the use of Surrogate Evaluation Models in Multi-Objetive Evolutionary Algorithms, European Congress on Computational Methods in Applied Sciences and Engineering, 2004.

[16] H. S. Kim and S. B. Cho, "An efficient genetic algorithms with less fitness evaluation by clustering," IEEE Congress on Evolutionary Computation, pp. 887–894, 2001.

[17] L. Ma, K. Xin, and S. Liu. Using Radial Basis Function Neural Networks to Calibrate Water Quality Model. Proceedings of World Academy of Science, Engineering and Technology, pp. 385–393, 2008.

[18] R. Myers, D. Montgomery, Response Surface Methodology, John Wiley and Sons, 1995.

[19] A. Ratle, "Kriging as a surrogate fitness landscape in evolutionary optimization,"Artif. Intell. Eng. Design Manufact., vol. 15(1):37–49, May 2001.

[20] M. Reyes-Sierra, C.A. Coello, Dynamic fitness inheritance proportion for multiobjective particle swarm optimization, Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation, 2006.

[21] J. Sacks, W.J. Welch, T.J. Michell, H.P. Wynn, Design and analysis of computer experiments, Statistical Science 4(4):409–435, 1989.

[22] M. Salami, T. Hendtlass, The fast evaluation strategy for evolvable hardware, Genetic Programming and Evolvable Machines 6(2):139–162, 2005.

[23] M. Salami, T. Hendtlass, "A Fitness Estimation Strategy for Genetic Algorithms", 15th Int. Conf. on Industrial and Engineering. Applications of Artificial Intelligence and Expert Systems, Vol. 2358, pp. 319-326, 2002.

[24] A. Schmitz, E. Besnard, and E. Vivies. Reducing the cost of computational fluid dynamics optimization using multilayer perceptrons. In IEEE 2002 World Congress on Computational Intelligence, IEEE.

[25] G. Schneider, W. Schrödl, and G. Wallukat. Peptide design by artificial neural networks and computaer-based evolutionary search. Proceedings of National Academy of Science, 95:12197-12184, 1998.

[26] R. Smith, B. Dike, and S. Stegmann, "Fitness inheritance in genetic algorithms," inProc. ACM Symp. Appl. Comput., pp. 345–350, 1995.

[27] L.X.Wang,Adaptive Fuzzy Systems and Control, Prentice-Hall, Englewood Cliffs, NJ,1994.

[28] G.S.Watson,Smooth regression analysis, Sankhya Ser. A 26 (1964) 101–116.

[29] M. Vijayalakshmi and M. Renuka Devi. A survey of different issues of different clustering algorithms used in large data sets. International Journal of Advanced Research in Computer Science and Software Engineering, 2:305–307, 2012.

[30] Y. Jin, B. Sendhoff, Reducing fitness evaluations using clustering techniques and neural networks ensembles, Proceedings of the Genetic and Evolutionary Computation Conference - GECCO, LNCS 3102 688–699, Springer, 2004.

[31] Y. Jin, "A comprehensive survey of fitness approximation in evolutionary computation,"Soft Comput., vol. 9, no. 1, pp. 3–12, Jan. 2005.

[32] Y. Jin, M. Olhofer, and B. Sendhoff, "On evolutionary optimization with approximate fitness functions," Proc. Genet. Evol. Comput. Conf., 2000, pp. 786–792.

[33] L. F. Zadeh, Toward a theory of fuzzy information granulation and its centrality in human reasoning and fuzzy logic, Fuzzy Sets and Systems, 90(2):111–127, 1997.

[34] Zhang J., Chowdhury S., Messac A., An Adaptive Hybrid Surrogate Model, Structural and Multidisciplinary Optimization, 46(2):223–238, 2012.

[35] Z. Zhou, Y. S. Ong, P. B. Nair, A. J. Keane, and K. Y. Lum, Combining global and local surrogate models to accelerate evolutionary optimization, IEEE Trans. Syst., Man, Cybern. C, 37(1):66–76, 2007.