

## Genetic Fuzzy System for Financial Management

*Penka V. Georgieva*

*Burgas Free University, 62 San Stefano Str., 8001 Burgas, Bulgaria*

*E-mail: pgeorg@bfu.bg*

**Abstract:** *This paper discusses genetic fuzzy systems – hybrid systems of artificial intelligence combining the potential of fuzzy sets for modeling approximate reasoning with the abilities of genetic algorithms for finding optimal solutions. The use of genetic algorithms for optimizing the parameters of a fuzzy system is demonstrated on GFSSAM.*

**Keywords:** *Genetic fuzzy system, intelligent hybrid systems, financial management, artificial intelligence.*

### 1. Introduction

This paper looks into designing and implementation of a hybrid system, based on genetic algorithms and fuzzy sets theory. The main concepts of hybridization are applied on Genetic Fuzzy Software System for Asset Management (GFSSAM) which is a hybrid software system, built on previously created and tested Fuzzy Software System for Asset Allocation Management (FSSAM), published in [1]. In GFSSAM, a genetic algorithm is implemented for finding optimal values of the parameters of FSSAM.

Artificial Intelligence (AI) aims at developing various techniques for knowledge presentation and knowledge discovery; intelligent search; dealing with inaccurate and/or uncertain data and knowledge; machine learning etc. The hybrid systems of AI are designed and created so that the advantages of one or more of AI computing paradigms are used to compensate the disadvantages of the others. Genetic fuzzy systems are integrated hybrid systems of AI that combine the potential of fuzzy sets theory for modeling the reasoning process with the abilities of genetic algorithms for finding optimal solutions.

### 2. Hybrid systems based on fuzzy sets and genetic algorithms

Conceptually fuzzy systems are considered to be able to solve non-linear problems in a variety of applications such as classification, modeling, management and others. Their key feature is the ability to model expert human knowledge but the main deficiency of such systems is the lack of ability to learn and adapt. The first attempts to add other techniques to fuzzy systems start around 1990. Two types of

hybridization are considered to be the most successful and they are neural-fuzzy systems and genetic fuzzy systems.

Genetic processes are applied to fuzzy systems for solving problems of different degree of complexity: From the simplest case for optimization of the parameters of a fuzzy system to the most complex – for training its rule-base. A reverse approach (namely hybridization for improving genetic algorithms with fuzzy tools) is proposed by Herrera et al. [2].

Two basic approaches exist in creating hybrid systems based on fuzzy sets theory and genetic algorithms: 1) fuzzy sets theory is used for modeling the components and adjusting the parameters of genetic algorithms and these are called *fuzzy genetic algorithms* (GA\_FIS = Genetic Algorithms with Fuzzy Inference System); 2) genetic algorithms are used to solve optimization or search problems related to fuzzy systems and thus Genetic Fuzzy Systems (GFS) are obtained.

The process of designing a fuzzy rule-based system can be defined as an optimization problem for finding most suitable built-in variables, parameters and rules. Moreover, genetic algorithms are a widely used technique for global extrema search, as they show the ability to find nearly optimal solutions with the possibility of using a priori knowledge concerning the search space. For a fuzzy rule-based system a priori knowledge is the information about the type of membership functions, fuzzy rules and the architecture of the fuzzy system itself [3].

### **Genetic fuzzy rule-based systems**

Fuzzy rule-based systems mimic the decision-making process by handling the available information in a human-like manner. The behavior of a fuzzy rule-based system depends on three sets of parameters: 1) fuzzy sets, associated with linguistic variables that define the semantics of the rules; 2) fuzzy rules, determining how the output variables are to be derived, and 3)  $t$ -norms and  $s$ -norms used for aggregation and defuzzification.

The process of creating a fuzzy system starts with designing the system's architecture, which is a relatively easy task. The most difficult, and requiring significant amount of resources, is the stage of setting up the parameters of the system so that the obtained output results are feasible.

Following the general structure of a fuzzy system, each fuzzy software rule-based system consists of a knowledge base, incorporating the data base and the rule base, and an inference machine. A precise description of the inference machine of FSSAM is published in [1].

The explicit structure of a fuzzy rule-based system follows the logic of the fuzzy reasoning process. In the process of hybridization it is a must to determine the number of parameters to be optimized. The following are the considerations about that number.

Let  $N$  be the number of the input fuzzy variables  $K_i$ ,  $i = 1, 2, 3, \dots, N$ , and  $n_i$  be the number of the terms  $X_{ij}$  of  $K_i$  for each  $i$ , where  $j = 1, 2, 3, \dots, n_i$ .

Let  $S$  be the number of the output fuzzy variables  $Q_s$ ,  $s = 1, 2, 3, \dots, S$ , and  $p_s$  be the number of the terms  $Y_{sp}$  of  $Q_s$  for each  $s$ , where  $p = 1, 2, 3, \dots, p_s$ .

Let  $\mu_{ij}(x)$  be the membership function of the term  $X_{ij}$  and  $\mu_{sp}(y)$  be the membership function of the term  $Y_{sp}$ .

Then the total number of the membership functions in the knowledge base is

$$(1) \quad N \cdot \sum_{i=1}^N n_i + S \cdot \sum_{s=1}^S p_s.$$

The numerical values of the input data form a  $N$ -dimensional vector  $x^* = (x_1^*, x_2^*, \dots, x_N^*)$  with crisp values. This vector is fuzzified by calculating the values of the membership functions  $\mu_{ij}(x_i^*)$  for each  $i$  and  $j$ . Thus the number of the values of the membership functions that are stored in the data base is

$$(2) \quad N \cdot \sum_{i=1}^N n_i.$$

The next procedure is the aggregation. Let the  $t$ -norm, defined by the *min* operator, be used for the logical *AND* operator and the  $s$ -norm is defined by the *max* operator. The  $m$ -th rule  $R_m$  in the rule base has the form

$$\begin{aligned} & \text{IF} \\ & \{K_{m_1} \text{ is } X_{m_1 j_{m_1}}\} \text{ AND } \{K_{m_2} \text{ is } X_{m_2 j_{m_2}}\} \text{ AND } \dots \text{ AND } \{K_{m_k} \text{ is } X_{m_k j_{m_k}}\} \\ & \text{THEN} \\ & \{Q_{m_1} \text{ is } Y_{m_1 j_{m_1}}\} \text{ AND } \{Q_{m_2} \text{ is } Y_{m_2 j_{m_2}}\} \text{ AND } \dots \text{ AND } \{Q_{m_l} \text{ is } Y_{m_l j_{m_l}}\} \end{aligned}$$

for  $m = 1, 2, \dots, M$ , where  $M$  is the number of rules.

After the choice and execution of the  $m$ -th rule the values of  $\Theta_m$  and  $\Theta_m^o$  are calculated sequentially according to the following formulae:

$$(3) \quad \Theta_m = \min_k \{ \mu_{m_1 j_{m_1}}(x_1^*), \mu_{m_2 j_{m_2}}(x_2^*), \dots, \mu_{m_k j_{m_k}}(x_k^*) \},$$

$$(4) \quad \Theta_m^o = \Theta_m,$$

where  $w_m$  is the weight of the  $m$ -th rule for  $m = 1, 2, \dots, M$ .

Once all the rules in the rule-base are executed, the membership degrees  $\mu_{sp}^m = \Theta_m^o$  are derived for each term  $Y_{sp}$  of the output variables.

The aggregation is then obtained by calculating the values:

$$(5) \quad P_{sp} = \max_{s,p} \{ \mu_{sp}^1, \mu_{sp}^2, \dots, \mu_{sp}^M \},$$

for each  $Y_{sp}$ , where  $s = 1, 2, 3, \dots, S$  and  $p = 1, 2, 3, \dots, p_s$ .

The last procedure is defuzzification. Here, a crisp output value is obtained and defuzzification is derived by implementing *Center of gravity method*, *Median method*, *Center of max method*, *Min of max method* or *Max of max method*.

The applications of fuzzy rules-based systems are various and numerous and some of them can be studied in detail in [5-12], etc.

### 3. GFSSAM

The ultimate goal of every decision making process is to find an optimal solution for a certain problem when a number of possible solutions exists. Historically, Bellman and Zadeh are the first to propose a fuzzy model for decision-making. They treat

objectives and goals as fuzzy sets and the solution is calculated by aggregating these sets. There are various algorithms for building fuzzy systems and detailed overview is proposed in [13-19] and others.

At the core of genetic fuzzy systems lies the idea that the advantages of evolutionary computation and fuzzy systems can be combined. There are different ways to apply evolutionary computing to fuzzy systems – genetic algorithms, genetic programming or evolutionary strategies can be used in order to obtain a genetic fuzzy system.

GFSSAM aims at an automatic set up of the system's parameters. In this case the fuzzy system architecture, the domains of the linguistic variables and the used operators are known in advance. Three distinctive situations exist with regard to the fuzzy system knowledge base:

- The rule base is known, but the fuzzy variables are either unknown or only approximated and need optimization;
- The semantics of fuzzy rules is at least approximately known, but the rules themselves are not defined precisely;
- Neither the fuzzy variables nor the fuzzy rules are known.

In each of these situations, the genetic algorithm is applied differently.

In the preliminary stages of designing a genetic fuzzy system the main goal is to automate the process of generating knowledge base, which is actually an optimization problem or a search problem for an optimal solution. This process consists of finding a suitable for the situation knowledge base, and after the parameterization of this knowledge base – adjusting the values of the parameters so that they are optimal depending on the optimization criteria.

In case of unknown fuzzy variables and known fuzzy rules, the problem is reduced to an optimization problem with constraints on an unlimited domain. If the fitness function is smooth, the conventional methods for optimization are applicable and generally they are faster than the genetic algorithms. The reason for using genetic algorithms in this situation is that first – the fitness function is not smooth, and second – a genetic algorithm gives several optimal solutions as an output result.

### 3.1. Designing a model of a genetic fuzzy system

The main objective in designing the model of GFSSAM is to use a genetic algorithm optimization on an already existing fuzzy system. The preliminary created system is FSSAM which econometric base, architecture and performance are described in details in [1, 7].

As can be seen from the general scheme of a genetic fuzzy system, presented on Fig. 1, the genetic algorithm is used for optimization of the fuzzy system's knowledge base. The knowledge base of FSSAM consists of the membership functions of the terms of input and output fuzzy variables, as well as the rule-base.

The design of the hybrid system has to provide the simultaneous tuning of all parameters of the membership functions by finding optimal values without changing their type or the rule-base.

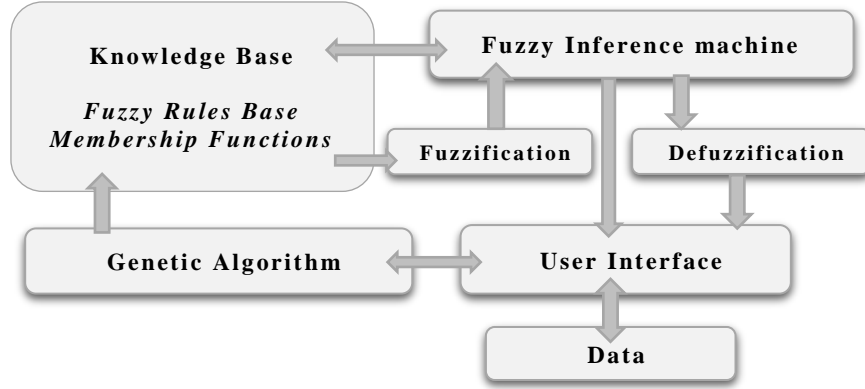


Fig. 1. Genetic fuzzy system for tuning the knowledge base

### 3.2. Genetic algorithm

Genetic Algorithm (GA) is an adaptive algorithm, defined by an ordered septenary of operators and parameters:

$$GA = (\mathcal{P}, \mathbf{PP}, \mathcal{F}, \mathcal{S}, \Omega, \Psi, \zeta),$$

where  $\mathbf{PP}$  is a population with size  $\mathcal{P}$  consisting of chromosomes  $\mathbf{c}^j$ :

$$\mathbf{c}^j = (c_1^j, c_2^j, \dots, c_l^j) \in \mathbf{PP}, \quad j = 1, 2, \dots, \mathcal{P},$$

which are  $l$ -dimensional binary vectors (GFSSAM is defined and realized on binary chromosomes presentation);

$\mathcal{F}$  is a function of  $l$  variables over  $\mathbb{R}^+$ , called *fitness function*, and

$$\mathcal{F}: \mathbf{c}^j \rightarrow \mathbb{R}^+, \quad j = 1, 2, \dots, \mathcal{P};$$

$\mathcal{S}$  is a selection operator for choosing  $u$  parents  $\mathbf{p}^k$  from the population  $\mathbf{PP}$ , and

$$\mathcal{S}: \mathbf{PP} \rightarrow \{\mathbf{p}^1, \mathbf{p}^2, \dots, \mathbf{p}^u\};$$

$\Omega$  is a set of genetic operators,

$$\Omega = \{\Omega_{\text{Cross}}, \Omega_{\text{Mut}}, \dots\},$$

where  $\Omega_{\text{Cross}}$  is a crossover operator,  $\Omega_{\text{Mut}}$  is a mutation operator, generating  $v$  children  $\mathbf{q}^m$  from  $u$  parents  $\mathbf{p}^k$ ,

$$\Omega: \{\mathbf{p}^1, \mathbf{p}^2, \dots, \mathbf{p}^u\} \rightarrow \{\mathbf{q}^1, \mathbf{q}^2, \dots, \mathbf{q}^v\};$$

$\Psi$  is a removal operator for  $v$  chromosomes from the  $i$ -th population and then the  $i + 1$  population is obtained according to the formula

$$\mathbf{PP}(i + 1) = \mathbf{PP}(i) - \Psi(\mathbf{PP}(i)) + \{\mathbf{q}^1, \mathbf{q}^2, \dots, \mathbf{q}^v\};$$

$\zeta$  is a criterion for end.

The  $\mathcal{S}$  and  $\Omega$  operators are always probabilistic, while  $\Psi$  can be either probabilistic or deterministic.

Every GA is a consecutive computation of populations

$$\mathbf{PP}(0), \mathbf{PP}(1), \dots, \mathbf{PP}(i), \mathbf{PP}(i + 1), \dots$$

with a randomly chosen initial population  $\mathbf{PP}(0)$ .

GA can also be described as a procedure for solving an optimization problem:

$$\max\{F(\mathbf{c}) \mid \mathbf{c} \in \{0, 1\}^l\} \quad \text{or} \quad \min\{F(\mathbf{c}) \mid \mathbf{c} \in \{0, 1\}^l\},$$

where  $F$  is the objective function,  $\mathbf{c} \in \mathbf{PP}$  is a binary vector of length  $l$ ,  $\mathbf{PP}$  is the space of possible solutions of size  $\mathcal{P}$ .

The performance of every GA depends on several parameters (INPUT), calculates the best population (OUTPUT) and has eight steps.

INPUT: *number of generations, population size, number of variables, number of bits, crossover probability, probability mutation, elitism, the upper limits, lower limits*

Generating the initial population

INPUT: population size

OUTPUT: initial population, value of fitness function

Encoding of the population

INPUT: population, population size, number of generations, number of bits, upper limits, lower limits

OUTPUT: population

Calculating fitness

INPUT: population, population size, number of variables

OUTPUT: values of fitness function

Selection

INPUT: population, values of fitness function, population size

OUTPUT: selected population

Crossover

INPUT: Selected population, crossover probability

OUTPUT: population after crossover

Mutation

INPUT: population after crossover, mutation probability

OUTPUT: population after mutation, number of mutations, matrix of mutated bits positions

Calculating fitness

INPUT: population after mutation, population size

OUTPUT: values of fitness function of the mutated population

Elitism

INPUT: values of fitness function of the mutated population, population after mutation

OUTPUT: BEST POPULATION AFTER MUTATION

OUTPUT: BEST POPULATION

### 3.3. Structure of FSSAM

FSSAM is an independent software system which consists of procedures for data collection and data storage, asset evaluation and investment portfolios construction. The system consists of three modules – Data Managing Module (DMM),  $Q$ -measure Fuzzy Logic Module (QFLM), Portfolio Construction Module (PCM), that are described in full detail in [1]. However, a brief overview of the second module is needed in this paper.

QFLM is an application based on fuzzy sets theory. Input data are the crisp numerical values of asset characteristics obtained from DMM – return, risk and  $q$ -ratio. These crisp values are fuzzified and after applying the aggregation rules, a fuzzy variable ( $Q$ -measure) for each of the assets is derived. The output is the defuzzified crisp value of  $Q$ -measure.

The linguistic variables are four: Three input variables and one output variable. The input linguistic fuzzy variables  $K_i$  are three ( $N = 3$ ) and their names correspond to the characteristics of an asset:  $K_1 \triangleq$  return,  $K_2 \triangleq$  Risk and  $K_3 \triangleq$   $q$ -ratio. The term-sets are  $T(K_1)=\{X_{1j}\}$ ,  $T(K_2)=\{X_{2j}\}$ ,  $T(K_3)=\{X_{3k}\}$  with  $j=1, \dots, 5$ ,  $k=1, 2, 3$ , and thus  $n_1 = n_2 = 5$  and  $n_3 = 3$ . The output linguistic fuzzy variable is one ( $S = 1$ ) and it is  $Y \triangleq Q$ -measure with a term-set  $T(Y)=\{Y_p\}$  for  $p=1, \dots, 5$  and that means that  $p_1 = 5$ .

$$X_{ij} \triangleq \begin{pmatrix} \text{Very low} & i = 1, 2 & j = 1 \\ \text{Low} & i = 1, 2 & j = 2 \\ \text{Neutral} & i = 1, 2 & j = 3 \\ \text{High} & i = 1, 2 & j = 4 \\ \text{Very high} & i = 1, 2 & j = 5 \\ \text{Small} & i = 3 & j = 1 \\ \text{Neutral} & i = 3 & j = 2 \\ \text{Big} & i = 3 & j = 3 \end{pmatrix}, \quad Y_j \triangleq \begin{pmatrix} \text{Bad} & j = 1 \\ \text{Not bad} & j = 2 \\ \text{Neutral} & j = 3 \\ \text{Good} & j = 4 \\ \text{Very good} & j = 5 \end{pmatrix}.$$

According to (1) the total number of the membership functions in the knowledge base is

$$N \cdot \sum_{i=1}^N n_i + S \cdot \sum_{s=1}^S p_s = 3 \times (5 + 5 + 3) + 1 \times 5 = 44.$$

The universes of discourse of the linguistic variables are  $U_{K_1} = U_{K_2} = U_{K_3} = U_Y = R$  and the types of membership functions of the terms, used in the system, are *Gaussian membership function*  $\mu_G(x) = e^{-\frac{1}{2}(\frac{x-\beta}{\alpha})^2}$ , *Bell membership function*  $\mu_B(x) = \frac{1}{1 + |\frac{x-\gamma}{\alpha}|^{2\beta}}$  and *Sigmoid membership function*  $\mu_S(x) = \frac{1}{1 + e^{-\alpha(x-\beta)}}$ .

The corresponding type of Membership Functions (MF) and the initial values of the parameters are shown on Table 1. The overall number of parameters of the membership functions of the fuzzy functions terms is 37.

Table 1. Type and parameters of membership functions of terms

Terms	$X_{11}$	$X_{12}$	$X_{13}$	$X_{14}$	$X_{15}$	$X_{21}$	$X_{22}$	$X_{23}$	$X_{24}$	$X_{25}$
MF	$\mu_S(x)$	$\mu_G(x)$	$\mu_G(x)$	$\mu_G(x)$	$\mu_S(x)$	$\mu_S(x)$	$\mu_G(x)$	$\mu_G(x)$	$\mu_G(x)$	$\mu_S(x)$
$\alpha$	-10	-0.4	0	0.4	10	-20	0.1	0.3	0.5	20
$\beta$	-0.5	0.35	0.42	0.35	0.5	0	0.085	0.1	0.085	0.6
Terms	$X_{31}$	$X_{32}$	$X_{33}$			$Y_1$	$Y_2$	$Y_3$	$Y_4$	$Y_5$
MF	$\mu_S(x)$	$\mu_B(x)$	$\mu_G(x)$			$\mu_G(x)$	$\mu_G(x)$	$\mu_G(x)$	$\mu_G(x)$	$\mu_G(x)$
$\alpha$	-10	0.25	10			0	0.25	0.5	0.75	0
$\beta$	0	2	0.5			0.1	0.1	0.1	0.1	0.1
$\gamma$		0.25								

For each input variable the degree of membership to the corresponding term is calculated and also the numerical values of the input data form a 3-dimensional vector  $\mathbf{x}^* = (x_1^*, x_2^*, x_3^*)$ . Each coordinate of  $\mathbf{x}^*$  is then fuzzified by calculating the values of the membership functions  $\mu_{ij}(x_i^*)$  for each  $i$  and  $j$ .

The  $t$ -norm, used in the aggregation procedure, is defined by a min operator while the  $s$ -norm is defined by a max operator. There are  $M = 24$  fuzzy rules

implemented in the system, for the  $t$ -norm is used for the logical AND operator. Under these considerations the  $m$ -th rule  $R_m$  in the rule base has the form

$$\text{IF } \{K_{m_1} \text{ is } X_{m_1 j_{m_1}}\} \text{ AND } \{K_{m_2} \text{ is } X_{m_2 j_{m_2}}\} \text{ AND } \{K_{m_k} \text{ is } X_{m_k j_{m_k}}\} \\ \text{THEN } \{Q_{m_1} \text{ is } Y_{m_1 j_{m_1}}\},$$

where  $m = 1, 2, \dots, M$  and  $M = 24$  is the number of the rules.

After the execution of the  $m$ -th rule the values of  $\theta_m$  and  $\theta_m^o$  are calculated sequentially according to (3) and (4) for  $m = 1, 2, \dots, 24$ . The rule weights are  $M = 24$  additional parameters in the fuzzy system.

Finally, FSSAM depends on  $37 + 24 = 61$  parameters.

After the execution of all the rules in the rule-base the membership degrees are derived for each term  $Y_{1p}$  of the output variables.

The aggregation is obtained by calculating the values of  $P_{sp}$  for each  $Y_{sp}$  from (5) for  $s = 1$  and  $p = 1, 2, 3, 4, 5$ .

In the last procedure, which is defuzzification, a crisp output value is obtained. In FSSAM the centre of gravity method is implemented by composite trapezoidal numerical method for approximating the integrals [4].

### 3.4. Optimization of parameters

The system's parameters are stored in the knowledge base of the fuzzy system and form the population of the GA – each of them being a chromosome, stored as a vector.

As the genetic algorithm is used for optimizing the parameters of the fuzzy system, the objective function is connected with the defuzzified value of the output variable  $Q$  of FSSAM. The objective function  $F$  for GFSSAM is defined as a sum of the squared differences of  $k$  consecutive values of  $Q$ :

$$(6) \quad F = \sum_{\substack{i,j=1 \\ j>i}}^k (Q_i - Q_j)^2.$$

The optimization is focused on finding the values of the parameters of FSSAM aiming at achieving stability of the derived estimation and so the optimization problem is in its essence searching a minimum value of  $F$ .

**Coding fuzzy variables.** All linguistic variables are defined on intervals and consist of a finite number of terms with corresponding membership functions. These variables are initially represented as binary strings.

**Coding a finite closed interval  $[a; b]$ .** The coding of the interval  $[a; b]$  is realized with two calculations:

(1) applying the coding function  $c_{m,[a;b]}$  defined as follows:

$$c_{m,[a;b]} : [a; b] \rightarrow \{0, 1, 2, \dots, 2^m - 1\} \\ c(x) = \left\lfloor (2^m - 1) \cdot \frac{x - a}{b - a} \right\rfloor,$$

where  $m$  is the length of the binary string and  $\lfloor \cdot \rfloor$  is the integer part of the number;

(2) the integer  $c(x)$  obtained in (1) is converted into a binary string.



**Decoding a finite closed interval  $[a; b]$ .** Decoding is also realized with two calculations:

- (1) the binary string is converted to a decimal number;
- (2) the corresponding decoding function is  $\tilde{c}_{m,[a;b]}$  defined as follows:

$$\tilde{c}_{m,[a;b]} : \{0, 1, 2, \dots, 2^m - 1\} \rightarrow [a; b],$$

$$\tilde{c}(x) = a + \frac{b - a}{2^m - 1} x .$$

Thus the condition  $c \circ \tilde{c} \equiv id_{\{0,1,2,\dots,2^m-1\}}$  is met.

**Coding membership functions defined on finite closed interval  $[a; b]$ .** One direct method for coding membership functions, defined on a finite closed interval of real numbers  $[a; b]$ , is by applying a linear interpolation on equidistant nodes as illustrated on Fig. 2. The nodes are encoded by applying the function  $c_{m,[a;b]}$  and for any membership function the functional values in the corresponding nodes and their encoding by  $c_{m,[0;1]}$  are used.

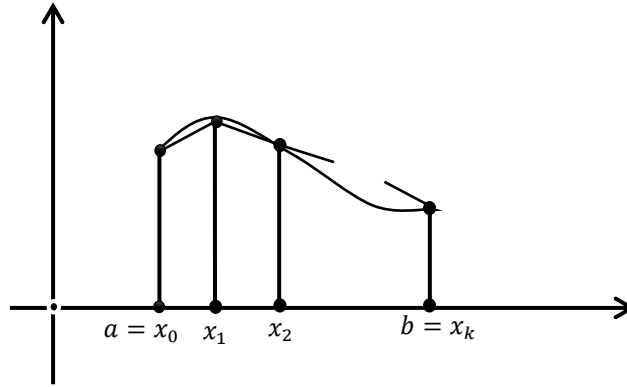


Fig. 2. Linear interpolation with equidistant nodes for coding a membership function

The problem that occurs after such interpolation is related with the length of each of the binary strings. For this reason, it is appropriate to look for another approach. If the membership function is triangular, its encoding may be simplified by simply coding the three real numbers that define it. If the membership function is trapezoidal, four parameters are encoded.

**Coding the linguistic variables of FSSAM.** In order to apply a genetic algorithm to FSSAM, it is necessary to encode differentiable functions – *Gaussian*, *Bell* and *sigmoidal*. Instead of encoding functional values as in interpolation, only the parameters of these functions can be coded which means encoding only two parameters  $(\alpha, \beta)$  for the Gaussian and Sigmoid functions and three parameters  $(\alpha, \beta, \gamma)$  for the Bell function. The coding functions  $c_{m,[a;b]}(\alpha)$  and  $c_{m,[\varepsilon;\delta]}(\beta)$  are used for coding the parameters of Gaussian function;  $c_{m,[a;b]}(\gamma)$ ,  $c_{m,[\varepsilon;\delta]}(\alpha)$  and  $c_{m,[\varepsilon';\delta']}(\beta)$  are used when coding the parameters of Bell function. Similarly,

$c_{m,[a;b]}(\beta)$  and  $c_{m,[\varepsilon;\delta]}(\alpha)$  are used to encode the sigmoidal function parameters. For each of the functions  $\varepsilon$ ,  $\delta$ ,  $\varepsilon'$  and  $\delta'$  are the lower and upper limits of the corresponding parameter.

The range, the coding functions and the binary representation of the terms  $X_{11}, X_{12}, X_{13}, X_{14}$  and  $X_{15}$  of the input variable  $T(X_1)$  of FSSAM are shown in Table 2. The terms of the other input variables  $T(X_2)$  and  $T(X_3)$  and the output variable  $T(Y)$  are encoded analogously.

Table 2. Coding the terms of the input variable  $T(X_1)$  of FSSAM

Linguistic variable	Term	Membership function	Coding function	Binary presentation
			$[a; b] = [0; 3]$	
		$[\varepsilon; \delta] = [-30; 30]$	$c_{8,[-30;30]} = c_1(x) = \lfloor 4.25x + 122.5 \rfloor$	
$T(X_1)$	$X_{11}$	$\text{sig}(x; -20; 0.08)$	$c_1(-20) = 37$ $c(0.08) = 6$	00100101 00000110
	$X_{12}$	$\text{gaussian}(x; 0.5; 0.4)$	$c(0.5) = 42$ $c(0.4) = 34$	00101010 00100010
	$X_{13}$	$\text{gaussian}(x; 1.5; 0.6)$	$c(1.5) = 127$ $c(0.6) = 51$	01111111 00110011
	$X_{14}$	$\text{gaussian}(x; 2.5; 0.4)$	$c(2.5) = 212$ $c(0.4) = 34$	11010100 00100010
	$X_{15}$	$\text{sig}(x; 20; 2.92)$	$c_1(20) = 207$ $c(2.92) = 248$	11001111 11111000

In genetic algorithms, chromosomes form a set of genes that are values of variables, encoded in advance. Each chromosome is actually an acceptable solution of the problem in interest. The different chromosomes make up the current generation. Through evolutionary operations (selection, recombination and mutation) the next generation is reached. In creating a fuzzy genetic system, besides encoding the variables, it is necessary to encode the fuzzy rules and to define the fitness function, then repeat the operators of the genetic algorithm.

Before the genetic algorithm is started, parameters that affect its operators are initially declared. These parameters are the number of generations, population size, number of variables, number of bits for coding a variable, crossover probability  $p_c$ , mutation probability  $p_m$ , degree of elitism, upper and lower limits. The upper and lower limits are intervals in which the variables of the fitness function can change.

Then a random population of candidate-solutions for the optimization problem is generated. Chromosomes need to be decoded from binary to real values. Once a population has been formed, the fitness of each of its individuals is calculated according to (6).

According to the obtained values of the fitness function, the most suitable individuals are selected and the operator selection is performed on them.

**Selection.** In nature, the selection of individuals is done through natural choice – the more an individual is adapted to the environment, the greater is his chance of surviving and creating offspring, thus transmitting his genetic information to the next generation. In evolutionary algorithms, the selection of the best individuals (parents)

is based on the fitness function, which gives an assessment of the individual. There are a variety of selection methods. In GFSSAM the roulette wheel selection algorithm is chosen for selection operator and thus the probability for selecting a chromosome equals the quotient of the value of the fitness function for this specific chromosome and the sum of the values of the fitness function of all the chromosomes in the population. In this method, a circle is divided into  $N$  sectors proportionally to the probability of selection of the individuals. Each time the roulette rotates, an individual is selected and accordingly individuals with higher fitness values have proportionately greater probabilities of being selected.

Once the parents are selected, reproduction for creating a new generation begins by performing recombination.

**Crossover.** The first operator in creating a new generation is crossover (recombination), in which the genes of the parents form an entirely new chromosome. Typically, the crossover operator needs two parent and aims at designing a new hromosome that is better than the two parents. Crossover is performed at a point chosen with crossover probability  $p_c$  where parents exchange their gene information and so form a new individual (child).

**Mutation.** The newly created by selection and crossover generation can be mutated. In biological species mutations alter DNA, these changes being caused mainly by errors in copying the parents genes. In genetic algorithms, mutation is a change of the value in a gene in the offspring, randomly selected with probability  $p_m$ . Additionally, a matrix, indicating on which gene the mutation is performed, is also created.

For the newly created generation, the fitness function values are re-calculated and the elitism check is performed because some fit individuals and their genetic material may be lost due to selection, crossover and/or mutation. In order to preserve their good genes, the genetic material must be kept in the algorithm. This process is called elitism.

## 4. Results

After the successful implementation of the GFSSAM hybrid system in MatLab, experimental tests have been carried out. The tests aim at studying the behavior of the parameters of the fuzzy system while changing the parameters of the genetic algorithm and finding the optimal solutions. The results presented in this paper are conducted on real data from BSE-Sofia.

A variety of tests have been performed for all 37 parameters of the therms of the input and output fuzzy variables. In the tests four parameters of GA have been changed and at the same time the changes of the respective change in the fuzzy system have been followed up. In addition, the convergence of GA has been recorded.

In this section, the behavior PAR1, which is the value of  $\alpha$  of the Gaussian function of term  $Y_1$  of the fuzzy variable  $Q$ -measure is presented in details.

At the end of the section the obtained results for PAR1, PAR2, PAR3, PAR4, PAR5 are the values of the parameter  $\alpha$  of the Gaussian function of the corresponding terms  $Y_1, Y_2, Y_3, Y_4$  and  $Y_5$  of the fuzzy variable  $Q$ -measure are summarized.

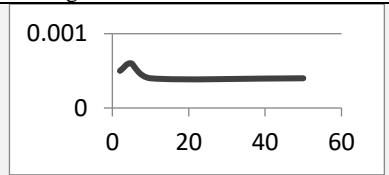
The rest of the parameters of the fuzzy variables have been studied analogically.

The initial values are PAR1 = 0; PAR2 = 0.25; PAR3 = 0.5; PAR4 = 0.75; PAR5 = 1 and the corresponding calculated defuzzified value of the output variable is  $Q = 0.7746$ . The value of the fitness function  $F$  is denoted with  $y$ .

**Results obtained after changing  $nGen$ .** The number of generations ( $nGen$ ), needed for finding optimal solutions, is one of the most important parameters of GA, because it is directly connected with the cost of the algorithm. Values of  $nGen$  have been increased from 2 up to 2000, while the other three parameters ( $nPop, p_c, p_m$ ) of GA have been kept constant. On Table 3 the change of PAR1 and  $y$  for  $nGen=2, 5, 10, 50$  are presented.

Table 3. Changes in PAR1 and  $y$  corresponding to changes in  $nGen$

$nGen$	$nPop$	$p_m$	$p_c$	PAR1	$y$
2	6	0.1	0.75	0.0005	0.0955
5	6	0.1	0.75	0.0006	-0.0192
10	6	0.1	0.75	0.0004	0.0292
50	6	0.1	0.75	0.0004	-0.0295



As can be seen from the table above, the increase in number of generations does not improve the accuracy of the obtained result. What is more, after the tenth generation the value of PAR1 practically does not change with respect to the desired precision (four decimal digits).

The conclusion here is that there is no need to continue the genetic algorithm after the 50th generation.

The convergence of the process for obtaining optimal values for PAR1 corresponding to the change in  $nGen = 2, 5, 10, 50$  is shown in Fig. 3.

When  $nGen = 5$  the value of  $y$  is closest to 0 and this is reached in the third generation which is actually the best result.

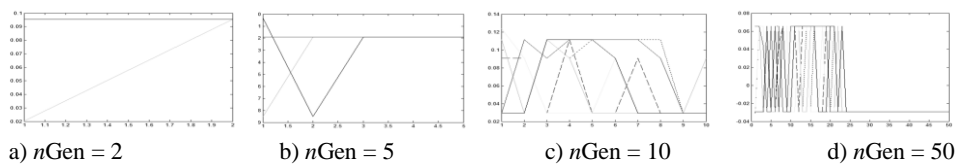
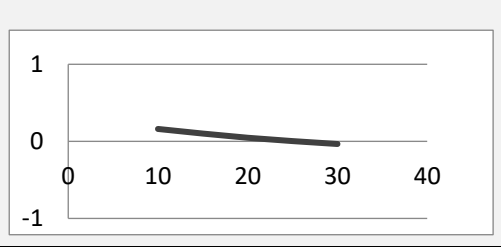


Fig. 3. Convergence in obtaining optimal values of PAR1 for  $nPop = 6; p_m = 0.1, p_c = 0.75$

**Results obtained after changing  $nPop$ .** The change in values of PAR1 and  $y$  corresponding to increase in the population size, while the other factors are constant) is shown on Table 4. It is important to note that a crossover error occurs for  $nGen = 10, nPop = 5, p_m = 0.1, p_c = 0.75$  the algorithm cannot be executed. Tests with other odd values of  $nPop$  (such as 7, 9, 19) have been also conducted and the crossover operator could not be executed as well.

Table 4. Changes in PAR1 and y corresponding to changes in nPop

nGen	nPop	$p_m$	$p_c$	PAR1	y
10	5	0.1	0.75	Impossible to execute	
10	10	0.1	0.75	0.161	0.003
10	20	0.1	0.75	0.0467	0.003
10	30	0.1	0.75	-0.0344	0.006



The convergence of the process for obtaining optimal values for PAR1 corresponding to the change in  $nPop = 10, 20, 30$  is shown in Fig. 4.

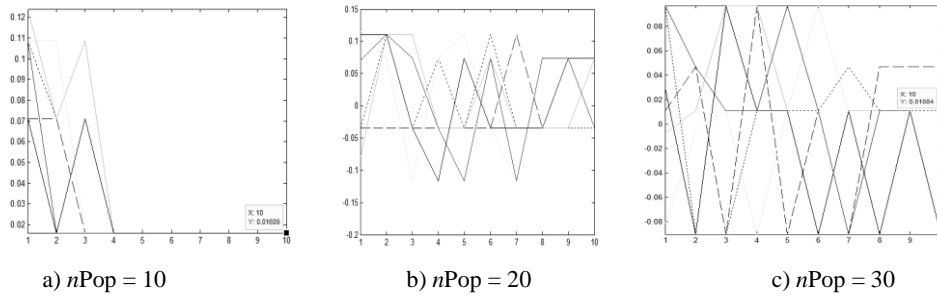


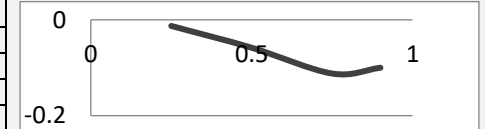
Fig. 4. Convergence in obtaining optimal values of PAR1 for  $nGen = 10$ ;  $p_c = 0.75$ ;  $p_m = 0.1$

The conclusion is that the genetic algorithm performs well enough even with relatively small population size.

**Results obtained after changing  $p_c$ .** Table 5 shows the change in values of PAR1 and y according to the crossover probability  $p_c$ . Smaller values of  $p_c$  lead to better results both in minimal value of y and in convergence.

Table 5. Changes in PAR1 and y corresponding to changes in  $p_c$

nGen	nPop	$P_m$	$P_c$	PAR1	y
10	6	0.1	0.25	-0.0127	0.0002
10	6	0.1	0.5	-0.0582	0.0004
10	6	0.1	0.75	-0.1121	0.0004
10	6	0.1	0.9	-0.1	0.0006



The convergence of the process for obtaining optimal values for PAR1 corresponding to the change in  $p_c = 0.25, 0.5, 0.75, 0.9$  is shown in Fig. 5.

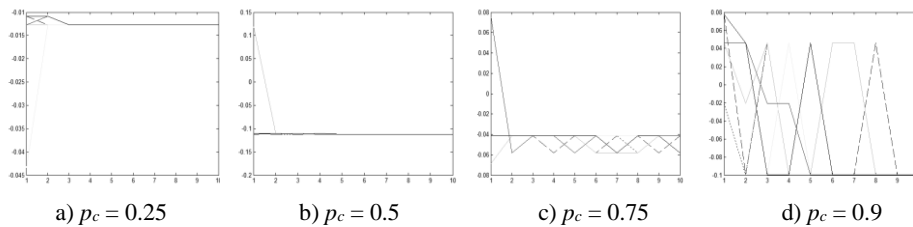


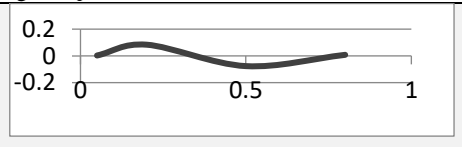
Fig. 5. Convergence in obtaining optimal values of PAR1 for  $nGen = 10$ ,  $nPop = 6$ ;  $p_m = 0.1$

The test results lead to the conclusion that smaller values of crossover probability lead to faster convergence, for instance the convergence when  $p_c = 0.25$  is reached just after the third generation.

**Results obtained after changing  $p_m$ .** The change in values of PAR1 and  $y$  due to change of the mutation probability  $p_m$  is shown in Table 6.

Table 6. Changes in PAR1 and  $y$  corresponding to changes in  $p_m$

$nGen$	$nPop$	$p_m$	$P_c$	PAR1	$y$
10	6	0.05	0.75	0.003	-0.0275
10	6	0.2	0.75	0.0837	0.0003
10	6	0.5	0.75	-0.0765	0.003
10	6	0.8	0.75	0.0079	0.0005



The convergence of the process for obtaining optimal values for PAR1 corresponding to the change in  $p_m = 0.05, 0.2, 0.5, 0.8$  is shown in Fig. 6.

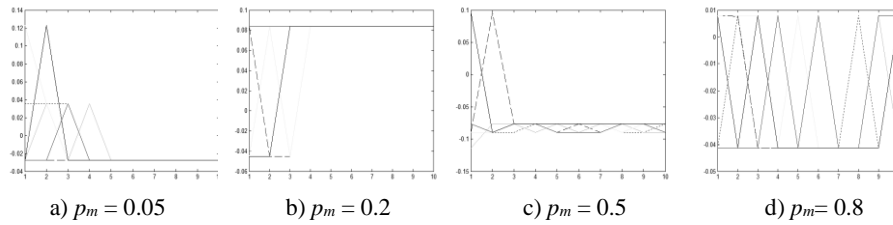


Fig. 6. Convergence in obtaining optimal values of PAR1 for  $nGen = 10, nPop = 6; p_c = 0.75$

The conclusion that smaller values of mutation probability lead to faster convergence is in line with the theoretical research in the area of evolutionary computation.

The convergence of the vectors of the fitness function  $F$  due to change in population size is shown in Fig. 7.

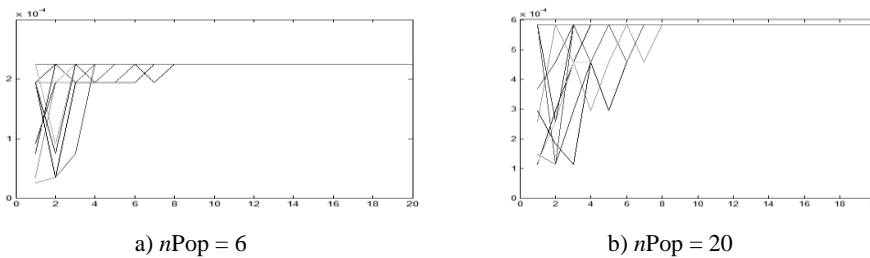


Fig. 7. Convergence in obtaining optimal values of PAR1 for  $nGen = 10, p_c = 0.75, p_m = 0.2$

After the conducted experimental tests the conclusion is drawn: the fastest convergence for PAR1 and  $y$  is reached at  $nGen = 10, nPop = 6, p_m = 0.1$  and  $p_c = 0.25$ , which means that the genetic algorithm finds satisfactory results at lower values of its parameters: number of generations, population size, mutation probability and crossover probability.

**Summarized results.** The optimal values of the parameters PAR1, PAR2, PAR3, PAR4, PAR5 and  $y$ , the corresponding minima of the fitness function  $F$  and the parameters of GA at which least time is reached, are shown in Table 7.

Table 7. Optimized values of PAR1, PAR2, PAR3, PAR4, PAR5 and  $y$

$nGen$	$nPop$	$p_m$	$p_c$	PAR1	PAR2	PAR3	PAR4	PAR5	$y$
10	20	0.2	0.75	0.0549	0.3435	0.4027	0.8634	1	0.0006
20	20	0.05	0.5	-0.0137	0.1944	0.3785	0.8727	1.0824	0.0006
12	8	0.02	0.2	-0.1225	0.1480	0.4074	0.7670	1.0653	0.0005
10	8	0.5	0.05	0.0764	0.1679	0.3866	0.8616	1	0.0006

When the results are analysed only with regard to the obtained of the fitness function  $F = y$ , the conclusion is as follows: an optimal value ( $y \leq 0.0006$ ) and the corresponding optimized values of the parameters of the fuzzy system are reached at several combinations of GA parameters and they are:

- $nGen = 10$ ,  $nPop = 20$ ,  $p_c = 0.75$  and  $p_m = 0.2$ ;
- $nGen = 20$ ,  $nPop = 20$ ,  $p_c = 0.5$  and  $p_m = 0.05$ ;
- $nGen = 12$ ,  $nPop = 8$ ,  $p_c = 0.2$  and  $p_m = 0.02$ ;
- $nGen = 10$ ,  $nPop = 8$ ,  $p_c = 0.05$  and  $p_m = 0.5$ .

Generally, in order to reach optimal values the number of generations does not need to exceed 20, the population volume does not need to exceed 20, but with smaller values of these two parameters it is necessary to decrease the crossover probability of crossing  $p_c$  and to increase the mutation probability  $p_m$ .

With regard to the last conclusion, it is advisable to use the following values (or close to them) as parameters of the genetic algorithm:

- $nGen = 50$ ,  $nPop = 8$ ,  $p_c = 0.75$  and  $p_m = 0.1$ .

## 5. Conclusion

GFSSAM, presented in this paper, is a genetic fuzzy rule-based system, implemented in MatLab. The major goal in creating the system is finding optimal values of the fuzzy system parameters. Genetic algorithms are successfully applied to various problems in different areas. Because their potential to bring flexibility, they are suitable for optimizing fuzzy systems for decision-making in diagnostics, monitoring and management.

The hybridization of the genetic algorithm and the fuzzy system is considered successful because the behavior of the system with the found optimal values of the parameters is stable and reliable on the basis of the results obtained. Another important result from this research, undoubtedly proven by the experimental tests, is that a large number of generations or population size is not a requirement for solving this optimization problem.

An interesting follow-up to this research is the creation of a genetic fuzzy system aiming at optimization of the architecture of the fuzzy system and in particular the number and type of fuzzy rules.

**Acknowledgements:** This research is partially supported by the project D14-2305 “Modern software engineering products for scientific research – training for lecturers and students in Faculty of Computer Science and Engineering and Faculty of Business Studies, Burgas Free University”, “University Research Fund”, BFU 2017/2018.

## References

1. Georgieva, P. V. FSSAM: A Fuzzy Rule-Based System for Financial Decision Making in Real Time. Handbook of Fuzzy Sets Comparison – Theory, Algorithms and Applications. Science Gate Publishing, 2016, pp. 121-148.
2. Herrera, F., M. Lozano, E. Herrera-Viedma, J. Verdegay. Fuzzy Tools to Improve Genetic Algorithms. – In: Proc. of European Congress on Intelligent Techniques and Soft Computing, Aachen, Germany, 1994, pp. 1532-1539.
3. Peneva, V., I. Popchev. Fuzzy Logic Operators in Decision-Making. – International Journal Cybernetics and Systems, Robert Trappl, Ed., Vol. **30**, 1999, No 6, pp. 725-745.
4. Georgieva, P. V., I. Popchev, S. Stoyanov. A Multi-Step Procedure for Asset Allocation in Case of Limited Resources. – Cybernetics and Information Technologies, Vol. **15**, 2015, No 3, pp. 41-51.
5. Peneva, V., I. Popchev. Multicriteria Decision Making Based on Fuzzy Relations. – Cybernetics and Information Technologies, Vol. **8**, 2008, No 4, pp. 3-12.
6. Georgieva, P. V. Applying FSSAM for Currency Rates Forecasting. – In: Transactions on Machine Learning and Artificial Intelligence, Manchester, SSE UK, Vol. **4**, 2016, No 3, pp. 30-40.
7. Georgieva, P. V. Fuzzy Rule-Based Systems for Decision-Making. – Engineering Sciences, BAS, Vol. **LIII**, 2016, No 1, pp. 5-16.
8. Mavrov, D., I. Radeva, K. Atanasov, L. Doukovska, I. Kalaykov. InterCriteria Software Design: Graphic Interpretation within the Intuitionistic Fuzzy Triangle. – In: International Symposium on Business Modeling and Software Design (BMSD’15), Milano, 2015, pp. 279-283.
9. Peneva, V., I. Popchev. Fuzzy Multi-Criteria Decision Making Algorithms. – Compt. Rend. Acad. bulg. Sci., Vol. **63**, 2010, No 7, pp. 979-991.
10. Zafari, A. Developing a Fuzzy Inference System by Using Genetic Algorithm and Expert Knowledge. Netherlands, Enschede, 2014.
11. Zadeh, L. A Theory of Approximate Reasoning. – Machine Intelligence, Vol. **9**, 1979, pp. 149-194.
12. Popchev, I., P. Georgieva. A Fuzzy Approach for Solving Multicriteria Investment Problems. – In: Innovative Techniques in Instruction Technology, e-Learning, e-Assessment, and Education. M. Iskander, Ed. Springer Science+Business Media B. V., 2008, pp. 427-431.
13. Sugeno, M. Industrial Applications of Fuzzy Control. Japan, Elsevier Science Pub, Co., 1985.
14. Melin, P., O. Castillo, E. Ramirez. Analysis and Design of Intelligent Systems Using Soft Computing Techniques. – Series: Advances in Soft Computing, Vol. **41**, 2007.
15. Jan, R. Fuzzy Inference Systems. NJ, Prentice-Hall, 1997.
16. Goldberg, D., K. Deb. A Comparative Analysis of Selection Schemes Used in Genetic Algorithms. – In: Foundations of Genetic Algorithms, Los Altos, Morgan Kaufmann, 1991, pp. 69-93.
17. Popchev, I., V. Peneva. An Algorithm for Comparison of Fuzzy Sets. – Fuzzy Sets and Systems, Elsevier Science Publishers, North-Holland, Amsterdam, Vol. **60**, 1993, No 1, pp. 59-65.
18. Radeva, I. Multicriteria Fuzzy Sets Application in Economic Clustering Problems. – Cybernetics and Information Technologies, Vol. **17**, 2017, No 3, pp. 29-46.
19. Radeva, I. Multi-Criteria Models for Cluster Design. – Cybernetics and Information Technologies, Vol. **13**, 2013, No 1, pp. 18-33.

*Received 11.01.2018; Second Version 26.02.2018; Accepted 15.03.2018*