POLISH MARITIME RESEARCH 3 (95) 2017 Vol. 24; pp. 3-14
10.1515/pomr-2017-0085

COMPARISON OF FUZZY SYSTEM WITH NEURAL AGGREGATION FSNA WITH CLASSICAL TSK FUZZY SYSTEM IN ANTI-COLLISION PROBLEM OF USV

Piotr Szymak
Polish Naval Academy, Poland

ABSTRACT

The paper presents the research whose the main goal was to compare a new Fuzzy System with Neural Aggregation of fuzzy rules FSNA with a classical Takagi-Sugeno-Kanga TSK fuzzy system in an anti-collision problem of Unmanned Surface Vehicle USV. Both systems the FSNA and the TSK were learned by means of Cooperative Co-evolutionary Genetic Algorithm with Indirect Neural Encoding CCGA-INE.
The paper includes an introduction to the subject, a description of the new FSNA and the tuning method CCGA-INE, and at the end, numerical research results with a summary. The research includes comparison of the FSNA with the classical TSK system in the anti-collision problem of the USV.

Keywords: neuro-fuzzy system,neural aggregation of fuzzy rules,cooperative co-evolution,anti-collision of USV

INTRODUCTION

Unmanned Surface Vehicles (USVs) are vessels which can perform many different missions both civilian and military. Civilian usage of USVs is mainly connected with different inspections of underwater environment, especially for oceanography and marine biology purposes. Military applications of USVs are focused on mine countermeasure, anti-submarine warfare and Intelligence, Surveillance and Reconnaissance (ISR) tasks.

USVs can be operated remotely and/or autonomously. USVs moving in a marine environment are exposed to collisions with stationary obstacles as well as moving obstacles, mainly other vessels occurring in USV's operation area. Therefore, a significant problem appearing in the USV motion is counteracting possible collisions [5,7,20]. Software responsible for the anti-collision usually cooperates with navigation devices such as: a radar system, an AIS receiver, GPS, a speed log, a gyrocompass, etc., and actuators such as: a propeller, a rudder, and additionally cooperates with software for route planning [14] and an electronic navigation map [9].

In the paper, the anti-collision problem is considered as a problem of selecting proper trajectory (i.e. desired waypoints achieved in desired time) for the USV operating in an area with other vessels which can be located on a collision course with the USV's course. Taking into account control task for the USV, proper changes of a course and a velocity have to be generated for avoiding the collision during motion of USV from starting to target position.

To resolve the USV anti-collision problem, an innovative Fuzzy System with Neural Aggregation of fuzzy rules (FSNA) was proposed. The FSNA was compared with the classical TSK fuzzy system. To tune the structure and parameters of both systems, Cooperative Co-evolutionary Genetic Algorithm

with Indirect Neural Encoding (CCGA-INE) was applied. To simulate the USV and other vessels motion, Control-Oriented Model of motion of Unmanned Marine Vehicle (COMUV) was used. A detailed description of the COMUV was included in [13]. The model parameters were applied for USV Edredon (Fig. 1) [6,16].

Fig. 1. Unmanned Surface Vehicle Edredon [6]

The Edredon was built by a consortium, whose leader was the Polish Naval Academy [6]. The vehicle can be controlled remotely from a Mobile Command Centre (MCC) (Fig. 2), or can be controlled locally from on board of the vehicle. As can be seen in Fig. 2, the MCC simulates an operator console located aboard USV with a classical steering wheel and a set of shifters and switches. To visualize the space around the USV, a set of three monitors that receive signals from daylight and thermal cameras, installed aboard USV is used.

Fig 2. Mobile Control Centre of USV Edredon [6]

During previous research devoted to the anti-collision system of the USV Edredon [15], the classical TSK system tuned by the new CCGA-INE method was used. The anti-collision problem was defined by 30 scenarios including trajectories of ten other vessels, which can be on collision course with the USV. The achieved TSK system successfully (without collision) controlled the USV in all the 30 collision scenarios [15]. Then, the 30 more complicated scenarios were created for testing the TSK anti-collision system. Unfortunately, the tests with 30 additional more complicated scenarios did not end successfully. Therefore, in this paper, an improved FSNA system was used to solve the anti-collision problem defined by the 60 scenarios (initial simpler 30 scenarios and additional more complicated 30 scenarios). Moreover, this paper included comparison of working TSK and FSNA anti-collision systems. Both systems were verified by means of 30 validating scenarios. It is worth underlying that the anti-collision problem defined by 60 scenarios is more difficult than the same problem defined by 30 scenarios. The first 30 scenarios are simpler and the next 30 scenarios are more complicated. The complexity is connected with trajectories of the other ships. The trajectory selection influences a greater number of possible collision situations.

The proposed FSNA is based on the classical TSK fuzzy system with two improvements. The first one is based on using an artificial neural network instead of classical operator for calculation of crisp value in the fuzzy system output (called in this paper fuzzy rules aggregation). Based on the literature [18], the FSNA can be classified as a concurrent neuro-fuzzy system. The second improvement depends on integration of the fuzzy rules and fuzzy sets. Both improvements allow to introduce more nonlinearity in the fuzzy system and consequently to achieve desired solution.

The CCGA-INE is based on Cooperative Coevolution Genetic Algorithm CCGA proposed by Potter and De Jong [11]. It was improved by adding indirect encoding of the fuzzy system by means of an artificial neural network. The CCGA depends on an evolution of cooperating subcomponents of an overall solution. The subcomponents evolve in different populations of species, which have to cooperate to achieve a desired solution.

The paper is as follows: Section 2 includes details of the Fuzzy System with Neural Aggregation (FSNA). Section 3 explains details of tuning method of FSNA called Cooperative Co-evolutionary Genetic Algorithm with Indirect Neural Encoding (CCGA-INE). Section 4 includes description of the anti-collision problem used as a testbed and section 5 presents the selected numerical research. The last 6th section includes a summary of the research. Detailed description of the classical TSK system for anti-collision problem is presented in [15,17], and control-oriented model of the motion used for the USV is the same, which was applied for an underwater vehicle (UV) [13]. The motion of UV is considered in 6 degrees of freedom, while the motion of USV usually in 3 degrees of freedom. Therefore, the UV model is useful to simulate motion of USV.

DESCRIPTION OF FSNA

ASSUMPTIONS

The new FSNA is based on the classical TSK fuzzy system proposed in [17]. The TSK system is well known and often used especially in control applications. Comparing to another a classical Mamdani type fuzzy system [8], the TSK system is computationally simpler but similarly efficient [4].

Comparing to the TSK system, following assumptions were made for the FSNA:

1) using a logical or an algebraic product for a rules' prerequisites aggregation (a conjunction of prerequisites),
2) input variables represented by gaussian membership functions (two parameters for each fuzzy set) and output variables represented by singletons (one parameter for each rule's consequent).

Modifications that led to the creation of the FSNA are as follows:

1) integration of the fuzzy sets and rules (the system is in the form of a matrix of integrated fuzzy sets and rules),
2) using an artificial neural network for the aggregation of the fuzzy rules instead of e.g. weighted sum [1].

The modifications are described in more details in the following subsections.

MATRIX OF INTEGRATED FUZZY SETS AND RULES

In the classical TSK system, each variable is defined by the specified number of fuzzy sets (represented by membership functions). The fuzzy sets are usually set by an expert. If the fuzzy sets are tuned automatically, usually, the expert specifies the number of fuzzy sets for each variable. In the TSK system, each rule's prerequisite can operate on one fuzzy set selected from all the fuzzy sets defined for the variable. In this case, a prerequisite is defined by a linguistic expression, e.g.:

$$X_1 \text{ is HIGH}$$

(here: X_1 is an input, HIGH determines one of the fuzzy sets of input X_1).

In the FSNA, fuzzy sets are integrated with the fuzzy rules. It means that instead of using linguistic expressions each prerequisite is defined by parameters of the fuzzy set (in the case of gaussian membership function, two parameters define this function: an expected value and a variance). In the FSNA, the same prerequisite (relating to the same input) in different rules can operate on different fuzzy sets. In an extreme FSNA case, each variable is defined by the number of fuzzy sets equal to the number of fuzzy rules. This approach is very useful in the situation, when the rules or all the fuzzy system parameters are tuned in an automatic way, e.g. by means of an evolution. In this case, division of an input-output space is only limited by the number of fuzzy rules, and the tuning method decides on the number (and parameters) of fuzzy sets needed to represent a specified variable.

In the research presented in the following section, the following representation of the FSNA in the form of a matrix of integrated fuzzy sets and rules $\mathbf{VB_I}$ was applied:

$$\mathbf{VB_I} = \begin{bmatrix} r_{11}^1 & r_{12}^1 & \dots & r_{1n_m}^1 & r_{21}^1 & r_{22}^1 & \dots & r_{n_k n_m}^1 \\ r_{11}^2 & r_{12}^2 & \dots & r_{1n_m}^2 & r_{21}^2 & r_{22}^2 & \dots & r_{n_k n_m}^2 \\ \dots & & & & & & & \\ r_{11}^{n_n} & r_{12}^{n_n} & \dots & r_{1n_m}^{n_n} & r_{11}^{n_n} & r_{12}^{n_n} & \dots & r_{n_k n_m}^{n_n} \end{bmatrix} \quad (1)$$

where r_{km}^n is m-th parameter of fuzzy set or singleton of k-th input or the output variable in n-th fuzzy rule, n_k is the number of input and output variables, n_m is the maximal number of parameters describing a fuzzy set or a singleton, and n_n is the number of fuzzy rules.

In the matrix $\mathbf{VB_I}$, some elements can be zero. In this case, appropriate prerequisites or conclusions will be removed, e.g. if the first and second elements in the first row are equal to zero, it means that prerequisite relating to the first input is removed in the first rule.

NEURAL AGGREGATION OF RULES

The next step of modification of the TSK fuzzy system is to apply an artificial neural network for the aggregation of implications of fuzzy rules. In the classical TSK system, the conclusion of i-th implication of a fuzzy rule is in the form of a functional dependence of the rule's predecessors. In this case, the aggregation of the implications is typically calculated using a weighted sum of individual rules [1].

Often (in engineering practice), due to the need of reduction the number of parameters necessary to tune and, consequently, to simplify the system, the functional dependence of the rule's predecessors is simplified into singletons. This leads to a reduction of a non-linearity of the system, which in turn may lead to the inability to match a problem. The possibility of using an artificial neural network to aggregate rule outputs, results mainly from the fact that they are successfully used to approximate non-linear functions [10]. Thus, it seems that the application of a neural network, in this case, is more flexible in obtaining a satisfactory solution fitted to the nonlinear control object.

It was assumed that an artificial neural network in the FSNA performs the duty of rule aggregation, i.e. inputs of the network are weights of rules w_i, and weights are determined using a logical or an algebraic product. Weights are calculated based on the membership function of the individual fragments of each rule's predecessor. In this case, the output of the whole TSK system is a crisp value of the neural network output y. The network architecture is always related to a number of rules (a number of neural network inputs) and generally a solved problem (internal network topology, weights and types of an activation function).

Fig. 3 shows the exemplary FSNA structure formed by a connection of a neural network with a TSK fuzzy system. In Fig. 3, the *i*-th neuron of the network is represented by N_i.

In the research presented in the following part of the paper, a feed-forward artificial neural network was applied for the aggregation of fuzzy rules [10]. The architecture of the network was determinated by the evolutionary method CCGA-INE.

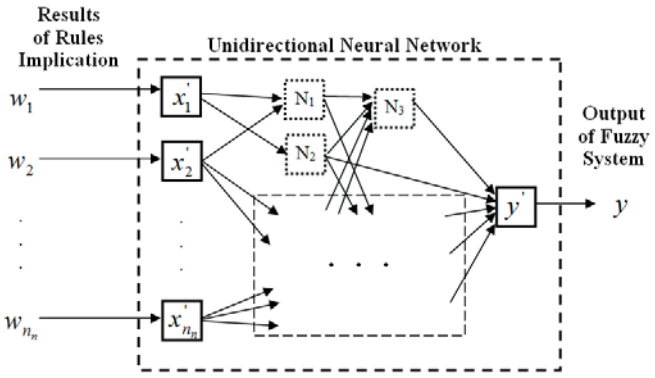

Fig. 3. The connection of neural network with TSK fuzzy system for the aggregation of fuzzy rules (Ni defines the i-th neuron)

The method of encoding an artificial neural network and its tuning by means of evolution is described in the next section.

DESCRIPTION OF CCGA-INE

CO-EVOLUTION

Co-evolution is a specific type of closely related species evolution. In the basic evolutionary algorithm, the process of evolution is seen as an attempt to adapt a population of individuals to a specific environment. Meanwhile, in the co-evolutionary approach, the process of co-evolution is seen as an attempt to adapt the population (or a subgroup of individuals from the population) to the specific environment that is affected by a population of another species (or another subgroup of individuals from the population). Usually, in the co-evolution, a complex solution is divided into sub-component solutions to evolve independently, i.e. there are many populations of individuals (multiple species), wherein each population encodes one sub-component solution.

A good example of co-evolution comes from the natural world in the form of relation between a predator and prey. The predator hunting the prey eliminates the weaker individuals from the population of prey. It causes those individuals which survive to have better features that can be transferred to their offspring. Similarly, the predators which achieve "worst results" in catching preys, have also less chance to transfer features to their offspring.

OVERVIEW OF CCGA

In general, a genetic algorithm (GA) is a heuristic search that mimics the process of natural selection. The GA is based on an iterative evolutionary procedure involving selection of genotypes for reproduction based on their fitness, and then introducing genetically changed (by means of mutation, crossover and other genetic operators) offspring into the next population. The procedure is finished after achieving satisfactory genotypes (a set of features of an individual) which correspond to phenotypes with high fitness function (the individual from a population) [3].

The CCGA is a specific Cooperative Coevolution Genetic Algorithm proposed by Potter and De Jong [11]. Generally, the CCGA solution is divided into sub-components that evolve in separate populations. There is no possibility of exchanging genetic information between populations of separate species, but individuals of different populations have to work together to achieve a satisfactory overall solution. Division into the sub-components is carried out by the following method. Initially, the solution is encoded in a single chromosome, which evolves in a single population. If the evolution of this population, after a specified number of iterations, does not lead to a satisfactory solution, then the next population is created, and next two populations evolve, etc. Sometimes, the CCGA algorithm may find that a particular species (population) does not make a significant contribution to the overall solution. In this case, the population is removed from the evolutionary algorithm. In the CCGA, to evaluate the overall solution a single individual of the first population must be connected with individual from each of the other populations [11].

When evaluating an individual from the given population, it is always combined with the fittest individual from each of the other populations, based on the evaluation (a fitness function) obtained in the previous iteration of an evolutionary algorithm (EA). During the first iteration of the EA, an individual is combined with the randomly selected individual from each of the other populations.

GENERAL IDEA OF CCGA-INE

Because the fuzzy system is described by a large number of parameters, the chromosomes coding these parameters should be very long. Evolution of long chromosomes is connected with complicated calculations, and in consequence, problems with achieving a final solution within assumed finite time. Due to potentially long chromosomes for the system defined by the large number of parameters, the indirect encoding of the fuzzy system is proposed. In the indirect encoding method, information from the chromosomes is used to generate other systems (neural network, nonlinear function, etc.), which in turn generates parameters of the FSNA. Such way of encoding is applied to create large fuzzy systems using relatively short chromosomes.

Generally, in the CCGA-INE a single chromosome encodes a neural network called coding network, defined by a Coding

Neural Network Definition Matrix **cNDM** [12], while the coding network or networks encode the FSNA (i.e. coding networks fills elements of matrices representing this system). In the case of a neural network for aggregation of fuzzy rules, coding networks generate elements of Aggregation Neural Network Definition Matrix **aNDM**, defining structure and parameters of the aggregation network. It should be noted that in the CCGA many populations can evolve, i.e. many chromosomes can generate many coding networks (Fig. 4). For many coding networks, each element of the matrices representing the FSNA is generated by one of the coding networks according to the algorithm described in the following subsection.

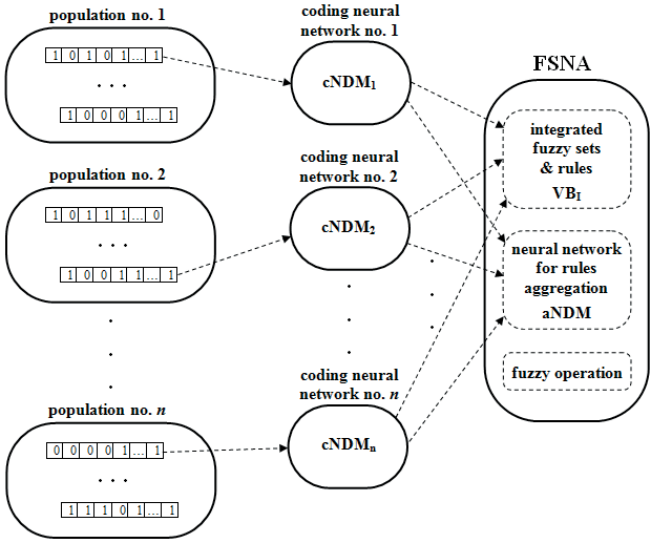

Figure 4. Generation of the FSNA using CCGA-INE

In conclusion, it should be noted that the task of CCGA-INE is to find the best structure and parameters of coding neural networks (one or several depending on the progress of co-evolution) which, in turn, encode integrated matrix of fuzzy sets and rules $\mathbf{VB}_I$ and a matrix defining an artificial neural network for aggregation of fuzzy rules **aNDM**.

GENERATION OF CODING AND AGGREGATING NEURAL NETWORKS

Fig. 5 shows a method for generating a coding neural network (defined by **cNDM**) [12] using the information stored in the chromosome, consisting of four components. Each component is composed of 7 bits, i.e. the whole chromosome is built from 28 bits. During the research, co-evolution produced chromosomes consisting of four to more than thirty components, i.e. chromosomes consisting of more than two hundred bits. The number of components is depended on the complexity of the problem and the co-evolution. Division into components is a decomposition of the problem. Always the first component of each chromosome is considered as a string of bits, while the next components represent integer values (scaled to real values), which are subsequent elements of the matrix. In the illustrated example (Fig. 5), the first component of the chromosome determines the topology of the neural network by indicating the elements of matrix **cNDM**, which should be reset (white boxes), and other which should adopt the values determined by the successive components of the chromosome c_1, c_2 and c_3 (black boxes).

Consecutive bits included in component „topology" determine if the following elements of the matrix (beginning from the first column and row, and ending on the last row and column) are zero or non-zero (Fig. 5). Bit string „topology" is too short to determine all the elements of the matrix, therefore, the string is repeated, i.e. after the last bit is the first bit of the same string, then second bit, etc., until all the elements are calculated. The bit which has a zero value determines zero value of the relating element in the matrix. This element is illustrated by white boxes in the table in Fig. 5. Bit which has value "1" determines non-zero value of the relating element in the matrix. The non-zero element is marked by grey boxes in the table in Fig. 5. The precise values of non-zero elements are determined by other components of the chromosome „coefficient no. 1", „coefficient no. 2" and „coefficient no. 3". Assignment of values c_1, c_2 and c_3 for successive elements of matrix **cNDM** is carried out according to the same principle as it is used for the bits of the component „topology".

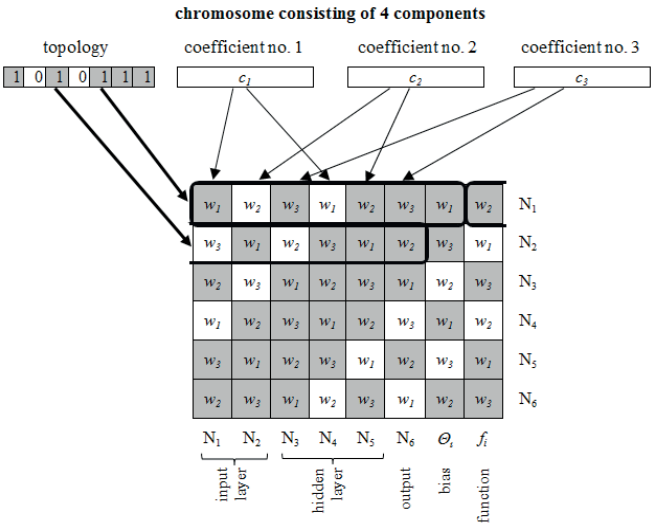

Figure 5. Generation of matrix cNDM by the chromosome consisting of four components

Matrix **cNDM** shown in Fig. 5 has n rows and $n + 2$ columns, where n is the number of neurons in the network layers, sequentially: input, hidden and output. Elements of matrix **cNDM** from the first element to the element of n-th row and n-th column determine the weights of connections between neurons. Column $n + 2$ determines type of the activation function, and the column $n + 1$ is a bias, i.e. a constant added to the total weight of input neurons.

Fig. 6 shows the architecture of an artificial neural network generated by means of information included in the chromosome and the relating matrix **cNDM**.

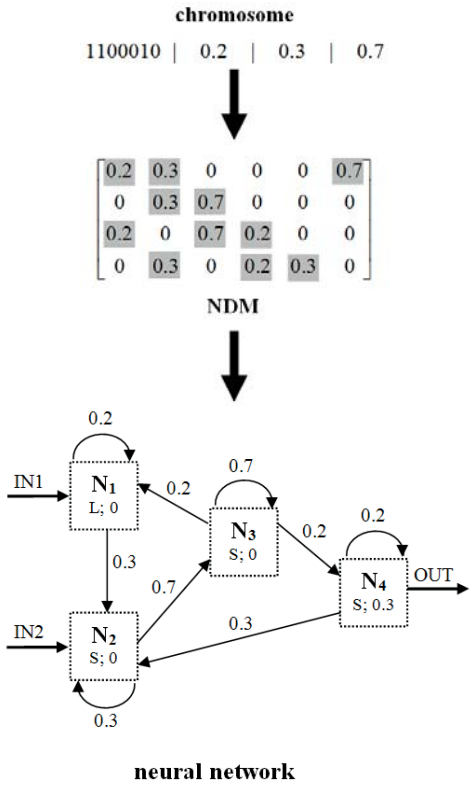

Fig. 6. Generation of the neural network based on NDM, formed on the basis of the chromosome

In Fig. 6, the chromosome components are presented in different forms: the first component in the form of a binary sequence, the other component in the form of real numbers (chromosomes include integers, which become real numbers after scaling). Individual neurons were visualized by succeeding numbers N_1, N_2, …, N_n and the type of the activation function: S - sigmoid, L - linear and additionally numerical value of the bias. The resulting neural network, presented in the Fig. 6 contains 4 neurons. The distribution of these neurons to the input and output layers, and possibly hidden, is determined by the designer of the system. In this case, it is assumed that two neurons are in the input layer, one is a hidden neuron and one is located in the output layer. As mentioned previously, the matrix **NDM** can define the coding neural network **cNDM** and also the neural network for the rules aggregation **aNDM** in the way as it was described for **cNDM**.

In the research, it was assumed that the coding neural network is composed of nine neurons: three in the input layer, three in the output layer, and three are the hidden neurons. Therefore, matrix **cNDM** is composed of 9 rows and 11 columns [12].

GENERATION OF FSNA MATRICES

Fig. 7 shows how to fill the matrices elements representing the FSNA. The values of elements are produced by the coding networks. The coding networks have three inputs and three outputs. Network inputs determine parameters of the element, whose value is produced by the network on its output. The first and second inputs determine, respectively, the row and the column of the matrix, and the third input determine the ordinal number of the matrix.

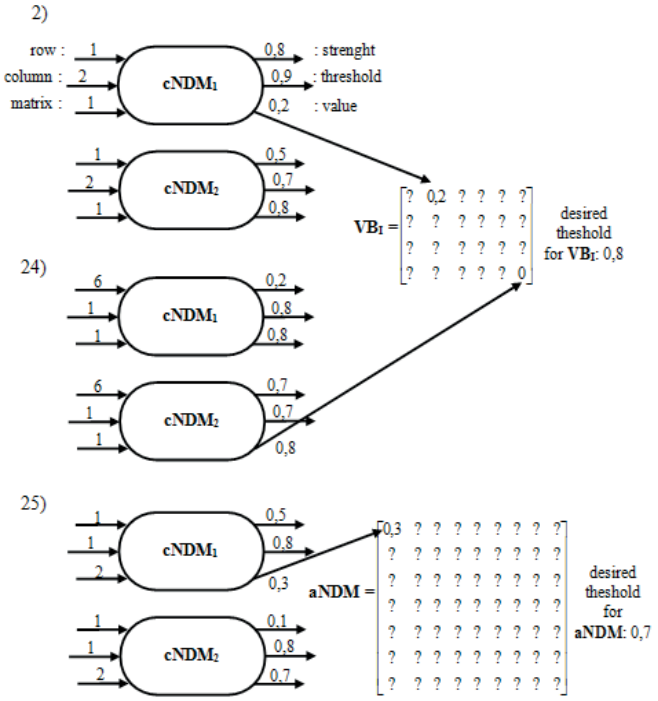

Fig. 7. Generation of the FSNA matrices by the coding neural networks in selected steps 2), 24), 25)

The coding network produces following outputs values:

1) strength: in a situation, where there is more than one network, this parameter determines which coding network should be used to „fill" the matrix in the current step (the coding network „wins", which has the highest strength value),
2) threshold: this parameter determines whether the value should be written to a specific element of the matrix, or the item should be zero (the element is reset if the threshold value is less than the desired threshold for the matrix),
3) value: assigned to the specific matrix element, defined by the coding network inputs, if the network has the highest strength value, and the threshold output is greater than the desired threshold for this matrix.

Fig. 7 shows selected steps of „filling" the FSNA matrices by one of two coding networks, formed on the basis of information included in the chromosomes, evolving in two populations. Step 2 illustrates a situation in which the first coding neural network (defined by $\mathbf{cNDM}_1$) has a higher strength value and its threshold output is greater than the desired threshold for $\mathbf{VB}_1$. Therefore, the first coding network writes a value to a specified element of the matrix. Step 24 illustrates the case in which the second coding network is „stronger" (has higher strength value), but the network's threshold value is less than the desired threshold for this matrix. In this case, the element specified by the coding network inputs obtains zero value.

Step 25 illustrates a situation similar to that which occurred in step 2, with the difference that in this case, the second element of the second matrix **aNDM** is filled. Generation of all the elements of the matrices requires iterations equal to the sum of elements in these matrices.

It can be seen that when using the CCGA-INE, the matrices consisting of even hundreds of parameters can be filled by means of information included in several chromosomes (depending on the number of populations).

In the next section, the anti-collision problem used to compare the TSK and the FSNA systems is described in details.

ANTI-COLLISION PROBLEM

ASSUMPTIONS

It was assumed that the information from an onboard navigation system, in particular about the detection of obstacles, was discretized in such a way that the USV anti-collision system received the distances from the obstacles located in the seven sectors around the USV. The four sectors with 45° view angle were located in the fore part and the three sectors with 60° view angle located in the aft part of the USV (Figure 8). Therefore, the anti-collision system obtained information about other vessels in the form of distances to the closest vessels in designated sectors from x_1 to x_7 (Figure 8). The sectors of detecting obstacles were limited in bearing and range of their view. The value of the view range was chosen experimentally. The anti-collision system provided desired course to the target ψ_s. Based on information about the obstacles and the target, the anti-collision system calculated the change of course $\Delta\psi$ and the change of advance velocity ΔV_a. The advance velocity is a velocity measured in a longitudinal axis of symmetry of the USV.

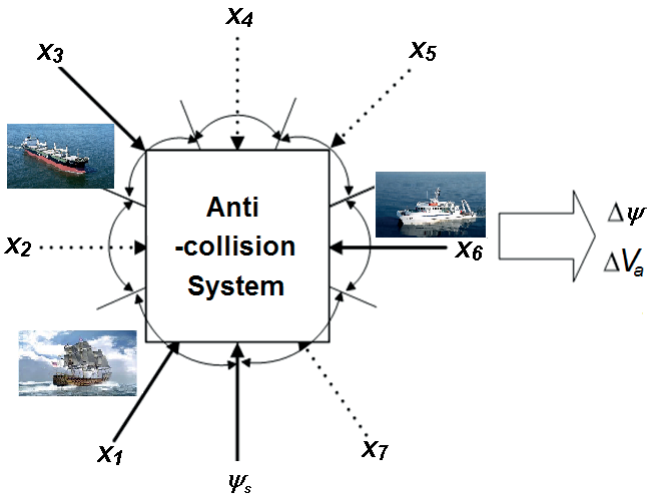

Fig. 8. Inputs and outputs of anti-collision system: x_1, x_2 ... x_7 – distances from obstacles in sectors, ψ_s – desired course to the target, $\Delta\psi$ – the change of course and DV_a – the change of velocity USV

SCENARIOS

In order to tune and then validate the TSK and the FSNA anti-collision systems, scenarios with increased difficulty level were designed. Each scenario contained information about starting and target positions of the USV and motion vectors of the ten other vessels, operating in the same area. The distance that had to be overcome by the USV was approximately 2 nautical miles. This distance is sufficient to deploy the collision obstacles. Due to USV dynamics [16], it was assumed that the anti-collision system took the decisions every 20 seconds.

Each of ten potential collision vessels moved with one fixed course and velocity. In the first part of the scenarios, the vessels moved along safe trajectories (not on collision course with the USV). At the beginning, the USV had to „learn" to reach the target. In the next part of the scenarios, the number of units moving on collision courses were gradually increased. Moreover, the number of vessels that moved near the starting position and the target were also increased. Such vessels are not on collision course at the beginning of the simulation, but may be on collision course in subsequent moments after various changes of the USV course.

In the following scenarios, the starting and the target positions were changed in such a way that potential trajectories ran in different directions: north, north east, south, etc.

In order to avoid too complex trajectories of the USV, including incorrect maneuvers, e.g. multiple passage on circular trajectory, an excessive descent from desired trajectory, etc., timeout for the each scenario was implemented. The timeout was equal to 150% of the time needed for movement along a straight route from the starting point to the target with an average velocity of 10 knots.

To tune and then validate the anti-collision systems, respectively, 60 learning scenarios (including 30 simpler and 30 additional scenarios), and 30 validating scenarios were designed. In the validating scenarios, an additional difficulty was implemented, i.e. the changes of course of the vessels were added.

EVALUATION FUNCTION

The scenarios are used for training and then validating sequentially, i.e. the first scenario was followed by a second, then the third, etc. Scenario finished at the moment of collision or after achieving the maximum time for the scenario (the maximal number of decisions).

The behavior of the n-th anti-collision system in 60 learning scenarios was evaluated using the fitness function $F(FL_n)$. The function $F(FL_n)$ was calculated as the sum of the rewards gained in all the scenarios. The following form of the reward function f in m-th scenario was applied [15]:

$$F_m(FL_n)=\begin{cases} 0 \quad , \quad case\ a \\ 0.5d_t-0.5\dfrac{k}{I_{\max}} \quad , \quad case\ b \\ 0.5+0.5d_t-0.5\dfrac{k}{I_{\max}} \quad , \quad case\ c \\ R_t+\left(I_{\max}-I\right)\Big/ I_{\max}-50\dfrac{k}{I_{\max}} \quad , \quad case\ d \end{cases} \tag{2}$$

where k is a scenario number, FL_n is an estimated n-th FSNA, d_t is a distance to the target at the end of scenario, and k is a penalty for change of course greater than 90°, if the distance to the closest obstacle is larger than 0.15 nautical mile. Additionally, I_{max} is a maximal number of decision, which the USV can take moving to the target, I is a number of decisions taken by the anti-collision system, and R_t is a reward for the USV reaching target (R_t = 100).

The function (2) is calculated for the following cases:

a) collision occurred,
b) USV did not reach the target, but did not collide with other vessels; at the end of simulation, the USV was located at the distance larger than 1 nautical mile from the target,
c) USV did not reach the target, but did not collide with other vessels; at the end of simulation, the USV was located at the distance less than or equal to 1 nautical mile from the target,
d) USV reached the target.

The occurrence of collisions automatically stops the process of evaluating the TSK or the FSNA systems in the m-th scenario with the value of the function $F_m(FL_n)$ equal to zero. In the case where the USV did not collide with other vessels and did not reach the target, the evaluation is dependent on the distance to the target at the end of simulation, and the number of forbidden maneuvers (change of course greater than 90°, if the distance to the closest obstacle is larger than 0.15 nautical miles). Tuning process of the anti-collision system is determinated by the evaluation function. The greater the function result is, the more effective anti-collision system is tuned.

The whole process of tuning the TSK or the FSNA systems is terminated, when the evaluation function reaches a value greater than or equal to 6000, i.e. when the USV reached the target without collision in 60 learning scenarios. In the research presented in the next section, 60,000 iterations were applied as the maximum number of iterations for the tuning process.

NUMERICAL RESEARCH

MODEL OF THE VESSEL MOTION

To simulate horizontal plane motion of the USV and other vessels control-oriented model of marine object was applied [13]. The model was described in the following matrix form:

$$\mathbf{S}_V = [\mathbf{P}_{ij}]_{6\ x\ 3} \tag{3}$$

where $\mathbf{P}_{i1}$ includes the changes of course $\Delta\psi_{kl}$, $\mathbf{P}_{i2}$ contains the changes of coordinate x: Δx_{kl}, and $\mathbf{P}_{i3}$ includes the changes of coordinate y: Δy_{kl}.

Elements of the matrices were registered in response to desired course $\psi_k^z = k \cdot \Delta\psi$ (where k = 1..36 and $\Delta\psi = 5°$), in l-th time step $t_l = l \cdot \Delta t$ (where l = 1..60 and Δt = 0,5 s), for advance velocities $V_i = i \cdot \Delta V$ (where i = 1..5 and ΔV = 5 knots).

For USV and the other vessels motion simulation, 5 discrete advance velocities were applied (5, 10, …, 25 knots), and the state vector of the vehicle and vessels were reduced to 3 parameters: course, coordinate x and coordinate y. The more details about the model was included in [13].

Parameters of the model were registered based on the classical nonlinear model [2]. To control the course of the USV and the vessels, slide mode controllers were used, which were described in details in [16].

STRUCTURE OF THE TSK AND THE FSNA SYSTEMS

In the structure of the fuzzy system, the following three components should be determined: fuzzy sets, fuzzy rules and fuzzy operation [1,19]. Based on the earlier research [15], fuzzy sets for inputs and outputs, illustrated in Figure 9, were applied. The same universe of discourse was used for all the inputs ($x_1, x_2 \ldots x_7$ – distances from obstacles in sectors, ψ_s – desired course to the target) and the same for all the outputs ($\Delta\psi$ – the change of course and ΔV_a – the change of velocity USV). The whole structure of the anti-collision system of USV is illustrated in Figure 8.

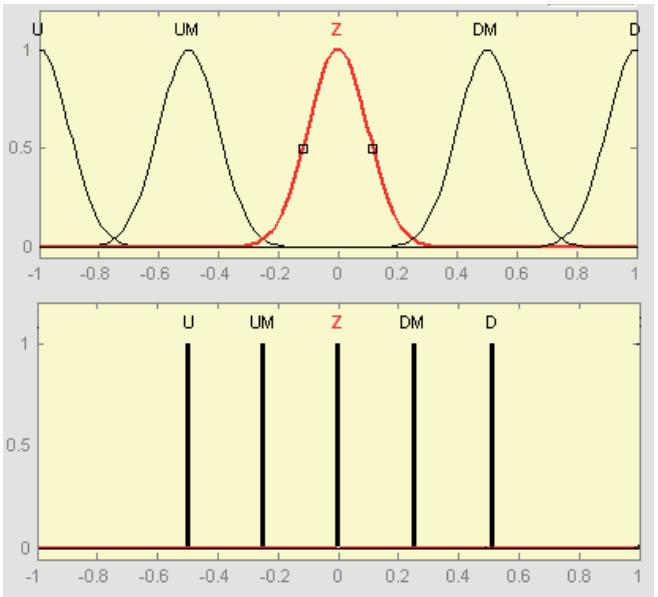

Figure 9. Fuzzy sets for inputs (gaussian functions) and output (singletons) of the TSK anti-collision system

In the TSK anti-collision system, fuzzy rules were tuned by the evolutionary method.

It was assumed that crisp values of the outputs were calculated using a weighted sum of the rules (wtsum) [1]. For this purpose, it is important to calculate the weight of the *i*-th rule and the crisp value of the output of the *i*-th implication. Calculation of rules weights is usually carried out using a logical product (min) or an algebraic product (product) [1]. Due to the lack of a proper solution of the TSK system using operation (product), in addition to the operation (min), an algebraic sum (sum) was applied [1] (usually (sum) is used for an alternative of prerequisites). In the case of using singletons on the outputs, the crisp value of the output of the *i*-th implication is reduced to the constant value of the proper singleton.

For the TSK system mentioned above preliminary studies were carried out for the simpler 30 learning scenarios. Satisfactory results were not received. Therefore, in the TSK system, additional improvement was introduced, i.e. rules with the same singletons were aggregated. For this purpose, two optional operations were applied: a logical sum (max) and the algebraic sum (sum) [1]. In this case, the weighted sum of rules operation (wtsum) was also modified in such a way that it did not work on all the rules, but on the aggregated rules for the same singletons in conclusions. In this case, the maximum number of components of the weighted sum is equal to the number of singletons in the output variable.

According to the description included in Section 2, in the FSNA, both fuzzy sets and rules were tuned in the evolutionary way.

In the FSNA, crisp values on the outputs were achieved in the result of operation of the aggregating neural network. The neural network is "fed" on the inputs with the weights of the fuzzy rules. To calculate the weight of rule, the same operations were applied as for the TSK system, i.e. the logical product (min) for a conjunction of prerequisites or the algebraic sum (sum) for an alternative of prerequisites.

TSK SYSTEM LEARNING AND VALIDATION

The learning phase of the TSK system was divided into two parts. The result of the first part was that the fuzzy systems learned by means of 30 simpler learning scenarios, described in [15]. The results of the second part were presented in the Table 1. The learning process was performed based on 30 simpler and 30 additional learning scenarios (in total 60 learning scenarios).

The main aim of the process was to compare different variants of the fuzzy system. All variants had the same distribution of the fuzzy sets, presented in the Figure 9. The variants differed in the maximum number of fuzzy rules (10 or 20) and applied fuzzy operations (min – sum, sum – sum, min – max, sum – max), respectively for (the calculation of the rules weights – the aggregation rules with the same singletons).

Tab. 1. Results of the learning for TSK system for the anti-collision problem defined by 60 learning scenarios

	Variants of TSK system			Learning	
	Aggregation of prerequisities	Aggregation of identical singletons	Number of rules	Average total evaluation function	Maximal total evaluation function
1	min	sum	10	3416	3607
2	min	max	10	3252	3605
3	sum	sum	10	3299	3606
4	sum	max	10	3393	3606
5	min	sum	20	3466	3506
6	min	max	20	3605	3606
7	sum	sum	20	3112	3606
8	sum	max	20	3436	3606

According to the results of the first part of the learning process, following parameters of the evolutionary method were used: the mutation probability equal to 0.045 and the crossover probability equal to 0.4. Similarly to the previous research [15], each variant of the fuzzy system was evolutionary tuned 30 times. Therefore, the Table 1 presents the results of the evolution in the form of the average and the maximum values of the total evaluation function achieved in 30 runs.

Tab. 2. Results of validating the TSK system for the anti-collision problem in 30 scenarios

	Variants of TSK system			Validation	
	Aggre-gation of prerequi-sities	Aggre-gation of identical singletons	Number of rules	Average total evaluation function	Maximal total evaluation function
1	min	sum	10	1215	1702
2	min	max	10	1462	1704
3	sum	sum	10	1170	1805
4	sum	max	10	1248	1805
5	min	sum	20	1556	1706
6	min	max	20	1605	2303
7	sum	sum	20	1269	2103
8	sum	max	20	1229	1902

Based on the results of evolutionary tuning of the TSK system for 60 learning scenarios (Table 1), this fuzzy system was not able to "learn" new scenarios. Evolution in the best case, stopped at scenario no. 36 and was not able to cope with the collision situation defined by scenario no. 37.

Despite the lack of even one TSK system that would have successfully avoided a collision in all 60 scenarios, to compare the TSK system with the FSNA, it was decided to validate the obtained TSK solutions by means of 30 validating scenarios.

Based on the results of validating tests presented in the Table 2, it is worth noting that the inability to tune the TSK fuzzy system in the learning phase, resulted in achieving poor evaluation in the validation phase.

FSNA LEARNING AND VALIDATION

The learning and validating processes of the FSNA were carried out by means of the learning and validating scenarios, the same as were used for the TSK system. Based on the results of the previous research [15], the subsequent parameters for validation tests were used:

1) the mutation probability 0.045,
2) the crossover probability 0.4,
3) the operation (min) for the aggregation of prerequisites.

The results of tuning 16 FSNA variants were presented in Table 3.

Tab 3. Results of learning the FSNA for the anti-collision problem defined by 60 learning scenarios

	FSNA variants		Learning		
	Number of fuzzy rules	Number of hidden neurons	Average total evaluation function	Maximal total evaluation function	Number of success-full runs
1	3	0	4094	5008	4
2		2	4580	5309	0
3		5	5095	6009	0
4		8	1464	2404	2
5	6	0	4521	6009	2
6		2	4735	6011	2
7		5	4581	6011	4
8		8	1957	2404	6
9	8	0	4085	5008	0
10		2	4464	5819	0
11		5	4264	6011	0
12		8	1638	2304	1
13	10	0	4093	5608	0
14		2	3993	4308	0
15		5	5066	6013	0
16		8	1549	2403	2

The variants differed in the number of fuzzy rules (3, 6, 8 and 10) and the number of hidden neurons in the aggregating neural network (0, 2, 5 and 8 neurons).

As in previous studies, each FSNA variant evolved during 30 runs. The results of tuning various options FSNA fuzzy system were illustrated in Table 3 as the average and the maximum overall evaluation functions. In addition, each FSNA variant was evaluated by an additional index, i.e. the number of successful runs (ended without collision).

The best learning result was achieved for the FSNA variant with 6 fuzzy rules and 8 hidden neurons in the aggregation network. For 30 runs of evolution for this variant, 6 runs were successful (the collision was avoided in all 60 learning scenarios), i.e. efficiency of tuning method was 20%.

Based on the results of evolution (Table 3), large influence of neural network for the fuzzy rules aggregation on the operation of the entire FSNA can be seen. Increasing the number of hidden neurons in these networks leads to better results (the average total evaluation function and the number of successful runs increased). Due to the condition of finishing the tuning process in a specified finite time, no research to a larger number of hidden neurons was performed.

Tab. 4. Results of validating the FSNA for the anti-collision problem in 30 scenarios

	FSNA variants		Validation	
	Number of fuzzy rules	Number of hidden neurons	Average total evaluation function	Maximal total evaluation function
1	3	0	735	1012
2		2	1917	2503
3		5	1946	2604
4		8	1946	2604
5	6	0	1977	2604
6		2	1977	2705
7		5	2037	2606
8		8	2037	2606
9	8	0	1550	2504
10		2	1550	2206
11		5	1449	2103
12		8	1449	2103
13	10	0	2060	2703
14		2	1531	2105
15		5	2018	2704
16		8	2018	2704

The verification tests were carried out for the obtained FSNA by means of 30 validating scenarios, the same as for the TSK system (Table 4). No FSNA was positively verified in all 30 validating scenarios. The best solutions for anti-collision systems managed to avoid collisions in the 27 validating scenarios, i.e. the best solutions of the FSNA achieved an effectiveness of 90%.

It can be concluded that the solutions of FSNA that have evolved in 60 learning scenarios, are able to work effectively on a different data set than the training set. As mentioned earlier, an important element for success of the learning process is the selection of training data. In this case, the learning scenarios could be improved to represent a wider range of learning data.

EXAMPLES OF FSNA OPERATION

In Fig. 10 and 12, the trajectories of the USV and 10 other vessels were illustrated for scenarios no. 14 and 10, respectively with collision and without collision. The USV starting position was marked by a circle, and the starting positions of the other vessels were marked by asterisk. Target position of the USV is the position with coordinates (4000 m, 4000 m), which is placed outside the space visualized in Figure 10.

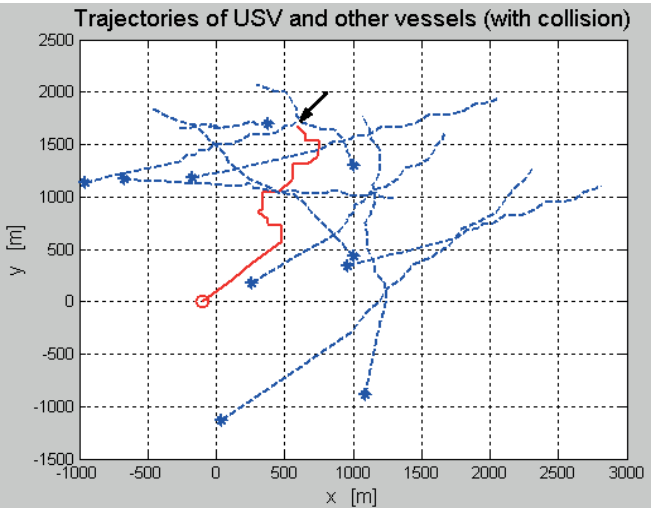

Fig. 10. Trajectories of the USV (red solid line, a circle – starting position), and the vessels (blue dotted lines, asterisks – starting positions) in scenario with collision (black arrow)

In Fig. 11, it can be seen that the USV initially moved straight to the target, then at approx. 170 s of simulation performed a maneuver avoiding a collision with one of the other vessels. Next, for longer than 200 seconds, the USV maneuvered to the port and to the starboard, trying to avoid collision and to cover the shortest path to the target. At approx. 460 s of simulation, the USV collided with one of the other vessels.

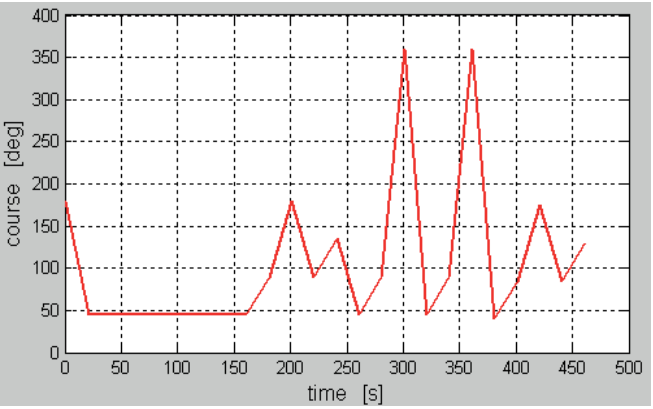

Fig. 11. Change in time of the USV desired course in the scenario with collision

In Fig. 12, it can be seen that the USV made several changes of its course during approx. 500 s of simulation. These manoeuvers enabled the USV to leave the area of potential collisions, and then to reach the target without collision.

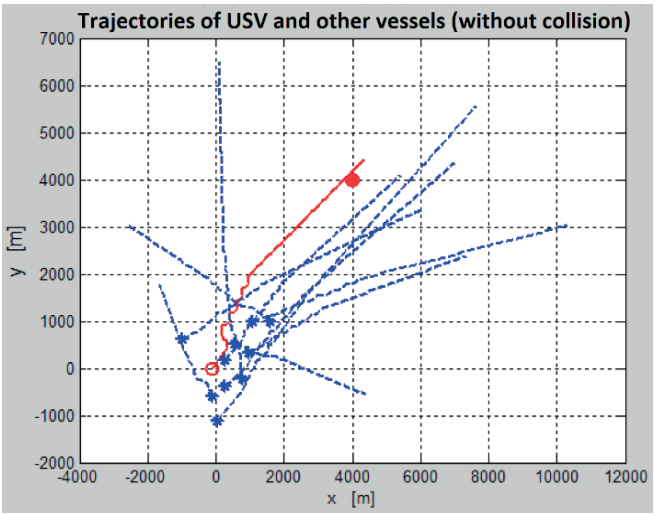

Fig. 12. Trajectories of the USV (red, solid line, a circle – starting position, a wheel – target position) and the vessels (blue, dotted lines, asterisks – starting positions) in the scenario without collision

CONCLUSIONS

In the paper, the new neuro-fuzzy system called FSNA was presented. The FSNA is an improvement of the classical TSK system enriched with (1) integration of fuzzy rules with membership functions, and (2) aggregation of fuzzy rules by an artificial neural network. The FSNA was tuned by the evolutionary method named CCGA-INE. The FSNA correctly learned and then was verified by means of respectively, the learning and validating scenarios in the anti-collision problem.

It is worth mentioning that the CCGA-INE is a quite efficient but a time-consuming method taking into consideration the learning process. The one variant of FSNA was received after 12-24 hours of a one core 3 GHz processor work. The research was conducted using BSD Operating System. Despite the long process of learning, the taught and verified variant of the FSNA can be used as a control system in time close to real using a medium class hardware platform.

The FSNA was compared with its predecessor the classical TSK system. The classical TSK system with base of rules tuned in evolutionary way (CCGA-INE) poorly generalized learned anti-collision behaviour. The classical system showed less effectiveness, both in the learning and validating phases. The selected solutions of FSNA obtained during an evolution process (Table 4) guarantee good behaviour for the validation scenarios. It should be noted that in the case of new scenarios, they always can be used for precise tuning of the FSNA (an additional learning process).

In the future, the following improvements and research are proposed to implement and test:

1) Implementation of the FSNA system based on Mamdani type fuzzy system,

2) Examination of the impact of an artificial neural network at the input of the fuzzy system, e.g. to aggregate prerequisites of the fuzzy rules,
3) FSNA testing in other control problems, e.g. to control a new marine control object – biomimetic underwater vehicle.

REFERENCES

1. D. Driankov, H. Hellendoorn, M. Reinfrank, *An Introduction to Fuzzy Control*, Springer-Verlag, 1996.
2. T.J. Fossen, *Guidance and Control of Ocean Vehicles*, John Wiley and Sons Ltd., 1994.
3. D.E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison Wesley, Reading, Massachusetts, 1989.
4. K. Guney, N. Sarikaya, Comparison of Mamdani and Sugeno Fuzzy Inference System Models for Resonant Frequency Calculation of Rectangular Microstrip Antennas, *Progress In Electromagnetics Research B*, Vol. 12, p. 81–104, 2009.
5. C. Hwang, "The integrated design of fuzzy collision-avoidance and H∞-autopilots on ships", *The Journal of Navigation*, Vol. 55(1), pp.117-136, 2002.
6. Z. Kitowski, "Autonomous unmanned surface vehicle Edredon", *Polish Hyperbaric Research*, Vol. 3(40), 2012, s. 7-22.
7. J. Lisowski, "Sensitivity of Computer Support Game Algorithms of Safe Ship Control", *International Journal of Applied Mathematics and Computer Science*, Vol. 23, No. 2, 439–446, 2013.
8. E. H. Mamdani, S. Assilian, "An experiment in linguistic synthesis with a fuzzy logic controller", *International Journal of Man-machine Studies*, Vol. 7, p. 1-13, 1975.
9. K. Naus, M. Wąż, A simplified navigational chart pyramid dedicated to an autonomous navigational system, *Polish Hyperbaric Research*, Vol. 3(40), pp. 99-118, 2012.
10. S. Osowski, *Neural networks for data processing*, in polish, Publishing House of Technology University in Warsaw, 2006.
11. M. A. Potter, K. A. De Jong, "Cooperative coevolution: An architecture for evolving coadapted subcomponents", *Evolutionary Computation*, Vol. 8(1), p. 1–29, 2000.
12. T. Praczyk, "Neural anti-collision system for Autonomous Surface Vehicle", *Neurocomputing*, Vol. 149, Part B, p. 559–572, 2015.
13. T. Praczyk, P. Szymak, "Decision System for a Team of Autonomous Underwater Vehicles – Preliminary Report", *Neurocomputing*, Vol. 74 (17), pp. 3323-3334, 2011.
14. T. Praczyk, P. Szymak, "Using Genetic Algorithms to Fix a Route for an Unmanned Surface Vehicle", in *Proceedings of the 17th International Conference on Methods and Models in Automation and Robotics*, pp. 487-492, 2012.
15. P. Szymak, T. Praczyk, "Using Neural-Evolutionary-Fuzzy Algorithm for Anti-collision System of Unmanned Surface Vehicle", in *Proceedings of the 17th International Conference on Methods and Models in Automation and Robotics*, pp. 286-290, 2012.
16. P. Szymak, "Course Control of Unmanned Surface Vehicle", *Solid State Phenomena*, Vol. 196, pp. 117-123, 2013.
17. T. Takagi, M. Sugeno, "Fuzzy Identification of Systems and its Application to Modelling and Control", *IEEE Transactions on Systems, Man and Cybernetics*, vol. 15, pp. 116-132, 1985.
18. J. Vieira, F.M. Dias, A. Mota, Neuro-Fuzzy Systems: A Survey, *WSEAS Transactions on Systems*, 3(2), 2004.
19. L. Zadeh, "Fuzzy sets", *Information and Control*, vol. 8, pp. 338–353, 1965.
20. Y. Zhuo, "An intelligent decision support system to ship anti-collision in multi-ship encounter", in *Proceedings of the Intelligent Control and Automation 2008*, pp. 1066–1071, 2008.

CONTACT WITH THE AUTHOR

Piotr Szymak
e-mail: p.szymak@amw.gdynia.pl
Polish Naval Academy
Smidowicza 69, 81-127 Gdynia
Poland