Using Neuro-Evolutionary-Fuzzy Method to Control a Swarm of Unmanned Underwater Vehicles

Piotr Szymak

Polish Naval Academy, Institute of Electrical Engineering and Automatics, 81-103 Gdynia, Poland, (e-mail: p.szymak@amw.gdynia.pl).

Abstract: The paper presents the research whose the main goal was to build a control system for a swarm of Unmanned Underwater Vehicles UUVs for predator-prey problem. To control a swarm of the vehicles new Fuzzy System with Neural Aggregation of the fuzzy rules FSNA was proposed. To learn the FSNA innovative Cooperative Co-evolutionary Genetic Algorithm with Indirect Neural Encoding CCGA-INE was used. At the beginning of the paper, the introduction to the subject of the paper is included. Next, the principles of operation of new FSNA and its tuning method CCGA-INE are presented. In the end, the results of numerical research of FSNA controlling a swarm of the underwater vehicles in a predator-prey problem are presented.

Keywords: neuro-fuzzy system, co-evolution, indirect encoding, control of unmanned vehicles swarm.

1. INTRODUCTION

Autonomous Underwater Vehicles (AUVs) are robots which can perform many different underwater missions both civilian and military, (Yong and Bishop, 2004; Menafo, 2011). Civilian usage of AUV is mainly connected with different inspections of underwater environment, especially for oceanography and marine biology purposes. Military applications of AUV are focused on mine countermeasure, anti-submarine warfare and Intelligence, Surveillance and Reconnaissance ISR tasks.

More often, the missions are performed by swarm of AUVs instead of single vehicles, which can enlarge the inspected area, increase the speed of execution, improve the accuracy of the performed operations. In the case of the swarm operation, all AUVs usually cooperate to achieve a common aim of mission (Leonard et al., 2010; Birk et al., 2011; Healey and Horner, 2008). In this case, selection of an appropriate method for a swarm control is an essential problem.

The paper takes the issue of the development of a new control approach for AUVs, which is based on evolutionary method, fuzzy system and artificial neural network. The approach is complementary and includes three components: a new Control-Oriented Model of motion of Unmanned Marine Vehicle COMUV, a Fuzzy System with Neural Aggregation of fuzzy rules FSNA for the control of AUV swarm and a tuning method for FSNA called Cooperative Co-evolutionary Genetic Algorithm with Indirect Neural Encoding CCGA-INE. The COMUV is described in more details in (Praczyk and Szymak, 2011). This paper does not include details about the new model of motion. However, it includes details of the FSNA and CCGA-INE, which are described in the next sections, also in reference to the literature. The whole approach was improved during past research, which are briefly presented in the next subsection.

1.1 Past research

The first results of the own research carried out in the field of control of AUVs team is included in (Szymak and Praczyk, 2010). In the paper, the results of the research with 'pure' fuzzy system in predator-prey problem are included. The system was tuned by the expert in the experimental way. Additionally in the paper, verification of the Control-Oriented mathematical Model of Unmanned underwater Vehicle COMUV, compared with the 'classical' nonlinear model was presented (Fossen, 1994). The obtained results of the verification tests shown that both models behave similarly. Moreover, using the control-oriented model reduces significantly simulation time what is very interesting especially in the case of using time-consuming evolutionary algorithms. Therefore, during the next research also the ones included in this paper, the COMUV was used.

In the paper (Szymak, 2012), comparison of centralized, dispersed and hybrid multi-agent system used for control of the vehicle team is presented. The compared multi-agent systems with different architectures were simple fuzzy expert systems. To examine the systems, the predator-prey problem was used. The achieved results were satisfied but not in all scenarios.

In the next paper, the fuzzy control systems mentioned above were compared with the systems based on neural network built by a neuro-evolutionary method called assembler encoding (Praczyk and Szymak, 2011). The paper shown more better efficiency of the neural network built by assembler encoding method than the classical fuzzy expert system (Praczyk and Szymak, 2011). The assembler encoding

method used for building neural networks is described in more details in (Praczyk and Szymak, 2013). During a next research devoted to an anti-collision problem of Unmanned Surface Vehicle USV (Szymak and Praczyk, 2012), the architecture of the classical fuzzy system TSK (Takagi and Sugeno, 1985), was tuned by the new CCGA-INE. The achieved TSK system successfully (without collision) controlled single USV in different collision scenarios. Based on the preliminary unpublished research, the TSK system tuned by CCGA-INE could not successfully control the AUV swarm in predator-prey problem. Control without success in predator-prey problems means that the prey was not caught by the predators in all scenarios. Therefore, in this paper the new FSNA system tuned by CCGA-INE is applied to control the AUV swarm.

1.2 General ideas of the FSNA and CCGA-INE

The FSNA is based on classical TSK fuzzy system with two improvements. The first one is connected with using artificial neural network instead of classical operator for calculation of crisp value on the fuzzy system output (called in the paper fuzzy rules aggregation). The second one 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, 2000). It was improved by adding indirect encoding of the fuzzy system by means of artificial intelligence network. The CCGA depends on the evolution of cooperating subcomponents of the overall solution. The subcomponents evolve in different populations of species, which have to cooperate to achieve desired solution.

1.3 Content of the paper

The paper is as follows: Section 2 includes details of new 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 predator-prey problem used as a test problem and section 5 presents selected results of numerical research. The last Section 6 includes a summary for the research.

2. FUZZY SYSTEM WITH NEURAL AGGREGATION

2.1 Assumptions

The new FSNA is based on classical TSK fuzzy system proposed by (Takagi, Sugeno, 1985). The TSK system is well known and often used especially in control applications. Comparing to another classical Mamdani fuzzy system (Mamdani and Assilian, 1975), TSK system is computationally simpler but similarly efficient (Guney and Sarikaya, 2009).

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

(1) using logical or algebraic product for aggregation of the rule's prerequisites (conjunction of the 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 (parameters of the fuzzy sets are included directly in fuzzy rules);
(2) using artificial neural network for aggregation of the fuzzy rules (e.g. instead of weighted sum (Driankov et al., 1996)).

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

2.2 Integration of fuzzy sets with fuzzy 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 number of fuzzy sets for each variable. In TSK system, each rule's prerequisite can operate on one fuzzy set, selected from all fuzzy sets defined for the variable. In this case, prerequisite is defined by linguistic expression, e.g.:

X_1 is HIGH

(here: X_1 is a input, HIGH determines one of fuzzy set of input X_1).

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

2.3 Matrix representation of integrated fuzzy sets and rules

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

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

where:

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

2.4 Neural aggregation of the rules

The next step in the modification of the classical TSK fuzzy system is an artificial neural network to aggregate implications of fuzzy rules. In the classical TSK system, conclusion of i-th implication of fuzzy rule is in the form of 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 (Driankov et al., 1996).

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 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 (Osowski, 2006). Thus, it seems that the application of 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 process of rule aggregation, i.e. inputs of the network are weights of rules w_i, and weights are determined using logical or algebraic product. Weights are calculated on the basis of the membership function of the individual fragments of the predecessor of each rule. In this case, an output of the whole TSK system is a crisp value of the neural network output y. The network architecture is always related to the number of rules (the number of inputs of neural network) and generally a solved problem (internal network topology, weights and types of activation function).

Fig. 1 shows an example of the FSNA structure formed by the connection of the neural network with the TSK fuzzy system. In the Fig. 1, the i-th neuron of the network is represented by N_i.

In the research presented in the further part of the paper, a feed-forward artificial neural network was applied for aggregation of the fuzzy rules (Osowski, 2006). The architecture of the network was determinated by the evolutionary method CCGA-INE.

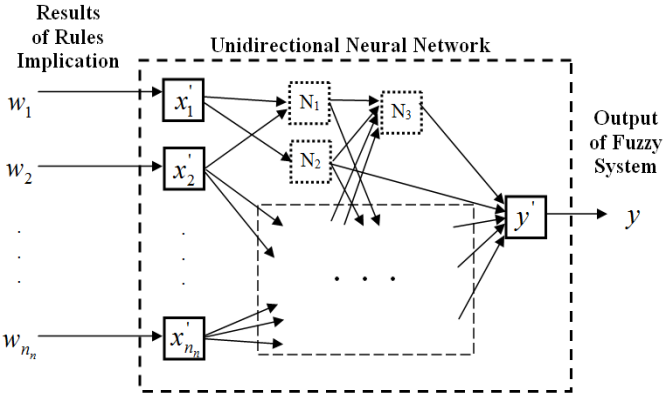

Fig. 1. The connection of neural network with TSK fuzzy system for the aggregation of fuzzy rules (N_i defines the i-th neuron).

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

3. CCGA WITH INDIRECT NEURAL ENCODING

3.1 Evolution and co-evolution

Co-evolution is a specific type of the evolution of closely related species. 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 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 natural world in the form of relation between predator and prey. A predator hunting a prey eliminates the weaker individuals from the population of prey. It causes that individuals which survive have better features that transfer to their offspring. A predator which achieve "worst results" in catching preys, has also less chance to deliver its features to its offspring.

3.2 Cooperative Coevolution Genetic Algorithm CCGA

In general, a genetic algorithm (GA) is a search heuristic that mimics the process of natural selection. The GA is based on iterative evolutionary procedure involving selection of genotypes for reproduction based on their fitness, and then by introducing genetically changed offspring (mutation, crossover and other genetic operators) into a next population. The procedure is finished after achieving satisfactory genotypes (a set of features of an individual) which correspond to high fitness phenotypes (the individual from a population) (Goldberg, 1989).

The CCGA is a specific Cooperative Coevolution Genetic Algorithm proposed by (Potter and De Jong, 2000). Generally, the CCGA solution is divided into sub-components that evolve in separate populations. There is no possibility of an exchange of genetic information between populations of separate species, but individuals of different species must work together to achieve a satisfactory overall solution. Division into the sub-components, in this case, is generally carried out by the following method. Initially, the solution is encoded in a single chromosome, which evolve 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 species is created, which follows the evolution of the two populations, etc. At some point, 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 evolutionary algorithm. In the CCGA, for the assessment of the overall solution a single individual of the first population must be connected with individuals from other populations (Potter, 1997).

3.3 General overview of CCGA-INE method

Because the fuzzy system is described by large number of parameters, therefore 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 large number of parameters, indirect encoding of the fuzzy system is proposed. In the indirect encoding method, information from the chromosomes is used to generate other system (neural network, nonlinear function, etc.), which in turn generates parameters of the FSNA. Such way of encoding is used 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** (Praczyk, 2015), 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. 2). For many coding networks, each element of the matrices representing the FSNA is generated by one of the coding network according to the algorithm described in the following subsection.

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 the 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 the rules aggregation **aNDM**.

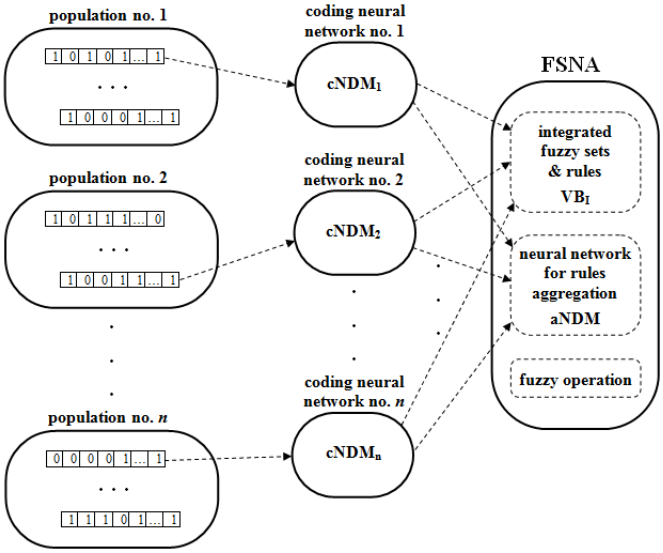

Fig. 2. Generation of the FSNA using CCGA-INE.

3.4 Generation of coding and aggregation neural networks

Fig. 3 shows a method for generating a coding neural network (defined by **cNDM**) (Praczyk, 2015), using the information stored in the chromosome, consisting of four parts (components). Each component is composed of 7 bits, i.e. the chromosome includes 28 bits in total. During the research, co-evolution produced chromosomes consisting of 4 to more than 30 components, i.e. chromosomes consisting of more than 200 bits. 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 further elements of the matrix. In the illustrated example (Fig. 3), 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 "topology" component determine, if the following elements of matrix (beginning from the first column and row, and ending on the last row and column) are zero or non-zero (Fig. 3). 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 string, then second bit, etc., until the all elements are determinated. Bit which has a zero value determines zero value of the relating element in the matrix. This element is illustrated by white box in the table in Fig. 3. Bit which has value “1” determines non-zero value of the relating element in the matrix. The non-zero element is marked by grey box in the table in Fig. 3. The precise values of non-zero elements are determinated by another components of the chromosome "coefficient no. 1", "coefficient no. 2" and "coefficient no. 3". The 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 applied to the bits of the component "topology".

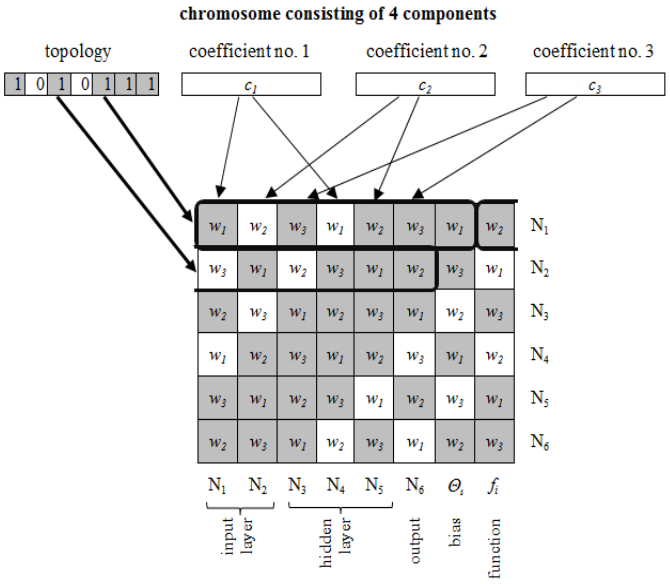

Fig. 3. Generation of matrix cNDM defining coding neural network by chromosome consisting of four components.

Matrix **cNDM** shown in Fig. 3 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 the type of activation function, and the column $n + 1$ is a constant added to the total weight of input neurons called bias.

Fig. 4 shows the architecture of an artificial neural network generated by means of information included in the chromosome and the relating matrix **cNDM**.

In the Fig. 4 chromosome components are presented in a different forms: the first component in the form of a binary sequence, the other in the form of real numbers (in fact, 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 in the Fig. 4 comprises 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 hidden neuron and one is located in output layer. As mentioned previously, the matrix **NDM** can define the coding neural network **cNDM** and also neural network for the rules aggregation **aNDM** in the same 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 (Praczyk, 2015).

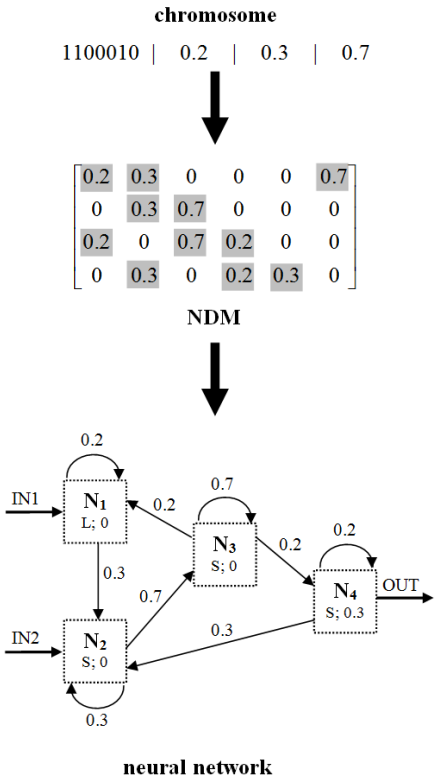

Fig. 4. Generation of neural network based on neural Network Definition Matrix NDM, which in turn, was obtained based on information included in the chromosome.

3.5 Method of filling FSNA matrices

Fig. 3 shows how to fill the matrices elements representing the FSNA. The values of the elements are produced by the coding networks. The coding networks have three inputs and three outputs. The inputs of the network determine parameters of the element, whose value is produced by the network on its output. The first and second inputs determine the row and the column of the matrix, and the third input determine the ordinal number of the matrix. The coding network produce on its outputs following values:

(1) the strenght: 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 ("wining" coding network, which has the largest strength),

(2) the 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) the value: assigned to the specific matrix element, defined by the coding network inputs, if the network has the largest strength, and the threshold output is greater than the desired threshold for this matrix.

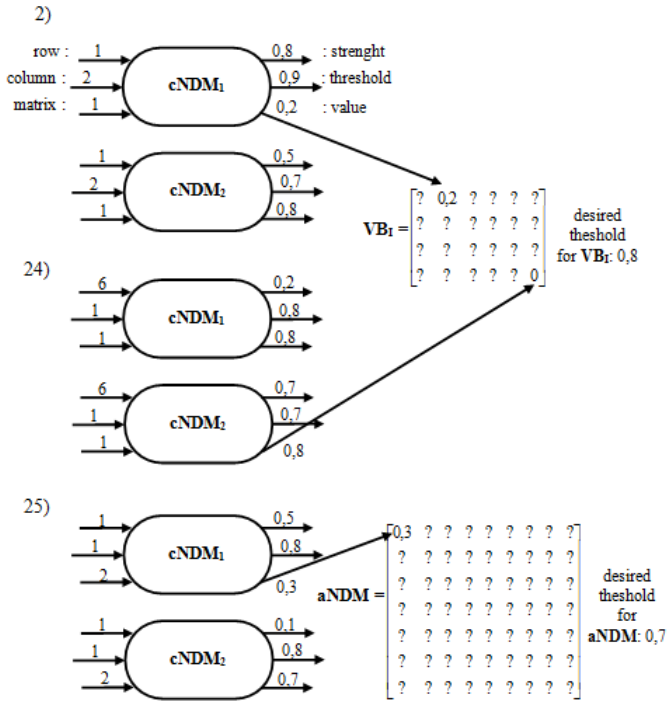

Fig. 5. Method of filling elements of FSNA matrices by coding neural networks in selected steps 2), 24), 25).

Fig. 5 shows selected steps of "filling" the FSNA matrices by coding network, generated based on chromosomes evolving in two populations (two coding networks). Step 2 illustrates a situation in which the first coding neural network (defined by **cNDM$_1$**) has a larger strength and its threshold output is greater than the desired threshold for **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 a "stronger" (has a larger value at the output strenght), 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 the all elements of the matrices requires iterations equal to the sum of elements in these matrices.

It can be seen that using the described indirect encoding method CCGA-INE, matrices defining FSNA consisting of even hundreds of parameters can be filled by means of information included in several chromosomes (depending on the number of the population).

In the next section predator-prey problem used as a test problem is described in details.

4. PREDATOR-PREY PROBLEM

4.1 Assumptions

During the previous research, the predator-prey problem was applied to compare different methods, e.g. fuzzy expert system tuned in experimental way by an expert (Praczyk and Szymak, 2011). Therefore, using the same test problem in the research presented in this paper enable to indicate if the new method is effective and how much it is effective comparing to the fuzzy expert system.

In general, in the predator-prey problem, underwater vehicles acting as predators have to catch another underwater vehicle, which plays a role of a prey. In the research presented further, the three predators had to catch the single prey. Both the prey and the predators were underwater vehicles moving in the horizontal plane. To simulate the motion of the vehicles, control-oriented model of the underwater vehicle "Ukwial" was used (Praczyk and Szymak, 2011). The vehicles moved in artificial environment – the square of 200x200 meters. The environment did not contain any obstacles, but it was open at each side. Therefore, the vehicle which "disappeared" on the left edge of the environment, appeared at this time on the right edge, and vice versa. Similarly, the vehicle which "disappeared" on the top edge of the environment, it appeared at this moment on the bottom edge, and vice versa. This simple mechanism allowed an infinite environment to be created.

During the experiments, the motions of predators were controlled by designed fuzzy system. The predators moved with constant velocity 0.5 m/s, but their directions of motion were determined by the fuzzy system. The courses of the vehicles could be changed by 0,5,10,...,355 degrees. It was assumed that the predators captured the prey, if the distance between the nearest predator and prey was lower than 5 meters. The prey was able to move with higher velocity (even two times) than the predators. The prey used the following strategy:

(1) Do not move if there is no predators in your field of vision,

(2) Run in the direction opposite to the course of predator when there is only one predator in your field of vision,

(3) In the case when there are two or three predators in the field of vision, move the following course:

$$\psi_s = \frac{\sum_{i=1}^{n} \frac{d_{\min}}{d_i} \psi_i}{n} \tag{1}$$

where ψ_s is a set value of the prey course (for maneuver to escape from predators), d_i is a distance between i-th predator and the prey, d_{min} is a minimum distance between the predators and the prey, ψ_i is a course of i-th predator (in the range ±180°) and n is a number of predators in the field of vision of the prey.

The ratio d_{min}/d_i determines what is the influence of the distance between the prey and predators on desired course of the prey in the case, when the prey sees two or more predators, i.e. the predator which is closer to the prey should decides more on set value of the prey course, or otherwise, the prey should escape in the opposite direction, in particular to the direction of motion of the closest predator.

Taking into account especially larger velocity of motion of the prey and the artificial infinite environment without

obstacles, it can be seen that the prey can be caught only as a result of coordinated effort of all the predators.

4.2 Scenarios

In order to tune the FSNA by CCGA-INE, and then verify the achieved solutions of FSNA in different situations, respectively 30 learning scenarios and further 60 validating scenarios were designed. The scenarios differ in the following parameters:

(1) the prey velocity (Table 1),

(2) the vision range of predators by the prey (Table 1),

(3) the starting position of the prey (Table 2).

Table 1. Velocity and vision range of the prey for learning (no. 1-30) and validating (no. 31-90) scenarios.

Scenario no.	Prey velocity [m/s]	Vision range of prey [m]
1-5	0,5	40
6-10	0,75	40
11-15	1	40
16-20	0,5	50
21-25	0,75	50
26-30	1	50
31-40	0,5	40
41-50	0,75	40
51-60	1	40
61-70	0,5	50
71-80	0,75	50
81-90	1	50

In all the scenarios, the underwater vehicles acting as predators start from position with coordinates (0, 0).

Table 2. Starting position of the prey for learning (no. 1-30) and validating (no. 31-90) scenarios.

Scenario no.	Coordinates of prey starting position [m; m]	Scenario no.	Coordinates of prey starting position [m; m]
1, 6, 11, 16, 21, 26	50; 50	31, 36, 41, 46, 51, 56	0; 100
2, 7, 12, 17, 22, 27	150; 50	32, 37, 42, 47, 52, 57	30; 170
3, 8, 13, 18, 23, 28	50; 150	33, 38, 43, 48, 53, 58	40; 180
4, 9, 14, 19, 24, 29	100; 100	34, 39, 44, 49, 54, 59	50; 100
5, 10, 15, 20, 25, 30	150; 150	35, 40, 45, 50, 55, 60	100; 0
		61, 66, 71, 76, 81, 56	100; 50
		62, 67, 72, 77, 82, 87	100; 150
		63, 68, 73, 78, 83, 88	170; 30
		64, 69, 74, 79, 84, 89	150; 100
		65, 70, 75, 80, 85, 90	180; 40

Scenarios presented in Tables 1 and 2 were designed in the way to increase their difficulty level. In the first scenario, the prey has a smaller range of vision and the same velocity of its movement as the predators. In the following scenarios, the prey velocity is increased by 50%, and in the part of the scenarios, even by 100%. Also in the scenarios, the range of vision of the prey is increased.

Tuning the FSNA is performed for 5 different starting positions, and testing the final solution of the FSNA is carried out for 10 various starting positions. Moreover, tuning and testing are made for different velocities and various field of vision of the prey.

4.3 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 catching the prey by the predators or after the maximum number of decisions taken by the FSNA. During research, the maximum number of decisions for a single scenario was assumed to 100 decisions.

The behaviour of the n-th FSNA in 30 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 k-th scenario was applied (Praczyk and Szymak, 2013):

$$f_k(FL_n) = \begin{cases} 0 \quad , \quad case\,a \\ d_{\max} - \min\limits_{p} d_k(p), \quad case\,b \\ f_{cap} + \left(m_{\max} - m_k\right)/l \quad , \quad case\,c \end{cases} \tag{2}$$

where k is a scenario number, FL_n is the estimated n-th FSNA, d_{max} is a maximal distance between two points in artificial environment and $d_k(p)$ is a distance between prey and p-th predator at the end of k-th scenario without success. Additionally, f_{cap} is a reward for catching prey, equal to 100, m_{max} is a maximal number of decisions taken by FSNA, equal to 100, m_k is a number of decisions taken by FSNA to catch the prey and l is a number of all scenarios.

The cases are as follows:

(a) the UUV – prey was captured in the previous scenario,

(b) the prey was captured in the current scenario,

(c) the prey was caught in the current scenario.

5. RESULTS OF NUMERICAL RESEARCH

5.1 Assumptions

To model motion of UUVs imitating both the prey and the predators, control oriented simulation model of the underwater vehicle Ukwial (Fig. 6) was used (Praczyk and Szymak, 2011). Ukwial is a Remotely-Operated underwater Vehicle ROV, equipped with a cable called an umbilical cord, which is used to power supply and control of the robot. For simplicity and making research more realistic (vehicles

should be autonomous) the control-oriented model does not take into account the effect of the affecting umbilical cord.

Fig. 6. Underwater vehicle Ukwial designed by Gdansk University of Technology.

Due to the fact that the proposed tuning method CCGA-INE is complicated and time-consuming and the propellers system of UUV Ukwial consists of four thrusters located in a horizontal plane and two in a vertical plane (acting independently in two planes of motion), the following simplifications was applied:

(1) the motions in the horizontal and vertical planes are considered independently – only motion in the horizontal plane is considered in the research of FSNA (motion in horizontal plane is more complicated than in vertical plane – greater manoeuvrability),

(2) the motion parameters are discreted, what is associated with the used motion model and relating limitation of the set of control signals, e.g. the velocity is (0.5, 0.75, 1 m/s), the course is (0, 5, 10, ... 355°),

(3) based on initial research, the time between following decisions is equal to 10 sec,

(4) the effects of sea current was omitted due to the fact that when the current is small and has a constant direction the indirect methods for measuring and effectively counteract the sea current work properly (Praczyk, Szymak, 2013); however, when the current is relatively large (i.e. close to the underwater vehicle velocity) and/or with variable direction of affecting, there are no effective methods to counteract (in such situations usually mission is not continued).

5.2 Input and output variables

In the system control of the swarm of underwater vehicles, it was assumed that the system produced change of course for each predator on its outputs based on distances vectors determined from each predator to the prey. For applied simplification, only motion in horizontal plane is considered, and each distance vector is represented by two scalar components: the distance in axis x and the distance in axis y. In this case, the FSNA has six fuzzy input variables and three output variables (Fig. 7).

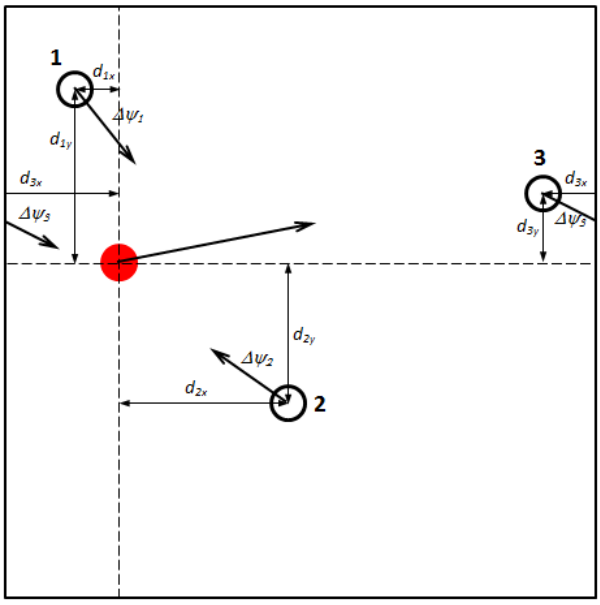

Fig. 7. Input (d_{1x}, d_{1y}, d_{2x}, d_{2y}, d_{3x}, d_{3y}) and output ($\Delta\psi_1$, $\Delta\psi_2$, $\Delta\psi_3$) variables of FSNA controlling the swarm of underwater vehicles no. 1, 2 and 3 trying to catch underwater vehicle – the prey.

5.3 Tuning of FSNA by CCGA-INE

In the process of tuning the FSNA, nine preliminary variants of the system were examined. The variants differed in the maximum number of fuzzy rules and the number of hidden neurons (Table 3).

In the first stage of the tuning phase, parameters of evolutionary method were selected: the mutation and the crossover probabilities, and the number of iterations between generation of the next population participating in the co-evolution. In accordance with the general description of the CCGA contained in the previous section, coevolution starts from one population, and when there is a lack of progress in evolution (no increase of a fitness function value), next population is created. In this case, more coding neural networks (generated from chromosomes) are involved in process of filling the FSNA matrices $\mathbf{VB_I}$ and **aNDM**.

Table 3. Nine initial variants of FSNA.

Ordinal number of FSNA variant	Maximal number of fuzzy rules	Number of hidden neurons
1	8	0
2	8	2
3	8	5
4	16	0
5	16	2
6	16	5
7	24	0
8	24	2
9	24	5

Based on the initial research, following values of the evolutionary parameters were used: the crossover probability equal to 0.7 and the number of iterations between generation of the next population equal to 10 000. Due to obtaining

different results for initial variants of the FSNA, it was decided to carry out research for the four selected mutation probability equal to: 0.0005, 0.002, 0.05 and 0.01. Therefore, each FSNA variant was examined in 120 runs of evolutionary algorithm (30 runs for each of four mutation probability).

The basic comparison of the FSNA variants was carried out based on an evaluation index named a number of runs without success. Runs without success is observed when predators do not catch the prey in one or more scenarios. In the case that two FSNA variants have the same values of the number of runs without success, the average number of the iterations is taken into account. The average number of iterations is calculated as an arithmetic average of the number of iterations of tuning runs ended successfully. Successful run means that the prey was captured in all 30 learning scenarios.

During the next research, which results were inserted in Table 4, the number of hidden neurons and the maximum number of fuzzy rules were examined. Increasing the value of the both parameters increases the calculations connected with the FSNA tuning. In this case, it was expected that an increase in the number of rules will increase the "system intelligence". At the beginning, the best number of hidden neurons was determined. As it can be seen, based on evaluation of the operation of different FSNA variants (Table 4), the most runs of evolutionary algorithm finished with success for variants with two hidden neurons, and therefore, this value of hidden neurons was used in the further research, in order to determine the best number of rules. Based on the results inserted in Table 4, the best evaluation indexes were obtained for variant no. 8, i.e. the FSNA with the largest maximal number of rules. Therefore, for the initial variants 24 rules is the best maximal number of rules.

In the next step of tuning phase, two additional FSNA variants were developed (Table 5) with greater number of rules than before (Table 4), respectively 32 and 40 rules (variant no. 10 and 11), with two hidden neurons in aggregation neural network (based on previously achieved results).

Table 4. Results of the evolution for the nine initial FSNA variants in predator-prey problem.

FSNA variant			FSNA evaluation (learning)	
No.	**Number of hidden neurons**	**Number of rules**	**Average number of iterations**	**Number of scenarios without success**
1	**0**	8	24749	**67**
2	2	8	28077	81
3	5	8	**21341**	85
4	0	16	17927	18
5	**2**	16	14927	**14**
6	5	16	**14673**	21
7	0	24	12360	2
8	**2**	24	17826	**0**
9	5	24	**10960**	3

Based on the obtained results (Table 5), it can be seen that during tuning the FSNA control system for the swarm of underwater vehicles:

(1) The best results (all 30 runs with success) were obtained for variants with 24 or more fuzzy rules,
(2) Increasing number of rules (above 24) reduced the average number of iterations needed to obtain solutions, which worked effectively in all 30 learning scenarios,
(3) The best performance of the FSNA in the predator-prey problem was achieved for the FSNA variants with two hidden neurons in aggregation neural networks.

Table 5. Results of evolution for two additional FSNA variants in predator-prey problem.

FSNA variant			FSNA evaluation (learning)	
No.	**Number of hidden neurons**	**Number of rules**	**Average number of iterations**	**Number of scenarios without success**
10	**2**	32	**11909**	0
11	**2**	40	8336	0

In the next subsection, a verification of obtained FSNA solutions is presented.

5.4 Validation tests of FSNA solutions

Validation tests of obtained FSNA solutions (for all 11 variants) were performed for 60 validating scenarios (Table 6). Table 6 presents the fitness functions (mean, minimum and maximum) obtained for all FSNA solutions achieved in 30 runs of the evolutionary algorithm, i.e. presents the average results of 30 FSNA with different structure obtained in 30 runs of evolutionary tuning in each of the tested variant. For verification tests, only FSNA solutions generated in the process of evolution with mutation probability equal to 0.01 were selected.

Table 6. Results of validation tests for twelve FSNA variants in predator-prey problem.

FSNA variants			FSNA evaluation (generalization)			
No.	**Number of hidden neurons**	**Number of rules**	**Average number of iterations**	**Fitness function**		
				mean	**min**	**max**
1	0	8	11	1105	3545	4941
2	2	8	29	2919	151	5141
3	5	8	22	2263	2947	5042
4	0	16	45	4595	49	5141
5	2	16	44	4495	2046	5338
6	5	16	46	4628	151	5439
7	0	24	45	4549	46	5042
8	2	24	44	4498	0	5239
9	5	24	44	4475	46	5340
10	**2**	**32**	**47**	**4728**	3844	**5539**
11	2	40	44	4462	3144	5239

Based on the results of validation phase illustrated in Table 6 following conclusions could be formulated:

(1) The best results were obtained for the FSNA variant no. 10; similar results were received for the FSNA variant no. 11 and 8, which confirmed that FSNA for predator-prey problem should have a neural network for the aggregation rules with two hidden neurons,
(2) The best results were achieved for the FSNA with 32 fuzzy rules (variant no. 10);
(3) The best solution of the control system of underwater vehicle allowed predators to catch prey in 55 scenarios (FSNA variant no. 10 obtained in 21st test with fitness function equal to 5539).

Comparing FSNA variant no. 10 with variants no. 8 and 11 (respectively 24 and 40 rules), it is worth underlining that the FSNA should have the structure possibly simplest (e.g. 32 rules instead of 40 rules), which guarantees generalization properties.

It is also worth noting that the FSNA variant no. 10 has a fuzzy inference base consisting of 32 rules and an aggregation neural network with two hidden neurons. In this case, the system is encoded in two matrices: $\mathbf{VB_I}$ with size 32x14 and **aNDM** with size 37x39. Together these two matrices have 1891 elements. In practice, some elements are zero, and some elements of the matrices have the same value.

5.5 Example of FSNA operation

For visualization of the exemplary operation of the system controlling the swarm of underwater vehicle, the FSNA variant no. 10 achieved in 22nd test was selected. This system has achieved the fitness function equal to 4740, which means that predators caught the prey in 47 scenarios (one of them is illustrated in Fig. 8).

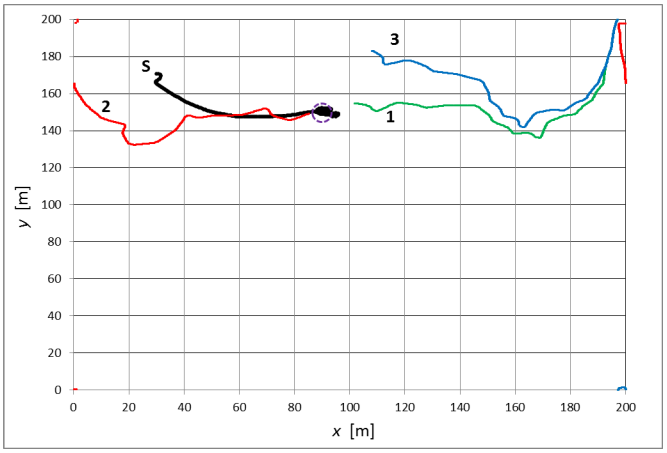

Fig. 8. Visualization of the FSNA operation in the vertical plane (variant no. 10 obtained in 22nd test), performing with success scenario no. 32: S - start position of the prey, 1, 2, 3 - predators no. 1, 2, 3, which start from position (0, 0).

Fig. 8 shows the strategy for control of underwater vehicles learned during the process of evolution. The vehicles acting as predators caught the prey in the position with coordinates (90, 150). Place of catching the prey was marked by a circle with a dotted line. Starting position of the prey was marked by "S", and trajectories of predators by their ordinal numbers 1, 2, 3. Predators in all scenarios started from the position specified by the coordinates (0, 0). The next Fig. 9 illustrates changes of course of all four underwater vehicles.

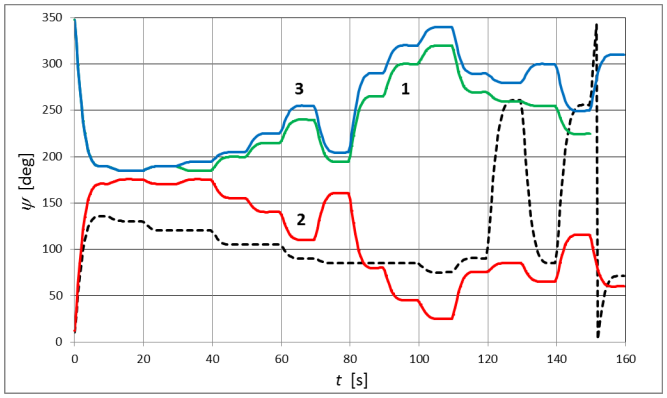

Fig. 9. Changes of course: the prey (black dashed line) and predators (number: 1, 2 and 3) – the FSNA variant no. 10 obtained in 22nd run in scenario no. 32.

As can be seen, predators made changes in every step (10 s), while the predator decided on changes of course, even at 3 decision steps (30 s). The difference is due to a range of vision of the prey equal in this scenario to 40 m, i.e. the prey 'saw' only predators that were on distance no more than 40 m.

6. CONCLUSIONS

In the paper, new neuro-fuzzy system called FSNA was presented. The FSNA was tuned by the evolutionary method named CCGA-INE. The FSNA was correctly learned and then verified by means of validating scenarios in the predator-prey problem.

Comparing the results for the FSNA with the results obtained earlier for the fuzzy expert system (Praczyk, Szymak, 2011), significant improvement in the effectiveness of the new FSNA is observed. The best fuzzy expert system effectively chased the prey only in 12 successive learning scenarios.

Generally, the selected solutions of FSNA obtained during evolution (Table 6) 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 (additional learning).

REFERENCES

Birk, A., Antonelli, G., Caiti, A., Casalino, G., Indiveri, G., Pascoal, A., Caffaz, A. (2011). The CO3AUVs (Cooperative Cognitive Control for Autonomous Underwater Vehicles) project: Overview and current progresses, *IEEE Proceedings of the Oceans Conference*, pp. 1 – 10.

Driankov, D., Hellendoorn, H., Reinfrank, M. (1996). *An Introduction to Fuzzy Control*, Springer-Verlag.

Fossen, T.J. (1994). *Guidance and Control of Ocean Vehicles*, John Wiley and Sons Ltd.

Goldberg, D.E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison Wesley, Reading, Massachusetts.

Guney, K., Sarikaya, N. (2009). Comparison of Mamdani and Sugeno Fuzzy Inference System Models for Resonant Frequency Calculation of Rectangular Microstrip Antennas, *Progress In Electromagnetics Research B*, Vol. 12, pp. 81–104.

Healey, A.J., Horner, D.P. (2008). Collaborative Unmanned Systems for Maritime Interdiction and Riverine Operations, *Proceedings of 17th World Congress of International Federation of Automatic Control*, Seoul.

Leonard, N.E., Paley, D.A., Davis, R.E., Fratantoni, D.M., Lekien, F., Zhang, F. (2010). Coordinated control of an underwater glider fleet in an adaptive ocean sampling field experiment in Monterey Bay, *International Journal of Field Robotics*, Vol. 27(6), pp. 718–740.

Mamdani, E. H., Assilian, S. (1975). An experiment in linguistic synthesis with a fuzzy logic controller," *International Journal of Man-machine Studies*, Vol. 7, pp. 1-13.

Menafo, A., Simetti, E., Turetta, A., Caiti, A., Casalino, G. (2011). Autonomous underwater vehicle teams for adaptive ocean sampling: a data-driven approach, Springer-Verlag, *Ocean Dynamics*, pp. 1981–1994.

Osowski, S. (2006). Neural Networks for data processing, Publishing House of Warsaw University of Technology.

Potter, M. (1997). *The Design and Analysis of a Computational Model of Cooperative Coevolution*, PhD Thesis, George Mason University, USA.

Potter, M. A., De Jong, K. A. (2000). Cooperative coevolution: An architecture for evolving coadapted subcomponents. *Evolutionary Computation*, Vol. 8(1), pp. 1–29.

Praczyk, T. (2015). Neural anti-collision system for Autonomous Surface Vehicle, Neurocomputing, Vol. 149, Part B, pp. 559–572.

Praczyk, T., Szymak, P. (2011). Decision System for a Team of Autonomous Underwater Vehicles – Preliminary Report, Elsevier, *Neurocomputing*, Vol. 74 (17), pp. 3323-3334.

Praczyk, T., Szymak, P. (2013). Using Assembler Encoding to build neuro-controllers for a team of autonomous underwater vehicles, Systems Research Institute Polish Academy of Sciences, *Control and Cybernetics*, Vol. 42, No. 1, pp. 267-286.

Szymak, P. (2012). Comparison of Centralized, Dispersed and Hybrid Multiagent Control Systems of Underwater Vehicles Team, Trans Tech Publications, *Solid State Phenomena*, Vol. 180, pp. 114-121.

Szymak, P., Praczyk, T. (2012). Using Neural-Evolutionary-Fuzzy Algorithm for Anti-collision System of Unmanned Surface Vehicle, IEEE, *Proceedings of the 17th International Conference on Methods and Models in Automation and Robotics*, pp. 286-290.

Szymak P., Praczyk, T. (2010). Control of a Team of Underwater Vehicles in Predator-Prey Problem, Polish Society of Theoretical and Applied Mechanics, *Scientific Aspects Of Unmanned Aerial Vehicle*, pp. 591-605.

Takagi, T., Sugeno, M. (1985). Fuzzy Identification of Systems and its Application to Modelling and Control, *IEEE Transactions on Systems, Man and Cybernetics*, vol. 15, pp. 116-132.

Yong, Ch.T., Bishop B.E. (2004). Evaluation of robot swarm control methods for underwater mine countermeasures, IEEE, *Proceedings of the Thirty-Sixth Southeastern Symposium on System Theory*, pp. 294 – 298.