Yulian FEDIRKO[1], Olha SVATIUK[2]

Scientific supervisor: Olexander BELEJ[3]

DOI: https://doi.org/10.53052/9788366249868.03

ZASTOSOWANIE ALGORYTMÓW NEUROEWOLUCYJNYCH W SYSTEMACH DO WYKRYWANIA ATAKÓW NA SYSTEMY CYBERFIZYCZNE

Streszczenie: W prezentowanych badaniach zaproponowano technikę zastosowania algorytmów neuroewolucyjnych w procesach wykrywania ataków sieciowych na systemy cyberfizyczne. Podczas wdrażania tej techniki oceniono dokładność opracowanego systemu wykrywania ataków sieciowych. Zasadą działania takiego systemu jest identyfikowanie odchyleń pomiędzy aktualnymi wartościami stanu systemów cyberfizycznych, a przewidywanymi wynikami. Prognoza opiera się na neuroewolucyjnym algorytmie rodziny topologii komplementarnych NeuroEvolution.

Słowa kluczowe: systemy cyber-fizyczne, algorytmy neuroewolucyjne, wykrywanie ataków sieciowych

APPLICATION OF NEUROEVOLUTIONARY ALGORITHMS IN SYSTEMS FOR DETECTING ATTACKS ON CYBER-PHYSICAL SYSTEMS

Summary: In the presented research the technique of application of neuroevolutionary algorithms in processes of detection of network attacks on cyber-physical systems is offered. During the implementation of this technique, the accuracy of the developed system of network attack detection was assessed. The principle of operation of such a system is to identify deviations between the current values of the state of cyber-physical systems and the predicted results. The prediction is based on the neuroevolutionary algorithm of the NeuroEvolution of Augmenting Topologies family.

Keywords: cyber-physical systems, neuroevolutionary algorithms, network attack detection

[1] Lviv Polytechnic National University, Institute of Computer Science and Information Technologies, Computer Sciences: +380506852005, 23 Saharova Str., Lviv yulian.fedirko.mknm.2020@lpnu.ua

[2] Lviv Polytechnic National University, Institute Of Computer Science And Information Technologies, Computer Sciences: +380676790080, FRANKO STR. 105 / 9 Lviv, olhasva@gmail.com

[3] Ph.D., Lviv Polytechnic National University, Department of Computer-Aided Design, Oleksandr.I.Belei@lpnu.ua

1. Introduction

In today's reality, the use of the concept of the industrial Internet of Things in the environment of cyber-physical systems (CPS) is not in doubt. Considering this area, one of the priority tasks is the methods used to present the data circulating in the CPS, the possible advantages, disadvantages, and scope of these methods [1, 2].
Now we have discusses new ways of presenting CPS data, but there are a huge number of models and methods for meeting end-user requests in the field of CPS information security [3].
This document provides a detailed analysis and comparison of existing CPS data description methods, CPS network attack detection methods, analysis of key CPS security approaches and solutions, and advisory additions to existing approaches based on the new unit.
Further areas of improvement include the introduction into theory and practice of attack detection systems (ADS), methods of the theory of synthesis and analysis of information systems, and a specific apparatus of pattern recognition theory, as these sections of the theory provide specific research methods for ADS systems.

2. Problem formulation

Based on all of the above, when considering CPS, which is subject to increased requirements in the field of information and cyber-physical security, in the presence of sufficient computing power, the authors recommend focusing on solutions based on machine learning, due to increased variability and analysis analyzer cycle. For additional in-depth analysis of CPS for maximum system security [4].
In other cases, for example, if it is not possible to meet the criteria of sufficient consumption of system computing resources and the ability to allow either only short-term or only long-term attacks, it is permissible to use both statistical tools and self-similarity criteria. The latter, in turn, is recommended in cases of small heteromorphic systems for greater efficiency and reliability, or in cases of multifractal, when you can apply the criteria for each subsystem, or if necessary to identify various anomalies, but the impossibility of using a learning machine [5].
In some special cases of deployment of a system for peripheral computing, the creation of networks, military operational networks, and other special cases, it is recommended to use modifications of graph structures due to the ease of letter conversion. This solution will provide maximum flexibility and binding to a very narrow task in a system with excessively high heteromorphism. In the case of low computing power or a large delay, it is recommended to use statistical tools in such networks to analyze the states of intermediate devices and logic nodes. In cases of excessively low computing power, it is worth thinking about the model of behavioral events and level agents described [6].

3. The main material

The "sandwich" structure was chosen as the starting configuration of the neural network: two two-dimensional flat grids with input and output nodes, where one layer can build connections in the direction of another.
As the primary substrate is used in the form of a multidimensional time series from time t_0, obtained from the data of the 7 devices described previously, and has a 28 dimension.
Multidimensional time series from time to time are used as input data.
As the initial data at the top of the hypercube, we obtain a multidimensional time series of the future state of the system in time t_{i+1} [7].
The addition of any arbitrary node or bond gene during the evolution of the network leads to the emergence of a new global dimension of the variation of bond patterns, ie to the emergence of new traits through the substrate of the phenotype. A new way to change the connection pattern is ultimately to modify the genome of the hypercube structure - to change the parameters by the method of simulated evolution of the rearrangement of connections and nodes. Additionally, previously created connections in the network can be reused as a basis for creating a new connection pattern for a substrate with a higher resolution than the original, which is used for training. Thus, this approach allows obtaining a solution to the problem at any resolution of the hypercube grid [8].
The above properties have made the hypercube algorithm a powerful tool for the development of large-scale artificial neural networks that mimic biological objects, and also corrected the problem of stagnation of neural network solutions by introducing variability in the location of nodes.
After modification of the algorithm, there is no need to strictly specify the structure of the neural network, as it can change during evolution due to the genetic component - the growth of intermediate layers, changes in the number of active neurons, and existing connections.
Figures 1-2 show the crossover operators and mutation operators used in conjunction with their operating principles. Since changing the neural network layout can be reduced to changing the location of the bus vertices and changing the connections between the vertices, it was decided to limit this set.
Finally, the use of crossover and mutation operators is as follows:
1. Inversion - bit change of communication, its weight or activity of neurons.
2. Change of order - transfer of an existing node and connections to another area. In the end, it comes down to changing the configuration of the links.
3. Cost change - a change in the weight of connections and activity of neurons.
4. Change of expression - the creation of new neurons, connections, construction of additional inverse dependences, or their removal.
5. Single, two-point and unified crossovers ultimately allow you to "mix" solutions between neural network nodes, reconfigure existing connections and their weight without changing their number and weight.
Also, figures 1-5 show real mutations in the phenotype of the neural network topology. In figures, 1-2 shows the initial population of the substrate, and Fig. 3 shows a new phenotype of the substrate obtained after mutation.

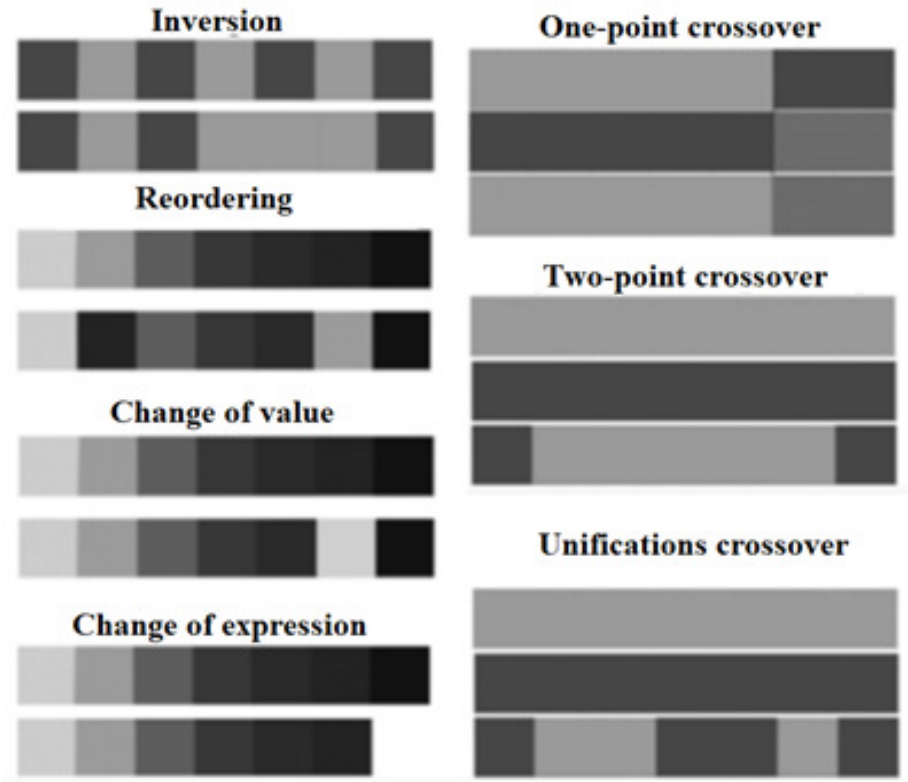

Figure 1. Used crossover operators

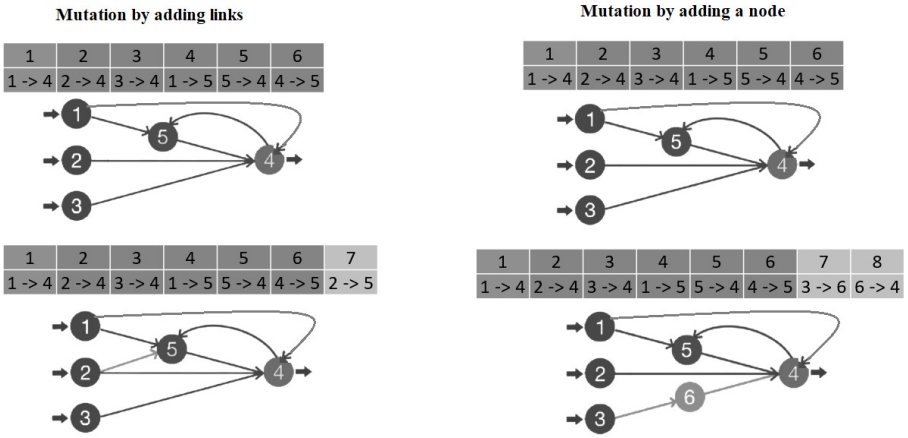

Figure 2. The initial population of the substrate

1	2	3	4	5	6	7	8
1 -> 4	2 -> 4	3 -> 4	1 -> 5	5 -> 4			2 -> 5

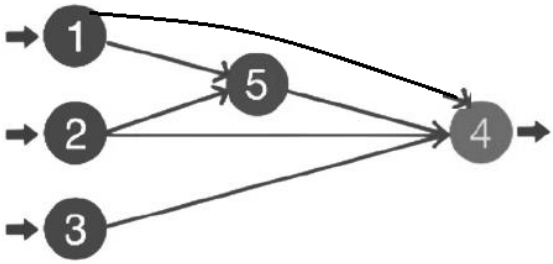

Figure 3. A phenotype mutation

1	2	3	4	5	6	7	8	9
1 -> 4	2 -> 4	3 -> 4	1 -> 5	5 -> 4	3 -> 6	6 -> 4		2 -> 6

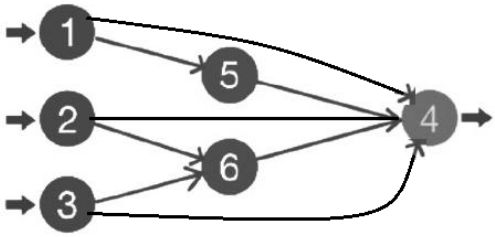

Figure 4. A real mutation in the phenotype of the neural network topology

The described method is based on processing the obtained multidimensional time series composed of data circulating within the CPS, predicting the future state of the system employing a modified neuroevolutionary algorithm NEAT-hypercube and analysis of errors - discrepancies between real values of the system and predicted.
The methods include 2 stages - preparatory and working. The preparatory stage is aimed at automatically configuring the optimal topology of the neural network and involves the following steps:
1. Preparation of test data - normalization and compilation of multidimensional time series.
2. Transmission of the obtained multidimensional series to the input of the neural network, initially configured by the user.
3. Training of the neural network on the transmitted data and its reconfiguration of the genetic component of the neuroevolutionary algorithm until the specified accuracy is obtained on the test data.

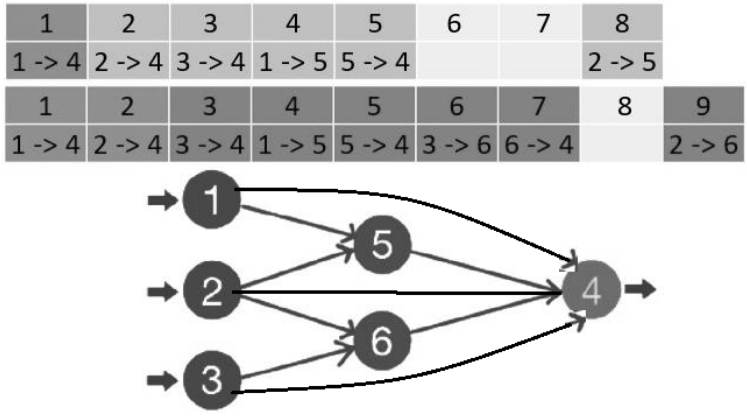

Figure 5. A new phenotype of the substrate obtained after mutation

The working stage involves the direct detection of network attacks aimed at the CPS, and includes the following steps:

1. Preparation of real data of the functioning CPS - normalization, and compilation of multidimensional time series.
2. Transfer of the obtained multidimensional series to the input of the neural network, optimally configured genetic component of the neuroevolutionary algorithm.
3. Prediction of the future state of the system by a neural network based on the obtained multidimensional time series.
4. Calculation of the error between the predicted state of the system and the real one.
5. Recording the presence or absence of attacks on the CPS based on the received error.

The scheme of operation of the method is presented in figure 6.

In step 1, the generated multidimensional time series S (t_i) from time ti is fed to the input of the neural network. In step 2, the prediction operation of the future series pred_S(t_{i+1}) is performed based on the series S(t_i), where Pred() is a prediction function performed by the neural network. In steps 3.1 and 3.2, the multidimensional series pred_S(t_{i+1}) and the multidimensional series obtained from the real indicators of the system S(t_{i+1}) are supplied to the comparison unit. In step 3.3, the difference between the indicators is calculated and an error is accumulated. In step 4, based on the comparable data in block 3.3, we get the answer about the presence or absence of attacks. In step 5, the value of the multidimensional time series from time ti is replaced by the values from time t_{i+1}, after which the algorithm is repeated.

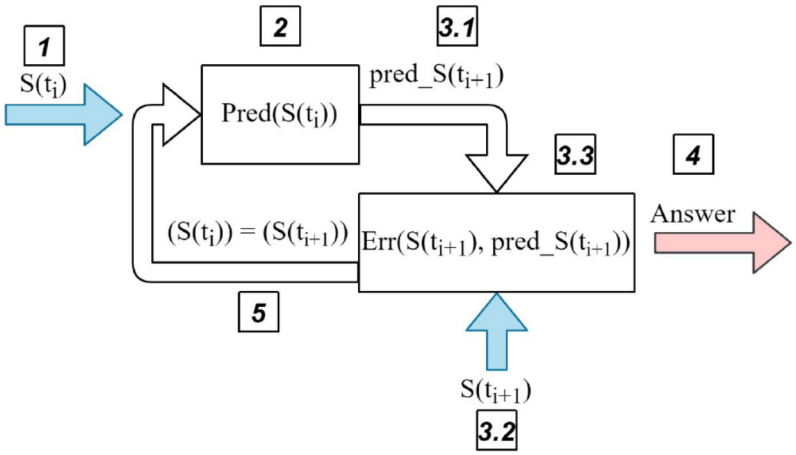

Figure 6. Schematic diagram of the method of detecting network attacks

As mentioned earlier, the data that have passed the normalization procedure must be pre-processed: each point in the time series is determined by the predicted value, as shown in figure 7.

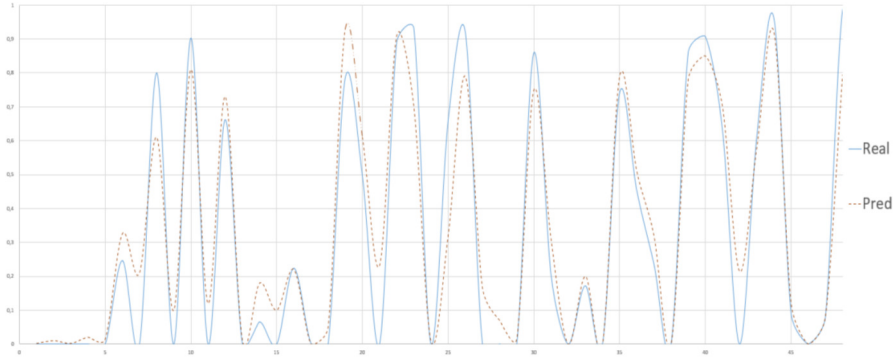

Figure 7. Predicting the state of the cyber-physical system

To predict the further value of the state of the system through the time series, you must operate:

$$\gamma_{pred} = pred(x_{t-1}, \dots, x_{t-n}). \quad (1)$$

The prediction operation is performed employing a neural network configured by the hypercube algorithm.

The states of the system predicted by the neural network may differ to some extent from the actual values, so it is necessary to take into account the error - the possible difference between the indicators.

To calculate the error between the predicted state of the system and the actual one, the following series of actions are performed:

- calculation of the difference between the predicted and actual value:

$$err_t = \left|\gamma_{pred} - \gamma_{real}\right| \quad (2)$$

- recording the presence or absence of an attack based on the condition of exceeding the value of the error of the real state and provided for more than a fixed amount:

$$Max(err_t) > T, \tag{3}$$

where T – the limit value of the manifestation of abnormal behavior in the system. However, there is a possibility of false positives due to short-term "emissions" of large prediction errors at short intervals, so it is necessary to take into account the average error over some time:

$$ERR_i = \langle \sum_{t=i-k}^{i} Max(err_t) \rangle > T \tag{4}$$

Figures 8-11 give examples of the magnitude of the error between the predicted and actual state of the system in the presence and absence of attacks.

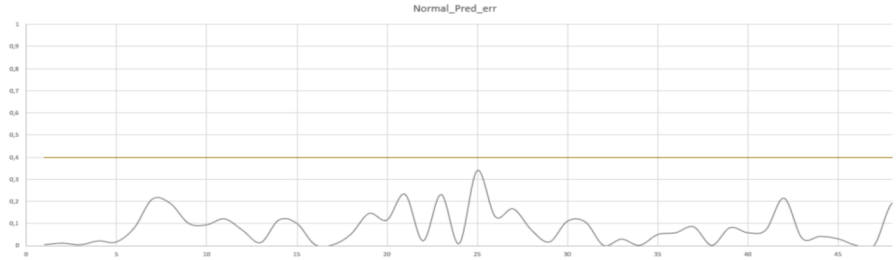

Figure 8. A system state prediction error in the absence of attacks

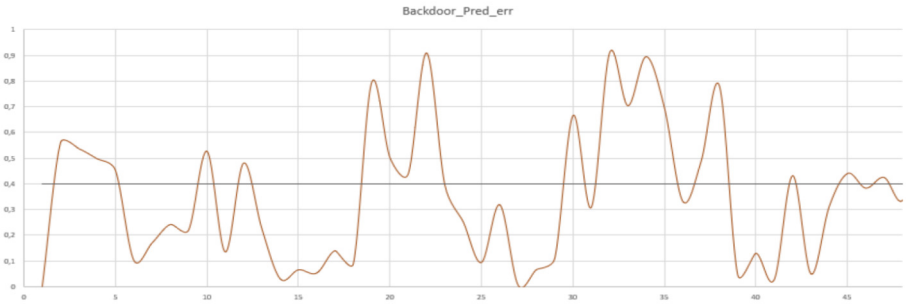

Figure 9. A system state prediction error in the event of a Backdoor attack

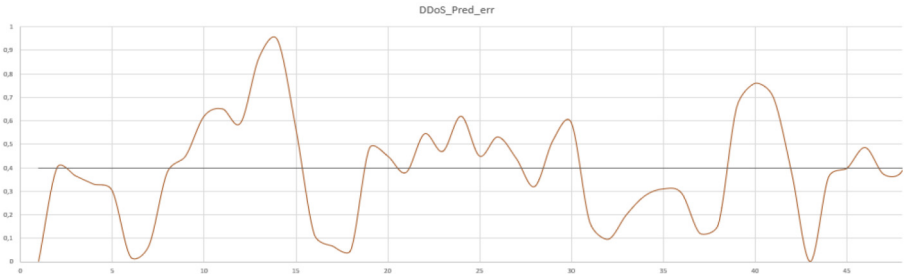

Figure 10. A system status prediction error in case of a DDoS attack

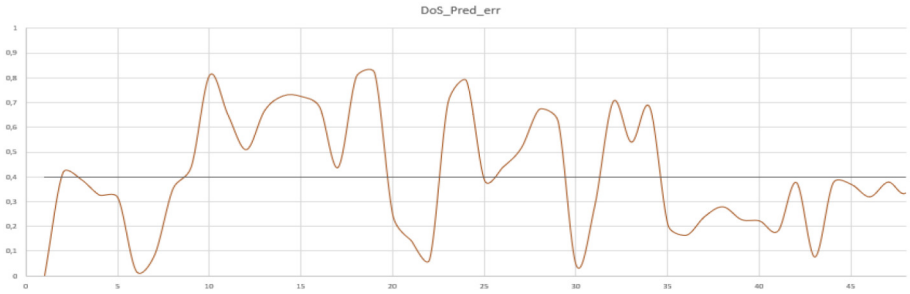

Figure 11. A system state prediction error in a DoS attack

The software implementation was performed using Python. Primary data processing was performed by a standard library that allows you to work with ".csv" files.
The creation of multidimensional time series was performed using the mathematical library Pandas.

4. Discussion

The neural network was built and trained according to the algorithm of a modified hypercube using the NEAT-Python library of the Python language. The NEAT-Python library uses a set of hyperparameters that affect the performance and accuracy of the NEAT algorithm.
The following hyperparameters were used in the work (the most important ones are given):
1. The activation function of all network nodes is sigmoidal, and the inputs of the nodes are aggregated by the sum function: activation_default = sigmoid, aggregation_default = sum. This choice is due to the desire to highlight weak signals and try to avoid saturation and oversaturation of strong signals.
2. Encrypted network type - fully connected reverse propagation network: feed_forward = True, initial_connection = full_direct. The basis of this solution is the desire to optimize the rate of ascent of the neurogenetic algorithm of the neural network topology. The reluctance to use the inverse distribution network topology in the initial substrate is justified by the lack of need to restore the shape and frequency of the primary data. However, in the future, the substrate is allowed to evolve into a feedback network, which is observed in practice.
3. In the course of evolution, new network nodes and connections are added f, j is removed with a certain probability. The probability of adding and removing nodes was set to a lower value than the probability of the appearance and removal of links. This decision is based on the desire to spread feedback, to optimize the network topology employing interconnected data, the emergence, and availability of minimizing the creation of "dead" nodes.
4. The probability of adding or removing a node - node_add_prob = 0.05, node_delete_prob = 0.05.
5. The probability of adding or removing communication - conn_add_prob = 0.3, conn_delete_prob = 0.3.

6. All connections are enabled by default with a very low probability of disconnection due to mutations: enabled_mutate_rate = 0.01. Although the topology of the neural network is generated relatively arbitrarily, it is not necessary to abandon the operations of screening and "drop-layer". The introduction of mutations disabling arbitrary nodes and connections allows you to get rid of parasitic indirect dependencies, which are not always able to affect the forecast of the system. Again, in the case of successful non-parasitic feedback, fitness function will not allow the death of populations with such a successful mutation.
To stimulate species diversity, we set the strong influence of redundant parts of the parent genomes on the distance between genomes: the parameters of the distance between genomes - compatibility_disjoint_coefficient = 1.0. This solution allows you to first create the maximum possible pseudo-random population of individuals for further crossover. Otherwise, when creating identical individuals, we have to wait for the additional time of occurrence of the initial mutations required for successful crossover operations.
Species stagnation can last up to 50 generations, and unique species are partially protected from extinction: species_fitness_func = min, max_stagnation = 50, species_elitism = 4. The choice of these values is due to the desire to maintain a smooth trend of the topology of the system. In the case of underestimation of the existence of stagnant populations, there is an abrupt development of the system topology, but in such moments the system solution is difficult to call stable and accurate - from time to time the convergence of the system is quite arbitrary and there can be solutions on many not very valid and optimal structures.
The number of species protected from extinction, in contrast to the duration of stagnation, was underestimated to maintain the possibility of the existence of more different individuals per unit time.

5. Conclusions

The result of this work is the creation, implementation, and experimental research of the implemented method for detecting network attacks carried out on the CPS. The method includes the use of a neuroevolutionary algorithm of the NEAT family: modified NEAT-hypercube.
After the modification, the algorithm allows almost completely configuring the target neural network without user intervention according to the specified parameters, including additionally creating intermediate network layers that were previously unavailable in the primary version of the algorithm.
The detection of network attacks carried out on the CPS was carried out in several stages:
1. Primary data processing and presentation of them in the form of multidimensional time series.
2. Configuring the neural network of the genetic component
3. Training a configured neural network on a test set.
4. Predict the future state of the system based on current data.
5. Calculation of the error between the predicted and real states of the system.
6. Comparison of the received error with the minimum threshold value T.
Testing was performed on the TON_IOT DATASETS dataset [14].

The obtained overall accuracy (Precision; 0.9152) and the proximity of solutions (Accuracy; 0.8861), as well as the values of False Positive Rate (0.1206) and False Negative Rate (0.1094), indicate the absence of model overfitting and high reliability of this method.
A further direction of the development of the topic is the creation of a data flow model of cyber-physical systems based on a hypercube with the possibility of self-healing according to an adaptive graph structure.

LITERATURA

1. KIM S.; PARK K.-J.: A Survey on Machine-Learning Based Security Design for Cyber-Physical Systems. *Appl. Sci,* 2021, *11*, 54-58.
2. DE SOUSA C. A. R.: An overview on weight initialization methods for feedforward neural networks. *2016 International Joint Conference on Neural Networks (IJCNN),* 2016, 52-59.
3. HUNDMAN K., CONSTANTINOU V., LAPORTE CH., COLWELL I., SODERSTROM T.: Detecting Spacecraft Anomalies Using LSTMs and Nonparametric Dynamic Thresholding. KDD '18: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2018, 387–395.
4. FILONOV P., LAVRENTYEV A., VORONTSOV A.: Multivariate Industrial Time Series with Cyber-Attack Simulation: Fault Detection Using an LSTM-based Predictive Data Model. NIPS Time Series Workshop 2016.
5. NANDURI A., SHERRY L.: Anomaly detection in aircraft data using Recurrent Neural Networks (RNN). Integrated Communications Navigation and Surveillance (ICNS), IEEE 2016, 5C2-1-5C2-8.
6. Yi S., Ju J., Yoon M.-K., Choi J.: Grouped Convolutional Neural Networks for Multivariate Time Series, *https://arxiv.org/pdf/1703.09938.pdf,* 1.04.2020.
7. STOUFFER K., FALCO J., SCARFONE K.: Guide to Industrial Control Systems (ICS) Security – Supervisory Control and Data Acquisition (SCADA) systems, Distributed Control Systems (DCS), and other control system configurations such as Programmable Logic Controllers (PLC), Special Publication (NIST SP). National Institute of Standards and Technology, Gaithersburg, *https://doi.org/10.6028/NIST.SP.800-82*, 11.03.2021.
8. NOUR M.: ToN_IoT datasets. IEEE Dataport, *https://dx.doi.org/10.21227/fesz-dm97*, 16.10.2019.