Advances in Science and Technology Research Journal
Volume 7, No. 18, June 2013, pp. 20–27
DOI: 10.5604/20804075.1049490

Original Article

DISTRIBUTION OF THE TREE PARITY MACHINE SYNCHRONIZATION TIME

Michał Dolecki[1], **Ryszard Kozera**[1, 2]

[1] Faculty of Mathematics, IT and Landscape Architecture, The John Paul II Katholic University of Lublin, ul. Konstantynów 1H, 20-708 Lublin, Poland, e-mail: michal.dolecki@kul.pl

[2] Faculty of Applied Informatics and Mathematics, Warsaw University of Life Sciences – SGGW, ul. Nowoursynowska 159, 02-776 Warsaw, Poland, e-mail: ryszard_kozera@sggw.edu.pl, ryszard.kozera@gmail.com

Received: 2013.03.08
Accepted: 2013.04.12
Published: 2013.06.10

ABSTRACT

Neural networks' synchronization by mutual learning discovered and described by Kanter et al. [12] can be used to construct relatively secure cryptographic key exchange protocol in the open channel. This phenomenon based on simple mathematical operations, can be performed fast on a computer. The latter makes it competitive to the currently used cryptographic algorithms. An additional advantage is the easiness in system scaling by adjusting neutral network's topology, what results in satisfactory level of security [24] despite different attack attempts [12, 15]. With the aid of previous experiments, it turns out that the above synchronization procedure is a stochastic process. Though the time needed to achieve compatible weights vectors in both partner networks depends on their topology, the histograms generated herein render similar distribution patterns. In this paper the simulations and the analysis of synchronizations' time are performed to test whether these histograms comply with histograms of a particular well-known statistical distribution. As verified in this work, indeed they coincide with Poisson distribution. The corresponding parameters of the empirically established Poisson distribution are also estimated in this work. Evidently the calculation of such parameters permits to assess the probability of achieving both networks' synchronization in a given time only upon resorting to the generated distribution tables. Thus, there is no necessity of redoing again time-consuming computer simulations.

Keywords: neural networks, neurocryptography.

INTRODUCTION

Neural networks represent a model of functioning living organisms' brains and human brain in particular. Currently available technologies allow simulating the work of merely relatively small networks in comparison to the real brain size. There is still ongoing work on increasing the scale of possible experiments. In particular, the work of IBM engineers on the creation and the use of chips operating like the brain [10] and The Human Brain Project [18] form the most advanced research topics within the discussed field. Functioning of artificial neural networks copying human brain, is based on processing of incoming signals and classifying them to one of the several groups. Each of the networks' input signals is amplified or reduced, accordingly by the corresponding weight value which determines the significance of a given input. Such classification mechanism is successfully applied to the recognition of bank or shop customers' behavior patterns or to analyze the results of medical examinations to discover patterns characteristic for some disease entities [17]. Before one uses neural network this network must first undergo a pertinent training process [21, 22]. In particular, the so-called supervised learning step of the feed-forward neural network consists of modifying the network's weights via specific optimization process. Algo-

rithms used for finding the optimal weights for a given neural network modify (via the updating procedure – see [8]) the corresponding values of multiple weights, which consequently leads to a very different output of the entire network. Additionally, the neural network is considered to be trained, if the number of misclassified input vectors falls below an arbitrarily admitted threshold. Thus upon completing the training phase, the network can still provide erroneous classification. Typically, the initial values of weights are determined randomly, and although the learning process is described by deterministic algorithm, the trained network depends on initial guesses for network's weights. In particular, for different values of optimal weights the same accuracy in final classification can still be reached. This can be easily seen in a very simple example of a network implementing a logical AND function. Figure 1 shows the separation of the input signal space for the AND function achieved by a single artificial neuron with selected two different sets of weights. The so-called decision boundaries representing two collections of weights are illustrated by red and green colors, respectively. The dotted black point (1.1) stands for the AND value equal to 1, while the remaining three encircled points correspond to the values equal to 0.

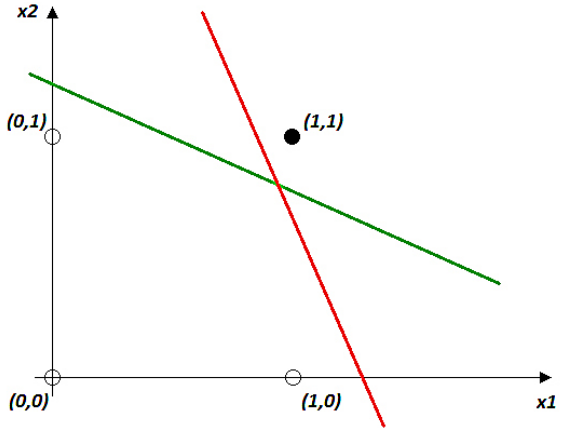

Fig. 1. The AND function realized by two weights sets

Evidently, in both cases this single neuron network classifies input vectors {(0,0), (0,1), (1,0), (1,1)} correctly, although the computed weights vary – each one is geometrically represented by different decision boundary (modulo k, $w=kw_1$, $k>1$). Additionally, for this particular example (forming the so-called linearly separable set of data), it is visible that we have infinitely many straight lines separating the above points. Thus, there are also infinitely many different weights for our single neuron network to classify the Boolean AND function. Such ambiguity yielding a correct network output, extends also to the general class of networks with more complex topologies. The existence of multiple correct weights in neural networks shifts them aside to the fringe of cryptographers' attention. The latter comes from the fact that in order to encrypt and to sign messages it is usually necessary for both sender and receiver to possess the key number with the same value. This problem can also be seen using newer neuro-fuzzy systems [3]. However, despite the above mentioned disadvantage, in certain applications neural networks are still applied as a sophisticated tool for crypto-analysis [9].

The research conducted in [11-15] indicates the possibility of using artificial neural network to create a secure cryptographic *key exchange protocol*. This work introduces specific conditions and modifications imposed on the network topology, weight values, networks learning procedure and finally on the activation function within the output neurons. Such modified network is called the TPM (*Tree Parity Machine*) – see e.g. [12, 13]. A characteristic fact for TPM is that in the process of mutual network training a static learning set is substituted by randomly generated input vectors. These restrictions make the so-far used methods to evaluate the accuracy of the network in question (by examining the error of the pertinent energy functions) inapplicable. There is no a priori given learning set, which could be used as reference for such analysis. The proposed key exchange protocol is based on the phenomenon of synchronization of the *mutual learning of neural networks*. The sender and receiver create networks with the same topology and start with randomly chosen different weights' values, which also remain confidential. In a sequel, both networks receive the same input vector and evaluate their outcome values, which are then exchanged. Sender's network treats the result of the recipient's network as the expected result, and in a similar fashion the recipient's network exploits the result of sender's network. In the next step both networks modify their weights in accordance with the pre-selected learning method. Commonly used methods coincide with Hebbian rule, Anti-Hebbian rule or with the Random Walk rule [8, 21-23]. In the subsequent step of the algorithm new input vectors common to both networks are randomly chosen.

As previously, both networks' results are calculated and mutually exchanged. The networks' weights are modified accordingly. Upon certain number of iterations of this procedure, two networks reach synchronization state, guaranteeing the same respective values of two collections of weights. The latter can be used directly as cryptographic keys, or as the seed of the algorithm that generates pseudo-random numbers, forming the role of respective keys [2, 13].

It is shown in [15], that bidirectional interaction between sender and receiver, which is achieved by exchange of TPM's outputs, allows faster synchronization than unidirectional network learning, which can perform a potential attacker. This difference in time needed to finish synchronization is crucial for security of the created key exchange protocol. Another strengthening of the proposed schema can be the best precise determination of the point at which TPM's are already synchronized, what in turn allows a quick termination of the learning process [4]. It makes it less susceptible for potential attack to occur by a third party. The latter holds as the time available for being attacked is reduced together with the amount of information potentially accessed by the attacker.

Computers and data stored on them are exposed to many attacks [6], and their protection is the main research area of cryptology. Cryptographic keys are numbers used in the algorithm as an additional input to the encryption and authentication of documents. In symmetric cryptography systems, the same key for encryption and decryption is applied. The security is based here on ensuring that the key is known only to the sender and receiver, who are both trusted parties. Asymmetric cryptography presents a different approach [20] in which both of these operations use a pair of keys. One of them is secret and refers to the private key and the other one, which is known, is coined as the public key. Using this system, first the sender retrieves the recipient's public key and encrypts the message. In sequel the receiver, having obtained the ciphertext, transforms it using his private own key to the plaintext. Asymmetric algorithms are slower in action. Therefore, in practice they are used for establishing the key that is applied in further communication with the aid of symmetric algorithms and also to encrypt small parts of data [25]. In addition, depending on various applications, the keys can be divided into different classes. Namely, they are applied to either encrypt messages, to decrypt cryptograms, to compute the digital signature, to verify signatures, to compute authentication code from messages, to verify this code and finally also to establish keys in further communication [1].

OBJECTIVE AND METHODOLOGY

Synchronization of the TPMs is a stochastic process, and the time needed to reach the same values of weights of the network with a given structure, depends on randomly selected initial weights and on randomly generated input vectors, respectively. In fact, the size of the network affects also the network synchronization time. Naturally, bigger TPMs synchronize longer. The simulation results [12] (see also figure 4) show that the distribution of a synchronization time for TPM networks with a given topology is asymmetric. Namely, the respective frequencies measuring how often both nets synchronize in a given number of steps are high on the left and skewed toward the right. In addition, the network's size does not affect the built-in distribution characteristics. The main task of this work is to determine the type of the observed distribution and its parameters for the network with different structures. A comparison of this generated distribution is accomplished here with the Poisson distribution, which is well-known and can be exploited by using e.g. standard distribution tables. The latter, permits in turn to determine the probability for TPMs to be synchronized in a given number of steps. Mathematical description of TPM synchronization with simple Poisson distribution permits to continue further theoretical research of this process, similarly to hte application mathematical models depicting various physical phenomena [16].

The simulations of networks' synchronization are carried out with different topologies by the authors' computer program. The obtained results of the synchronization measurements are analyzed with MS Excel. Due to the fact that the range of the analyzed sample is very high, it is divided into respective intervals. Therefore, the analysis of the TPMs' synchronization time is contained within a particular interval instead of specifying the probability of exact number of synchronization steps.

TREE PARITY MACHINE

Artificial neural network used in synchronization is similar to the tree-like structure with cer-

tain selected disjoint perceptron's receptive input fields. An example of a TPM network is shown in Figure 2. Here a feed-forward, multi-layer network, has in the output layer always only one perceptron. Alternatively, disjoint input fields can be linked with all neurons in the hidden layer but the unmarked connections between input impulses and first hidden neurons' layers should have weights always equal to zero.

TPM's hidden layer consist of K neurons, $K \in \mathbb{N}$. Each of them is built on the basis of the model of McCulloch-Pitts [19] with bipolar, step activation function, given by the following formula:

$$\sigma_{j,t} = f_t(\varphi_t) = \begin{cases} -1, if\ \varphi_t \leq 0 \\ 1, if\ \varphi_t > 0 \end{cases}$$

$$j = 1, 2, 3, ..., K.$$

The value of this function is the value of the output σ neuron j in time t. Argument φ_t is an adder block value at time t determined according to the formula:

$$\varphi_t = \sum_{i=1}^{N} w_{ki} x_{ki},$$

where N is the number of input impulses to a single neuron, x_i – input signals ($i \in [1; n]$, $n \in \mathbb{N}$), w_i – weights assigned to corresponding inputs ($i \in [1; n]$, $n \in \mathbb{N}$). Each weight belongs to the set $\{-L, -L + 1, ..., L - 1, L\}$.

TPM network structure can be thus expressed with three parameters as the network of type K-N-L. The last layer neuron performs the multiplication operation, and its outcome is the output of the entire network:

$$\tau = \prod_{j=1}^{K} \sigma_j$$

Learning TPM is in accordance with one of the three methods [23]:

1. Anti-Hebbian rule – weights are modified if the outputs of both networks are different. This process leads to network synchronization with opposite vectors of weights. Weights' modification complies here with the following formula:
$$w_{ki}^{(t+1)} = w_{ki}^{(t)} - x_{ki}\sigma_k .$$
This above mentioned method is originally used by Kanter et al. in [12]. However, as shown in [15] it is easier to apply a normal Hebbian rule, and the results generated by both methods are similar.
2. Hebbian rule – weights are modified if the TPM's results are equal. Then synchronization process ends with the same weights' values. Weights are modified now according to:
$$w_{ki}^{(t+1)} = w_{ki}^{(t)} + x_{ki}\sigma_k .$$
3. Random Walk rule – similar to Hebbian rule, modification occurs when the results of the networks are equals. The latter leads to equal weights' values for both nets. In this method, the weights' adjustment does not depend on the output of the hidden layer neuron, but only on the input signal:
$$w_{ki}^{(t+1)} = w_{ki}^{(t)} + x_{ki} .$$

If the new weight value $w_{ki}^{(t+1)}$ is greater than L, it is replaced by L. Analogously, if the weight

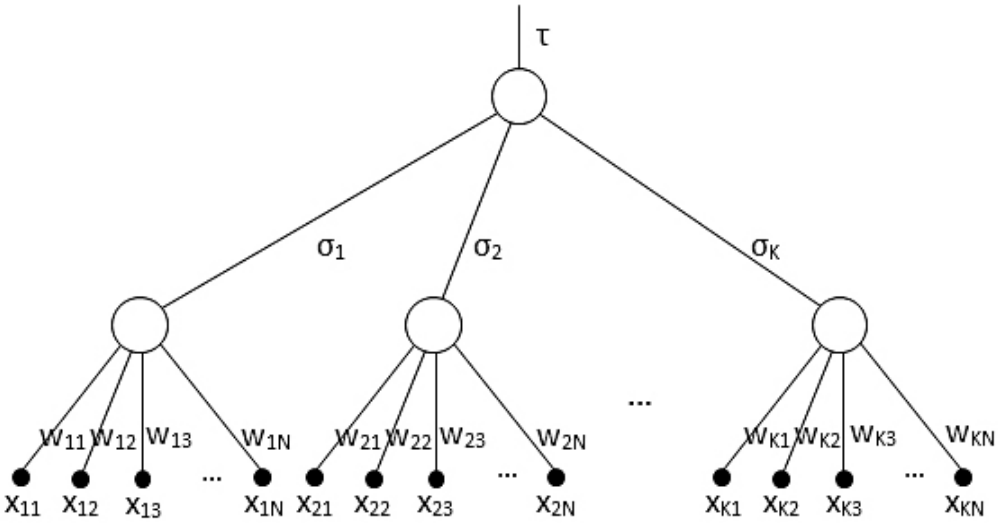

Fig. 2. Tree Parity Machine topology

value is less than -*L*, it's substituted with -*L*. Networks that reached the synchronization status are synchronized regardless of further learning time.

Random Walk rule modifies weights using only input vector, which is randomly chosen at each step of synchronization so the values obtained in this method are closer to uniform distribution as compared to the weights gained by using Hebbian method. Figure 3 shows a comparison of the distribution of weight after synchronization of 1000 TPMs of the structure 3–16–5 learned with Hebbian and Random walk rule.

The time required to achieve synchronization by the network depends on the initial values of weights and the random input vectors chosen in every step of the synchronization. A typical histogram showing the number of synchronized networks in a specified number of learning cycles is shown in Figure 4.

In this work we analyze 10 000 synchronizations for the networks of type 3-16-3. The shortest observed synchronization time (measured in the corresponding number of steps) is 46 steps, and the longest one is 1156, which gives the range of possible 1110 cycles. Average synchronization time equals 238 steps**.** The number of classes to create a histogram was counted using Huntsberger formula [5] $k = 1 + 3.32 \cdot \log 10000 = 14.28$, which is subsequently doubled for better graph readability.

RESULTS

The histograms showing the number of synchronized networks in a specified number of steps are similar for all networks with different parameters K-N-L. They all remind the histogram from Figure 4. The main difference is just the number of steps needed to synchronize TPMs.

For a network with the number of neurons in the hidden layer equal to $K = 3$ and with the respective number of input signals for each of them equal to $N = 16$, 1000 synchronizations with different values of $L \in \{1,2,\ldots,5,10,15,\ldots,50\}$ are carried out in our tests. A Random walk rule

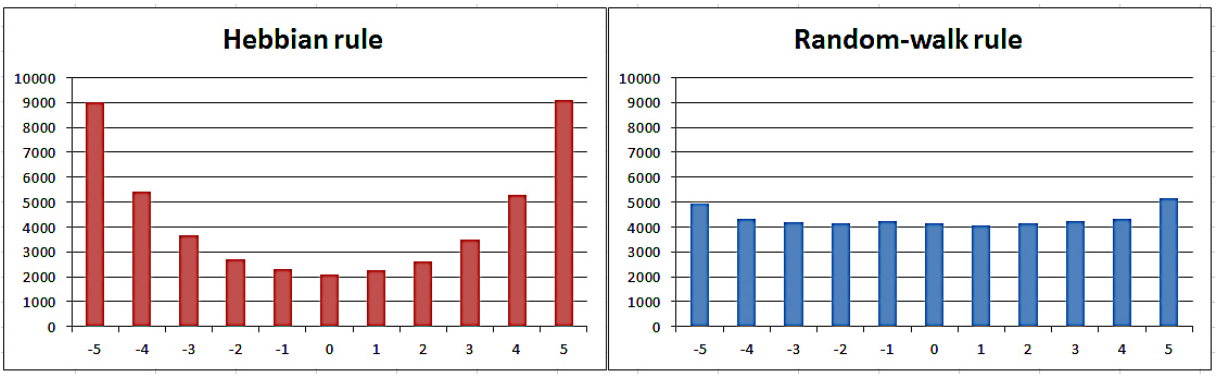

Fig. 3. Weights distribution for Hebbian and Random Walk rule

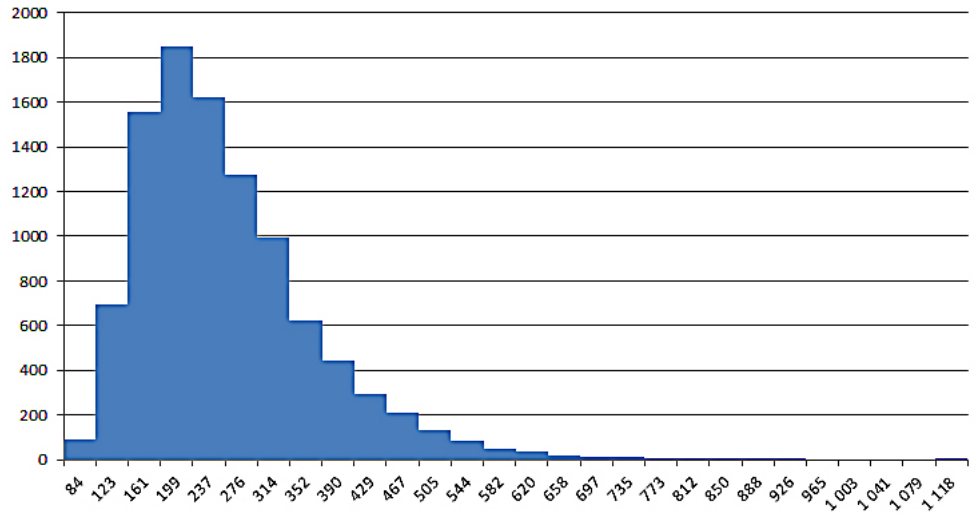

Fig. 4. Synchronization time histogram for TPM 3–16–3

used as a learning method is invoked here. Table 1 shows a summary of the numbers of steps needed for achieving synchronization: namely, the shortest and longest which are observed, and the average ones.

Table 1. Summary of the number of TPM's synchronization steps

TPM parameters	Min	Max	Average
3-16-1	7	132	33.4
3-16-2	37	394	118.3
3-16-3	80	702	264.2
3-16-4	160	1469	469.7
3-16-5	263	2849	755.6
3-16-10	1159	8579	3239.3
3-16-15	2991	20263	7657.9
3-16-20	5044	46057	14444.6
3-16-25	8242	62486	23033.2
3-16-30	14520	92569	34104.3
3-16-35	19209	124496	48443.2
3-16-40	23919	161440	64018.9
3-16-45	26191	251386	82656.0
3-16-50	42856	288219	102382.7

Given the above listed experimentally generated data, we determined next for each sample the range, the number of classes for histogram preparation, the width and the multiplicity of each of the classes' intervals. Dividing the multiplicity of a given class by the size of the sample, the empirical probabilities of TPM's synchronization in each time interval are consequently determined.

For TPM of type 3-16-2 our results are as follows:

1. Sample size $n = 1000$.
2. Range $R = 357$.
3. Number of intervals $k = 11$.
4. Interval size $d \approx 32.6$.

The corresponding boundaries of intervals, their multiplicity and network TPM 3-16-2 synchronization probabilities for the number of steps belonging to the interval are shown in Table 2.

Given the shape of the histogram, we conjecture the hypothesis that the Poisson distribution defined by the formula

$$P[X = k] = \frac{\lambda^k e^{-\lambda}}{k!}$$

is a function of the probability for TPM synchronization time. The parameter λ is the weighted average, which is calculated on the basis of empirical results using class number as a value and its multiplicity as weight. Hence, the sum of intervals' multiplicities is equal to the number of analyzed nets, so weighted average for this net is equal to $\lambda = 2{,}979$. In sequel, the above mentioned hypothesis is verified in compliance with chi-square test, in which the sum is calculated:

Table 2. TPM 3-16-2 synchronization time summary

Interval	Boundaries		Multiplicity	Probability
1	37.0	69.6	96	0.096
2	69.6	102.1	338	0.338
3	102.1	134.7	292	0.292
4	134.7	167.3	137	0.137
5	167.3	199.9	75	0.075
6	199.9	232.4	38	0.038
7	232.4	265.0	12	0.012
8	265.0	297.6	8	0.008
9	297.6	330.2	2	0.002
10	330.2	362.7	1	0.001
11	362.7	395.3	1	0.001

$$\chi^2 = \sum_{i=1}^{r} \frac{(O_i - E_i)^2}{E_i},$$

where r is the number of classes, O_i is the observed probability and E_i stands for the expected probability. Recall that the requirement for the chi-squared test is that the sample size is not less than 8 (see [7]). For this reason, the intervals 8–11 in Table 2 are merged to form a range of multiplicity 12 with the respective probability $0.012 = 0.008 + 0.002 + 0.001 + 0.001$. Finally, the number of classes reads as 8, and the sum $\chi^2 \approx 0{,}12$. The number of classes is 8, thus, the number of degrees of freedom is $7 = 8 - 1$. From the chi-square statistical distribution table, the critical value for 7 degrees of freedom and reliability level 0.999 reads as 0.59849. Since the sum equal to 0.12 is much lower than critical value 0.59849, even for restrictive level of reliability, there is no reason to reject the hypothesis. Hence the analyzed distribution is a Poisson distribution.

Similar statistical analysis is performed for other networks contained in Table 1. The results of such research are presented in Table 3. For each of the analyzed networks we specify here the observed minimal number of steps required to synchronize the network, the approximate width

Table 3. Chi-square test summary

TPM parameters	Min	Interval width	Interval count	Average (parameter λ)	Sum χ^2	Critical value
3-16-1	7	11.4	7	2.81	0.13	0.381
3-16-2	37	32.6	8	2.98	0.12	0.598
3-16-3	80	56.8	10	3.74	0.09	0.152
3-16-4	160	119.4	9	3.09	0.08	0.857
3-16-5	263	236	7	2.58	0.13	0.381
3-16-10	1159	677	9	3.57	0.06	0.857
3-16-15	2991	1576	9	3.47	0.04	0.857
3-16-20	5044	3742.1	8	3.00	0.13	0.598
3-16-25	8242	4949.3	10	3.48	0.08	1.152
3-16-30	14520	7121.3	9	3.24	0.07	0.857
3-16-35	19209	9606.5	10	3.55	0.09	1.152
3-16-40	23919	12547.5	9	3.70	0.09	0.857
3-16-45	26191	20547	8	3.27	0.15	0.598
3-16-50	42856	22387.1	8	3.16	0.08	0.598

of the interval and the number of intervals used in the chi-squared test. As for the TPM described above, the last intervals are collated in one group, in order to satisfy requirements of minimum multiplicity of elements in interval. The latter impacts on some variation in the number of intervals for the different networks. The following columns contain empirically determined weighted averages, which are used as parameter λ of the Poisson distribution, the sum of chi-square and its critical value for reliability level equals to 0.999. The resulting value of the sum is less than the value of the distribution χ^2 for all tabled levels of reliability. Thus the table includes the most restrictive level.

The generated data listed in Table 3 permits to answer the question, what is the probability for a given structure of TPMs to synchronize in a certain time interval. For example, let the network parameters be 3-16-4 and the question is: "What is the probability that this TPM synchronizes in 600 cycles of learning". First, one verifies, using the shortest time and the width of the interval, to which the interval belongs at a given time of synchronization. The first interval is here (160.274), and our chosen number 600 does not belong to this interval. The fourth interval coincides with (519.637) and contains the selected number 600, which is the searched length of the synchronization. Then, from the Poisson distribution tables the value for $k = 4$ and $\lambda = 3.09$ can be read, yielding a probability of 0.1728. This renders the answer to the above stated question. In addition, one can also calculate the cumulative probability value for $k \leq 4$. The latter yields the answer to the question what is the probability of network synchronization at up to 600 steps. It should be mentioned here that an alternative is to compute the probability of the opposite event by calculating the probability of network synchronization in more than 600 steps.

CONCLUSIONS

This work showed that the distribution of TPM network's synchronization time is a Poisson distribution with parameter λ, which can be estimated as an weighted average using the observed synchronization times. The outcomes of the chi-square test of conformity lead to the recognition of the empirical synchronization time distribution for the Poisson distribution at significance level equal to 0.999. The simulations and analysis performed herein allow determination of parameters λ for the networks in question with different structures in a simple tables. The results generated in the presented research permit further analysis of the TPM synchronization process based on the value of the Poisson distribution without necessity of undergoing long-term computer simulations and again performing the analysis of the obtained results.

REFERENCES

1. Barker E., Barker W., Burr W., Polk W., Smid M.: Recommendation for Key Management, Part 1: General (Revision 3). National Institute of Standards and Technology Special Publication 800-57, 2012.
2. Bisalapur S.: Design of an efficient neural key distribution center. International Journal of Artificial Intelligence & Applications, 2(1), 2011, 60–69.
3. Charlak M., Jakubowski M.: Porównanie systemów rozmytych i sztucznych sieci neuronowych. Advances in Science and Technology – Research Journal, 4, 2010, 54–64.
4. Dolecki M.: Tree Parity Machine synchronization time – statistical analysis. Труды БГТУ, Серия Математика, Физика, Информатика, 6(153), Mińsk, 2012, 149–151.
5. Gardiner V., Gardiner G.: Analysis of Frequency Distributions. Geo Abstracts, University of East Anglia, 1979.
6. Gil A., Karoń T.: Analiza środków i metod ochrony systemów operacyjnych. Advances in Science and Technology – Research Journal, 12, 2012, 149–168.
7. Greń J.: Statystyka matematyczna modele i zadania. Warszawa, 1978.
8. Hassoun M.: Fundamentals of Artificial Neural Networks. MIT Press, 1995.
9. Ibrachim S., Maarof M.: A review on biological inspired computation in cryptology. Jurnal Teknologi Maklumat, 17(1), 2005, 90-98.
10. Imam N., Cleland T., Manohar R., Merolla P., Arthur J., Akopyan F., Modha D.: Implementation of olfactory bulb glomerular-layer computations in a digital neurosynaptic core. Frontiers in Neuroscience, 6(83), 2012
11. Kanter I., Kinzel W.: The theory of neural networks and cryptography. Proceedings of the XXII Solvay Conference on Physics on the Physics of Communication, 2002, 631-644.
12. Kanter I., Kinzel W., Kanter E.: Secure exchange of information by synchronization of neural networks. Europhysics Letters, 57, 2002, 141-147.
13. Kinzel W., Kanter I.: Neural cryptography. Condmat 0208453, 2002.
14. Klein E., Mislovaty R., Kanter I., Ruttor A., Kinzel W.: Synchronization of neural networks by mutual learning and its application to cryptography. Advances in Neural Information Processing Systems, 17, MIT Press, Cambridge, 2005, 689-696.
15. Klimov A., Mityagin A., Shamir A.: Analysis of neural cryptography. In: Y. Zheng (ed.), Advances in Cryptology – ASIACRYPT 2002, Springer, 2003, 288-289.
16. Lenik K., Korga S.: FEM applications to model friction processes in plastic strain conditions. Archives of Materials Science and Engineering, 41, 2010, 121–124.
17. Lula P.: Sztuczne sieci neuronowe jako narzędzie analiz typu data mining. Data mining: metody i przykłady, StatSoft, 2002.
18. Markram M.: The Blue Brain Project. Nature Reviews Neuroscience 7, 2006, 153-160.
19. McCulloch W.: A logical calculus of the ideas immanent in nervous activity. Bulletin of Mathematical Biophysics, 5, 1943, 115-133.
20. Menezes A., Vanstone S., Van Oorschot P.: Handbook of Applied Cryptography, CRC Press, 1996
21. Osowski S.: Sieci neuronowe w ujęciu algorytmicznym, WNT, 1996.
22. Rutkowski L.: Metody i techniki sztucznej inteligencji, PWN, Warszawa 2006.
23. Ruttor A.: Neural Synchronization and Cryptography, PhD thesis, Wurzburg 2006.
24. Ruttor A., Kinzel W., Naeh R., Kanter I.: Genetic attack on neural cryptography, Phys. Rev. E, 73(3):036121, 2006.
25. Stokłosa J., Bilski T., Pankowski T.: Bezpieczeństwo danych w systemach informatycznych, PWN, Warszawa 2001.