Smart Crossover Mechanism for Parallel Neuroevolution Method of Medical Diagnostic Models Synthesis

Serhii Leoshchenko[1[0000-0001-5099-5518]], Sergey Subbotin[2[0000-0001-5814-8268]], Andrii Olinyk[3[0000-0002-6740-6078]], Viktor Lytvyn[4[0000-0003-4061-4755]] and Matviy Ilyashenko[5[0000-0003-4624-4687]]

[1,2,3,4] Dept. of Software Tools, National University "Zaporizhzhia Polytechnic", 69063 Zaporizhzhia, Ukraine
[5]Dept. of Computer Systems and networks, National University "Zaporizhzhia Polytechnic", 69063 Zaporizhzhia, Ukraine
[1]sergleo.zntu@gmail.com [2]subbotin@zntu.edu.ua
[3]olejnikaa@gmail.com [4]lytvynviktor.a@gmail.com
[5]matviy.ilyashenko@gmail.com

Abstract. Information technologies significantly expand the capabilities of modern medicine. Using of artificial neural networks is becoming particularly promising. They are actively used for diagnostics based on patient data. However, the problem of network synthesis with a satisfactory topology and accurate diagnosis remains important. Neuroevolution methods allow to solve this problem without much involvement of an expert. Moreover, these methods make it possible to effectively use the parallel power of modern computer systems. On the other hand, parallelization raises a number of new problems. The paper suggests mechanisms for improving the crossover operator. The proposed solution allows you to reduce resource consumption and improve the synthesis process.

Keywords: Medical Diagnosis, Forecasting, Neuroevolution, Synthesis, Adaptive Mechanism, Genetic Algorithm, Parallel Genetic Algorithm, Crossover.

1 Introduction

The human factor often causes a number of problems. This also applies, of course, to medicine. A doctor's mistake can mean the loss of a patient's health or even their life, and doctors make mistakes not infrequently. Even the highest-level professional can make mistakes, because the specialist can be tired, irritated, concentrating on the problem worse than usual [1-3].

In this case, information technologies can come to the rescue. For example, the IBM Watson cognitive system [4-7] copes with work in the medical field at a fairly high level (oncology, reading x-rays, etc.) [4], [5]. But there are other solutions proposed by independent researchers. A number of scientists regularly present successful results of using artificial intelligence systems in medical diagnostics [8-11].

Artificial neural networks (ANN) have come into practice wherever it is necessary to solve problems of forecasting, classification or management. Problems of medical diagnostics are a sub-division of problems classification and forecasting of the object's condition. The following reasons determine the impressive success of the use of ANNs [10-13]:

- a huge number of opportunities [12], [13]. ANNs are a powerful modeling method that allows reproducing extremely complex dependencies. In particular, neural networks naturally are nonlinear. In problems where linear application is unsatisfactory (most medical diagnostics problems), linear models do not work well. In addition, neural networks cope with modeling linear dependencies in the case of a large number of variables;
- leraining on examples (data about the object) [14], [15]. Neural networks are trained using examples. Initially, representative data is selected, and then a training algorithm is run that automatically perceives the data structure.

On the other hand, a user who trains a neural network needs a certain set of knowledge about how to select and prepare data, select the appropriate network architecture, and interpret the results [14-20]. Therefore, it can be assumed that the probability of error passes from the expert in knowledge of the problem area to the expert in the design of the ANN [16], [18].

Neuroevolution methods of ANN synthesis are based on an evolutionary approach [21]. These methods can simultaneously configure the network structure and weights. This allows getting ready-made neural network solutions with only data from the training sample, and in most cases does not require the user to have deep knowledge of the ANN theory [22], [23].

Advantages of using neuroevolution methods [24-27]:

- a wide variety of resulting topologies – possible solutions with" non-standard " ANNs structures;
- adaptivity;
- universality;
- the use does not require a deep knowledge about the ANN;
- possibility of using a parallel approach.

The most well-established neuroevolution method is the genetic algorithm (GA).

Unlike other optimization technologies, GA contain a population of trial solutions that are competitively managed using defined operators [28-30]. GA is inherent in iterative training of a population of individuals. The capacity of GA increases with the use of distributed computing. Such algorithms are called parallel genetic algorithms. They are based on splitting a population into several separate subpopulations, each of which will be processed by GA independently of other subpopulations.

However, when it comes to parallelizing, it should be taken into account that the main disadvantage of GA is a constant desire for a population containing one local optimum, which leads to a constant decrease in the genetic diversity of the population, which reduces the ability of GA to search for a global optimum and/or adapt to chang-

ing parameters of target functions [31], [32]. Therefore, the task of improving the methods of evolutionary optimization based on increasing the adaptive characteristics of the method is urgent.

2 Literature review

As noted above, GA is one of the most acceptable ways to synthesize ANN. This is due to the fact that at the initial stage there is absolutely no information about the direction of movement in terms of setting the matrix weights. Under conditions of uncertainty, evolutionary methods, including GA, have the highest chances to achieve the necessary results.

At the same time, there is a significant disadvantage of using genetic algorithms-the multi – iterative nature of the methods, serious time costs. To solve such problems, parallelization can be used to branch out the execution of calculations between the cores of modern computer systems. However, when designing a parallel approach, you should consider the most common problems encountered:

- selecting or developing a strategy for interaction between the components of the algorithm;
- choosing the frequency of migration between populations;
- determination of migrating individuals and their number;
- determining the structure of the evolution of individual populations.

Consider the problems in more detail. The structure of a parallel system is an important factor in the performance of a parallel algorithm, since it determines how quickly (or how slowly) a good solution spreads to other populations. If the system is strongly connected, then good solutions will quickly spread to all streams and can quickly "saturate" the population [33]. On the other hand, if the network is loosely connected, solutions will spread more slowly and threads will be more isolated from each other. Further parallel development and recombination of different solutions can occur to obtain potentially better solutions.

A common trend in promising parallel genetic algorithms (PGA) [34], [35] is the using of static system structures that are defined before the algorithm is run and remain unchanged.

Another method of constructing a structure is to create a dynamic system [33-35]. In this case, the flow is not limited to links with a certain fixed number of threads; instead, migrants are directed to threads that meet a certain criterion. As a similar criterion, the degree of population diversity or the measure of genotypic distance between two populations (or the distance from a characteristic individual of the population, for example, a favorite) is taken. This structure requires mechanisms for tracking events in neighboring populations, and if an event occurred in one of the neighboring populations, then an event should be expected in the second population.

The frequency of migrations also has a big impact on the final decision [33], [34]. As it known, too frequent migrations lead to degeneration of populations, and rare

ones, on the contrary, to a decrease in convergence. Various methods are used to regulate the frequency of migration, which can be divided into two types: adaptive and event-based. In the first case, adaptation methods are used to adjust the migration frequency during the algorithm operation. In the second case, methods are used that determine the need for migration, that is, migration is performed only when an event occurs.

Selection mechanisms are used to select individuals for migration [36-40]. It is known that individual chromosomes may contain "good" fragments of the genetic code, but these parts may be in chromosomes that are poorly adapted. But at the same time, excluding similar ones can lead to premature convergence, or skipping the global optimum.

Using different strategies imposes the main restriction: the necessity of forming the same type of chromosome structure. But the effect that is possible with successful formation can be much greater than when using a single GA structure in all populations [37].

It is also worth noting that a large number of migrations and even dynamic exchange of intermediate information between threads requires additional overhead, which sometimes largely negates the achievement of parallel execution of calculations [36], [38].

Moreover, we should not forget that even the sequential version of GA imposes significant resource requirements (especially RAM), compared to gradient methods [39], [40]. This is because in most cases a population of neural networks is used. Hence, when parallelizing, keep in mind that the requirements will increase according to the number of threads involved.

Therefore, the use of additional mechanisms and the introduction of hybrid methods, at different stages of operation, will improve the performance of the PGA.

3 Parallel genetic algorithm with smart crossover

The paper proposes a parallel genetic method. Parallelization will occur in the following way. Initially, we will generate a population of individuals on the main core of the system, where each individual is a separate neural network:

$$P = \{NN_1, NN_2, ..., NN_n\}, \tag{1}$$

where, P is population,

NN_n is neural network,

n is the size of population.

Also, at the initial stage, the main free parameters of the method will be set: the stop criteria, population size, and so on.

The next step is to divide the population into subpopulations and distribute subpopulations between the cores of the multi-core system.

The cores will perform the same sequence of actions with subpopulations: evaluating the genetic information of individuals, sorting and selecting the best individuals, crossing and selecting the new best individuals.

Then the selected best individuals are sent to the main core, where they are re-sorted and selected the best representatives. The new best individual is evaluated for the stop criterion. If the resulting individual is satisfied, it becomes a solution and the method ends. If the solution is not satisfactory, the best individuals from the subpopulations are crossed and a new population is obtained, which is re-distributed between the system cores.

Diversity is maintained in the process of evolution by the fact that each species (subpopulation) develops without exchanging genetic material with other species. This is an important aspect of this model. The exchange of genetic material between two different species usually produces non-viable offspring. In addition, mixing of genetic material can reduce the diversity of populations. This approach will also significantly reduce the share of overhead associated with the transfer of information between the system cores.

Now let's look at the mechanisms that allow optimizing the use of RAM, while maintaining the adaptive characteristics of the method. In the proposed method, it is recommended to use a smart crossover, which is based on uniform crossing and rank selection enhanced by criteria conditions.

Uniform crossover is one of the most effective recombination operators in standard GA [41], [42].

Uniform crossover is performed according to a randomly selected standard that specifies which genes should be inherited from the first parent (other genes are taken from the second parent) [42]. In other words the general rule of uniform crossover can be represented as follows:

$$\begin{aligned} &Ind_3 = Crossover(Ind_1, Ind_2, DataofCros); \\ &g_{Ind_3} = \{g_1 = Rand\left(g_{1Ind_1}, g_{1Ind_2}\right), \\ &g_2 = Rand\left(g_{2Ind_1}, g_{2Ind_2}\right), \ldots, \\ &g_i = Rand\left(g_{iInd_1}, g_{iInd_2}\right)\}. \end{aligned} \tag{2}$$

It has long been known that setting the probability of passing a parent gene to a descendant in a uniform crossover can significantly increase its efficiency [35], [36], and also allows emulating other crossing operators (single-point, two-point). It is also known that the use of the uniform crossing operator makes it possible to apply the so-called multi-parent recombination, when more than two parents are used to generate one offspring. Despite this, in most studies, only two parents are used and the fixed probability of transmitting the gene is 0.5 [41].

Even crossing gives greater flexibility when combining rows, which is an important advantage when working with GA.

However, it should be noted that even crossing requires additional computing power. On the other hand, uniform crossover makes it possible to emulate the operation of simpler types of crossing, such as two-point crossover. Therefore, we will use a two-

point crossover to work on threads. This approach will allow you to implement this method in the future on computing systems running graphics processors (GPUs).

It is proposed to strengthen the ranking selection by introducing additional criteria that will help to track various characteristics of neural networks more subtly, namely: excessive memory usage and approximation properties of the neural network.

The general scheme of the method is shown in Fig. 1.

4 Experiment

4.1 Description of the experiment

A sample of Parkinson's Disease Classification Data Set will be used for the experiment [43].

The data used in this study were gathered from 188 patients with PD (107 men and 81 women) with ages ranging from 33 to 87 (65.1Â±10.9). The data used in this study were gathered from 188 patients with PD (107 men and 81 women) with ages ranging from 33 to 87 (65.1Â±10.9) at the Department of Neurology in Cerrahpasa Faculty of Medicine, Istanbul University.

The control group consists of 64 healthy individuals (23 men and 41 women) with ages varying between 41 and 82 (61.1Â±8.9). During the data collection process, the microphone is set to 44.1 KHz and following the physician's examination, the sustained phonation of the vowel was collected from each subject with three repetitions. Table 1 shows the main characteristics of the data sample.

Table 1. Main characteristics of the Parkinson's Disease Classification Data Set

Criterion	Characteristic	Criterion	Characteristic
Data Set Characteristics	Multivariate	Number of Instances	756
Attribute Characteristics	Integer, Real	Number of Attributes	754

The following hardware and software have been used for experimental verification of the proposed parallel genetic method for ANN synthesis: the computing system of the Department of software tools of Zaporizhzhya national technical university (National university "Zaporizhzhia politechnic"), Zaporizhzhia: Xeon processor E5-2660 v4 (14 cores), RAM 4x16 GB DDR4, the programming model of Java threads.

The results of the proposed method will be compared with the results of the method considered in the previous works [44]. Old modification will be called PGM (Parallel genetic method) and new variant – PGM SC (Parallel genetic method with smart crossover). Note that when working, the size of the parent pool for even crossing will be equal to the number of system cores involved.

4.2 The results of the experiment

In the Fig. 2 is graph of the execution time (in minutes) of the proposed method on computer systems, which depends on the number of cores involved.

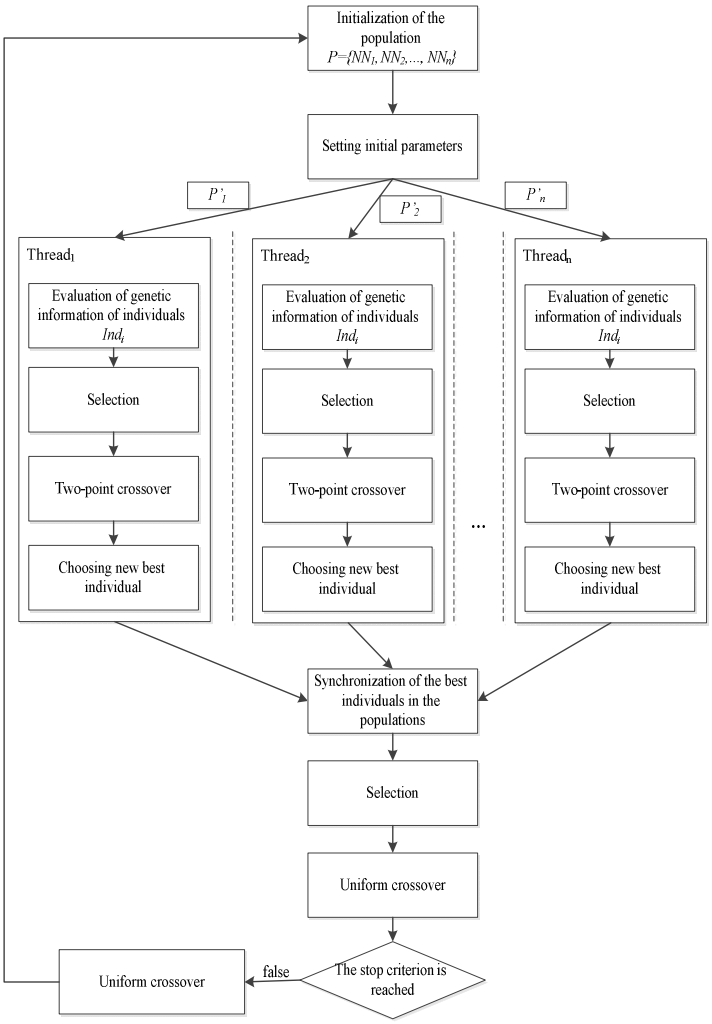

Fig. 1. Parallel genetic method for ANN synthesis with smart crossover

It can be seen from the graphs that the proposed method has an acceptable degree of parallelism and is effectively performed on MIMD parallel system. In addition, the processor in multi-core computer supports Turbo Boost technology [45–47], making the time of the method execution on the single core much less than on the core of which does not support this technology.

Fig. 3 shows graphs of changes in communication overhead. Since the new method offers the transfer of only the best individuals, this allows you to significantly reduce the transfer of excess information. This is explained by the fact that communication

overhead of the proposed method execution on computer systems is relatively small, and the number of parallel operations significantly exceeds the number of serial operations and synchronizations. In communication overhead, is understood the ratio of the time spent by the system for transfers and synchronization among cores to the time of target calculations on a given number of cores.

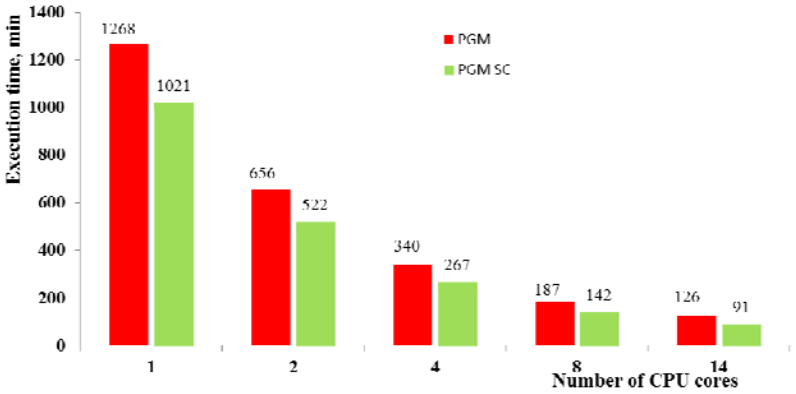

Fig. 2. Dependence the execution time of the proposed method to the number of involved cores of the computing system

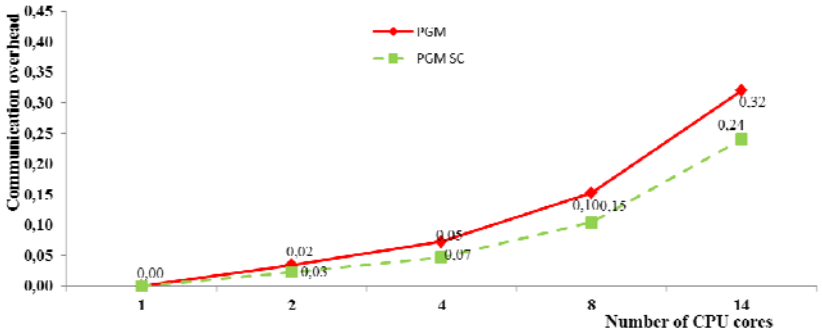

Fig. 3. Communication overhead performing the proposed method to the number of cores involved of the computing system

Fig. 4 shows graphs of speedup changes. Based on the fact that the communication overhead has decreased, the speedup of execution increases significantly.

The graph of efficiency of computer systems is presented in Fig. 5. It shows that the using of even 14 cores of computer systems for the implementation of the proposed method retains the efficiency at a relatively acceptable level and indicates the potential, if necessary and possibly, to use even more cores.

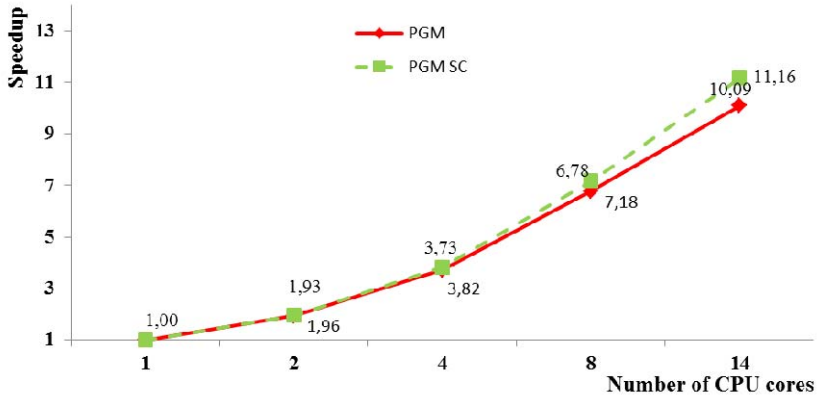

Fig. 4. The speedup graphics of calculations on computing system

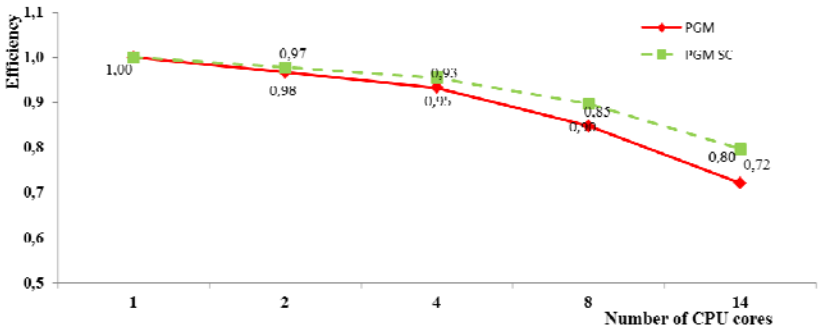

Fig. 5. The efficiency graph of computing systems when executing the proposed method

5 Conclusion

Based on the results of experiments, it can be argued that the proposed method can be used in the synthesis of neural network diagnostic models. The proposed smart crossover mechanism significantly optimizes the synthesis process by using an adjustable uniform crossover and additional criteria at the selection stage.

However, we are not talking about large-scale implementation of neural networks in hospitals around the world. The main problem in terms of the spread of these technologies is that neural networks are a kind of "black box". Specialists enter data and get a certain result. But the creators of such systems may not fully understand how this result was obtained, what algorithms and in what sequence are involved. If neural networks could be made more transparent, and the principle of their operation could be easily explained to medical practitioners, then the rate of spread of this technology would be much higher.

Acknowledgment

The work is supported by the state budget scientific research project of National University "Zaporizhzhia Polytechnic" “Intelligent methods and software for diagnostics and non-destructive quality control of military and civilian applications” (state registration number 0119U100360).

References

1. Einarson, T.R., Acs, A., Ludwig, C., Panton, U.H.: Prevalence of cardiovascular disease in type 2 diabetes: a systematic literature review of scientific evidence from across the world in 2007–2017. Cardiovasc Diabetol 17 (1), 1-19 (2018). https://doi.org/10.1186/s12933-018-0728-6
2. Petrie, J.R., Guzik, T.J., Touyz, R.M.: Diabetes, Hypertension, and Cardiovascular Disease: Clinical Insights and Vascular Mechanisms. The Canadian journal of cardiology 34(5), 575-584 (2017). https://doi.org/10.1016/j.cjca.2017.12.005
3. Jungen, C., Scherschel, K., Eickholt, C., Kuklik, P., Klatt, N.: Disruption of cardiac cholinergic neurons enhances susceptibility to ventricular arrhythmias. Nature Communications 8 (2017). https://doi.org/10.1038/ncomms14155
4. Bakkar, N., Kovalik, T., Lorenzini, I. et al.: Artificial intelligence in neurodegenerative disease research: use of IBM Watson to identify additional RNA-binding proteins altered in amyotrophic lateral sclerosis. Acta Neuropathol 135, 227–247 (2018). https://doi.org/10.1007/s00401-017-1785-8
5. Gomes, C., Dietterich, T., Barrett, C. et al.: Computational sustainability: computing for a better world and a sustainable future. Communications of The ACM 62 (9), 56–65 (2019). https://doi.org/10.1145/3339399
6. Kolpakova, T., Oliinyk, A., Lovkin, V.: Improved method of group decision making in expert systems based on competitive agents selection. UKRCON: IEEE First Ukraine Conference on Electrical and Computer Engineering (UKRCON), 939–943. KPI, Kiyv (2017). https://doi.org/10.1109/UKRCON.2017.8100388
7. IBM Watson Homepage, https://www.ibm.com/watson
8. Pranav, R., Anirudh, J., Anuj, P. et al.: CheXpedition: Investigating Generalization Challenges for Translation of Chest X-Ray Algorithms to the Clinical Setting (2020), https://arxiv.org/pdf/2002.11379.pdf
9. Artificial intelligence rivals radiologists in screening X-rays for certain diseases, https://med.stanford.edu/news/all-news/2018/11/ai-outperformed-radiologists-in-screening-x-rays-for-certain-diseases.html
10. Stanford researchers develop artificial intelligence tool to help detect brain aneurysms, https://news.stanford.edu/2019/06/07/ai-tool-helps-radiologists-detect-brain-aneurysms/
11. Leoshchenko S, Oliinyk A., Subbotin, S, Zaiko T: Using modern architectures of recurrent neural networks for technical diagnosis of complex systems. In: Proceedings of the 2018 international scientific-practical conference problems of infocommunications. science and technology (PIC S&T), 411–416. IEEE, Kharkov (2018). https://doi.org/10.1109/infocommst.2018.8632015
12. Schmidt, J., Marques, M.R.G., Botti, S. et al.: Recent advances and applications of machine learning in solid-state materials science. npj Comput Mater 5, 83 (2019). https://doi.org/10.1038/s41524-019-0221-0

13. Bradley, J.R., Holan, S.H., Wikle, C.K.: Multivariate spatio-temporal models for high-dimensional areal data with application to Longitudinal Employer-Household Dynamics. The Annals of Applied Statistics, 9(4), 1761-1791 (2015).
14. A Tour of Machine Learning Algorithms, https://machinelearningmastery.com/a-tour-of-machine-learning-algorithms/
15. Sabater-Mir, J., Torra, V., Aguiló, I., González-Hidalgo, M.: Artificial Intelligence Research and Development. IOS Press, Amsterdam (2019).
16. Scher, S., Messori, G.: Generalization properties of feed-forward neural networks trained on Lorenz systems. Nonlinear Processes in Geophysics 26, 381–399 (2019).
17. How to Scale Data for Long Short-Term Memory Networks in Python, https://machinelearningmastery.com/how-to-scale-data-for-long-short-term-memory-networks-in-python/
18. Learning process of a neural network, https://towardsdatascience.com/how-do-artificial-neural-networks-learn-773e46399fc7
19. Oliinyk, A.O., Zayko, T.A., Subbotin, S.O.: Synthesis of Neuro-Fuzzy Networks on the Basis of Association Rules. Cybern Syst Anal 50, 348–357 (2014). https://doi.org/10.1007/s10559-014-9623-7
20. Oliinyk, A.O., Skrupsky, S.Y., Subbotin, S.A.: Using parallel random search to train fuzzy neural networks. Aut. Control Comp. Sci. 48, 313–323 (2014). https://doi.org/10.3103/S0146411614060078
21. Stanley, K.O., Clune, J., Lehman, J. et al.: Designing neural networks through neuroevolution. Nature Machine Intelligence 1, 24–35 (2019). https://doi.org/10.1038/s42256-018-0006-z.
22. Baldominos, A., Saez, Y. & Isasi, P.: On the automated, evolutionary design of neural networks: past, present, and future. Neural Comput & Applic 32, 519–545 (2020). https://doi.org/10.1007/s00521-019-04160-6
23. Gaier, A., Asteroth, A., Mouret, J.-B.: Data-efficient Neuroevolution with Kernel-Based Surrogate Models. GECCO '18: Proceedings of the Genetic and Evolutionary Computation Conference, 85–92 (2018). 10.1145/3205455.3205510.
24. Albrigtsen, S.I., Imenes A.: Neuroevolution of Actively Controlled Virtual Characters. University of Agder, Kristiansand & Grimstad (2017).
25. Bohrer, J., Grisci, B., Dorn, M.: Neuroevolution of Neural Network Architectures Using CoDeepNEAT and Keras (2020), https://arxiv.org/abs/2002.04634
26. Phillips B.: Deep Neuroevolution: Genetic Algorithms are a Competitive Alternative for Training Deep Neural Networks for Reinforcement Learning – Paper Summary, https://towardsdatascience.com/deep-neuroevolution-genetic-algorithms-are-a-competitive-alternative-for-training-deep-neural-822bfe3291f5
27. Gradient descent vs. neuroevolution, https://towardsdatascience.com/gradient-descent-vs-neuroevolution-f907dace010f
28. García-Martínez C., Rodriguez F.J., Lozano M. (2018) Genetic Algorithms. In: Martí R., Pardalos P., Resende M. (eds) Handbook of Heuristics. Springer, Cham
29. Lee, W.-P., Hsiao, Y.-T., Hwang, W.-C.: Designing a parallel evolutionary algorithm for inferring gene networks on the cloud computing environment. BMC Systems Biology 8(1), 23-28 (2014). https://doi.org/10.1186/1752-0509-8-5.
30. Gonçalves, I., Gomes, A., Henggeler, C.: Optimizing residential energy resources with an improved multi-objective genetic algorithm based on greedy mutations. GECCO '18: Proceedings of the Genetic and Evolutionary Computation Conference, 1246–1253 (2018). https://doi.org/10.1145/3205455.3205616

31. Zhang, F., Mei, Y., Zhang, M.: A two-stage genetic programming hyper-heuristic approach with feature selection for dynamic flexible job shop scheduling. GECCO '19: Proceedings of the Genetic and Evolutionary Computation Conference, 347–355 (2019). https://doi.org/10.1145/3321707.3321790
32. Stork, J., Eiben, A., Bartz-Beielstein, T.: A new Taxonomy of Continuous Global Optimization Algorithms (2018), https://arxiv.org/pdf/1808.08818.pdf
33. Potuzak, T.: Optimization of a genetic algorithm for road traffic network division using a distributed/parallel genetic algorithm. 2016 9th International Conference on Human System Interactions (HSI), 2016, pp. 21-27. https://doi.org/10.1109/HSI.2016.7529603
34. Hoseini Alinodehi, S.P., Moshfe, S., Saber Zaeimian, M., et al.: High-Speed General Purpose Genetic Algorithm Processor. IEEE Transactions on Cybernetics 46(7), 1551–1565, (2016). https://doi.org/10.1109/TCYB.2015.2451595
35. Hou, N., He, F., Zhou, Y., Chen, Y., Yan, X.: A Parallel Genetic Algorithm With Dispersion Correction for HW/SW Partitioning on Multi-Core CPU and Many-Core GPU. IEEE Access 6, 883–898 (2018). https://doi.org/10.1109/ACCESS.2017.2776295
36. Varun Kumar, S.G., Panneerselvam, Dr.R.: A Study of Crossover Operators for Genetic Algorithms to Solve VRP and its Variants and New Sinusoidal Motion Crossover Operator. International Journal of Computational Intelligence Research 13(7), 1717–1733 (2017).
37. Hassanat, A., Alkafaween, E.: On Enhancing Genetic Algorithms Using New Crossovers, https://arxiv.org/ftp/arxiv/papers/1801/1801.02335.pdf
38. Umbarkar, Dr.A., Sheth, P.: Crossover operators in genetic algorithms: a review. ICTACT Journal on Soft Computing 6(1), 1083–1092 (2015). https://doi.org/6. 10.21917/ijsc.2015.0150.
39. Oliinyk, A.O., Skrupsky, S.Y., Subbotin, S.A.: Experimental investigation with analyzing the training method complexity of neuro-fuzzy networks based on parallel random search. Aut. Control Comp. Sci. 49, 11–20 (2015). https://doi.org/10.3103/S0146411615010071
40. Leoshchenko, S., Oliinyk, A., Skrupsky, S., Subbotin, S., Zaiko T.: Parallel Method of Neural Network Synthesis Based on a Modified Genetic Algorithm Application. Workshop Proceedings of the 8th International Conference on "Mathematics. Information Technologies. Education", MoMLeT&DS-2019, 11–23. Lviv Polytechnic National University, Lviv (2019). https://dblp.org/rec/conf/momlet/LeoshchenkoOSSZ19
41. Baeck, T., Fogel, D.B., Michalewicz, Z.: Evolutionary Computation 1: Basic Algorithms and Operators. CRC Press, Boca Raton (2000).
42. Das, A.K., Pratihar, D.K. A directional crossover (DX) operator for real parameter optimization using genetic algorithm. Appl Intell 49, 1841–1865 (2019). https://doi.org/10.1007/s10489-018-1364-2
43. Parkinson's Disease Classification Data Set, https://archive.ics.uci.edu/ml/datasets/ Parkinson%27s+Disease+Classification
44. Leoshchenko, S., Oliinyk, A., Skrupsky, S., Subbotin, S., Lytvyn V.: Parallel Genetic Method for the Synthesis of Recurrent Neural Networks for Using in Medicine. In: Proceedings of the Second International Workshop on Computer Modeling and Intelligent Systems (CMIS-2019), pp. 1-17 (2019). https://dblp.org/rec/conf/cmis/LeoshchenkoOSSL19
45. Intel Turbo Boost Technology 2.0, https://www.intel.com/content/www/us/en/architecture-and-technology/turbo-boost/turbo-boost-technology.html
46. Oliinyk, A., Subbotin, S., Lovkin, V., Leoshchenko, S., Zaiko, T. Feature Selection Based on Parallel Stochastic Computing. 13th International Scientific and Technical Conference

on Computer Sciences and Information Technologies (CSIT), 347-351. IEEE, Lviv (2018). https://doi.org/10.1109/STC-CSIT.2018.8526729.
47. Shkarupylo, V., Skrupsky, S., Oliinyk, A., Kolpakova, T.: Development of stratified approach to software defined networks simulation. Eastern-European Journal of Enterprise Technologies, 5. 67-73 (2017). https://doi.org/10.15587/1729-4061.2017.110142.