Implementation of swarm intelligence methods for preprocessing in neuroevolution synthesis

Serhii Leoshchenko[a], Andrii Oliinyk[a], Sergey Subbotin[a] and Tetiana Kolpakova[a]

[a] *National University “Zaporizhzhia Polytechnic”, Zhukovskogo street 64, Zaporizhzhia, 69063, Ukraine*

Abstract

Using swarm intelligence techniques for data preprocessing over neuroevolution synthesis of artificial neural networks (ANNs) can provide a number of advantages. Therefore, swarm analysis techniques such as particle swarm optimization (PSO) or ant colony optimization (ACO) can effectively identify the most relevant traits from multidimensional data. This reduces the dimension of the input space, mitigating the Curse of dimension and improving the effectiveness of ANNs training. In addition, swarm analysis methods can filter out noisy or mismatched data points and detect outliers, increasing ANNs resistance to noisy input data and expanding generalization capabilities.

In general, the introduction of swarm intelligence techniques for data preprocessing prior to neuroevolution ANNs synthesis results in improved model performance, reduced computational complexity, and improved generalization capabilities, making it an appropriate approach in machine learning tasks.

This paper is proposed to consider the implementation of swarm intelligence methods for preprocessing to improve the performance of neuromodel synthesis.

Keywords 1

Preprocessing, swarm intelligence, neurosynthesis, neuromodels, machine learning, artificial neural networks

1. Introduction

Today, ANN-based neuromodels are widely used to automate and solve many processes and tasks of human activity (industry, medicine, operational processes, etc.). ANNs are computational models based on the structure and functioning of biological neurons in the human brain. They consist of interconnected nodes (neurons) organized in layers, with each neuron performing simple calculations and transmitting signals to connected neurons [1].

Let's look at how ANNs are used in diagnostic tasks.

1. Medical image analysis: Anns are used to analyze medical images such as X-rays, magnetic resonance imaging, computed tomography, and histopathological images. Convolutional neural networks (CNNs), a special type of deep ANNs designed for processing spatial data, are particularly effective in tasks such as tumor detection, organ segmentation, and pathology classification. CNNs can automatically study image features, providing accurate diagnostics and anomaly detection [1].
2. Disease diagnosis: Anns are used to diagnose various diseases by analyzing patient data such as symptoms, medical history, laboratory tests, and genetic information. Recurrent neural networks (RNNs) and long-term short-term memory networks (LSTMs), thanks to their structural features (the presence of feedback and gates, which allows to store the context/history of states), are able to

The Seventh International Workshop on Computer Modeling and Intelligent Systems (CMIS-2024), May 3, 2024, Zaporizhzhia, Ukraine
EMAIL: sergleo.zntu@gmail.com (S. Leoshchenko); olejnikaa@gmail.com (A. Oliinyk); subbotin@zntu.edu.ua (S. Subbotin); t.o.kolpakova@gmail.com (T. Kolpakova)
ORCID: 0000-0001-5099-5518 (S. Leoshchenko); 0000-0002-6740-6078 (A. Oliinyk); 0000-0001-5814-8268 (S. Subbotin); 0000-0001-8307-8134 (T. Kolpakova)

process sequential data, used for tasks such as predicting disease progression, identifying risk factors, and diagnosing conditions such as heart disease, diabetes, and cancer [1].

3. Medical signal processing: Anns are used to process physiological signals such as an electrocardiogram (ECG), electroencephalogram (EEG), and electromyogram (EMG). They are used for tasks such as detecting arrhythmias, predicting seizures, classifying sleep stages, and analyzing muscle activity. Given the nature of the input data that is the result of clinical trials, models that best represent time series are used, in particular the RNNs and LSTM networks, which perfectly capture time dependencies and patterns in sequential data, making them suitable for analyzing medical signals [2].
4. Drug discovery and development: Anns are used in drug discovery and development processes, including virtual screening, molecular modeling, and drug toxicity prediction. They are used to analyze chemical structures, predict drug-target interactions, and develop new compounds with desired pharmacological properties. ANN-based models help speed up the drug search process and reduce the time and cost associated with developing new drugs [2].
5. Predictive and predictive modeling: ANNs are used to build predictive and predictive models in healthcare that help clinicians make informed decisions about patient management and treatment strategies. These models predict outcomes such as disease progression, treatment response, and patient survival based on clinical data, biomarkers, and other relevant factors [3].

Overall, ANNs are versatile diagnostic tools that offer the ability to analyze different types of data, extract meaningful patterns, and make accurate predictions, thereby improving medical decision-making and patient care.

Preprocessing refers to a set of methods applied to raw data before it is passed to a machine learning model, such as an ANN. These methods are aimed at converting data to a format that is more appropriate to the chosen training method, and at improving the overall performance and effectiveness of the model. Preprocessing includes steps such as data cleaning, normalization, object scaling, dimensionality reduction, and processing missing values or outliers [4].

Pretreatment is crucial for Ann synthesis and training for several reasons.

1. Normalization and scaling: Anns often work better when input objects are normalized or scaled to a similar range. Preprocessing techniques, such as minimum-maximum scaling or Z-score normalization, ensure that features make an equal contribution to the learning process, preventing certain features from dominating the learning process due to differences in scale [5].
2. Preprocessing methods allow you to process missing values or outliers in a data set. Imputation methods can be used to fill in missing values, while outlier detection and removal methods help ensure that extreme values do not have an excessive impact on model behavior [5].
3. Object design and selection: preprocessing allows for object design and selection, which removes irrelevant or redundant objects and creates new objects to better represent basic patterns in the data. This helps improve Model generalization and reduce overtraining [5].
4. Dimensionality reduction: large-size data can create problems for ANNs in terms of computational complexity and retraining. Dimensionality reduction techniques, such as Principal Component Analysis (PCA) or feature selection algorithms, help reduce the number of input measurements while preserving as much information as possible, thereby improving the efficiency and speed of the model [5].
5. Data balancing: classification problems with unbalanced class distributions can use preprocessing techniques such as resampling or understated sampling to balance the data set, ensuring that the model learns from representative samples of each class [5].

In general, preprocessing plays a crucial role in preparing data for Ann synthesis and training, ensuring that the model assimilates data efficiently, generalizes well to invisible instances, and provides reliable and accurate predictions.

Swarm intelligence is an area of research inspired by the collective behavior of decentralized, self-organizing systems in nature, such as ant colonies, flocks of birds, and flocks of fish. Swarm intelligence techniques aim to solve complex problems by coordinating the actions of multiple agents, each following simple rules, without the need for centralized control or global knowledge.

Some key characteristics of swarm intelligence include [6].

1. Self-organization: swarm-based intelligence systems exhibit emerging behavior where global patterns and solutions arise from local interactions between individual agents without explicit coordination.
2. Swarm intelligence relies on distributed decision-making, where each agent makes decisions based on local information and interaction with nearby neighbors, rather than relying on centralized control or external leadership.
3. Adaptation and reliability: swarm intelligent systems are often adaptive and resistant to environmental changes or disturbances. They can self-adjust environments and reorganize in response to dynamic conditions, providing stability and flexibility.
4. Research and Operation: Swarm Intelligence techniques combine exploration of the search space to identify new solutions and use known solutions to effectively optimize performance.

Swarm intelligence techniques can be applied to neurosynthesis, which involves computer-aided design or synthesis of artificial neural networks (ANNs) using optimization techniques. These methods can help you explore the wide range of possible neural network architectures, optimize network parameters, and improve network performance. Some uses of swarm intelligence techniques for neurosynthesis include [7]:

1. Self-organization: swarm-based intelligence systems exhibit emerging behavior where global patterns and solutions arise from local interactions between individual agents without explicit coordination.
2. Swarm intelligence relies on distributed decision-making, where each agent makes decisions based on local information and interaction with nearby neighbors, rather than relying on centralized control or external leadership.
3. Adaptation and reliability: swarm intelligent systems are often adaptive and resistant to environmental changes or disturbances. They can self-adjust environments and reorganize in response to dynamic conditions, providing stability and flexibility.
4. Research and Operation: Swarm Intelligence techniques combine exploration of the search space to identify new solutions and use known solutions to effectively optimize performance.

5. Swarm intelligence techniques can be applied to neurosynthesis, which involves computer-aided design or synthesis of artificial neural networks (ANNs) using optimization techniques.

These methods can help you explore the wide range of possible neural network architectures, optimize network parameters, and improve network performance. Some uses of swarm intelligence techniques for neurosynthesis include [8]:

6. Swarm intelligence techniques such as particle swarm optimization (PSO), ant colony optimization (ACO), or genetic algorithms (GAs) can be used to find optimal neural network architectures by exploring the space of possible configurations, including the number of layers, neuron types, connectivity patterns, and more. hyperparameters.
7. Swarm intelligence techniques can optimize neural network hyperparameters, such as learning speed, activation functions, regularization parameters, and network topology, to improve performance and generalization.
8. Swarm intelligence algorithms can help optimize the learning process of neural networks by tweaking parameters related to optimization algorithms (such as gradient descent options), convergence criteria, and data preprocessing techniques, resulting in faster convergence and improved performance.
9. Ensemble training: swarm intelligence can be used to optimize the creation of ensemble models that train and combine multiple neural networks to improve prediction accuracy and reliability.

Overall, swarm intelligence techniques offer powerful neurosynthesis tools to automate the design and optimization of neural networks for a variety of applications, including pattern recognition, classification, regression, and management tasks [8].

2. Related Works

Let's analyze why preprocessing of input data is necessary for neuroevolution synthesis of analytical models based on ANN and what such processes are reduced to [9].

Normalization and scaling. Preprocessing techniques such as normalization and scaling ensure that the input characteristics are at the same scale. This is important because neural networks can be sensitive to the magnitude of input values. Data normalization prevents certain functions from dominating the training process solely because of their scale, resulting in more stable and effective learning [9].

Filling in missing values and monitoring emissions. Preprocessing allows you to process missing values and outliers in input data. Missing values can be calculated using methods such as calculating the Mean, median calculation, or interpolation. Outliers, if left untreated, can negatively affect the learning process, distorting the studied model parameters. Preprocessing techniques such as outlier detection and removal ensure that the neural network learns from clean and reliable data, resulting in improved performance [9].

Data reduction and feature selection. Preprocessing allows you to develop and select objects, while removing irrelevant or redundant objects and creating new informative objects. This process helps reduce the size of the input space by focusing the neural network's attention on the most relevant functions and improving its ability to generalize invisible data [10].

Data balancing. In classification problems with unbalanced class distribution, preprocessing methods can balance a data set by over-sampling minority classes, under-sampling majority classes, or using more advanced methods such as the Synthetic Minority over-sampling Technique (SMOTE). Data set balancing ensures that the neural network is trained on a representative set of samples from each class, preventing it from shifting to the majority class [10].

Noise reduction and improved data quality: preprocessing techniques can help filter out noise and improve the overall quality of input data. This is especially important in real data sets that may contain inappropriate or erroneous information. Noise removal ensures that the neural network focuses on basic data patterns, resulting in more accurate and reliable predictions [10].

Overall, input preprocessing plays a crucial role in neuroevolution synthesis, ensuring that the neural network learns from high-quality, well-structured data. This contributes to more efficient learning, faster convergence, and better generalization performance, which ultimately leads to more reliable and accurate neural network models [11].

Also, when studying the very practical implementation of input preprocessing, it should be noted that such processes can help save time during neurosynthesis as follows.

Obviously, preprocessing techniques such as selecting or extracting objects can reduce the dimension of input data by eliminating non-essential or redundant objects. With fewer input parameters, the neuroevolution process requires less computational resources and time to study the reduced feature space, which leads to faster processing [11].

Normalizing or scaling the input data ensures that all functions are on the same scale, preventing certain functions from dominating the learning process based solely on their size. Normalized data contributes to more efficient learning convergence by reducing the time required for the neuroevolution synthesis process [11].

Preprocessing methods for handling missing values and outliers ensure that the input data is clean and reliable. By removing or attributing missing values and outliers, the neuroevolution process can focus on learning based on high-quality data, resulting in faster convergence and more efficient use of computational resources.

In general, pretreatment of input data before neuroevolution synthesis helps optimize the learning process, increases the efficiency of model training, and reduces the overall processing time required to achieve the desired results [12].

Let's take a look at some of the most popular methods of swarm intelligence.

PSO method:

- PSO is inspired by the social behavior of flocks of birds or schools of fish;
- each candidate solution (particle) in the search space adjusts its position based on its own experience and the most known position of its neighbors;
- PSO is relatively easy to implement and can efficiently search in large-dimensional spaces;
- the method tends to approach local optima quickly, but may have difficulty going beyond local optima in multi-modal or deceptive landscapes.

ACO method [13], [14]:

- ACO is inspired by the search behavior of ants.
- artificial ants leave pheromone traces along the edges of the graph representing the problem space, and the intensity of pheromone traces reflects the desirability of the path;
- ACO is excellent in combinatorial optimization problems and is particularly effective in solving problems with discrete and undifferentiated objective functions;
- the method requires careful parameter adjustment and may suffer from slow convergence, especially in large and complex problem spaces.

Firefly algorithm (FA) [13], [14]:

- FA is inspired by the flashing behavior of fireflies, when brighter fireflies attract others;
- the method optimizes the set of solutions by iteratively moving brighter solutions to brighter ones in the search space;
- FA is easy to implement and requires no gradient information, making it suitable for optimization tasks with complex and multi-modal landscapes.;
- however, in some cases, FA can suffer from slow convergence and premature convergence, especially in multidimensional and nonlinear optimization problems.

Each of these swarm analysis methods has its own strengths and weaknesses, which makes them suitable for different types of optimization problems and subject areas. The choice of method depends on factors such as problem complexity, search space characteristics, computational resources, and desired convergence properties [13], [14].

Let's compare the most well-known methods of swarm intelligence according to the following criteria:

- basic idea: this criterion describes the natural phenomenon or behavior that inspired the development of each method. Understanding a biological or natural process can provide insight into how the method works and its suitability for various optimization tasks [15], [16];
- objective function: refers to the type of optimization problems that each method is primarily designed to solve. Some methods are better suited for continuous optimization problems, while others are excellent for combinatorial or discrete optimization problems [15], [16];
- aggregate initialization: this criterion describes how the initial set of candidate Solutions is generated in each method. Random initialization is common, but some methods may have specific initialization strategies [15], [16];
- environment vs. exploitation research: reflects the balance between research (finding new areas of the solution space) and exploitation (refining known promising solutions). Different methods may show different trends in research or exploitation [17];
- communication mechanism: this criterion refers to how information is exchanged between individuals in a swarm. This may include mechanisms such as pheromone traces in ACO or brightness-based attraction in FA [17];
- convergence properties: describes the speed and efficiency with which a method comes to a solution. So, for example, some methods can converge quickly, but run the risk of getting stuck in local optimums, while others can converge more slowly, but with better global optimization properties [17];
- this criterion evaluates how sensitive the method is to the choice of metaparameters. Methods that are less sensitive can be easier to configure and more reliable in various problem areas;
- implementation complexity: it reflects the ease of implementation and computational complexity of each method. Methods with simpler implementations may require less computing resources;
- areas of application: this criterion defines the types of optimization problems that each method is usually applied to. Understanding typical application areas can help you choose the most appropriate method for a particular task [18]-[21].

In the table. 1 shows the results of comparison by criteria.

Table 1
Results of comparing PSO, ACO and FA

Criteria	PSO	ACO	FA
Nature Inspiration	Social behavior of bird flocks or fish schools	Foraging behavior of ants	Flashing behavior of fireflies
Objective Function	Continuous optimization problems	Combinatorial optimization problems	Continuous optimization problems
Population Initialization	Random initialization	Random initialization	Random initialization
Exploration vs. Exploitation	Balanced exploration and exploitation	Exploration focused	Exploration focused
Communication Mechanism	No direct communication	Pheromone-based communication	Attraction based on brightness
Convergence Properties	May converge quickly to local optima	Slow convergence	May suffer from slow convergence
Robustness to Parameter Settings	Moderately sensitive	Sensitive	Moderately sensitive
Implementation Complexity	Relatively simple to implement	Moderate complexity	Relatively simple to implement
Application Domains	Continuous optimization problems	Combinatorial optimization problems	Continuous optimization problems

From the results of the comparison, we can conclude that the FA – method is the most universal in order to use it for preprocessing input data in the future.

2.1. Main features of applying the FA method

The FA method is a nature-inspired optimization method developed by Xin-she Yang in 2008. It is inspired by the blinking behavior of fireflies, where fireflies use bioluminescence to attract partners or prey. FA is a metaheuristic algorithm used to solve optimization problems in various fields [1], [6], [20].

The Firefly method was proposed by Xin-she Yang in 2008.

The method was developed as a population-based optimization method inspired by the blinking behavior of fireflies to solve optimization problems more efficiently.

Since its introduction, the FA method has gained popularity and has been applied to various optimization problems in Engineering, Computer Science, and other fields.

The main idea of the Firefly algorithm is to simulate the blinking behavior of fireflies to find optimal solutions to the optimization problem. The method works on the basis of the following principles [1], [6], [20]:

attraction: fireflies are attracted to each other depending on the brightness of their flashes. Similarly, in FA, the attractiveness of a firefly (solution) is determined by its suitability, with brighter solutions representing better solutions;

moving towards brighter solutions: Fireflies tend to move towards brighter fireflies nearby. In FA, each Firefly (solution) adjusts its position in the search space, moving towards brighter solutions, and the intensity of attraction is determined by the difference in brightness and distance between the Fireflies [1], [6], [20];

randomization and research: to help explore the search space, FA includes randomization by adding a random component to Firefly movement. This ensures that the method does not get hung up on local optimums and can explore different areas of the solution space.;

global optimization: FA aims to find a global optimal solution by iteratively updating Firefly positions based on their attractiveness and distance between them. The method converges when the Firefly positions no longer change significantly or the predefined completion criterion is met.

Overall, the FA method is a simple but effective optimization technique inspired by Firefly behavior. It is able to effectively solve a wide range of optimization problems and is successfully used in various fields due to its simplicity and efficiency [1], [6], [20].

Now we can dive into the intricacies of firefly optimization in more detail. The essence of the method is clearly shown in Fig. 1.

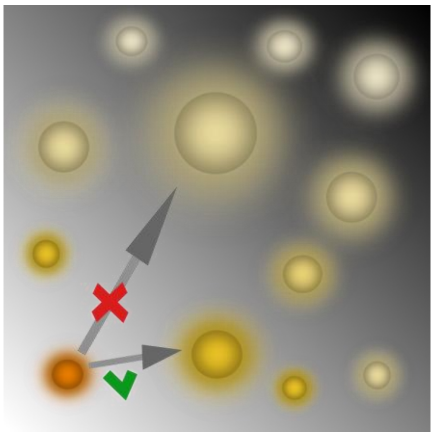

Figure 1: Fireflies in the search space. Visibility decreases with increasing distance

3. Proposed method

Levy's flight function can be incorporated into the Firefly Algorithm (FA) to enhance its exploration capability and improve its convergence speed. Levy's flight is a random walk process described by Levy distribution, which exhibits heavy-tailed behavior and allows for long-distance jumps in the search space. By incorporating Levy's flight function, the fireflies in the FA can explore the search space more effectively, leading to better global optimization performance [1], [6], [20].

Here's how Levy's flight function can be integrated into the Firefly Algorithm [22]-[25]:

1. Step 1: Levy Flight Generation:

- at each iteration of the algorithm, Levy flights are generated for each firefly to determine their movement direction and distance;
- Levy flights are generated using Levy's flight function, which produces step sizes according to the Levy distribution. The step sizes are typically drawn from a Levy distribution with a specified scale parameter.

2. Movement of Fireflies:

- the fireflies adjust their positions based on the generated Levy flights. Each firefly moves a distance determined by Levy's flight function in a random direction;
- the movement of fireflies is guided by their attractiveness, with brighter fireflies attracting others more strongly. Fireflies move towards brighter individuals while incorporating Levy flights for exploration.

3. Exploration and Exploitation:

- Levy flights facilitate exploration by allowing fireflies to make long-distance jumps in the search space, enabling the algorithm to escape local optima and explore new regions;
- at the same time, the attractiveness mechanism of the Firefly Algorithm ensures that fireflies tend to converge towards brighter solutions, promoting exploitation of promising regions of the search space.

4. Update Positions and Iteration:

- after adjusting their positions based on Levy flights and attractiveness, the positions of fireflies are updated, and the algorithm proceeds to the next iteration;
- the process continues until a termination criterion is met, such as a maximum number of iterations or convergence of solutions.

By incorporating Levy's flight function into the Firefly Algorithm, the algorithm gains enhanced exploration capabilities, enabling it to efficiently search for global optima in complex optimization problems. This modification can lead to improved convergence speed and better overall performance of the algorithm across various domains [22]-[25].

4. Experimental research

For the experimental research and comparison of proposed method with another approaches was be used the following as the training and testing data:

- RT-IoT2022 [26], [16]. The RT-IoT2022 is a data sample formed by scientists on the basis of the Internet of things infrastructure, when a wide range of Internet of Things devices and a range of network attacks are combined in real-time conditions. The sample covers normal and hostile network behaviors that model a general view of real-world scenarios;
- Visegrad Group companies data (VGCD) [27], [16]. This dataset contains information about companies from the Visegrad Group countries, which include the Czech Republic, Hungary, Poland, and Slovakia. The Visegrad Group is a cultural and political alliance of these four Central European countries;
- Influenza Outbreak Event Prediction via Twitter (IOEP) [28]. This dataset aims to predict influenza outbreaks using Twitter data

The general information about datasets presented in Table 2.

Table 2
General information about datasets

Datasets	Feature Type	Number of Features	Number of Instances
RT-IoT2022	Real	83	123117
VGCD	Real	83	85
IOEP	Real	523	75839

The meta-parameters for neuroevolution synthesis of models demonstrate at Table 4.

Table 3
The meta-parameters for neuroevolution synthesis

Metaparameter	Value
Population size	100
Elite size	5%
Activation function (fitness functions)	hyperbolic tangent
Mutation probability	25%
Crossover type	uniform
Types of mutation	deleting an interneuronal connection removing a neuron adding interneuronal connection adding a neuron changing the activation function
Clustering method	*k*-nearest neighbors
Neighbors number	7

The results of the work present at Table 4.

Table 4
The work results of proposed method

Datasets	ANN time synthesis (without feature selection)	ANN time synthesis (using PSO)	ANN time synthesis (using ACO)	ANN time synthesis (using FA)	Error on test sample
RT-IoT2022	8456 s	7225 s	6935 s	5689 s	0.91
VGCD	9555 s	8123 s	7456 s	6126 s	0.924
IOEP	9832 s	8526 s	7626 s	6751 s	0.942

5. Discussions of results

Analyzing the results obtained, it is worth noting that using any method for data preprocessing significantly reduces the time of subsequent synthesis. However, it is the use of FA that demonstrates the greatest optimization of synthesis time. This can be explained precisely by better selection of informative features. After all, if features can be shortened differently during preprocessing by different methods, the FA method helps to better track hidden relationships between data, and therefore significantly simplify the process of forming structural relationships between really dependent features.

6. Conclusion

At the paper examines the using swarm intelligence techniques for data preprocessing over neuroevolution synthesis of ANN can provide a number of advantages. Therefore, swarm analysis techniques such as particle swarm optimization or ant colony optimization can effectively identify the most relevant traits from multidimensional data. This reduces the dimension of the input space, mitigating the Curse of dimension and improving the effectiveness of ANNs training. In addition, swarm analysis methods can filter out noisy or mismatched data points and detect outliers, increasing ANNs resistance to noisy input data and expanding generalization capabilities.

The results of experiments prove that the new concept has quite acceptable (certifying) indicators of time use during the synthesis of neural networks.

7. Acknowledgements

The work was carried out with the support of the state budget research projects of the state budget of the National University "Zaporozhzhia Polytechnic" “Intelligent information processing methods and tools for decision-making in the military and civilian industries” (state registration number 0124U000250).

8. References

[1] X.-S. Yang, M. Karamanoglu, Z. Cui, A. H. Gandomi, R. Xiao, Swarm Intelligence and Bio-Inspired Computation: Theory and Applications, Elsevier, 2013.
[2] T. M. Shami, A. A. El-Saleh, M. Alswaitti, Q. Al-Tashi, M. A. Summakieh, S. Mirjalili, Particle Swarm Optimization: A Comprehensive Survey, IEEE Access 10 (2022) 10031–10061. doi:10.1109/access.2022.3142859.
[3] F. Marini, B. Walczak, Particle Swarm Optimization, Comprehensive Chemometrics, Elsevier, 2020, p. 649–666. doi:10.1016/b978-0-12-409547-2.14581-0.

[4] P. A. Gkaidatzis, A. S. Bouhouras, D. P. Labridis, Application of PSO in Distribution Power Systems: Operation and Planning Optimization, Applying Particle Swarm Optimization, Springer International Publishing, Cham, 2021, p. 321–351. doi:10.1007/978-3-030-70281-6_17.

[5] A. Vasuki, Ant Colony Optimization, Nature-Inspired Optimization Algorithms, Chapman and Hall/CRC, 2020, p. 99–114. doi:10.1201/9780429289071-8.

[6] X.-S. Yang, X.-S. He, Why the Firefly Algorithm Works?, Nature-Inspired Algorithms and Applied Optimization, Springer International Publishing, Cham, 2017, p. 245–259. doi:10.1007/978-3-319-67669-2_11.

[7] Ant Colony Optimization — Intuition, Code & Visualization, 2024. URL: https://towardsdatascience.com/ant-colony-optimization-intuition-code-visualization-9412c369be81

[8] Growth Optimizer (GO), 2024. URL: https://medium.com/@melika.akz97/growth-optimizer-go-dd6da47a4e93

[9] Wolf Search Algorithm (WSA): Harnessing the Social and Hunting Strategies of Wolves for Optimization, 2024. URL: https://python.plainenglish.io/wolf-search-algorithm-wsa-harnessing-the-social-and-hunting-strategies-of-wolves-for-9de4d1fc19c6

[10] Machine Learning with Particle Swarm Optimization by Sebastian Raschka, 2021. URL: https://sebastianraschka.com/

[11] M. F. Ahmad, N. A. M. Isa, W. H. Lim, K. M. Ang, Differential evolution: A recent review based on state-of-the-art works, Alex. Eng. J. (2021). doi:10.1016/j.aej.2021.09.013.

[12] A. Sharma, A. Sharma, J. K. Pandey, M. Ram, Swarm Intelligence Applications in Artificial Neural Networks, Swarm Intelligence, CRC Press, Boca Raton, 2021, p. 99–122. doi:10.1201/9781003090038-5.

[13] S. J. Anand, C. Tamilselvi, D. Sam, C. Kamatchi, N. Dey, K. Sujatha, Swarm Intelligence-based Framework for Image Segmentation of Knee MRI Images for Detection of Bone Cancer, Swarm Intelligence and Machine Learning, CRC Press, Boca Raton, 2022, p. 95–115. doi:10.1201/9781003240037-6.

[14] M. Ehteram, A. Seifi, F. B. Banadkooki, Structure of Particle Swarm Optimization (PSO), Application of Machine Learning Models in Agricultural and Meteorological Sciences, Springer Nature Singapore, Singapore, 2023, p. 23–32. doi:10.1007/978-981-19-9733-4_2.

[15] Ant Colony Optimization, Handbook of Machine Learning, WORLD SCIENTIFIC, 2019, p. 123–141. doi:10.1142/9789811205675_0007.

[16] A. Sharma, A. Sharma, J. K. Pandey, M. Ram, Swarm Intelligence Applications in Artificial Neural Networks, Swarm Intelligence, CRC Press, Boca Raton, 2021, p. 99–122. doi:10.1201/9781003090038-5.

[17] O. Nagaya, A. W. Pillay, E. Jembere, Comparative Study of Image Resolution Techniques in the Detection of Cancer Using Neural Networks, Artificial Intelligence Research, Springer Nature Switzerland, Cham, 2023, p. 187–202. doi:10.1007/978-3-031-49002-6_13.

[18] D. Kumar, B. G. R. Gandhi, R. K. Bhattacharjya, Firefly Algorithm and Its Applications in Engineering Optimization, Nature-Inspired Methods for Metaheuristics Optimization, Springer International Publishing, Cham, 2020, p. 93–103. doi:10.1007/978-3-030-26458-1_6.

[19] D. Varela, J. Santos, Crowding Differential Evolution for Protein Structure Prediction, From Bioinspired Systems and Biomedical Applications to Machine Learning, Springer International Publishing, Cham, 2019, p. 193–203. doi:10.1007/978-3-030-19651-6_19.

[20] H. Xu, S. Yu, J. Chen, X. Zuo, An Improved Firefly Algorithm for Feature Selection in Classification, Wirel. Pers. Commun. 102.4 (2018) 2823–2834. doi:10.1007/s11277-018-5309-1.

[21] W. Li, Z.-d. Lu, D.-L. Deng, Quantum Neural Network Classifiers: A Tutorial, SciPost Phys. Lect. Notes (2022). doi:10.21468/scipostphyslectnotes.61.

[22] L. Fernández-Brillet, O. Álvarez, J. L. Fernández-Martínez, The PSO Family: Application to the Portfolio Optimization Problem, Applying Particle Swarm Optimization, Springer International Publishing, Cham, 2021, p. 111–132. doi:10.1007/978-3-030-70281-6_7.

[23] H. Gao, B. Feng, Y. Hou, L. Zhu, Training RBF Neural Network with Hybrid Particle Swarm Optimization, Advances in Neural Networks - ISNN 2006, Springer Berlin Heidelberg, Berlin, Heidelberg, 2006, p. 577–583. doi:10.1007/11759966_86.

[24] Y. Geng, L. Zhang, Y. Sun, Y. Zhang, N. Yang, J. Wu, Research on Ant Colony Algorithm Optimization Neural Network Weights Blind Equalization Algorithm, Int. J. Secur. Its Appl. 10.2 (2016) 95–104. doi:10.14257/ijsia.2016.10.2.09.
[25] RT-IoT2022, 2022. URL: https://archive.ics.uci.edu/dataset/942/rt-iot2022
[26] E Visegrad Group companies data, 2021. URL: https://github.com/7ossam81/EvoloPy
[27] Influenza Outbreak Event Prediction via Twitter, 2021. URL: https://archive.ics.uci.edu/dataset/861/influenza+outbreak+event+prediction+via+twitter