Jerzy Rudolf Tchórzewski[1]
ORCID: 0000-0003-2198-7185

Maciej Zakrzewski[2]
ORCID: 0009-0003-1375-6771

[1] University of Siedlce
Faculty of Exact and Natural Sciences
Institute of Computer Science
ul. 3 Maja 54, 08-110 Siedlce, Poland

[2] Computer Science Graduate
(Master of Science Engineer)
at University of Siedlce
Faculty of Exact and Natural Sciences
Institute of Computer Science
ul. 3 Maja 54, 08-110 Siedlce, Poland

[1]jerzy.tchorzewski@uws.edu.pl,[2]maciej.zakrzewski@stud.uws.edu.pl

Analysis of the Possibilities of the Evolutionary Algorithm to Improve the Neural Model of the TGE S.A. Day Ahead Market System Using Selected Programming Environments

DOI: 10.34739/si.2024.31.07

Abstract. The article contains selected research results regarding the analysis of the possibility of using the Evolutionary Algorithm to improve neural models of intelligent systems using selected programming environments. Choosing an appropriate program-

ming language is one of the basic activities in the process of implementing complex algorithms, which include methods of artificial neural networks and evolutionary algorithms. Due to the fact that the object of the research was an intelligent Day Ahead Market system operating on the Polish Power Exchange and the modeling methods were artificial neural networks and evolutionary algorithms, it was decided to use very high-level programming languages such as Python, Matlab and C# for implementation and associated development environments. It turned out, among other things, that each of these languages and programming environments has its advantages and disadvantages, but all of them are very useful due to their useful syntax and rich included libraries. A thorough analysis of the implementation shows, among other things, that the choice of programming language affects the efficiency, speed and quality of the obtained implementations of system models. Against this background, the advantages and disadvantages of individual programming languages are shown, especially in the context of implementing evolutionary algorithms. The research results indicate directions for selecting an appropriate programming language and the associated programming environment for system modeling using artificial neural networks and evolutionary algorithms. In addition, the method of analysis, as well as the method of modeling and implementation was shown on the example of a specific system, which was the Day Ahead Market system of TGE S.A.

Keywords: Artificial Neural Networks, C#, Day Ahead Market (DAM), Evolutionary Algorithm, Matlab, Python, TGE S.A.

1 Introduction

One of the current problems in modeling intelligent systems is the search for adequate models using artificial intelligence and machine learning methods. Such an intelligent system is the Day Ahead Market (DAM) system operating on the Polish Power Exchange. (TGE S.A.). The search for the possibility of creating an DAM system model should be preceded by an analysis of selected programming environments and appropriate artificial intelligence methods, which include artificial neural networks and evolutionary algorithms. It was assumed that in the first phase of design and implementation, a neural model of the RDN system would be created by designing and training an Artificial Neural Network (ANN), which would then be improved using the Evolutionary Algorithm (EA) using three selected programming languages, i.e. Matlab, Python and C# [28-29].

The results of the analysis of the implemented evolutionary algorithms using the above-mentioned were used for comparative research. Three programming languages and related computing environments, i.e. C#, Matlab and Python as languages currently inspiring the best programming languages for artificial intelligence and machine learning methods that have been adopted in the academic and industrial environment [1-2, 4-6, 13, 18, 26]. Each of the above-mentioned programming languages has its own unique features, rich libraries and well-prepared tools that may affect the efficiency, quality and ease of implementation of evolutionary algorithms, taking into account on the one hand the fact of their availability, and on the other hand the degree of openness or the amount of costs. purchasing them.

Therefore, using the above-mentioned programming languages, i.e. Python, Matlab and C#, hybrid methods consisting of artificial neural networks improved using evolutionary algorithms were designed, implemented and then compared. In particular, attention was paid to examining how differences in syntax, libraries and programming tools affect the efficiency, speed and quality of the solutions obtained in the form of neural-evolutionary models of the TGE S.A. system [12, 15, 19-20, 25].

An important additional goal of the research is the possibility of comparing the advantages and disadvantages of languages and programming environments in the context of using, especially, evolutionary algorithms, which may be crucial for researchers involved in modeling systems of the same class as the TGE S.A. Day Ahead Market system, i.e. their design, implementation, and testing of correctness. functioning [12, 15, 19-20, 25].

The basic research was preceded by a discussion of the basic concepts of evolutionary algorithms, selected programming languages, etc. Through the lens of practical implementations and performance tests, the research conducted provides valuable information for all those interested in the subject of neural-evolutionary modeling [28-29].

2 Evolutionarily aided neural modeling

For many years, research has been conducted to evaluate selected programming languages used to implement evolutionary algorithms used to improve models of various types of systems. And so, among others: in [13] an attempt was made to compare and evaluate selected programming languages from the point of view of the speed of performing basic operations in evolutionary algorithms. This research aimed to determine the programming language and associated environment that could be considered best for specific applications in evolutionary algorithms.

The authors showed, among other things, that there is no universal programming language that would be perfect in all aspects, both in terms of chromosome and genotype sizes and related parameters, as well as the structure of the data used in a specific AE. Although some languages, such as C#, demonstrate high performance when using oriented operations, in other cases the language is Matlab, and in still others - Python. For these reasons, the cited work shows that the use of a hybrid approach can give significantly better results from the point of view of efficiency, with the C# language having great possibilities in terms of program efficiency, the Matlab language in terms of system modeling, and the Python language ensuring greater flexibility and speed of prototyping. [1-2, 4-6, 10, 13, 18, 26].

Combining languages such as C languages, which are known for their high performance in system modeling, with other languages such as Python, which offer greater flexibility and speed of prototyping, may prove to be an optimal solution for specific applications in the design and implementation of evolutionary algorithms Such a strategy would allow the advantages of different programming languages to be combined in one system, which could lead to better results in terms of both performance and flexibility [28-29].

3 TGE S.A. Day Ahead Market system as a modeling object

To compare the properties of selected programming languages and related programming environments, the Day Ahead Market (DAM) system of the Polish Power Exchange was selected, which is a key subsystem of the Polish Power Exchange system. The DAM system enables Exchange participants to submit purchase and sale offers for electricity (ee) for delivery on the next business day [12, 15, 19-20, 25].

Neural modeling of the DAM system in the context of analyzing the possibility of using evolutionary algorithms to improve system models is an interesting subject of research for several reasons. Firstly, the DAM system generates huge amounts of data regarding commercial offers, energy prices, the amount of available energy and other market factors. This data can be used to train neural models using evolutionary algorithms, primarily for modeling and subsequent forecasting using the obtained energy price models, and thus also to better match supply and demand in electricity trading, and thus to manage risk more effectively [12, 15, 19-20, 25].

Secondly, the DAM system operates in a dynamic market environment, where energy prices may be subject to rapid changes depending on many factors, not only such as demand and supply, but also due to unpredictable weather conditions, energy and environmental policy, etc. In these circumstances, the use of evolutionary algorithms may help adapt the neural models of the DAM system to changing market conditions, especially in conditions of quick response to new information and trends on the Stock Exchange.

Thirdly, the DAM system requires complex and very well-prepared models for effective management of electricity trade and thus minimizing risk. Evolutionary algorithms can be used in this regard to improve the hyper parameters of models and to automatically tune them to obtain the best trading results.

The description of data listed on the Stock Exchange is crucial from the moment of the need to train, validate and test the Artificial Neural Network in order to obtain a neural model of the DAM system, through the use of the Evolutionary Algorithm to improve its parameters, to the use of the obtained neural model in comparative studies and in sensitivity testing, and also in prognostic studies.

It was assumed that the data available on the Exchange will concern the volume of electricity delivered and sold on the Exchange on the input side of the DAM system in all separate hours of the day and average weighted by the volume of ee on the output side also in all hours of the day [12, 19-20, 25, 28-29] . Due to the above, the Day Ahead Market system of TGE S.A. is an interesting object of research in the context of selecting a programming language for research, implementing the model, and then analyzing the possibility of using evolutionary algorithms to improve system models, especially regarding sensitivity testing and forecasting electricity prices in shorter and longer time horizons.

3.1 Data used in research

Historical data quoted on TGE S.A. was used to model the DAM system in the years 2016-2019, including the volume of delivered and sold ee according to the structure presented in Table 1 and average volume-weighted ee prices according to the structure presented in Table 2 [12, 19-20, 25]. Therefore, the research experiments used hourly data from a period of four years, i.e. from 2016, 2017, 2018, 2019.

Table 1. Structure of input data regarding the volume of delivered and sold ee in individual hours of the day in 2016-2029 [MWh] used to train the ANN models of the DAM TGE S.A. system. Source: [12, 19-20, 25, 28-29].

Variable name [MWh]	Explanation of variable designation	1.01.2016	2.01.2016	...	30.12.2019
u1	ee volume in hour 0-1	89.10	154.90	...	915.80
u2	ee volume in hour 1-2	80.60	172.60	...	1287.40
...	...	...	...	...	...
u24	ee volume in hour 23-24	266.10	262.60	...	616.80

Table 2. Structure of output data regarding the volume-weighted average price of delivered and sold electricity in individual hours of the day in 2016-2029 [MWh] used to train ANN models of the RDN TGE S.A. system. Source: [12, 19-20, 25, 28-29].

Variable name [PLN/MWh]	Explanation of variable designation	1.01.2016	2.01.2016	...	30.12.2019
y1	ee fixing rate in 0-1	89.10	154.90	...	915.80
y2	ee fixing rate in 1-2	80.60	172.60	...	1 287.40
...	...	...	...	...	...
y24	ee fixing rate in 23-24	266.10	262.60	...	616.80

The expected models of the DAM system are neural models with 24 inputs and 24 outputs. It was determined that the optimal ANN structure to be used in research experiments is a structure with one hidden layer with 24 neurons. Moreover, it was assumed that the activation function tansig() occurs in the hidden layer and purelin() in the output layer, and the Levenberg-Marquardt (LM) algorithm was used to train the Artificial Neural Network. The analysis of volume data and average volume-weighted ee prices, e.g. for 6:00 a.m., 12:00 p.m., 6:00 p.m. and 24:00 p.m., showed, among other things, that: the highest average consumption is at 12:00, and the smallest at 24:00. An example of hourly price data is shown in Figure 1 for: 6:00.

4 Neural modeling of the DAM system in Matlab

The data used to train the ANN of the neural model of the DAM system were quoted on TGE S.A. in the period from September 1, 2016 to December 31, 2019. They concerned the volume of ee delivered and sold in individual hours of the day at the entrance to the system [MWh] and the average weighted by the volume of ee at the system output [PLN/MWh]. Using the

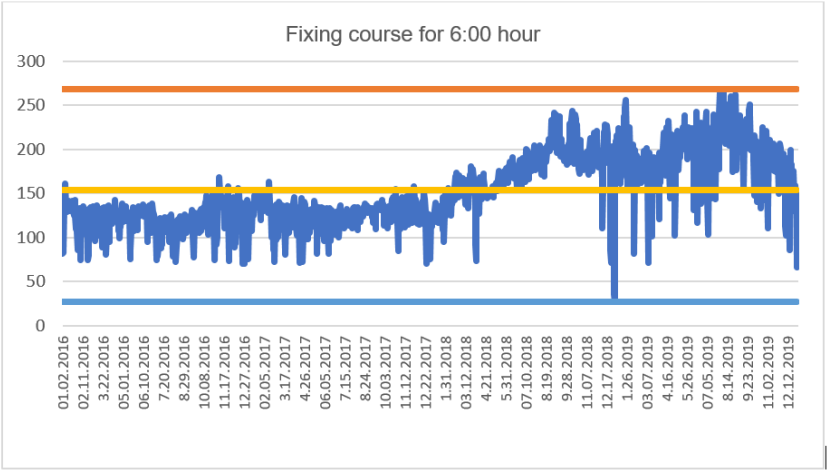

Figure 1. Example of data flow on volume-weighted average ee prices for 6:00 [PLN/MWh]. Markings: X axis – time [day], Y axis – volume-weighted average price ee [MWh], AVG = 153.48 [PLN/MWh], Max = 268.58 [PLN/MWh], Min = 26.4 [PLN /MWh]. Source: Own study [12, 19-20, 25, 28-29]., Source: Own study [28-29].

above-mentioned data, an ANN was designed and implemented in Matlab in order to teach it a neural model of the DAM system. This model was then the basis for testing the quality of AE designed and implemented in three programming languages, i.e. Matlab, Python and C#.

Both the data on 24 input variables corresponding to individual hours of the day, as well as the data on 24 output variables corresponding to individual hours of the day, but with a time shift of one day, taken from the period from September 1, 2016 to December 31, 2019, were normalized. Normalization consisted in replacing individual values, both input and output, with values determined by dividing the values of individual data by the sum of data values in a given hour of the day from the entire period adopted for learning, e.g. a monthly, quarterly, semi-annual or annual period.

4.1 ANN design

In order to create the ANN architecture for training the DAM system model, it was assumed that there were 24 input quantities and 24 output quantities, and the key issue was the selection of the number of hidden layers and the number of neurons in them, as well as the selection of activation functions related to individual layers. Moreover, the selection of the learning method was important, which was adopted as the Levenberg-Marquid algorithm [7, 11, 10, 21, 24].

Moreover, before implementing the model, a detailed data analysis was carried out, choosing the set of data recorded as part of the Fixing II exchange rate in the period from September

1, 2016 to December 1, 2019, both in terms of input and output values, only with a shift of one day between the entry and the output from the system, where the activation function of the hidden layer was assumed as a hyperbolic tangent (tansig), and a linear function (purelin) for the output layer.

4.2 ANN implementation

For the ANN training process, the data was divided into training, testing and validation data in the proportion [%] 70:15:15, respectively. After carrying out the learning process using the Levenberg-Marquid algorithm [11, 16, 24, 28-29], a neural model of the RDN system was obtained, in which the weights and biases of the two ANN layers play a key role: the layer located between the input neurons and the neurons of the hidden layer and the ANN layer located between neurons of the hidden layer and neurons of the output layer, as well as the types of neuron activation functions used. In further research experiments, the weights and biases of neurons in both ANN layers, as well as the parameters of the adopted neuron activation functions, whose values and mutual relations are improved using the Evolutionary Algorithm [3, 8, 14, 17, 22-24, 27], play an essential role.

4.3 Mathematical model of Perceptron ANN

Creating a mathematical model of a Perceptron ANN trained with an DAM system model is intuitively understandable. It is enough to use the values of appropriate weights and biases to determine weighted net adders at the output of both layers for individual neurons, and then use the net values obtained in this way as arguments in the appropriate neuron activation functions, i.e. tansig() and purelin() along with the appropriate function parameters. This mathematical approach allows for the processing of input data in the considered architecture by two layers of neurons, in accordance with specific activation functions, which is a key element in the process of data analysis and prediction in the context of the neural model of the DAM system obtained for forecasting.

Therefore, the first step in any programming environment is to determine the net value using operations on matrices and multidimensional arrays, which are the basic data type used in this programming language. In the case of AE improving the ANN, inputs and their corresponding weights are extracted based on data from the chromosome representing the ANN weight and bias matrices and the appropriate parameters of the activation function. Then, the weight matrix is multiplied by the corresponding input vectors and the biases are added, which leads to the net value.

4.4 Design and implementation of ANN in Python and C#

Designing and implementing an ANN in Python and C# is basically not much different from the way of creating a solution in Matlab, because, among other things, a similar method for determining the net value and, therefore, the value of the ANN outputs using newly generated chromosomes in individual AE iterations is similar. The research noted, among others, that the possibilities of performing operations on matrix and table data written in Matlab in comparison

with solutions written in C# and Python indicate that many operations have simple and convenient coding solutions in the language. Matlab require writing more extensive code in C# and Python, and often adding dedicated new methods to obtain the same equivalent functionalities.

The above-mentioned solutions, e.g. when determining the value of the tansig() activation function, require first the extraction of parameters from the chromosome, and then the previously calculated tab1net value is assigned to the net variable. Next, the tansig() function is used to calculate the result based on the previously prepared code. In the implementation carried out in Python, the code is basically similar to the solution used in Matlab, but it can be noticed, among other things, that there is a difference in the syntax of both languages, because in Python during the process of calculating the neuron activation function tansig() first, the parameters are downloaded from the chromosome. Also in the case of implementing this solution in C#, you can notice that it is almost identical to the codes in Matlab and Python, except for the syntax specific to this language.

Here, the process of calculating the tansig() function starts with the extraction of parameters from the chromosome. Then, the previously calculated tab1net value is assigned to the net variable. In the final step, the tansig() function is used to calculate the result using the previously prepared code. The situation is similar when calculating the value of the weighted net adder and the neuron activation function in the ANN output layer. It is worth noting that the input to this layer is the output from the hidden layer of neurons. This aspect is important when computing the weighted net adder and the purelin() activation function because it affects the connection structure in the ANN. The implementation in Python and C# is almost identical to Matlab, with minor syntactical differences. After calculating net, the output value from the ANN is calculated in the second layer [24, 28-29]. It turned out that the codes written in the three programming languages selected for comparison, i.e. Matlab, Python and C#, do not differ significantly, as they all operate practically on the same operations, but differ syntax.

5 AE implementation to improve neural model

Designing and implementing an ANN in Python and C# is basically not much different from the way of creating a solution in Matlab, because, among other things, a similar method for determining the net value and, therefore, the value of the ANN outputs using newly generated chromosomes in individual AE iterations is similar. The research noted, among others, that the possibilities of performing operations on matrix and table data written in Matlab in comparison with solutions written in C# and Python indicate that many operations have simple and convenient coding solutions in the language. Matlab require writing more extensive code in C# and Python, and often adding dedicated new methods to obtain the same equivalent functionalities.

The above-mentioned solutions, e.g. when determining the value of the tansig() activation function, require first the extraction of parameters from the chromosome, and then the previously calculated tab1net value is assigned to the net variable. Next, the tansig() function is used to calculate the result based on the previously prepared code. In the implementation carried out in Python, the code is basically similar to the solution used in Matlab, but it can be

noticed, among other things, that there is a difference in the syntax of both languages, because in Python during the process of calculating the neuron activation function tansig() first, the parameters are downloaded from the chromosome.

Also in the case of implementing this solution in C#, you can notice that it is almost identical to the codes in Matlab and Python, except for the syntax specific to this language. Here, the process of calculating the tansig() function starts with the extraction of parameters from the chromosome. Then, the previously calculated tab1net value is assigned to the net variable. In the final step, the tansig() function is used to calculate the result using the previously prepared code. The situation is similar when calculating the value of the weighted net adder and the neuron activation function in the ANN output layer.

It is worth noting that the input to this layer is the output from the hidden layer of neurons. This aspect is important when computing the weighted net adder and the purelin() activation function because it affects the connection structure in the ANN. The implementation in Python and C# is almost identical to Matlab, with minor syntactical differences. After calculating net, the output value from the ANN is calculated in the second layer [24, 28-29].

It turned out that the codes written in the three programming languages selected for comparison, i.e. Matlab, Python and C#, do not differ significantly, as they all operate practically on the same operations, but differ syntax.

5.1 Initialization of the Beginning Population

In the initial phase of the Evolutionary Algorithm, called population initialization, a key step is to assign random values to each gene in the chromosome if the genotype is composed of one chromosome as in the case under consideration. The following method was adopted: first, the range of values from which the values of individual chromosomes will be drawn is calculated, i.e. the upper value of each gene is determined by adding to the value of each parameter the adopted value related to the accuracy of the parameter or, in the case of the lower value of each gene, this value is subtracted. Then, the value of each parameter (weight and bias) is drawn from the range defined by its upper and lower values, thus ensuring diversity in the Initial Population (IP) called Beginning Population (BP).

PP is implemented in a similar way in Python using the numpy library, which is very helpful in array operations. It was assumed that 100 sets of chromosome values (100 individuals) were generated, each containing 1 296 genes. Each set consists of random values, generated on the basis of given minimum and maximum values for various parameters. The method implemented in C# is analogous to the described solution in Python, differing only in the use of a generic list and methods available in the C# environment. C# code is implemented in a way that maintains the same logic as Python code, while maintaining readability and efficiency.

All methods in three different programming languages that generate the Initial Population are basically similar to each other, but differ in the specifics of the programming languages in which they are implemented. The main goal of these methods is to generate 100 chromosomes

(individuals) with a length of 1,296 each, which will be subjected to the process of changes as part of the methods used in AE, i.e.: crossover, mutation, selection, etc.

5.2 Crossover and mutation

The modified single-point crossover method was adopted as the crossover method, which will be presented in each of the implemented programming languages and briefly discussed. The crossbreeding method is usually a key step in the AE algorithm, where units of the Parent Population are paired together to create offspring that will have traits from both parents. The first step is to create a loop that will be executed five times because five crossovers will be generated.

The second step is to draw two chromosomes numbered from 1 to 100, then cut both chromosomes at a randomly selected point and cross-assign the cut parts. In the next step, two points are selected: one for the first layer and the other for the second, to finally select a random split point for each type of data: for the first layer's weights, for the first layer's biases, for the arguments and for the second layer's weights and biases.

This process will be repeated for each parameter set. This method was chosen because it is a very effective way to achieve diversity in the Parent Population. By randomly selecting the split point for each type of data, it is ensured that the newly created units will contain characteristics of both parents, which can lead to more diverse offspring. Uniform mutation was adopted as the mutation method, so that each chromosome has an equal chance of mutation, which allows maintaining diversity in the population. If a mutation is drawn, the chromosome value is again drawn from the minimum and maximum range for a given parameter.

5.3 Fitness function

The fitness function enabling the assessment of the degree of adaptation of individuals (chromosomes) to the population was defined as the MSE error occurring between the output values obtained from the model and the values obtained from the system (Figure 2).

In order to visualize the performance of the AE algorithm, Figures 3-5 show the waveforms of the adaptation function defined as the MSE error value in individual programming languages, i.e. Matlab, Python and C#. It turned out, among others, that the algorithm implemented in Matlab shows the best value of the adaptation function, the next one is the algorithm implemented in Python, and finally the algorithm implemented in C#.

5.4 Selection using a roulette wheel

In AE, a roulette wheel method was used as a selection method for a population of chromosomes, where each chromosome is assessed based on its quality and then selected on the basis of a probability proportional to its assessment, with the selection probability determined for each chromosome based on its assessment.

```
for j = 1:24
    for i = 1:liczba_chromosomow
        purelineerror(i, j) = abs(( purelinetab(i, j) - srednie_kolumn(1, j)) / srednie_kolumn(1, j));
    end
end
```

```
for j in range(24):
    for i in range(liczba_chromosomow):
        purelineerror[i, j] = abs((purelinetab[i, j] - srednie_kolumn[j]) / srednie_kolumn[j])
```

```
for (int j = 0; j < 24; j++)
{
    for (int i = 0; i < liczba_chromosomow; i++)
    {
        purelineerror[i, j] = Math.Abs((purelinetab[i, j] - srednie_kolumn[j]) / srednie_kolumn[j]);
    }
}
```

Figure 2. Listing of fragment code regarding determining the MSE error as the value of the fitness function in a given epoch. Source: Own study [12, 19-20, 25, 28-29].

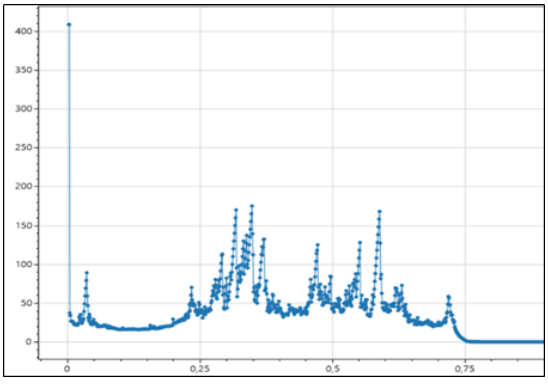

Figure 3. The course of the adaptation function generated in C#. Source: Own study [28].

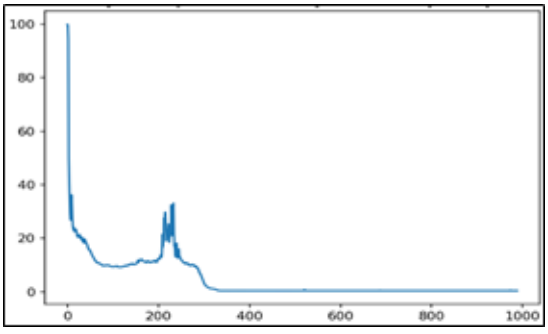

Figure 4. The course of the adaptation function generated in Python. Source: Own study [28-29].

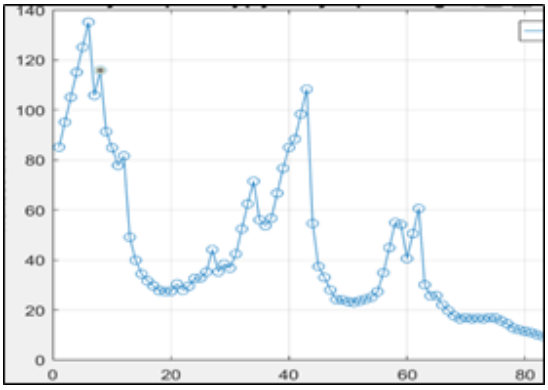

Figure 5. The course of the adaptation function generated in C#. Source: Own study [28].

6 Comparative studies of programming languages

Detailed research is aimed at assessing the effectiveness and execution time of the evolutionary algorithm implemented to improve the neural model of the DAM system in three programming languages, i.e. Python, Matlab and C#, and comparing them with each other. The main aim of the research was to assess how changes in the size of the population and the number of iterations (epochs) affect the speed of AE implemented in particular programming languages. Particular attention was paid to the quality of the model, i.e. the consumption of the smallest possible memory capacity by individual algorithms and the running time of individual algorithms.

6.1 Methodology and measurement tools

Three programming languages were used in the research process, i.e. Python due to its free availability for individual clients and rich equipment with ready-made various implementations of evolutionary algorithms, Matlab due to its adaptation to scientific and engineering research due to its rich possibilities of conducting calculations using matrices and multidimensional arrays, as well as rich library equipment with artificial intelligence algorithms, and the C# language due to the possibility of using it directly in the .NET environment with the possibility of memory and performance management.

Therefore, in order to compare the effectiveness of the evolutionary algorithm implemented in these three programming language environments, we mainly focused on two main aspects: the execution time measured from the start of the algorithm to its completion and the memory consumption measured by the amount of memory occupied by the process during its execution. In order to measure the execution time of Evolutionary Algorithm implemented in each of the three programming languages, appropriate measurement tools were used. Thus, Python uses the time module to measure the algorithm execution time, Matlab uses the tic and toc functions

to measure time, and C# uses the stopwatch class to precisely measure the code execution time. To measure memory consumption, the memory-usage function from the memory-profiler module was used in Python, the memory function to monitor memory consumption in Matlab, and the GC.GetTotalMemory(true) method in C# to measure memory consumption [1-2,4-5, 18, 26, 28-29].

6.2 Research procedure

The research was carried out by running an evolutionary algorithm using different sizes of the Parent Population (PP) and a different number of epochs (iterations) in each implemented programming language. The results were compared in terms of: algorithm execution time measured from the beginning to the end of the algorithm and memory consumption measured by the amount of memory occupied by the process while executing the algorithm. The research results are presented in Table 3 and in Table 4, which made it possible to precisely compare the effectiveness of the evolutionary algorithm implemented in each of the three programming languages.

Table 3. Comparison of memory consumption and execution time with different numbers of iterations and constant sizes of the Beginning Population. Source: Own study [28-29].

Iterations size	100		500		1000	
BP size	100		100		100	
A programming language	Memory usage [MB]	Execution time [s]	Memory usage [MB]	Execution time [s]	Memory usage [MB]	Execution time [s]
Python	63.28	14.16	63.05	67.94	63.5	133.24
Matlab	0.0152	8.36	0.089	4.25	0.35	8.59
C#	0.038	3.0	0.039	15.0	0,0349	30.0

Table 4. Comparison of memory consumption and execution time with constant numbers of iterations and different sizes of the Beginning Population. Source: Own study [28].

Iterations size	1000		1000	
BP size	1000		500	
A programming language	Memory usage [MB]	Execution time [s]	Memory usage [MB]	Execution time [s]
Python	72.21	2 118.49	67.35	420.52
Matlab	0.77	201.0	0.47	50.42
C#	0.77	375.0	0.033	161.0

7 Discussion of selected research results

In comparative studies of selected programming environments from the point of view of the duration of the execution time of the Evolutionary Algorithm, Python showed high efficiency with smaller sizes of the Parent Population in EA. This situation means that for a small number

of individuals in PP and a relatively small number of iterations (epochs), the time needed to perform EA was acceptable, which results from its simplicity in implementing this type of algorithms. However, as the PP size and number of iterations increased, the AE execution time increased significantly, and furthermore, the interpreted Python code execution approach resulted in higher CPU and memory resource consumption for more complex computational operations.

Therefore, Python's performance may be sufficient for smaller scale applications or for evolutionary problems with low computational complexity. However, for more computationally intensive tasks, such as large population sizes or long series of iterations in evolutionary algorithms, it is necessary to consider alternative programming environments with higher performance. In turn, Matlab showed higher efficiency with larger population sizes compared to Python and C#. One of the key factors influencing the higher performance of the Matlab language is the built-in support for matrix and array operations through the possibility of using numerous algorithms included in the MATLAB environment toolboxes, which contributes to faster data processing compared to the Python environment and the C# language environment.

In the case of evolutionary algorithms, which often operate on large data sets represented in matrix and table form, such efficiency significantly speeds up the computational process. Additionally, the MATLAB environment offers a wide range of built-in m-files for data analysis and modeling, which can facilitate the implementation and testing of evolutionary algorithms. Thanks to this, research with larger PR sizes may be easier to conduct and more effective. It is also worth noting that the performance of AE implemented in Matlab may also depend on specific operations and the type of calculations, and in some cases, especially with more complex computational operations, the performance may be similar to the performance of other programming languages, or even slightly lower due to the characteristics of the interpreted language.

It is also worth noting that C# showed a longer execution time for the evolutionary algorithm than Matlab, but significantly shorter than Python. In summary, the choice of programming language to implement evolutionary algorithms should take into account not only execution time but also memory consumption. For applications requiring intensive computation and large data management, MATLAB and C# may offer better memory efficiency compared to Python.

References

1. Abadi, M., TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems, preprint arXiv:1603.04467v2 [cs.DC], 2016, pp. 1-19.
2. Albahari, J., Albahari, B., C# 9.0 in a Nutshell: The Definitive Reference, O'Reilly Media, Inc, USA, 2021, pages 1041.
3. Arabas, J., Wykłady z algorytmów ewolucyjnych (in Polish), in English: Lectures on evolutionary algorithms. WNT, Warszawa, 2003, pages 303.
4. Attaway, S., MATLAB: A Practical Introduction to Programming and Problem Solving (5th ed.). Butterworth-Heinemann, Oxford, 2020, pages 626.

5. Deitel, P., Deitel, H. M., Liperi, D., Python for Programmers. Pearson, 2020.
6. Griffiths, I., Programming C# Build Cloud, Web, and Desktop Applications, Reilly Media, 2019.
7. Helt P., Parol M., Piotrowski p., Metody sztucznej inteligencji w elektroenergetyce (in Polish), in English: Artificial Intelligence Methods in Power Engineering, OW PW, Warszawa 2000.
8. Hongwei, H., Jin, X., New Hybrid Genetic Algorithm for Vertex Cover Problems. Journal of Systems Engineering and Electronics, 14(4)2023, pp. 90-94.
9. Kłopotek M., Tchórzewski J., The Concept of Discoveries Evolving Neural Net,. Advances in Soft Computing, Vol. 17, IPI PAN, Warszawa 2002, pp. 165-174.
10. Kokosa, K., Zaawansowane zarządzanie pamięcią w .NET: Lepszy ISBN, wydajność i skalowalność (in Polish), in English: Advanced Memory Management in .NET: Better ISBN, Performance and Scalability. Appres, 2020.
11. Korbicz J., Obuchowicz A., Uściński D., Sztuczne sieci neuronowe. Podstawy i zastosowania (in Polish), in English: Artificial Neural Networks. Fundamentals and Applications, Problemy Współczesnej Nauki, Teoria i zastosowania, Informatyka, AOW PLJ, Warszawa 1994, pages 251.
12. Marlęga R., Comparative Study of Selected Methods of the Analytical, Identification and Neural Modeling on example of the Day-Ahead Market Systems, Chapter 6 [in:] Modeling and Analysis of Intelligent Information Systems, Monograph No. 2 in Series: Intelligent Systems and Information Technologies [ed.] J. Tchórzewski, P. Świtalski, WN UwS, Siedlce, 2023, pp. 145-178.
13. Merelo-Guervós J. J., Blancas-Álvarez, I., et all, A comparison of implementations of Basic Evolutionary Algorithm Operations in Different Languages; IEEE Congress on Evolutionary Computation (CEC), 2016.
14. Michalewicz Z., Algorytmy genetyczne + struktury danych = programy ewolucyjne (in Polish), in English: Genetic algorithms+data structures=evolutionary programs. WNT, Warszawa 2004, pages 432.
15. Mielczarski W., Rynki energii elektrycznej. Wybrane aspekty techniczne i ekonomiczne (in Polish), in English: Electricity markets. Selected technical and economic aspects. ARE S. A., Warszawa, 2000, pages 321.
16. Osowski S., Sieci neuronowe do przetwarzania informacji (in Polish), in English: Neural networks for information processing, OW PW, Warszawa, 2013, pages 422.
17. Ostaszewski M., Seredyński F., Bouvry P., Coevolutionary-based mechanisms for network anomaly detection. Journal of Mathematical Modeling and Algorithms, Springer, 6(4), 2007, 411-431.
18. Price M. J., C# 8.0 and .NET Core 3.0. Modern Cross-Platform Development: Build applications with C#, .NET Core, Entity Framework Core, ASP.NET Core, and ML.NET. Packt Publishing, 2019.
19. Ruciński D., Neural modelling of electricity prices quoted on the Day Ahead Market of TGE S.A. shaped by environmental and economic Factors, Studia Informatica. Systems and Information Technology, Vol. 1-2(24)2020, pp. 25-35.
20. Ruciński D., Modelling of the Day-Ahead Market on the Polish Power Exchange on the Example of Selected Neural Network, Chapter 4 [in] Theory and Application of Artificial Intelligence Methods, Monograph No. 1 in Series: Intelligent Systems and Information Technologies [ed.] J. Tchórzewski, P. Świtalski, WN UPH, Siedlce, 2022, pp. 85-117.
21. Tadeusiewicz R., Elementarne wprowadzenie do techniki sieci neuronowych z przykładowymi programami (in Polish), in English: An elementary introduction to neural network techniques with sample programs, AOW PLJ, Warszawa 1998, pages 312.
22. Tchórzewski J., Systemowy algorytm ewolucyjny (SAE) (in Polish), in English: Systemic Evolutionary Algorithm (SAE) . Bio-Algorithms and Med-Systems, 2005.
23. Tchórzewski J., Systemic Method of Structure and Parameters Researching for Beginning Population Building of Evolving Algorithm SAE on the Example of Electricity Market. Polish Journal of Environmental Studies, 2008.
24. Tchórzewski J., Metody sztucznej inteligencji i informatyki kwantowej (in Polish), in English: Artificial Intelligence and Quantum Computing Methods WN UPH, Siedlce, 2021, pages 343.

25. Towarowa Giełda Energii S.A., www.tge.pl [dostęp: lata 2023 - 2024].
26. Troelsen, A., Japikse, P., Pro C# 8 with .NET Core: Foundational Principles and Practices in Programming. Apress, 2019.
27. Wang, Y., Zhu, Q., A Hybrid Genetic Algorithm for Flexible Job Shop Scheduling Problem With Sequence-Dependent Setup Times and Job Lag Times. IEEE Access, 9, 2021, pp. 104864-104873.
28. Zakrzewski, M., Analiza możliwości Algorytmu Ewolucyjnego do poprawy modeli systemów z wykorzystaniem wybranych języków programowania (in Polish), in English: Analysis of the Evolutionary Algorithm's Capability to Improve System Models Using Selected Programming Languages, Praca magisterska pod kierunkiem dr hab. inż. Jerzego Tchórzewskiego, prof. uczelni napisana w Instytucie Informatyki na Wydziale Nauk Ścisłych i Przyrodniczych, UwS, Siedlce, 2024.
29. Tchórzewski, J., Zakrzewski, M., Selection of the Programming Environment for EA Implementation in order to Improve ANN as a Neural Model of the TGE S.A. DAM System (Extended Abstract), [in:] Proceedings of the 2nd Conference on Intelligent Systems and Information Technologies Logic, Knowledge, and Reasoning in Intelligent Systems Extended abstracts, [ed.] Niewiadomski A., Wawrzyńczak-Szaban A., Wydawnictwo Naukowe Uniwersytetu w Siedlcach, Siedlce 2024, pp.54-63.