Mathematical Statistician and Engineering Applications
ISSN: 2094-0343
DOI: https://doi.org/10.17762/msea.v70i2.2458

Evolutionary approach in Assembler by Implementation of Neural Networks

Aditya Verma
Asst. Professor, Department of Comp. Sc. & Info. Tech., Graphic Era Hill University, Dehradun, Uttarakhand India 248002

Article Info
Page Number:1678 - 1685
Publication Issue:
Vol 70 No. 2 (2021)

Article History
Article Received*: 05 September 2021*
Revised*: 09 October 2021*
Accepted*: 22 November 2021*
Publication*: 26 December 2021*

Abstract

A neural network is represented using the Assembler Encoding approach as a straightforward computer programme called the Assembler Encoding Programme. The goal of this programme is to create a Network Definition Matrix, which has all the data required to build a neural network. In order to create the programmes and subsequently the neural networks, evolutionary techniques are used.Finding the ideal number of neurons for a neural network is one of the difficulties in Assembler Encoding. The current implementation of Assembler Encoding relies on an inefficient and time-consuming way to deal with this issue. The report offers four other approaches to address this problem, though. Experiments were performed utilising a predator-prey problem to assess various solutions. The network's design, including the number of layers, the number of neurons in each layer, and the connections between neurons, is specified by the Assembler Encoding Programme. These rules are expressed in a low-level language that resembles machine code very closely.The Network Definition Matrix is produced by running the Assembler Encoding Programme, and it acts as a design for the neural network. Typically, the matrix contains data on the biases and weights of the connections between neurons.Thepaper track the Assembler Encoding Programmes are frequently improved and evolved using evolutionary techniques like genetic algorithms. The programmes are iteratively enhanced through evolutionary processes to create neural networks that display desirable behaviour for certain tasks or issues.

I. Introduction

The application of evolutionary neural networks in the context of assembler encoding has drawn a lot of interest from researchers working in artificial intelligence and machine learning. By representing neural networks as assembly language programmes, Assembler Encoding enables the creation of Network Definition Matrices, which include the data required to build neural networks. The Assembler Encoding paradigm provides a strong framework for building and optimising neural networks by utilising evolutionary techniques like genetic algorithms.The implementation of neural networks presents a number of difficulties, and evolutionary methods have shown to be effective in overcoming these difficulties. Finding the ideal number of neurons for a network is a key issue. Due to the extensive search space of potential architectures, this effort has historically considered time-consuming and unfeasible.However, Assembler Encoding offers a way to automatically evolve the architecture of neural networks by utilising evolutionary approaches, doing away with the necessity for manual trial-and-error procedures [1].

In the Assembler Encoding (AE) technique [2], various aspects, including the number of neurons, the type of each neuron, connectivity, and the values of parameters for each neuron and connection, need to be determined in order to properly define an Artificial Neural Network (ANN). With the exception of the number of neurons, the majority of this data in AE is contained inside the elements of the Network Definition Matrix (NDM). The number of neurons in the ANN depends on the size of the NDM.

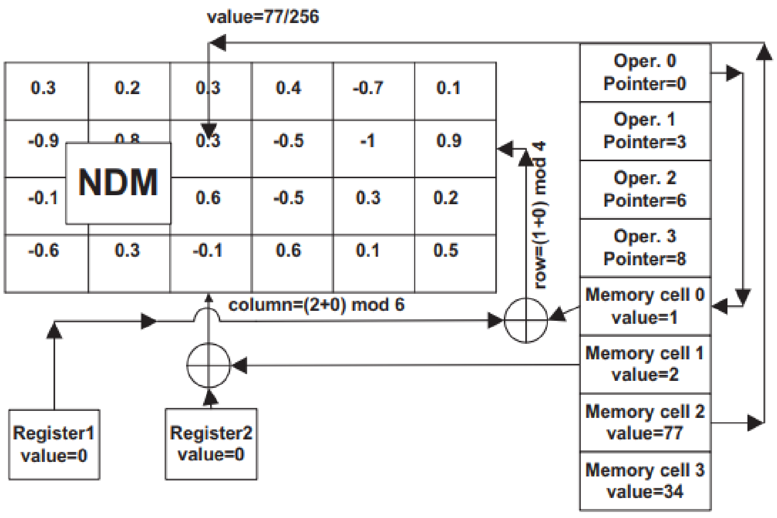


Figure 1: Fundamental of Assembler Encoding

The NDMs and ANNs increase gradually throughout the evolutionary process in the present version of AE. All Assembler Encoding Programmes (AEPs) are initially designed to run on NDMs with a minimum allowable number of rows and columns. As a result, the ANNs built at the start of evolution only have input and output neurons.The NDMs are expanded by adding one row and one column if, over a predetermined period, none of the AEPs are able to produce satisfactory solutions. One additional neuron has been added to the ANNs as a result of this expansion. The NDMs continue to grow until they are able to represent ANNs with the maximum permissible number of neurons.During the evolutionary process, AE enables the exploration of alternative network sizes by dynamically expanding the NDMs and ANNs. This method permits the evolution of ANNs with the right level of complexity to address certain tasks or issues. The steady development technique makes sure that the ANNs can adjust and learn as the problem's complexity rises[4],[5].

II. Basics of Assembler Coding

An Artificial Neural Network (ANN) is represented as an Assembler Encoding Programme (AEP) in Assembler Encoding (AE), which has two sections: the code part (which includes operations) and the memory part (which includes data). The Network Definition Matrix (NDM) generation and value input are the main responsibilities of the AEP. The activities listed in the code component of the AEP are used to accomplish this. The procedures are carried out in a sequential order, and as they do so, they make use of the data that was stored at the AEP's conclusion. The process of constructing the NDM is deemed complete once the final procedure has finished running.

Assembler Encoding (AE) [7] uses a jump operation known as JMP in addition to operations that change the Network Definition Matrix's (NDM) content. The Assembler Encoding Programme (AEP) can reuse the same code at several points throughout the NDM thanks to the JMP function. Once the jump has been done, this is accomplished by altering the values of the registers. Figure 2 shows an illustration of the jump procedure in use.The steps taken by the programme shown in the picture are as follows: First, the values in both registers are set to 0. The top left corner of the NDM changes after the first two operations have been completed. The jump operation designated as JMP(0,2,0,*) is carried out in the following phase.

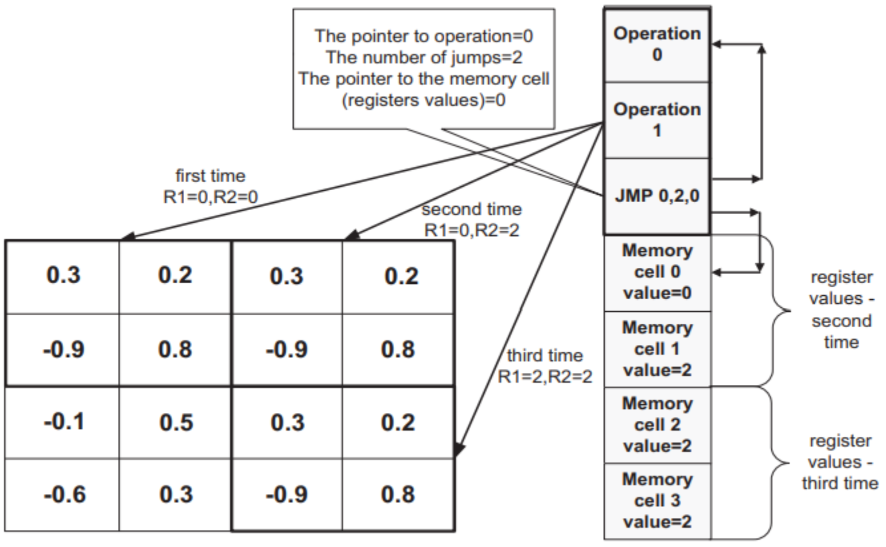


Figure 2: Systematic Example of JUMP Operation

The Network Definition Matrix (NDM) must contain all the data required to build the network in order to enable the building of an Artificial Neural Network (ANN) based on the NDM. The NDM can take the form of a traditional Connectivity Matrix (CM) when the objective is to construct the fundamental structure of an ANN without explicit weights for interneuron connections. The number of rows and columns in the CM are equal to the number of neurons in the network, and it is a square binary matrix. In the i-th column and j-th row of the CM, a value of "1" denotes a connection between the i-th and j-th neurons, whereas a value of "0" denotes the absence of such a link.However, the NDM should be a real-valued extension of the CM when the goal is to build a complete ANN with specified values for weights, neuron types, and neuron parameters. The definitions of individual neurons are included in additional columns or rows in this enhanced NDM. These [10] additional entries detail the particular traits of each neuron, such as its parameters, type, and weight values.By utilising this expanded version of the NDM, all ANN specifications can be included in the construction process, resulting in the development of a fully functional neural network with specified interneuron connections, weights, and other pertinent properties.

III. ANN's encoding size in Assembler Encoding

The number of neurons, the architecture of interneuron connections, the types of neurons, and the properties of both neurons and connections must all be established in order to fully define an Artificial Neural Network (ANN). All of the requirements for the ANN are completely stated in the Network Definition Matrix (NDM) of Assembler Encoding (AE). The maximum number of neurons is determined by the NDM size, while the remaining ANN parameters are determined by the NDM components [11].The Assembler Encoding Programme (AEP) makes it easier to create NDMs. The AEP must have information on the matrix size, or the intended number of neurons in the ANN, in order to start the NDM.

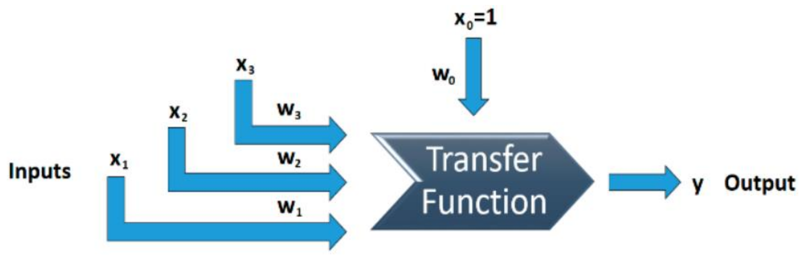


Figure 3: Representation of Artificial Neural Network

All of the NDMs are expanded by adding one row and one column if none of the AEPs are able to provide an acceptable ANN over an extended period of time. This growth is equivalent to adding one hidden neuron to each ANN. Up until an ANN is formed that fulfils the desired expectations and achieves the necessary network complexity and performance, the NDMs and ANNs are expanded.During the evolutionary phase, AE enables the exploration of alternative network sizes and configurations by gradually extending the NDMs and ANNs. Because of their adaptability, ANNs can develop over time, becoming more complicated and capable as they take on new challenges[12],[13].

In the context of Assembler Encoding, the study proposes four alternative methods for building Artificial Neural Networks (ANNs). From the very beginning of the evolutionary process, each scheme enables the creation of ANNs with a specific number of neurons. No matter the point of formation, it is assumed in all of these schemes that every Assembler Encoding Programme (AEP) has the capacity to produce an ANN that is distinct in terms of connectivity, the weights of interneuron connections, the types of neurons, and the overall number of neurons [14].

IV. Methodology

1. Neural Network Assumptions:

Artificial neural networks (ANNs) were used in the study to evaluate the order of merging assembly components. To do this, a predetermined set of training examples and input and output properties for the network were selected. A [15] number of characteristics, including the quantity of tool changes, the frequency of assembly direction changes, and assembly stability, were included in the input data. In contrast, the output data included a breakdown of assembly time into other categories.Making sure there were enough training samples and creating linkages between the data were important aspects of the study. This process was essential for getting the desired outcomes and increasing the neural network's effectiveness.By multiplying it by the range of numerical data and scaling it in accordance with the formula, the difference between the scaled value and minimum value is determined.

$$X^{*}= \frac{X-\min(X)}{Max(x)-Min(x)}\text{.........}(1)$$

The process of neural network training, which minimises the prediction error function based on the sum of squares (SOS) as specified by the formula:

$$\text{Sum Of Square } = \sum_{k=1}^{n} \text{Square}(\text{Yi} - \text{Yi})\text{.........}(2)$$

The learning algorithm that a neural network uses directly affects how effective it is. In order to minimise the error function, learning algorithms typically operate on the idea of iteratively modifying the weights given to neurons. By optimising the weight values across a number of iterations, the performance of the network is to be improved. The learning algorithm aims to increase the network's precision in modelling and predicting the input data by iteratively changing the weights depending on the error between the predicted and desired outputs [16].

2. **Evolutionary Method no 1:**

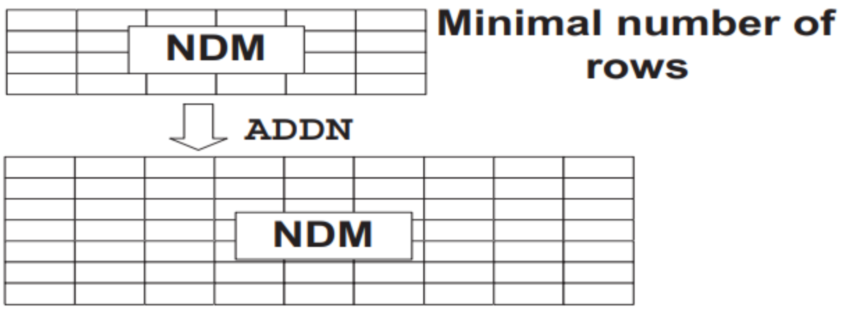


Figure 4: Description of Method no 1

In this instance, the Network Definition Matrices (NDMs) can be dynamically expanded by the Assembler Encoding Programmes (AEPs) as they run. Each AEP starts off by creating an NDM that corresponds to an Artificial Neural Network (ANN) made up only of input and output neurons. However, as shown in Figure 4, the AEPs can now use the ADDN operation to enlarge the NDM by include a predetermined number of rows and columns. New neurons that are initially disconnected from the rest of the network are incorporated into the ANN in the same way that new rows and columns are added to the NDM. A parameter of the ADDN operation controls the number of neurons that are added. It is significant to notice that the ANN's established connections are not disrupted by the insertion of new neurons. The network's structural integrity is further ensured by the subsequent removal from the network of any neurons that are isolated from the ANN's output.

3. Evolutionary Method no 2:

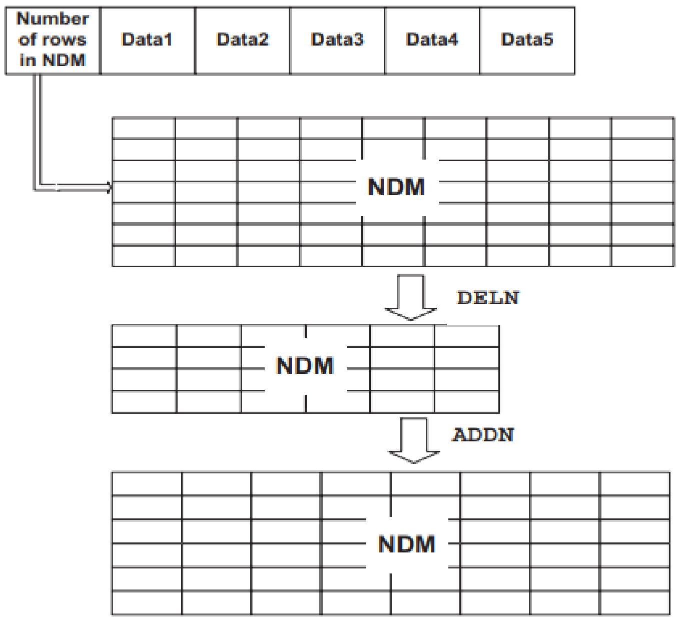


Figure 5: Description of Method no 2

Combining parts from techniques 2 and 3, this approach. The Network Definition Matrix (NDM) size is initially encoded in a chromosome with data. The [17] Assembler Encoding Programme (AEP) can then change the size of the NDM using the ADDN and DELN operations (as shown in Figure 5). This method enables the NDM and, in turn, the Artificial Neural Network (ANN) to be dynamically adjusted.The ANN's existing connections are unaffected as a result of the ADDN operation's addition of new neurons to the network. The DELN procedure, on the other hand, uses the neuron's unique identification number as a parameter to delete a specific neuron from the ANN.

4. Evolutionary Method no 3:

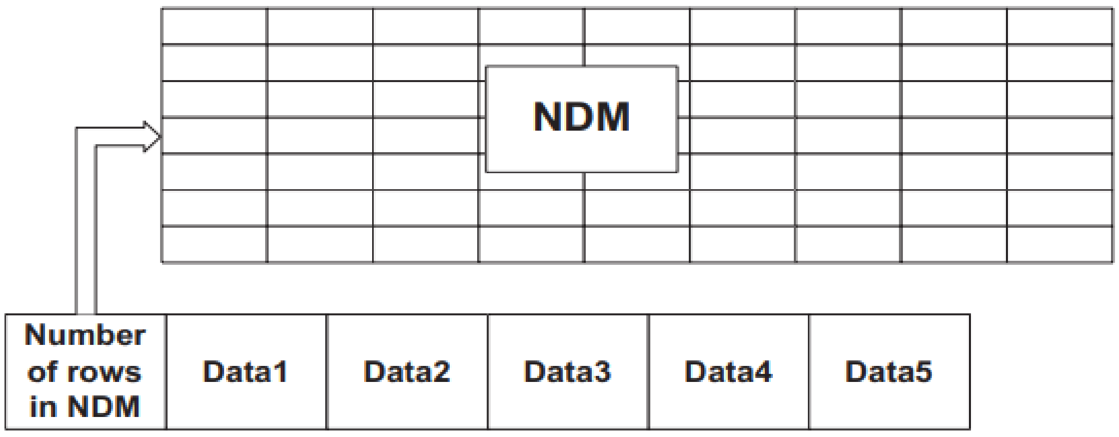
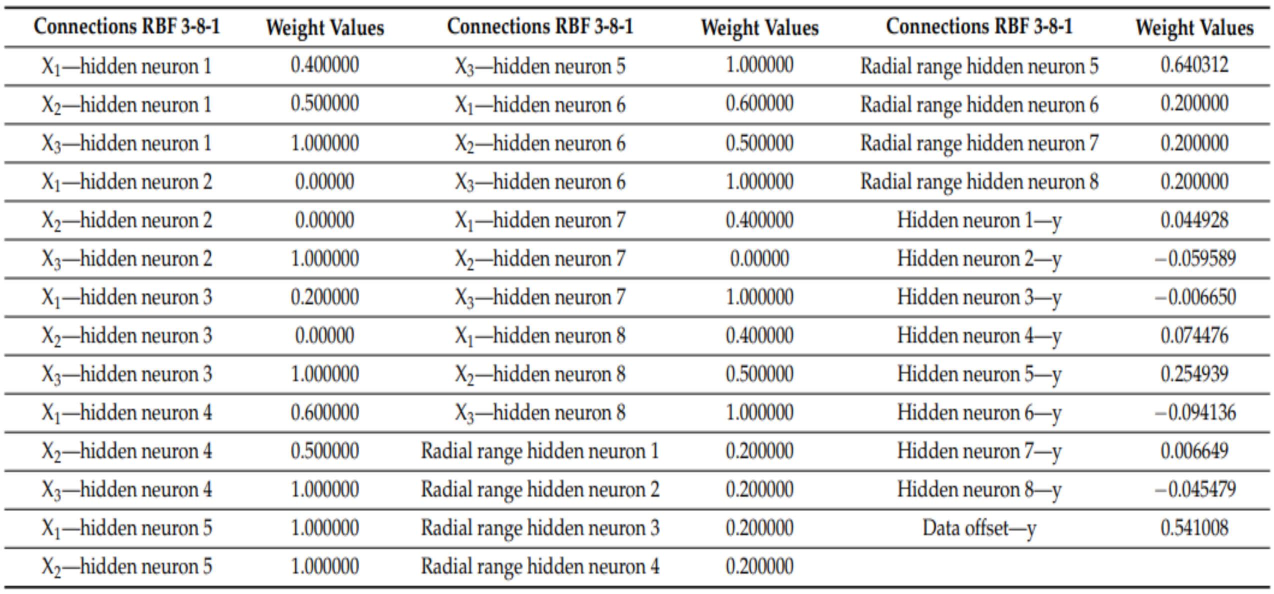


Figure 6: Description of Method no 3

The Network Definition Matrix (NDM) size information is incorporated in a data-filled chromosome in this manner, as shown in Figure 4. In Assembler Encoding, chromosomes typically only include information that is unique to the AEP. However, in the approach under consideration, the data chromosome contains the information regarding the NDM size.Combining a collection of chromosomes that represent operations with one chromosome that contains the data creates the AEP. The programme initialises the NDM after the AEP integration is finished. The first section of the data chromosome contains the size of the NDM [18].



Connections RBF 3-8-1	Weight Values	Connections RBF 3-8-1	Weight Values	Connections RBF 3-8-1	Weight Values
X_1—hidden neuron 1	0.400000	X_3—hidden neuron 5	1.000000	Radial range hidden neuron 5	0.640312
X_2—hidden neuron 1	0.500000	X_1—hidden neuron 6	0.600000	Radial range hidden neuron 6	0.200000
X_3—hidden neuron 1	1.000000	X_2—hidden neuron 6	0.500000	Radial range hidden neuron 7	0.200000
X_1—hidden neuron 2	0.00000	X_3—hidden neuron 6	1.000000	Radial range hidden neuron 8	0.200000
X_2—hidden neuron 2	0.00000	X_1—hidden neuron 7	0.400000	Hidden neuron 1—y	0.044928
X_3—hidden neuron 2	1.000000	X_2—hidden neuron 7	0.00000	Hidden neuron 2—y	−0.059589
X_1—hidden neuron 3	0.200000	X_3—hidden neuron 7	1.000000	Hidden neuron 3—y	−0.006650
X_2—hidden neuron 3	0.00000	X_1—hidden neuron 8	0.400000	Hidden neuron 4—y	0.074476
X_3—hidden neuron 3	1.000000	X_2—hidden neuron 8	0.500000	Hidden neuron 5—y	0.254939
X_1—hidden neuron 4	0.600000	X_3—hidden neuron 8	1.000000	Hidden neuron 6—y	−0.094136
X_2—hidden neuron 4	0.500000	Radial range hidden neuron 1	0.200000	Hidden neuron 7—y	0.006649
X_3—hidden neuron 4	1.000000	Radial range hidden neuron 2	0.200000	Hidden neuron 8—y	−0.045479
X_1—hidden neuron 5	1.000000	Radial range hidden neuron 3	0.200000	Data offset—y	0.541008
X_2—hidden neuron 5	1.000000	Radial range hidden neuron 4	0.200000		

Figure 6: Snapshot of Neural Network weight prediction for AE

V. Conclusion

The research compares various approaches for increasing the effectiveness of Assembler Encoding (AE) when building substantial Artificial Neural Networks (ANNs). The trials carried out in the context of the predator-prey problem shown that the gradual growth strategy employed in classical AE outperformed all offered methods for smaller ANNs, where the size of the network was sufficient to handle the given challenge.The experiments did show, however, that several of the techniques covered in the paper can be successfully integrated with the traditional form of AE. These techniques were used to estimate the initial size of the ANN. The progressive growth strategy was then used to further hone and optimise the ANN. The outcomes showed that this combination strategy can be equally successful.The outcomes showed that this combination technique can be just as successful and, more crucially, quicker than the conventional approach employed thus far.In conclusion, the combination of chosen approaches with the traditional AE approach proved to be a successful tactic, even though the steady development method continues to be superior for smaller ANNs. By determining an initial ANN size and then optimising it through steady growth, this hybrid technique enables the construction of effective and efficient networks.

References:

[1] Krawiec, K. and Bhanu, B. (2005) Visual Learning by Coevolutionary Feature Synthesis. IEEE Trans. on Systems, Man, and Cybernetics, Part B: Cybernetics 35, 409-425.
[2] Luke, S. and Spector, L. (1996) Evolving Graphs and Networks with Edge Encoding: Preliminary Report. In: J.R. Koza, ed., Late Breaking Papers at the Genetic Programming 1996 Conference. Stanford University, CA, 1214 T. PRACZYK USA. Stanford Bookstore (1996), 117-124
[3] Potter, M.A. and De Jong, K.A. (2000) Cooperative coevolution: An architecture for evolving coadapted subcomponents. Evolutionary Computation 8 (1), 1-29.
[4] Praczyk, T. (2007a) Evolving co-adapted subcomponents in Assembler Encoding. International Journal of Applied Mathematics and Computer Science 17(4).
[5] Praczyk, T. (2007b) Procedure application in Assembler Encoding. Archives of Control Science 17(LIII),1, 71-91.
[6] Praczyk, T. (2008) Modular networks in Assembler Encoding. Computational Methods in Science and Technology, CMST 14 (1), 27-38.
[7] Moriarty, D.E. and Miikkulainen, R. (1998) Forming Neural Networks Through Efficient and Adaptive Coevolution. Evolutionary Computation 5 (4), 373-399.
[8] Zeng, C.; Gu, T.; Zhong, Y.; Cai, G. A Multi-Agent Evolutionary algorIthm for Connector-Based Assembly Sequence Planning. Procedia Eng. 2011, 15, 3689–3693
[9] Suszy ´nski, M.; Zurek, J.; Legutko, S. Modelling of assembly sequences using hypergraph and directed graph. ˙ Teh. Vjesn. Tech. Gaz. 2014, 21, 1229–1233.
[10] Suszy ´nski, M.; Zurek, J. Computer aided assembly sequence generation. ˙ Manag. Prod. Eng. Rev. 2015, 6, 83–87.
[11] Murali, G.B.; Deepak, B.B.V.L.; Bahubalendruni, M.V.A.R.; Biswal, B.B. Optimal Assembly Sequence Planning To-wards Design for Assembly Using Simulated Annealing Technique. Res. Des. Communities 2017, 1, 397–407.
[12] Wang, D.; Shao, X.; Liu, S. Assembly sequence planning for reflector panels based on genetic algorithm and ant colony optimization. Int. J. Adv. Manuf. Technol. 2017, 91, 987–997.
[13]Li, X.; Qin, K.; Zeng, B.; Gao, L.; Su, J. Assembly sequence planning based on an improved harmony search algorithm. Int. J. Adv. Manuf. Technol. 2016, 84, 2367–2380
[14] Biswal, B.B.; Pattanayak, S.K.; Mohapatra, R.N.; Parida, P.K.; Jha, P. Generation of optimized robotic assem-bly sequence using immune. In Proceedings of the ASME, International Mechanical Engineering Congress and Exposition, Houston, TX, USA, 9–15 November 2012;
[15] Sinano ˘glu, C.; Börklü, H.R. An assembly sequence-planning system for mechanical parts using neural network. Assem. Autom. 2005, 25, 38–52
[16] Chen, W.C.; Tai, P.H.; Deng, W.J.; Hsieh, L.F. A three-stage integrated approach for assembly sequence planning using neural networks. Expert Syst. Appl. 2008, 34, 1777–1786
[17] S ˛asiadek, M. Planning and analysis of mechanical assembly sequences in design engineering—Part I: The Method. Teh. Vjesn. Tech. Gaz. 2015, 22, 337–342.
[18] Butlewski, M.; Suszy ´nski, M.; Czernecka, W.; Pajzert, A.; Radziejewska, M. Ergonomic criteria in the optimization of as-sembly processes, Cristina Feniser. In Proceedings of the 6th RMEE2018—Performance Management or Management Performance—Todesco; Publishing House: Cluj-Napoca, Romania, 2018; pp. 424–431
[19] Guo, J.; Sun, Z.; Tang, H.; Yin, L.; Zhang, Z. Improved Cat Swarm Optimization Algorithm for Assembly Sequence Planning. Open Autom. Control. Syst. J. 2015, 7, 792–799.