ZESZYTY NAUKOWE AKADEMII MARYNARKI WOJENNEJ
ROK LI NR 1 (180) 2010

Tomasz Praczyk
Akademia Marynarki Wojennej

PROPERTIES OF ASSEMBLER ENCODING

ABSTRACT

Assembler Encoding is a new neuro-evolutionary method. In the paper, characterization of the method is given. To characterize Assembler Encoding, the following properties were taken into consideration: Completeness, Closure, Compactness, Scalability, Multiplicity, Ontogenetic Plasticity, Modularity, and Redundancy.

Keywords:
evolutionary neural networks.

INTRODUCTION

Artificial Neural Networks (ANNs) are created by means of different techniques. One of them are Genetic Algorithms (GAs). GA processes a population of genotypes that typically encode one phenotype, although encoding several phenotypes is also possible. In the neuro-evolution, genotypes are encodings of corresponding ANNs (phenotypes). The evolutionary procedure involves selecting genotypes (encoded ANNs) for reproduction based on their fitness, and then by introducing genetically changed offspring into a next population. Repeating the whole procedure over many generations causes the population of encoded ANNs to gradually evolve into individuals corresponding to high fitness phenotypes (ANNs).

In principle, all the existing ANN encoding methods can be divided into two main classes, i.e. direct encodings and indirect encodings. In the direct encodings, all the information necessary to create ANN is directly stored in chromosomes. Thus, to encode larger ANNs larger chromosomes are necessary, which is the main drawback of the direct encodings. In the indirect encodings, chromosomes are recipes how to create ANN. Such encodings can be used to create larger ANNs by means of relatively short chromosomes.

The paper presents a new indirect method to encode ANNs. The method is called Assembler Encoding (AE). AE originates from the cellular [2] and edge encoding [3], although, it also has features common with Linear Genetic Programming presented, among other things, in [5]. AE represents ANN in the form of a program called Assembler Encoding Program (AEP). The structure of AEP is similar to the structure of a simple assembler program. AEPs are formed by means of GAs. The task of each AEP is to create Network Definition Matrix (NDM) which stores all the information necessary to create ANN. In AE, the process of ANN construction consists of three stages. First, GA is used to produce AEPs. Next, each AEP creates and fills up NDM. Then, the matrix is transformed into ANN.

In the paper, characterization of AE is given. The description of the method takes into consideration the following properties: Completeness, Closure, Compactness, Scalability, Multiplicity, Ontogenetic Plasticity, Modularity, and Redundancy. All the properties specified above were proposed in [1].

FUNDAMENTALS OF AE

In AE [7], [8], [9] ANN is represented in the form of AEP. AEP is composed of a part including operations and a part including data. The task of AEP is to create and to fill in NDM with values. To this end, AEP uses the operations. The operations are run in turn. When working the operations can use data located at the end of AEP. Once the last operation ends its work the process of creating NDM is completed and the matrix is transformed into ANN (fig. 1).

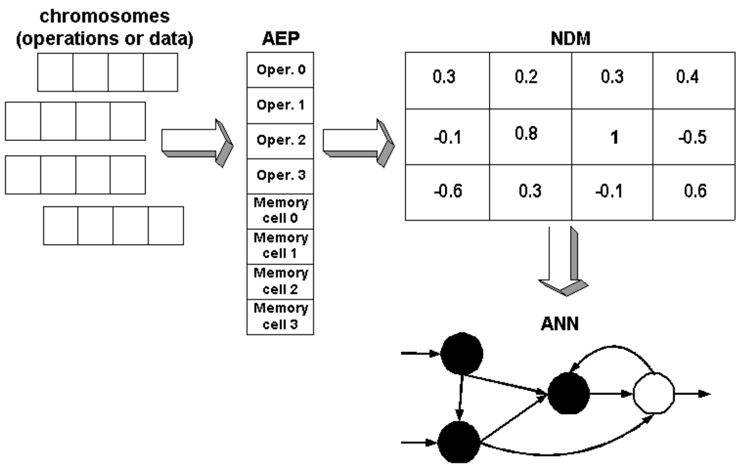

Fig. 1. Diagram of AE

AEPs can use various operations. The main task of most of operations is to modify NDM. The modification can involve a single element of NDM or group of elements. Fig. 2 presents example operation used in AE.

```
CHGC0(p0,p1,p2,*)
{
column=(abs(p0)+R2)mod NDM.height;
numberOfIterations=abs(p2)mod NDM.width;
for(i=0;i<=numberOfIterations;i++)
      {
      row=(i+R1)mod NDM.width;
      NDM[row,column]=D[(abs(p1)+i)mod D.length]/Max_value;
      }
}
```

Fig. 2. CHGC0 operation changing a part of column of NDM:
D[i] — ith data in AEP, *D.length* — the number of memory cells, *Max_value* — scaling value

CHGC0 (fig. 2) modifies elements of NDM located in a column indicated by parameter p_0 and the register R_2. The number of elements being updated is stored in a parameter p_2. An index of the first element being updated is located in the register R_1. To update elements of NDM CHGC0 uses data from AEP. An index to a memory cell including the first data used by CHGC0 is stored in p_1.

Once AEP finishes its work the process of transforming NDM into ANN is started. To make it possible to construct ANN based on NDM the latter has to include all the information necessary to create ANN. To create the same skeleton of ANN, i.e. ANN without determined weights of interneuron connections, NDM can take the form of the classical connectivity matrix (CM) [4], i.e. a square, binary matrix of a number of rows and columns equal a number of neurons. The value '1' in ith column and jth row of such a matrix means a connection between ith neuron and jth neuron. In turn, the value '0' means lack of the connection between these neurons. When the purpose is to create complete ANN with determined values of weights, types of neurons, parameters of neurons then NDM should take the form of a real valued variety of CM with extra columns or rows containing definitions of individual neurons.

In AE, the evolution of AEPs proceeds according to a scheme proposed by Potter and De Jong [6]. The scheme assumes a division of evolutionarily created solution into parts. Each part evolves in a separate population. The complete solution is formed from selected representatives of each population. To use the scheme above in relation to AEPs it is necessary to divide them into parts. For AEPs the division is natural. The operations and data make up natural parts of AEPs. Since the evolutionary scheme chosen assumes evolution of each part in a separate population AEP consisting of n operations and a sequence of data evolves in n populations with operations and one population with data (fig. 3).

During the evolution AEPs expand gradually. Initially, all AEPs include one operation and a sequence of data. The operations and the data come from two different populations. When the evolution stagnates, i.e. lack of progress in fitness of generated solutions is observed over some period, a set of the populations containing the operations is enlarged by one population. This procedure extends all AEPs created by one operation.

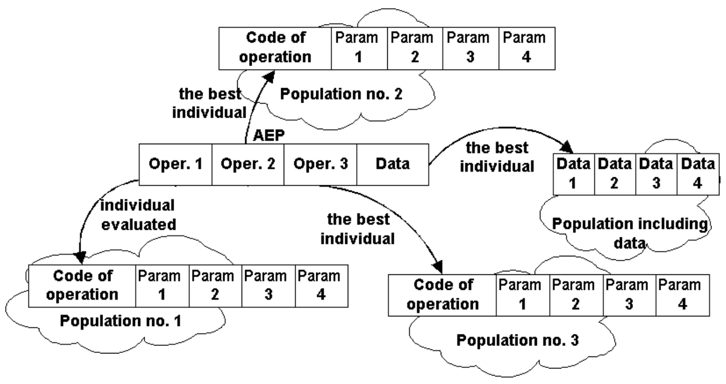

Fig. 3. Evolution in AE for $n = 3$

PROPERTIES OF AE

To start characterization of AE, first a few definitions have to be presented.

G — the space of genotypes representable in the encoding scheme, i.e. in the scheme used to encode an ANN into a genotype. In AE, G is the set of all possible AEPs.

$p = D(g, E)$ — the decoding procedure that produces the phenotype p, i.e. ANN, corresponding to the genotype g under the influence of the environment E. D may be stochastic, i.e. it can generate different phenotypes p for the genotype g.

P — the space of all phenotypes (ANN) that can be constructed by means of the encoding scheme:

$$\forall p \in P \ \exists g \in G \ \ p = D(g, E). \tag{1}$$

G_p — the set of genotypes representing the phenotype p:

$$\forall g \in G_p \ \ p = D(g, E). \tag{2}$$

S — the set of ANNs that satisfy a desired performance criterion.

A — the set of acceptable ANNs (e.g. feed-forward ANNs, two factors can influence the acceptance of ANN: the number of input and output neurons and the existence of feedback connections).

Property 1 (Completeness)

An encoding scheme is complete with respect to the set H if any element of the set can be encoded, i.e. if:

$$\forall s \in H \ \exists g \in G \ s = D(g, E). \tag{3}$$

Proposition 1 *AE is complete with respect to the set* S.

Because AE can produce any ANN, consequently, it can also generate ANN from S. In order to produce ANN composed of *M* neurons, the following AEP can be used: {*M*(*M*+3)*CHG|Data}, i.e. *M*(*M*+3) time execution of CHG. In AE, ANN under consideration is represented in the form of NDM of size *M*(*M*+3). Each cell in the matrix determines either synaptic weight between neurons or the value of a parameter of a neuron. To fill in the whole NDM with different values the AEP above runs CHG operation *M*(*M*+3) times. Each run fills other cell of NDM with a different value.

Property 2 (Closure)

An encoding scheme is completely closed with respect to the set H if every possible code develops ANN in this set, i.e. if:

$$\forall g \in G \ D(g, E) \in H. \tag{4}$$

Proposition 2 *AE is completely closed with respect to the set* A.

In AE, the number of neurons in ANN depends on size of NDM. The first rows and columns of the basic part of NDM correspond to input neurons while the last rows and columns correspond to output neurons. The remaining rows and columns store the information about hidden neurons. In AE, all AEPs created at the same moment of the evolution act on NDMs of the same size. In the beginning, NDMs are used of size corresponding to the smallest acceptable ANNs, i.e. ANNs including only input and output neurons. If all ANNs of minimal acceptable structure, created within an assumed period of the evolution, cannot accomplish a task, then, in the next stage of the evolution, all NDMs are augmented to represent ANNs including one neuron more. This process is continued to some reasonable limit of the number of neurons. Given all the process of creating ANNs described above, it is necessary to state that AE always produces acceptable ANNs in terms of the number of inputs and outputs.

AE assumes that each operation from AEP can change any element in NDM, i.e. both the element determining a feedback connection and the element defining a forward connection. Hence, each NDM produced by means of AE usually represents recurrent ANN (RANN). The simplest method to obtain feed-forward ANN (FFANN) is to ignore all feedbacks in NDM. Unfortunately, such solution leads to the situation in which a part of operations in AEP can be completely or partially unproductive. In order to avoid such undesirable situation we can force every operation which acts in the impermissible part of NDM, i.e. in the part defining feedback connections, to act in the acceptable area. We can use a special projection function which changes each impermissible address into address indicating the acceptable place in NDM. Assuming that the bottom fragment of NDM determines FFANN, we can define the projection function, for example in the following way:

$$F_{pr}((i,j)) = \begin{cases} (i,j) \text{ if } i > j \\ (j,i) \text{ if } i < j \\ (i+1,j) \text{ if } i = j \end{cases} . \tag{5}$$

Another method to produce NDMs defining exclusively FFANNs is to assign low fitness to each AEP generating RANN. As a result, all operations acting in the part of NDM defining feedbacks have a little chance to survive and once they emerge in corresponding populations they quickly disappear from them. In consequence, a massive majority of AEPs generated during the evolution solely use operations acting in the part of NDMs responsible for creating ANNs of feed-forward topology.

Property 3 (Compactness)

The encoding scheme ES^1 is more compact than the encoding scheme ES^2 with respect to the set H if the scheme ES^1 can encode each ANN from the set H by means of a shorter genotype than the scheme ES^2, i.e. if:

$$\forall p \in H \ \forall g \in G_p^2 \ \exists d \in G_p^1 \ |d| < |g|, \tag{6}$$

where $|g|$ and $|d|$ denote size of computer memory necessary to store genotypes g and d, respectively.

Proposition 3 *AE is more compact than CM with respect to the set of fully connected recurrent ANNs composed of at least eight neurons*.

Let us consider the case with a fully connected RANN composed of M neurons. Furthermore, let us assume that each connection in the ANN is described by a different weight. To represent such ANN and generally to represent each RANN consisting of M neurons CM always requires $(M+3)M$ elements (additional three columns correspond to the bias, the parameter and the type of each neuron). It seems that one of the shortest AEPs which may be used to fill in the whole $(M+3)M$ NDM with different values, is AEP of the following construction: {CHGR0|(M-1)*{CHGR3, CHGR2}|Data} (a description of CHGR0, CHGR3 and CHGR2 is included in Appendix 1 at the end of the paper). The program mentioned performs in the following manner. First, CHGR0 modifies a row of NDM using for this purpose data from memory. To execute this operation, 5 cells for the operation itself (one cell for code of the operation and the remaining four cells for arguments) and additionally M+3 cells in the data part of AEP are needed. In the next step, CHGR3 and CHGR2 are executed M-1 times. Each execution of the operations involves a different destination and source row. The task of CHGR3 is to transform elements from the most recently updated row to other blank row. CHGR2 adds a value to all elements of the row updated by means of CHGR3. To implement M-1 time execution of the two mentioned operations, 10(M-1) cells are required. All in all, the AEP presented above needs $5+M+3+10(M-1)=11M-2$ cells.

Comparing values $(M+3)M$ and $11M-2$ it is apparent that CM is a more compact representation than the AEP described above for each ANN consisting at most of 7 neurons. All the time, the fully connected RANN with all weights different is assumed. Eight neuron ANN is the smallest network for which the program described above is the shorter representation than CM. Generally, in the case considered above, the more neurons in ANN is the greater chance that AEPs to be more compact than CMs is.

Proposition 4 *AE is more compact than CM with respect to the set of two and more neuron ANNs in which all weights have the same value and all connections are directed to one output neuron (the output neuron is also connected with oneself).*

The ANN described above can be represented in the form of $(M+3)M$ CM in which the only elements unequal to zero are elements located in a column corresponding to the output neuron. According to the assumptions made above all elements in the column mentioned have the same value. To create such matrix, AE requires only one operation, e.g. CHGC0 (fig. 2), and one piece of data. All in all, it is 6 cells, i.e. 5 cells for the operation and 1 cell for data. Comparing values $(M+3)M$ and 6 it is necessary to state that in the currently considered case AE is able to produce shorter representations (AEPs) than CM already for ANNs consisting of 2 neurons.

Generally, it can be stated that smaller, fully connected ANNs of varied values of weights are represented in more compact form by means of CM than by means of AE. In turn, AE is more compact scheme than CM for sparsely connected ANNs, for modular ANNs, and for ANNs of greater number of neurons.

Property 4 (Scalability)

The encoding scheme ES^1 scales better than the encoding scheme ES^2 if enlargement of ANN by one element (neuron, connection) entails smaller growth of size of a genotype in the case of the scheme ES^1 than in the case of the scheme ES^2.

Proposition 5 *AE scales better than CM in the case of adding a neuron to ANN.*

In CM, the addition of a neuron to ANN entails increase in size of a genotype to $(M+4)(M+1)$ cells (enlarged CM is greater by $2M+4$ elements). To accomplish the same effect, AE does not require additional operations and data. In AE, ANNs can expand automatically after some assumed period of stagnation in the evolution (gradual growth of ANNs described above).

Proposition 6 *The scalability of AE and CM in the case of adding a connection to ANN is on the same level.*

In the case of the addition of an interneuron connection, neither CM nor AEPs have to be augmented. To add a connection, it suffices to change a single element of CM. The size of the matrix does not have to be changed. In AE, the procedure of adding a connection can, for example, consists in increasing parameter p_2 in CHGC0 (fig. 2) by one. This way, the operation can modify one element of NDM more.

Property 5 (Genotypic Multiplicity)

The encoding scheme ES exhibits a genotypic multiplicity if many different genotypes can represent the same ANN, i.e if:

$$\exists p \in P \left| \{ g \in G | (p = D(g, E)) \} \right| > 1 . \tag{7}$$

Proposition 7 *AE exhibits the genotypic multiplicity.*

AE shows genotypic multiplicity because two different AEPs can create the same ANN. For example, executing a sequence of CHG operations may yield the same result as using a single CHGC0 operation.

Property 6 (Phenotypic Multiplicity)

The encoding scheme ES exhibits a phenotypic multiplicity if one genotype can represent many different ANNs (phenotypes), i.e. if:

$$\exists g_1, g_2 \in G \; p_1 = D(g_1, E) \wedge p_2 = D(g_2, E) \wedge g_1 = g_2 \wedge p_1 \neq p_2 . \qquad (8)$$

Proposition 8 *AE does not exhibit the phenotypic multiplicity.*

AE is not phenotypically multiple because AEPs perform in a deterministic manner (no random factors have influence on behavior of AEPs; decoding procedure *D* is deterministic). They change NDMs always in the same way. One and the same AEP always produces the same NDM and in consequence the same ANN.

Property 7 (Ontogenetic Plasticity)

The encoding scheme ES exhibits an ontogenetic plasticity if determining the phenotype (ANN) corresponding to a given genotype is influenced by the environment.

Proposition 9 *AE exhibits the ontogenetic plasticity.*

The environment can influence the shape of ANN in two situations. The first of them is a decoding process of ANN. The second situation is a learning process of ANN. With regard to the decoding process, in AE, it is completely independent on the environment. To produce ANN, AE merely uses the information that is included in AEP. No other factors have the influence on the process of decoding ANN. In AE, the only method that enables the environment to influence the shape of ANN is to apply the learning process. Hebbian learning is an example of learning used in AE [10]. In the Hebbian learning, the architecture of ANN is variable. It undergoes changes according to Hebbian rules assigned by NDM to each interneuron connection. The Hebbian rules adjust weights of interneuron connections to input signals coming from the environment.

Property 8 (Modularity)

ANN_1 is modular if it incorporates several instances of a sub-network ANN_2 and the information necessary to create ANN_2 occurs in a genotype corresponding to ANN_1 only once.

Proposition 10 *AE has a potential to create modular ANNs.*

To create modular ANNs, AE can use solutions described in [9] (e.g. repeated use of data by various operations). Example of AEP creating modular ANN is presented in fig 4.

```
Operations:
0000001 0110110 1100100 1010011 1110111
0011010 1010011 0010110 1010111 0011101
Data:
0010110  0001101  0100001  0111011  1001110
0101100  1010100  1101001  0001001  1010101
1111000  0000000  0001110  0110110  0001001
0010010
```

a)

```
0 0 0 0 0 0 0 0 0 -0.15873 0.52381 0.428571 -0.666667 0.206349
0 0 0 0 0 0 0 0 0 -0.587302 0.873016 0.571429 -0.111111 0
0 0 0 0 0 0 0 0 0 0.571429 -0.444444 0.285714 0 0
0 0 0 0 0 0 0 0 0 -0.666667 0.206349 0.698413 0.444444 0
0 0 0 0 -0.793651 -0.793651 -0.793651 0 0 -0.111111 -0.15873 0.52381 0.428571 0
0 0 0 0 0 0 0 0 0 0.285714 0 -0.587302 0.873016 0.571429
0 0 0 0 0 0 0 0 0 0.698413 0.444444 0.571429 -0.444444 0.285714
0 0 0 0 0 0 0 0 0 0.52381 0.428571 -0.666667 0.206349 0.698413
0 0 0 0 0 0 0 0 0 0.873016 0.571429 -0.111111 -0.15873 0.52381
0 0 0 0 0 0 0 0 0 -0.444444 0.285714 0 -0.587302 0.873016
0 0 0 0 0 0 0 0 0 0.206349 0.698413 0.444444 0.571429 -0.444444
```

b)

Fig. 4 (a) Example of AEP representing modular ANN, (b) NDM generated by AEP presented in point (a) (the number of integer valued genes in AEP = 26; the number of elements different from 0 located above the diagonal of NDM = 64; NDM presented in point (b) represents FFANN and only elements above the diagonal decide about the architecture of ANN)

Property 9 (Genotypic Redundancy)

The encoding scheme ES exhibits a genotypic redundancy if genotypes can include redundant genes.

Proposition 11 *AE does not exhibit the genotypic redundancy.*

AE does not exhibit the genotypic redundancy since AEPs do not contain redundant operations and data.

Property 10 (Decoding Redundancy)

The encoding scheme ES exhibits a decoding redundancy if the decoding process can be repeated many times.

Proposition 12 *AE does not exhibit the decoding redundancy.*

AE does not exhibit the decoding redundancy since the decoding process is performed in AE only once. Initially, AEP fills up NDM which is next transformed into ANN. Once ANN is created, the decoding process is stopped.

Property 11 (Phenotypic Redundancy)

The encoding scheme ES exhibits a phenotypic redundancy if ANN can include redundant neurons, connections or sub-networks.

Proposition 13 *AE does not exhibit the phenotypic redundancy.*

AE does not exhibit the phenotypic redundancy since ANNs generated by AEPs do not contain redundant elements. In AE, ANNs are formed based on the information from NDMs which do not contain additional rows and columns for redundant connections or neurons.

SUMMARY

In the paper, characterization of AE was given. To characterize AE, the following properties were determined: Completeness, Closure, Compactness, Scalability, Multiplicity, Ontogenetic Plasticity, Modularity, and Redundancy. As a result of analysis made in the paper it appeared that:

- AE is complete with respect to the set of all ANNs that satisfy a desired performance criterion;
- AE is completely closed with respect to the set of all ANNs with acceptable architecture;
- AE is more compact than CM with respect to the set of fully connected recurrent ANNs composed of at least eight neurons;
- AE is more compact than CM with respect to the set of two and more neuron ANNs in which all synaptic weights have the same value and all connections are directed to one output neuron;
- AE scales better than CM in the case of adding a neuron to ANN;
- The scalability of AE and CM in the case of adding a connection to ANN is on the same level;
- AE exhibits the genotypic multiplicity;
- AE does not exhibit the phenotypic multiplicity;
- AE exhibits the ontogenetic plasticity;
- AE has a potential to create modular ANNs;
- AE does not exhibit the genotypic redundancy;
- AE does not exhibit the decoding redundancy;
- AE does not exhibit the phenotypic redundancy.

The following main conclusions spring from the analysis made in the paper:

1. Completeness and Closure of AE make it a tool which can be used to solve any problem solvable by means of ANNs.
2. AE seems to be more suited method for constructing larger ANNs than CM. In turn, CM appears to be a better solution when creating smaller ANNs. This results from comparing Scalability and Compactness of both methods.
3. Modularity is a next feature of AE which makes it an ideal tool to create ANNs of larger size.
4. A next significant characteristic of AE is its plasticity. It makes it possible to combine AE with different learning methods in the process of ANN construction.

APPENDIX

CHG — Update of an element. Both the new value and address of the element are located in parameters of the operation.

CHGR0 — Update of a certain number of elements in a row. Index of the row, index of the first element in the row to be changed, the number of changed elements and a pointer to data, where new values of elements are memorized, are located in parameters of the operation.

CHGCR — Update of a certain number of elements in a row. A new value of every element is a sum of the operation's parameter and the current value of this element. The second parameter of the operation is an index of the row. The third and fourth parameter of the operation determine the number of changed elements and index of the first element in the row to be changed, respectively.

CHGCR — A number of elements from one row are transformed to another row. Both rows are indicated by parameters of the operation. The number of transferred elements and index of the first element in the row to be transferred are also included in parameters of the operation.

REFERENCES

[1] Balakrishnan K., Honavar V., *Properties of Genetic Representations of Neural Architectures*, Proc. of the World Congress on Neural Networks (WCNN '95), 1995, pp. 807–813.

[2] Gruau F., *Neural Network Synthesis Using Cellular Encoding and the Genetic Algorithm*, PhD Thesis, Ecole Normale Superieure de Lyon, 1994.

[3] Luke S., Spector L., *Evolving Graphs and Networks with Edge Encoding: Preliminary Report*, in John R. Koza, ed., Late Breaking Papers at the Genetic Programming 1996 Conference (Stanford University, CA, USA, Stanford Bookstore, 1996), pp. 117–124.

[4] Miller G. F., Todd P. M., Hegde S. U., *Designing Neural Networks Using Genetic Algorithms*, Proceedings of the Third International Conference on Genetic Algorithms, 1989, pp. 379–384.

[5] Nordin P., Banzhaf W., Francone F., *Efficient Evolution of Machine Code for {CISC} Architectures using Blocks and Homologous Crossover*, Advances in Genetic Programming III (L. Spector and W. Langdon and U. O'Reilly and P. Angeline, 1999), pp. 275–299.

[6] Potter M. A., De Jong K. A., *Cooperative coevolution: An architecture for evolving coadapted subcomponents*, ‘Evolutionary Computation’, 2000, 8(1), pp. 1–29.

[7] Praczyk T., *Evolving co-adapted subcomponents in Assembler Encoding*, ‘International Journal of Applied Mathematics and Computer Science’, 2007, 17(4).

[8] Praczyk T., *Procedure application in Assembler Encoding*, ‘Archives of Control Science’, 2007, Vol. 17(LIII), No 1, pp. 71–91.

[9] Praczyk T., *Modular Neural Networks in Assembler Encoding*, ‘Computational Methods in Science and Technology’, 2008, 14(1).

[10] Praczyk T., *Forming Dynamic, Self-organizing Neural Networks by means of Assembler Encoding* (in review).

WŁASNOŚCI KODOWANIA ASSEMBLER

STRESZCZENIE

Kodowanie Assembler jest metodą neuro-ewolucyjną. W artykule scharakteryzowano ją, biorąc pod uwagę następujące własności: kompletność, zamknięcie, kompaktowość, mierzalność, plastyczność ontogenetyczną, modularność oraz redundancję.

Słowa kluczowe:
ewolucyjne sieci neuronowe.

Recenzent dr hab. inż. Wojciech Jędruch, prof. PG