DE GRUYTER OPEN

STUDIES IN LOGIC, GRAMMAR AND RHETORIC 40 (53) 2015

DOI: 10.1515/slgr-2015-0011

Paweł Stacewicz
Warsaw University of Technology

EVOLUTIONARY SCHEMA OF MODELING BASED ON GENETIC ALGORITHMS

Abstract. In this paper, I propose a populational schema of modeling that consists of: (a) a linear AFSV schema (with four basic stages of abstraction, formalization, simplification, and verification), and (b) a higher-level schema employing the genetic algorithm (with partially random procedures of mutation, crossover, and selection). The basic ideas of the proposed solution are as follows: (1) whole populations of models are considered at subsequent stages of the modeling process, (2) successive populations are subjected to the activity of genetic operators and undergo selection procedures, (3) the basis for selection is the evaluation function of the genetic algorithm (this function corresponds to the model verification criterion and reflects the goal of the model). The schema can be applied to automate the modeling of the mind/brain by means of artificial neural networks: the structure of each network is modified by genetic operators, modified networks undergo a learning cycle, and successive populations of networks are verified during the selection procedure. The whole process can be automated only partially, because it is the researcher who defines the evaluation function of the genetic algorithm.

Keywords: modeling, modeling schema (linear vs. populational), cognitive models, artificial intelligence, genetic algorithms, artificial neural networks.

Introduction

This paper elaborates on some of my earlier ideas concerning the methodological structure of the modeling process (see Stacewicz & Włodarczyk, 2011; Stacewicz, 2010). My aim here is to develop them into a proposal of a modeling schema that would allow for the automation of the modeling process. The schema I envisage is based on the genetic algorithm, an evolutionary computer-science method, inspired by the biological theory of natural selection, that has found many applications in a wide range of fields. The schema is also populational in character: whole populations of models (rather than single models) are considered at subsequent stages of the mod-

ISBN 978-83-7431-440-4 ISSN 0860-150X

eling process and the optimum model is produced by means of cyclically repeated operations of mutation, crossover, and selection.

Given the centrality of genetic algorithms to my account, I devote a whole section of this paper (section III) to discussing them in a semi-formal manner and characterizing in detail such procedures as mutation, crossover, and selection. As a hypothetical example of the use of the evolutionary modeling schema, I present a typical problem encountered in cognitive science of partial mind-brain modeling by means of *artificial neural networks*. The idea is that an artificial neural network that models a certain part of the brain undergoes artificial *evolution* governed by a genetic algorithm. The operations performed by the algorithm are aimed at obtaining the right pattern of connections between the neurons comprising the network.

I assume that the reader is familiar with artificial neural networks, so I only provide brief *general* information about them and illustrate it with a few examples. I concentrate on the idea of fine-tuning the network by means of genetic algorithms (as part of the so-called COGANN strategy), which I develop in the context of mind-brain modeling.

As this article is methodological in character, what I present are some general ideas and schemas rather than detailed computer science solutions. In fact, the basic assumption of this paper is that formal computer science structures, such as genetic algorithms and artificial neural networks, constitute an appropriate basis for methodological propositions, especially those concerning reconstruction or automation of modeling activities.

I. Modeling and computer science

Modeling is widespread in contemporary science in almost all fields of research, including the natural sciences, the social studies and the humanities. Understood empirically (as there is another concept of modeling used in the deductive sciences[1]), modeling consists in constructing a simplified, though cognitively useful, image (description) of a phenomenon under investigation. The image is based on a theory accepted in a given field and is created to effectively expand knowledge about a particular object (for example, the neuron, which is the object of neurobiology, or the atom in physics).

In terms of their relationship to theory, models are divided into *theoretical* and *real* ones. Theoretical models are narrow fragments of theory and take the form of symbolic descriptions of the phenomena. Real models are physical realizations of the aforementioned descriptions – material systems

based on them. Both types of models perform cognition assisting functions, that is, they help the researcher to better understand a given phenomenon and solve problems related to it (for example, connected with predicting the course of similar phenomena) more effectively.

The basic character of the constructions of the first type – that is, their theoretical nature – means that any type of modeling appeals (directly or indirectly) to *formal* sciences. These sciences provide general notions and structures without which it would be impossible to construct a theory that could serve as a basis for building a model (which, in turn, becomes part of the theory). The typical examples here are equations, functions, logical formulas, data structures, and algorithms. Depending on the scientific origin of the structures used, we can talk about various types of models, including: mathematical, logical, cybernetic, and computer science models. Usually, these structures comprise more than one formal *layer* – for instance, a deeper mathematical layer and a more superficial, computer science one.

This article is chiefly concerned with models constructed in *computer science*. This is an extremely complex topic, involving various planes on which contemporary computer science merges with mathematics and modeling theory. By way of an introduction, I shall outline three preliminary issues whose discussion will lead us to the main problem of the article.

The first issue is connected with the idea of the *control code*. According to it, the basic function of computer science consists in transforming mathematical schemata into *machine codes* and, going further, into the real actions of machines controlled by these codes. What is the connection with modeling here? It is quite simple. The indicated function allows us to move from a theoretical model to a real one – that is, to transform static mathematical descriptions into *computer programs*, which are run in time. These programs make it possible to computationally implement the theoretical assumptions of models, causing them to work and reflect – depending on the cognitive needs – a given variation of the phenomenon under investigation (see Stacewicz & Włodarczyk, 2010).

As an illustration, let us consider a typical mathematical structure such as a *set of linear equations*, assuming that the theoretical model of a certain phenomenon X (e.g., in the field of economics) takes the form of a set of linear equations with n unknown values $x_1, x_2, \ldots, x_n$.

$$\begin{aligned} &a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = b_1 \\ &a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n = b_2 \\ &\vdots \\ &a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n = b_n \end{aligned}$$

Such a model describes, first of all, certain interdependencies between the phenomenon under investigation (they are represented by the equations). Secondly, it comprises an unlimited number of solutions to different problems connected with phenomenon X. Substituting various values as coefficients of the unknown values, we obtain different variants of the problem, the solutions to which are particular values of the variables x_1 to x_n. What is essential here, however, is that the model itself, irrespective of the possibility of solving the set of equations defining it, is *static* in character: particular equations will represent the relations on which the researcher focuses (leaving aside the complexity of phenomenon X).

By adding a certain algorithmic description to the model, with information on how to mechanically solve a given set of equations (there may be many such descriptions – some more effective than others), and transforming the model into an appropriate computer program, we can obtain more than just a mathematical description of phenomenon X.

This expanded *programming and computer model* has a different methodological status than the set of equations itself. Contrary to a purely theoretical model (which describes only the relations defining the problem), it is a *real model*, the functioning of which may be observed and changed.

The assisting function of computer science, which makes it possible to transform a theoretical model into a real one, does not exhaust the model-making potential of the discipline. As it turns out, in the case of many models (both theoretical and real ones), it is the sphere of computer science which constitutes the proper *modeling plane*. In other words, the essence of the modeled phenomenon is reflected by the concepts of computer science (and not mathematical ones).

This is certainly the case in the field of cognitive science, with its so-called partial models of the mind – models of certain isolated cognitive acts, such as perception, reasoning, or learning. For example, modeling the act of perception by means of an artificial neural network (such as a perceptron), we reflect the essence of perception by means of computer science concepts: network data processing technology, network structure, artificial neuron, learning algorithms, etc. Naturally, this type of description has a deeper mathematical layer in which, for example, mathematical functions of artificial neuron activation are specified, but its specificity is determined by *computer science* formalisms (see Żurada, 1992).

Moving towards increasingly important computer science applications in modeling, we reach the point where the actual message of the article

starts. This is the issue of computer science applications in describing the very *modeling schema* – the general schema, that is, independent of the type of phenomena under investigation.

If we assume that the modeling process consists of a series of identifiable stages which take place in a sequence, irrespective of the subject of programming and which – again, irrespective of the subject – involves the same methodological procedures, we may wonder to what extent the process may be automated by means of appropriate algorithms. If the process cannot be automated, it can perhaps be described in a precise way – so that the modeler could take advantage of "IT support" at this or that stage of the modeling process.

II. Linear modeling schema

In Stacewicz and Włodarczyk, 2010, and Stacewicz, 2010, I presented a methodological reconstruction of the modeling process according to which the process comprises four cyclically repeating stages: *abstraction*, *formalization*, *simplification*, and *verification*.

At the first stage, the researcher identifies significant features of the modeled phenomenon, *abstracting away* from its other features. At the second stage, the researcher *formalizes* the model by choosing particular formal tools (in the computer science layer of the description these are, for example, data structures and algorithms).[2] At the third stage, she *simplifies* the formalized model by eliminating (with the use of formal transformations) some of the initially chosen features and relationships between them. At the fourth stage, the researcher *verifies* the model by investigating its properties, such as its correspondence with the described fragment of reality, non-contradiction or the desired level of non-contradiction, as well as predictive effectiveness and simplicity – the point of reference for the verification may be the domain modeled, as well as alternative models of the phenomenon under investigation.[3]

Based on the sequencing of the process described, and using the first letters of its particular stages, the schema presented here may be called a *linear AFVS schema*. The name aptly reflects the situation in which the researcher moves from stage to stage, transforming a single model (and not, for example, a series of test models).

It must be stressed that the fourth stage of the modeling procedure, consisting in a gradual movement towards the most adequate model, does not conclude the modeling process. Instead, the process enters into a loop:

it goes back to the first stage, in which – depending on the results of the verification – another set of features of the phenomenon under investigation is considered. Thus, a different type of abstraction is conducted, which becomes the basis for the subsequent stages. It has to be added that depending on the range of changes within the framework of the new abstraction procedure, the author of the model may choose new formal structures or keep the ones used before.

As an example illustrating the AFVS schema (and, at the same time, introducing the issues discussed in chapter V) we will briefly describe a cycle of modeling of perception by means of an artificial neural network, called a perceptron (see Rojas, 1996). A perceptron is a typical multilayer network usually used to recognize objects based on the value of their specified features (for example, letters, based on the location of segments constructing them). The network categorizes objects in the following way: a signal representing an object is fed to the initial layer of the network (the first layer of artificial neurons) – its components, corresponding to the specified features of the object, are fed to different neurons. The signal is propagated in parallel manner by subsequent layers of neurons (the so-called hidden layers). The propagation ends when the state of the neurons in the final layer stabilizes, interpreted as a decision about the object's belonging to a certain category (see, e.g., Rojas, 1996).

In the problem discussed here, the model of perception is a perceptron with a defined *structure of connections* between artificial neurons and the weights of these connections (their throughput). The structure unequivocally defines the operation of the model, that is, the process of assigning the objects presented to the proper categories. Assuming that seeking the proper model can be described by the AFVS schema, we obtain the following correspondences.

The stage of *formalization* corresponds to the choice of the type of network which constitutes the basis for the model. In this case, it is a perceptron with a particular number of neurons on each layer.

The stage of *abstraction* consists in determining what features of recognized objects subsequent neurons of the initial layer are to correspond to (for example, shape, color, toughness, etc.).

The stage of *simplification*, that is, of refining or adjusting the structure of the network, corresponds to the procedure of network *learning* based on a properly chosen set of examples. This is a purely formal procedure, adjusted to the type of network, described by an algorithm which is well justified mathematically (e.g., by the algorithm of error back propagation).

The stage of *verification* consists in the assessment of the trained network with respect to the quality of recognition of a certain set of test objects. Depending on the verification results, the researcher constructing the model may: (a) change the type of modeling network (at the level of formalization) (b) change the functions of neurons of the initial layer; that is, assign them to other features than the current ones (a change at the level of abstraction).

In what follows, the AFVS schema described above will become the starting point for a far-reaching broadening, in which we will introduce two new elements:

1) *populationality* – populations of models, rather than single ones, will be generated and verified within a single modeling cycle,
2) *random factor* – models comprising subsequent populations will be created partially at random.

Before we do that, however, we will describe the idea of genetic algorithms known from computer science, which will constitute the basis for the proposed broadening.

III. Genetic algorithms

Genetic algorithms are a perfect example of benefits derived from interdisciplinary studies. The concept of such algorithms was drawn from biology, specifically from the theory of natural selection. However, a problem solving schema was developed based on it in the field of computer science (see Holland, 1975; Goldberg, 1989). The name of the schema – *genetic algorithm* – alludes to the fact that the data processed using it are coded as sets of genes and certain mechanisms known from genetics are applied during their artificial evolution.

The practical equivalent of genetic algorithms – their realization, to be exact – are computer programs called *evolutionary* or *selective*. The principles of their operation differ significantly from the principles characterizing traditional programs. While the latter ones proceed step by step to the outcome, in accordance with a sequence instruction set beforehand, evolutionary programs generate whole *populations* of results and check their compliance with the requirements for solving the problem. Generating subsequent populations takes place based on principles resembling the natural evolution of organisms: potential solutions mutate, crossbreed, proliferate, and die. Most importantly, however, they are subjected to the selection

procedure, as a result of which more and more outcomes approximating the expected solution appear in subsequent populations. Drawing on biological terminology, one can say that subsequent populations become more and more adjusted to the environment – that is, to the pre-set requirements of the problem being solved.[4]

The principles described above are reflected by such a *general schema* of the genetic algorithm:

Genetic algorithm
(P_t – subsequent populations of trial solutions to a problem)

1. Generate initial population P_t; $t = 0$
2. Until the final condition is fulfilled perform 3 . . . 6
   3. Process (P_t) [for example, perform mutations and crossover]
   4. Evaluate (P_t)
   5. Perform_Selection (P_t)
   6. $t = t + 1$
7. The solution to the problem is the best element in P_t population

Referring to the subsequent points of the schema we will describe in more detail the functioning of a program consistent with it, that is, an evolutionary program. Firstly, the program is to solve a problem P associated with it (for example, to determine the roots of a certain complex non-linear set of equations). The general method it implements consists in searching the space of potential solutions of the problem P. These potential solutions are called *chromosomes* or *individuals* and are most often coded in the form of binary sequences.

Before starting the evolution (point 1) the program randomly generates a certain group of chromosomes, which are labeled as the initial population P_0. The subsequent steps (from 3 to 6) mark the course of the artificial evolution during which subsequent populations randomly change their content, so that, statistically, the average adjustment of the population to the requirements of the problem being solved grows.

In the third step (point 3 of the schema) some chromosomes of the current population are subjected to the activity of genetic operators, such as *mutation, crossover*, and *inversion*. Speaking in very general terms as yet, these operators cause random changes in the code of non-selectively chosen individuals. As a result, new individuals appear in the population, which are added to it or which replace the existing chromosomes.

In the fourth step (point 4 of the schema) each of the chromosomes in the newly created population is evaluated as to its adjustment to the problem P.

The evaluation consists in establishing the value of a certain function, which depends on the definition of the problem and usually constitutes a constant element of the program (for example, if the program solves a set of equations, a hypothetical function may "measure" to what extent potential roots fulfill subsequent equations).

The results of the evaluation procedure become the basis for *selection* (point 5 of the schema), which consists in choosing chromosomes with the highest evaluation function value in the current population. Strictly speaking, it is probability which determines the choice: better chromosomes are chosen with greater probability and worse chromosomes with lesser probability, so the higher the value of evaluation function of a chromosome, the greater representation it has among the selected chromosomes.

A new population results from the selection, which – just like the preceding one – will be processed in the subsequent step of the evolution. The program ends its work when one particular final criterion has been met – for example, when an individual with a sufficiently high evaluation function value appears in the population, or the average fitness of the population reaches a certain satisfactory value.

As can be seen from the description above, the specificity of the evolutionary method rests upon points 3 and 5 of the schema discussed. They make the search of the space of potential solutions (a) partially random, (b) globally directed by the evaluation function connected with the problem. Randomness at the level of processing and modification of subsequent populations is usually granted by the operations of mutation and crossover, performed on randomly chosen chromosomes, or, strictly speaking, on their fragments. In the case of binary representation of individuals, the idea of mutation is as follows:

For each individual the program generates a random number from the $(0, 1)$ interval. If the number is less than the probability of mutation p_M (for example, $p_M = 0,02$), the individual undergoes mutation (that is, in randomly chosen positions of the code zeros are exchanged for ones and reversely). Otherwise, the individual does not undergo mutation. Crossover takes place according to a similar schema. The program generates a number from the $(0, 1)$ interval for each chromosome. If the number is less than the probability of crossover p_C (for example, $p_C = 0,3$), the chromosome is chosen for crossover; that is, it is added to the set of parent chromosomes CR. Then subsequent pairs from the set CR are crossed-over; that is, they are divided into parts from which their offspring is composed.[5]

Algorithmical schemas of these operations are as follows:

Mutation

1. For $i = 1, \ldots, n$ perform steps 2 and 3
   2. Randomly generate number r from the interval $[0, 1]$
   3. If $(r < p_M)$, choose C_i to be mutated $(M = M \cup C_i)$
4. In subsequent C_i from the set M change randomly chosen zeros to ones and randomly chosen ones to zeros.

Crossover

1. For $i = 1, \ldots, n$ perform steps 2 and 3
   2. Randomly generate number r from the interval $[0, 1]$
   3. If $(r < p_C)$, choose C_i for crossover $(CR = CR \cup C_i)$
4. Cross-over subsequent pairs of chromosomes from CR.[6]

Irrespective of the genetic operators (although mutation and crossover are the most frequent), it is the *selection* procedure which determines the evolutionary character of the method presented. It is usually based on the so-called *roulette rule*, which, as its "gambling" name indicates, involves random choices and probabilities connected with them (see Goldberg, 1989).

Before the roulette rule starts to "work", subsequent chromosomes of the current population (C_i) are assigned selection probabilities, equaling the results of division of particular chromosome fitness by the sum of the values of evaluation function of all chromosomes. These probabilities may be illustrated as smaller or larger sections on the roulette wheel, covering an area amounting to one. The wheel is set in motion as many times as there are chromosomes in the population. Each time when the wheel stops and the roulette indicator shows a given section, a new individual corresponding to the section joins the new population. Because of the random nature of the whole procedure some individuals may be chosen a few times and others may never be chosen. The following schema reflects the description above:

Selection

1. Calculate the fitness of subsequent chromosomes C_i, that is, $O(C_i)$, $i = 1, \ldots, n$.
2. Calculate the sum of fitness values $F = \sum_{i=1,\ldots,n} O(C_i)$
3. Calculate the probability of choices $p_i = O(C_i)/F$
4. Calculate the joint probability of $q_i = \sum_{j=1,\ldots,i} p_j$
5. For $i = 1, 2, \ldots, n$ perform steps 6, 7, 8
   6. Randomly generate number r from the interval $[0, 1]$
   7. Search for j fulfilling the condition: $q_j \leq r < q_{j+1}$

8. Widen the set of selection results SEL (initially $SEL = \emptyset$). $SEL = SEL \cup C_j$[7]

Finally, it has to be stressed that the genetic algorithm is a certain non-standard – partially random and, above all, referring to biological analogies – method of searching the space of solutions to a certain problem. As the key element of each algorithm is the *evaluation function* (evaluation of potential solutions), in fact, we are dealing with a certain method of determining the maximum or local maximums of this function. Chromosomes which are selected as a result of artificial evolution are solutions in which the aforementioned function takes on the highest values, globally or locally.

Therefore, irrespective of certain general features of the schema described, it is this, and not any other evaluation function which determines the specific nature of a particular genetic algorithm (and the corresponding evolutionary program). The function sets the goal of the algorithm and, as the goal is the solution to a given problem, its definition contains the information on the most significant features of the problem (see Michalewicz, 1999). On the other hand, in terms of biological analogies, the evaluation function models the environment in which the potential solutions are selected.

IV. Evolutionary modeling schema

From a certain point of view, modeling can be understood as a problem solving procedure characterized as follows: *Find a model M which sufficiently fulfills the verification criterion.* In this light, the AFSV schema discussed in the second chapter constitutes the general basis for methods which, roughly speaking, consist in correcting certain parameters of the initial model, until it reaches the maximum adequacy. These parameters are determined and modified at the level of abstraction, while their temporary usefulness is checked at the stage of verification (when the model determined by them – formalized and then simplified – undergoes evaluation).[8]

With such an approach, modeling comes down to *searching* the set of possible values of the aforementioned parameters, aiming at determining the most adequate combinations of these values. Adopting such a perspective, we gain the interesting possibility of enjoying the merits of genetic algorithms, which are, after all, a certain non-standard method of searching the space of solutions to problems posed. Strictly speaking, we gain the possibil-

ity of joining the sequential AFSV schema with the populational algorithm. The basic ideas for the suggested solution are as follows:

(1) not single models, but whole *populations* of them are considered in the subsequent modeling steps;
(2) subsequent populations are subjected to the activity of genetic operators (such as mutation and crossover);
(3) populations created having used genetic operators undergo *selection* procedures, as a result of which subsequent populations are created;
(4) the basis for selection is a certain criterion which is identical with the accepted model *verification criterion*.

Summing up the points above, it can be said that the sequential ASFV scheme is embedded in a higher level scheme – a genetic algorithm (see Stacewicz, 2010). Writing our proposal in a quasi-computer science manner – in a form analogous to the genetic algorithm schema – we obtain the following *evolutionary modeling schema*.

Evolutionary modeling schema (AFSV/GA)

1. Generate an initial population of models P_k; $k = 0$
   $P_k = \{m_{k1}, m_{k2}, \ldots, m_{kn}\}$ (n is the number of models in the population)
2. Until the final condition is fulfilled perform 3 ... 6
   3. Process (P_k)[9] – using genetic operators with models $m_{k1}, m_{k2}, \ldots, m_{kn}$
   4. Evaluate (P_k) – evaluating subsequent models
   5. Perform_Selection (P_k) – based on the verification criterion
   6. $k = k + 1$
7. The best element in the final population P_c is the solution to the problem.

The realization of the proposed schema leads to the following sequence of events, connected with modeling a certain phenomenon Z:

In the first step of the procedure an *initial population* of test models P_0 is generated. Models contained in it have a strictly defined formal shape (depending on the initially accepted formalization method) and they differ from one another by the choice of features comprising the description of the modeled phenomenon Z (by definition, it is an incomplete description).

Having generated the P_0 population, the first evolution cycle begins. The P_0 population models undergo recombination operations such as mutation and crossover, as a result of which the set of features in the description of the phenomenon Z changes, largely at random. Thus, it is in fact the *abstraction* stage – only this time it is applied en masse to all models in the

current population. Detailed definitions of recombining operations depend on the formalism applied – the method of the mathematical representation of models.

In the next step of the method, models comprising the P_0 population are *simplified*, which means that dispensable features and other needless elements in the description of the phenomenon Z are eliminated (within particular models). Here the operation – termed *simplification* in the AFSV schema – constitutes an element of the Process (P_k) procedure.

Simplified models are then evaluated with respect to certain permanent and predefined criteria (verification criteria specific for a given modeling procedure). A set of values of evaluation function becomes the basis for selection, as a result of which some models in a modified P_0 population become part of the P_1 population, thus progressing to the subsequent cycle of evolution.

Let us finally emphasize that the new schema AFSV/GA differs from the AFSV schema in that its four basic stages of modeling – abstraction, formalization, simplification, and verification – are merged with a genetic algorithm. In combination with the algorithm, they perform slightly different functions than they initially did. Enumerated, it looks as follows:

(a) the stage of *formalization* precedes launching a genetic algorithm – the method of formalization does not change within the algorithm realization process;
(b) genetic operators are applied at the stage of abstraction – they are used with models in the current population;
(c) the stage of *simplification* concerns subsequent models in the population formed with the use of genetic operators – it is realized in the same way as envisaged in the AFSV) although parallelly and en masse (for all models from the current population);
(d) the stage of *verification* is replaced by a selection procedure – the verification criterion is retained, but it is used to evaluate particular models of the current population (based on the evaluation, models are added to the subsequent population or not).

V. Application of the evolutionary schema as part of the COGANN strategy

Two questions come into play in view of the proposition of the evolutionary modeling schema (EMS) presented in the previous section.

Firstly, is this schema actually used in *practice*; that is, do the scientists

working on a model of a given phenomenon actually use anything resembling an evolutionary strategy?

Secondly, even if the EMS schema is not present in the scientific practice, can it constitute a good basis for the automation of the modeling process (or automated support for researchers working on a model)?

We will not undertake to answer the first question in this paper. It is worth noting, however, that some leading methodologists of science, such as Popper, claim that the actual development of scientific theories is evolutionary in nature. In one of his lectures, Popper (1994) says explicitly that it is possible to view the development of knowledge as a fight for survival between competing theories. Only the best adjusted theories will survive, though even they may be annihilated at any time.[10]

If the theory is replaced by a model here (which seems justified, as a model constitutes a part of a theory) the view is analogous to ours, although more general.

As far as the second question is concerned – the one about the possible automation of the modeling process – we think, in line with the previous sections, that computer scientists provide valid premises for giving a positive answer to it. From the point of view of pure computer science, it is simply a fact that genetic algorithms exist – a technique which may be implemented using a computer (that is, with entire or partial automation) and which abounds in various mathematical and engineering applications.

From the point of methodology, however, the possibility of making a genetic algorithm the core of a modeling schema is the answer.

Accepting these premises, we will point to such development of them which leads directly to cognitive science and, specifically, to the mind/brain modeling procedure which is justified from the point of computer science and possible to realize with the use of the computer. We will refer not only to genetic algorithms, but to a broader strategy called the COGANN (*Combinantions of Genetic Alghorithms and Neural Networks*). This strategy consists in solving problems using methods involving both genetic algorithms and artificial neural networks. The possibility of cognitive interpretation, on the other hand, is determined by the fact that at least some neural networks are considered neurobiologically justified and mathematically grounded models of certain fragments of the brain (see, e.g., Churchland, 1986).

In this paper, we assume that the field of artificial neural networks (ANN) is quite well-known. In section II, we briefly presented an important example of a network (a perceptron), which may be used as a model

of neurobiological structures responsible for perception. We will collectively discuss here these features of ANN which determine their specificity (against other data processing systems) and, furthermore, which predispose them to become good candidates for partial mind-brain models.

Firstly, then, ANN are systems of data processing inspired by biology, which, both with respect to their structure and operating principles, resemble fragments of the nervous system in living creatures (some talk directly about the human brain here, but – because of the too highly specialized nature of ANN – it is better to use more prudent terms).

Secondly, the structure of each ANN consists of a vast number of identical processing units called artificial neurons, which are linked by a specially matched network of connections. Artificial neurons are frequently simplified realizations of real biological neurons. Each interneural connection has a certain throughput, a so-called weight, which determines how powerful a signal passing from one neuron to the other may be.

Thirdly, the whole dispersed or distributed "knowledge" of the network about how it should function is comprised in the total arrangement of weights. This means that a certain arrangement of weights makes the network react to the received stimuli in a particular way.

Fourthly, the aforementioned knowledge is not usually coded in advance, but is created in the learning process, when networks are presented with certain typical stimuli, for which particular responses are expected by the system designer. Technically speaking: most of the ANNs have the ability to self-organize, which leads to the generalization of patterns presented to it.[11] (See Żurada, 1992).

In computer science literature, the COGANN strategy is considered with respect to particular algorithmic issues, such as the travelling salesman problem, known from discrete mathematics. Despite that, at a certain level of generality, four general variants are differentiated, within which there is a different "division of roles" between connected genetic algorithms (GA) and artificial neural networks (ANN).[12] Let us enumerate:

(1) ANN and GA perform *the same task*, but in a different way (for instance, by constituting a part of a larger algorithm package). From the engineering point of view, such a solution is well grounded, as long as there are variations of the task; that is, such sets of initial data in case of which each method ensures different effectiveness of realization.

(2) GA performs the task and ANN is used outside GA as a *supporting tool*. For example, adequately chosen networks perform the operations of genetic material recombination (such as crossover).

(3) ANN performs the task but GA *prepares* ANN for action. In this variation genetic algorithm – appropriately to its main function – is responsible for forming the network adequately through simulated, artificial evolution.
(4) GA and ANN cooperate in performing one complex task. This cooperation may be manifested in two ways: (a) GA is responsible for one particular element of task completion, while ANN is responsible for another element of it, or (b) GA is responsible for the dynamic network self-organization which takes place during the completion of the task and is necessary for its completion.

Attempting to apply COGANN in the mind/brain modeling process automation, we may point to its third variation. In this variation, a certain *neurolike network* may be prepared to function as a partial model of the brain.[13] By a functioning model we mean an already formed network (that is, a network with an adequate structure of inter-neural connections and their weights), which may be observed instead of the modeled object. The *genetic algorithm* would be the tool generating the final model. The algorithm would be responsible for selecting the best model among the mutating, crossover, and competing trial structures.

Expressing the aforementioned idea in the spirit of the AFSV/GA scheme (see section IV), we obtain the following image of the evolutionary modeling procedure. Whole populations of models undergo evolution, while each model is an artificial neural network of a particular type (for example, a perceptron), differing from the other elements of the population in terms of the structure of connections between artificial neurons. Systems of connections and their weights which define the models are properly coded (for example, binarily) and all genetic operations coming into play concern their codes. In accordance with the schema above, modeling consists in the cyclical performance of procedures *Process*(P_k), *Evaluate*(P_k), and *Perform_Selection*(P_k). These procedures concern subsequent populations of neural networks and are embedded in the *genetic* algorithm.

In the first part of the procedure *Process*(P_k) networks randomly chosen from the current population P_k (generally termed chromosomes or individuals) undergo *recombination*, that is, operations such as mutation and crossover, These operations are conducted on their codes, specifically on those fragments of the code which correspond to the features of the objects modeled. As they lead to changes in the fragments of the code, they are nothing but a random realization of the stage of abstraction.

In the second part of the *Process*(P_k) procedure, each modeling network undergoes a learning cycle proper to it (called self-organization[14]), during

which weights of interneural connections are fine-tuned. Some of the weights may change to zero, which may be interpreted as the disappearance of corresponding connections. In the terminology accepted here (see section II) this procedure must be called simplification (the term is the more justified the more neurons and connections disappear).

Having processed the populations, the coupled procedures *Evaluate(P_k)* and *Perform_Selection(P_k)*, during which properly trained networks are evaluated with respect to a predetermined modeling criterion and then chosen (according to general selection rules) to become part of the subsequent population.

Summing up the description above, it has to be stated that in the sequence of procedures postulated in it there are two cycles of network perfecting: the external one – connected with genetic operations, that is, recombination and selection – and the internal one – connected with self-organization (that is, learning) of temporarily created networks. Details of both cycles depend, naturally, on the type of network; that is, on the accepted method of model formalization.

VI. Final remarks

The method of modeling process automation presented in the article is – as I have emphasized – a method of *partial* automation. This means that the researcher is a non-negligible "component" of the whole process. He not only initiates it, but also, above all, he determines the *criteria* which the final construction should fulfill. Technically speaking, the criteria correspond to the evaluation function of potential models – the function which constitutes a key element of the genetic algorithm and reflects the goal of modeling. Without direct determination of this function the evolutionary schema would be entirely useless.

The necessity to determine the goal of modeling does not contradict another feature of the proposed schema – its *randomness*. As we have explained above, because of the random operations characterizing artificial evolution (for example, mutations, crossover, and selection) the precise shape of the final model is never predetermined, despite the fixed evaluation function. This feature may be treated as an automated equivalent of human *inventiveness*: the researcher diplays inventiveness at the level of setting a goal, while at the stage of generating models fulfilling this goal the researcher's inventiveness is supported or replaced by the random mechanism of artificial evolution.[15]

The last of the remarks completing the text concerns the type and scope of justification for the evolutionary modeling strategy. The *methodological* justification refers to the fact that the core of the evolutionary strategy is the locally non-deterministic (and, as such, supporting the researcher's inventiveness) and globally directed (and, therefore, not chaotic) process of automatic generation of subsequent models based on a certain strictly defined computer science procedure. These three features – partial probabilism, global focus, and the possibility of automation – are highly beneficial for the modeling practice. From the theoretical point of view, however, the fact that genetic algorithms, which form the basis for evolutionary strategy, have not been encapsulated by one comprehensive mathematical theory describing, for example, the conditions for their concurrence, are not beneficial. With the lack of this sort of formal justification, one needs to rely on other arguments, properly empirical and supported by reasoning by analogy. The first of them refers to the practice of *computer experts* who more and more effectively use genetic algorithms in the field of optimization, which is a field related to modeling (see section IV). The second argument refers to the theory of biological darwinism – as this theory describes well the fact of the natural evolution of species, it is assumed that genetic algorithms inspired by it allow for an effective evolution of populations of models.

NOTES

[1] In deductive sciences, especially in logic, the concept of a *semantic* model is used (see Frigg & Hartmann, 2012, section 1.3). This type of model is a (formal) interpretation of an axiomatic system such that it makes its axioms (and, by the same token, all the statements of the system derivable from the axioms) true. Note that a fragment of a theory having a semantic model, that is, an interpreted model, may perform the function of a model in an empirical sense (it is then treated as a precise description of the phenomenon under investigation in terms of an adequately interpreted theory).

[2] It has to be noted that model formalization, that is, the choice of the adequate mathematical and/or computer science formalism, causes the model to become a certain type of *idealization* of the studied phenomenon. Each formalization makes it necessary for a phenomenon to be described by means of a certain ideal creation, generated by science, not encountered in reality. Thus an inevitable deformation of the phenomenon takes place, imposed by the formalization procedure itself. Preempting any remarks on that, we could say that while modeling the mind/brain by means of a particular neurolike network, we perform an idealization by appealing to certain features of a given network rather than to any other ones (which may be effective, but which doubtlessly distorts the original object).

[3] In fact, it would be adequate to talk here, following Popper, about attempts of falsification; that is, verification aimed at rejecting the model (see Popper, 1934).

[4] From the methodological point of view, research on genetic algorithms belongs to the field of research on artificial intelligence (that is, methods of solving complex tasks, which, in human beings, normally require the engagement of intellect). To be precise, it belongs to the *naturalist* trend in research. Within this trend – often set against logicism – it is initially assumed that human cognitive abilities have biological and social roots and, therefore, when realizing them artificially one has to refer to empirical data, connected with, for example, the development and functioning of the human brain (the biological basis of the mind). The typical example of naturalist systems are artificial neural networks, inspired directly by studies of the human brain, as well as evolutionary systems which process data in a way reminiscent of natural evolution. These latter ones are often based on genetic algorithms (see Russell & Norvig, 1994; Stacewicz, 2010).

[5] Classic crossover of two chromosomes C_i and C_j consists in: a) random choice of one point of cutting inside the chromosome code, b) generating two daughter chromosomes in such a way that the first of them contains the initial fragment of chromosome C_i (until the point of cutting) and the final fragment of chromosome C_j (starting from the point of cutting), while the other, reversely, contains the initial fragment of C_j and the final fragment of C_i.

[6] In the procedures presented – mutation, crossover, and then selection – the following symbols were accepted:

C_i – marks i-th chromosome in the current population,

p_M and p_c – the probability of mutation and crossover, respectively,

M and CR – a set of chromosomes chosen for mutation and crossover, respectively.

It is worth noticing that the probabilities of reconfiguring operations (here: p_M and p_c) either constitute a permanent element of the algorithm (which is most frequently the case) or are changed globally, together with the progress of the algorithm, or are added to particular chromosomes and are subject to evolution.

The probabilities of reconfiguring operations either constitute a permanent element of the algorithm (which is most frequently the case) or are globally changed together with the progress of the algorithm, or are added to particular chromosomes and undergo evolution.

[7] The method described here was modified in many ways, which, in general, consist in certain "refinements" of the random nature of selection. The simplest of them is limited to the chromosome with the highest evaluation value joining the population each time. In more advanced modifications the expected numbers of copies of particular individuals are added to the population (determined based on their selection probabilities). Only then are the remaining places filled based on the roulette rule (see Goldberg, 1989).

[8] From the perspective of computer science such a schema may be called a machine learning schema. Thus, its possible computer science realization could draw on learning algorithms known from computer science (see Mitchell, 1997).

[9] In the subsequent points of the schema we apply the symbol P_k, although, in fact, the population has already been modified (processed by Process (P_k) procedure).

[10] Popper (1994) combines the evolutionary interpretation of knowledge/science with a certain general schema of its development, which he calls the schema of attempts and error elimination. According to it, researchers solving a certain problem (or a group of problems) put forward a series of competing theories. During scientific competition the theories are purified of errors (perfected) and the one which proves the best is temporarily accepted as the basis for the solution (at the same time, however, new problems arise based on it, which require generating subsequent theories).

[11] It has to be noted that although in this paper we consider self-organizing networks, some networks do not have the ability to self-organize (e.g. McCulloch-Pitts' simple networks used for logic operations).

[12] See for example Rutkowska, Piliński & Rutkowski (1997, pp. 250–266).

[13] Speaking about a partial model we mean either a model of a certain specialized system inside the brain or a model of a certain isolated cognitive activity (such as the perception of a certain type of objects).

[14] Self-organization is a process characteristic for each network of changing weights of connections, as a result of which the network gains the ability to function effectively (the phase of learning usually precedes the phase of functioning). Various learning algorithms are elaborated for various networks. Most of them have a good mathematical justification.

[15] Here, a question may arise whether the evolutionary schema could not be used also at the stage of generating goals, not only seeking models fulfilling particular goals. Although this seems possible, immediately a question may arise, whether the evolutionary schema could not provide full automation of the modeling process. And so, could evolutionary strategy be used also at the stage of generating goals, not only seeking models fulfilling particular goals? Although this seems possible, immediately yet another question arises, concerning the criteria of assessment and choice of these goals. Who is to set them: the researcher or the machine? If these new criteria (in fact, new goals) were to be found also using some evolutionary strategy, another problem is posed of determining the new criteria. What we have got here is, in fact, a certain "computer science" version of the ancient skeptics' reasoning, which, in a way, enforces the statement (consistent with modeling practice) that in the modeling process the decisive and directional role of the researcher should not be overlooked.

REFERENCES

Churchland, P. S. (1986). *Neuropsychology: Toward a unified science of mind/brain*. Cambridge, MA: MIT Press.

Frigg, R., & Hartmann, S., (2012). Models in science. In E. N Zalta (Ed.), *The Stanford encyclopedia of philosophy* (Fall 2012 ed.). Retrieved from http://plato.stanford.edu/archives/fall2012/entries/models-science/.

Harel, D. (1987). *Algorithmics: The spirit of computing*. Reading, MA: Addison-Wesley.

Goldberg, D. E. (1989). *Genetic algorithms in search, optimization and machine learning*. Boston, MA: Addison Wesley

Holland, J. H. (1975). *Adaptation in natural and artificial systems*. Ann Arbor: University of Michigan Press.

Michalewicz, Z. (1992). *Genetic Alghorithms + Data Structures = Evolution Programs*. Berlin: Springer Verlag.

Michalewicz, Z. (1999). *The significance of the evaluation function in evolutionary algorithms*. In L. D. Davis, K. De Jong, M. D. Vose & L. D. Whitley (Eds.), *Evolutionary Algorithms* (Vol. 111 of The IMA Volumes in Mathematics and its Applications, pp. 151–166). New York: Springer.

Mitchell, T.M. (1997). *Machine learning*. Singapore: McGraw-Hill.

Popper, K. R. (1934). *Logik der Forschung*. Wien: Verlag von Julius Springer.

Popper, K. R. (1994). *Knowledge and the body-mind problem: In defence of interaction*. London – New York: Routledge.

Russell, S., & Norvig, P. (1994). *Artificial intelligence: A modern approach*. Englewood Cliffs, NJ: Prentice-Hall.

Rutkowska, D., Piliński, M. & Rutkowski, L. (1997). *Sieci neuronowe, algorytmy genetyczne i systemy rozmyte*. Warszawa – Łódź: PWN.

Shaffer, J. D., Withley L., & Eshelman J. (1992). Combinations of genetic algorithms and neural networks: A survey of the state of the art. *Proceedings of International Workshops on Combinations of Genetic Algorithms and Neural Networks (COGANN-92)* (pp. 1–37). Piscataway, NJ: IEEE Press.

Stacewicz, P. (2010). *Umysł a modele maszyn uczących się: Współczesne badania informatyczne w oczach filozofa*. Warszawa: Wydawnictwo EXIT.

Stacewicz, P. (1994), *Zastosowanie algorytmów genetycznych do wnioskowania z przykładów*. Warszawa: Prace IPI PAN.

Stacewicz, P., & Włodarczyk, A. (2010). Modeling in the context of computer science: A methodological approach. *Studies in Logic, Grammar and Rhetoric*, *33*, 155–180.

Rojas, R. (1996). *Neural networks*. Berlin: Springer-Verlag.

Żurada, J. M (1992). *Introduction to artificial neural systems*. St. Paul: West Publishing Company.