INFORMATION SYSTEMS IN MANAGEMENT

Information Systems in Management (2016) Vol. 5 (3) 377−388

IMPLEMENTING EVOLUTIONARY ALGORITHM INTO TRAINING SINGLE-LAYER ARTIFICIAL NEURAL NETWORK IN CLASSIFICATION TASK

STANISŁAW PŁACZEK

Faculty of Computer Science, Vistula University (AFiB Vistula)

The article proposes implementing a modified version of genetic algorithm in a neural network, what in literature is known as "evolutionary algorithm" or "evolutionary programming". An Evolutionary Algorithm is a probabilistic algorithm that works in a set of weight variability of neurons and seeks the optimal value solution within a population of individuals, avoiding the local maximum. For chromosomes the real value variables and matrix structure are proposed to a single-layer neural network. Particular emphasis is put on mutation and crossover algorithms. What is also important in both genetic and evolutionary algorithms is the selection process. In the calculation example, the implementation of theoretical considerations to a classification task is demonstrated.

Keywords: Genetic Algorithm, Evolutionary Algorithm, selection process, mutation, recombination, replacement.

1. Introduction

Genetic Algorithms (GA) and Evolutionary Algorithms (EA) are in some respect inspired by the process of natural selection. A given environment is filled with a population of individuals that strive for survival and reproduction. The fitness of these individuals represents their chances of survival.

Between the concepts of natural evolution and computer calculation one can appoint the following relation [1]:

Environment – Solving problem
Individuals – Candidate solution
Fitness value – Quality or target

Everybody should be very careful when interpreting the results achieved; taking into account the differences between a natural environment and a computer environment that is built with silicon and mathematics. From the historical point of view, three different implementations of the basic idea have been developed. In the USA, Fogel, Owen and Walsh introduced Evolutionary Programming (EP). Holand called his method a Genetic Algorithm (GA), while in Germany Rechenberg and Schwefel invented the Evolution Strategy. Since 1990 all streams following the general idea emerged as Genetic Programming (GP). Now the whole field of evolutionary computing is known as Evolutionary Algorithm (EA). In the article the last term is used. In the article, an EA is implemented to an Artificial Neural Network (ANN) in the teaching process and, in practice, implementation. An ANN has valuable characteristics, such as approximating any non-linear mapping and generalization, parallel and distributed computation, learning and adaptation. Parallel and distributed computation especially correlate with the genetic and evolutionary algorithm structures. Evolutionary Algorithms are interpreted as genetic algorithm generalization. In an EA the evolution principle and inheritance are implemented, as well as using the appropriate data structure according to the solving task. For an ANN a real figure matrix is applied. Using this, the appropriated variations of operators are used.

2. Evolutionary Algorithm structure in an ANN application

The simplest ANN structure is described as one layer Perceptron (Fig.1). The input vector data X are put into input neurons, which are multiplied by appropriate weight coefficients of an ANN structure. Using different activation function types, the output signal is calculated as vector Y. This vector is compared with the teaching vector Z calculating a target function value. As a teaching function the minimum of the mean square error (MSE) is usually used (1).

$$\Phi = \frac{1}{2}\sum_{p=1}^{p=N_p} \cdot \sum_{i=1}^{i=N_1} (z_i^p - f_i(\sum_{j=0}^{j=N_0} w1_{ij} \cdot x_j^p))^2 \qquad (1)$$

where:
N_0 – the dimensionality of the input vector X,
N_1 – the dimensionality of the output vector Y and the teaching vector Z,
N_p – the epoch dimensionality,
$w1_{ij}$ – matrix weight coefficients connecting input and output vectors,
f_i – the non-linear activation function.

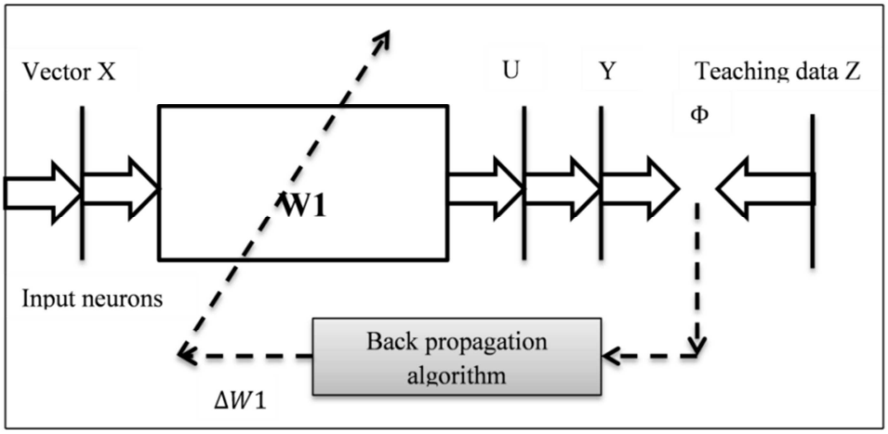

Figure 1. One-layer Perceptron scheme

The activation function could be sigmoid or tanh. The scheme presented in (Fig.1.) describes the teaching algorithm structure based on a target function gradient. Matrix weight coefficients are modified by each iteration (an off-line algorithm). A backpropagation algorithm is the most popular teaching algorithm. Only one exemplar of matrix weight coefficients is used (using the EA terminology: only one individual is used). Of course, the gradient has to be calculated, which is used to modify the weight coefficients. In an EA the set of individuals is generated. Evaluating the output vector Y is fulfilled in a parallel way, so the gradient calculation is unnecessary.

The basic model is different in an EA calculation. We have to define the set of ANN individuals with one input and output vectors (Fig.2). These individuals are the unit of selection. Their reproductive success depends on how well they are evaluated by the target function. The more successful individuals have a higher ability to reproduce in the next generation. Additionally, mutations give rise to new individuals to be tested. Thus, as the iteration passes, there is a chance in the constitutions of the population. The whole set of individuals is known as a population. In an ANN every individual is represented by the weight coefficient matrix W1, which connects the input signal X with the output one. In the EA terminology a matrix is defined as a chromosome – an individual calculation entity. A chromosome contains a set of weight coefficients known as genes. Genes are expressed in natural figures. It is the basic difference between EA and GA. In the GA all genes have bit representation.

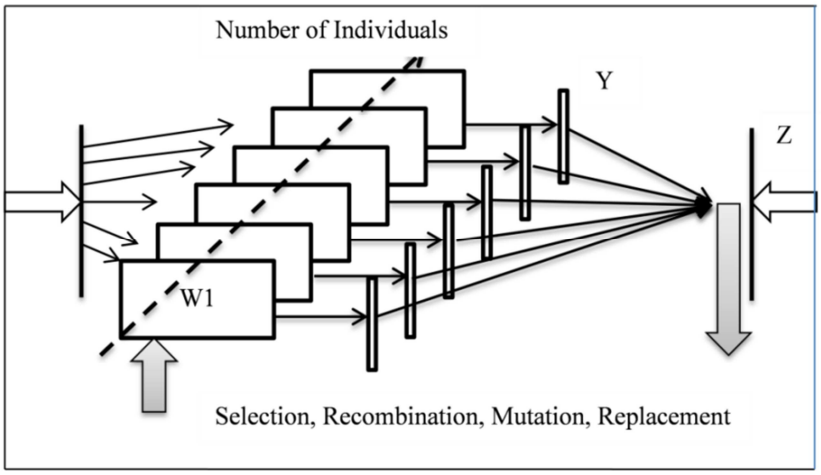

Figure 2. Chromosome and Population structure for an ANN

Where general data structure is the following:

$X^p[0: N_0]$ – the input vector from the epoch set,

$Y^p[0: N_1]$ – the output vector,

$Z^p[0: N_1]$ – the teaching vector,

$p[1: N_p]$ – the epoch size,

$W1^k[1: N_1][1: N_0][1: N_{popsize}]$ – the three-dimensionality matrix of individuals,

$popsize$ – the size of the population.

Thus, a set of genes creates a chromosome. According to Fig. 2, an EA structure contains the "number of individuals" dimensionality "*popsize*". The input vector $X^p[0: N_0]$, in a parallel way is sending into *popsize* matrix W1. All output vectors $Y^p[0: N_1]$ are evaluated and the fitness function Φ^k for every individual is calculated. One of the most important parameters which has to be taken is the number of individuals (or parents) μ, and the number of offspring (children) λ. The decision is not simple. If one decides to increase the number of both the individuals and the offspring, the algorithm will look through the solving space better. But, on the other hand, this process is time-consuming and finding the final solution could last too long. Comparing Fig.1 and Fig.2 one can see the main differences between a standard backpropagation algorithm and an EA structure. A backpropagation algorithm is serial. Solving space is indwelled by an algorithm using information contained in the target function gradient. An EA is a parallel algorithm. In a parallel way, the set of individuals indwells solving space, looking for the best individuals fitted to the target function (the solving task).

3. Evolutionary Algorithm components

In literature one can find many different variants of an EA. But there are common technics behind all of them. There is a population of individuals within some environment (giving target function). These individuals compete using the natural selection (survival of the fittest). This, in turn, causes a rise of the population. Giving the target function (the quality function to be maximized), an algorithm randomly creates a set of candidate solution. The target function is applied as an abstract fitness measure. On the basic of these fitness values some of the better candidates are chosen to seed the next generation, which is fulfilled by applying recombination and mutation to them. Therefore, by executing the variation operators on the parents, a new set of candidates is generated (the offspring). A new target function value is evaluated and competition is created. For competition a different algorithm is used – the fittest, the maximum age. This process can be iterated until a candidate with the maximum quality is found. To summarize one can conclude:

- EA are population-based. They process a whole collection of candidate solution in a parallel way.
- EA use recombination, mixing information from two or more candidates.
- EA are stochastic using a lot of randomly generated data [1].

Below, the most important components of EA are described:

- Individuals' representation. For an ANN a set of matrices will be applied.
- Evaluation function, target function or fitness function.
- Population dimensionality *popsize*.
- Individual (parent) selection.
- Variation operators. Recombination and mutation to the matrix structure will be applied.
- Replacement (survival selection mechanism).

The general block scheme of the EA is given in Fig. 3. It contains all the main components and the relations between them. Every block in the scheme could be realized in many ways and could be implemented by various algorithms, which are described in Fig. 3. As stated above, an ANN layer structure is described as the weight coefficients matrix W1. For a long input vector X and output vector Y, the matrix dimensionality could be big and contain a lot of weight coefficients.

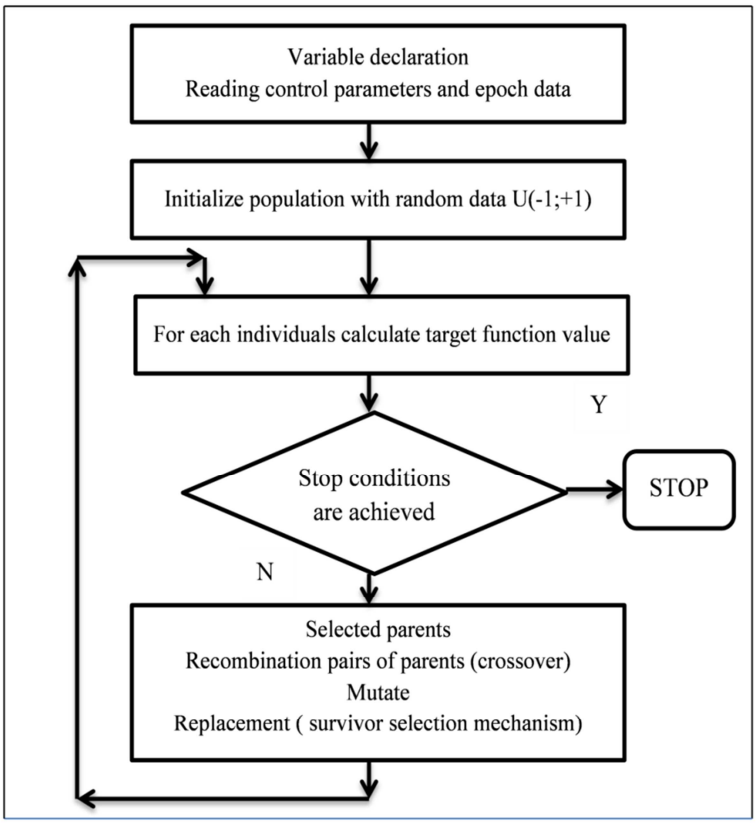

Figure 3. Bloch scheme of Evolutionary Algorithm structure.

3.1. The representation of an ANN weight coefficients

For $N_0 = 50$ and N_1= 30 variables one matrix W1 contains 1500 weight coefficients. Using binary representation, one has to decide on the weight coefficients range dimension. Say, it will be [-5 : +5]. Taking six positions for approximation precision every matrix weight coefficient w_{ij} will be represented by 10*1000000=10 000 000 subintervals. This means that the binary vector has to have minimum 24 bits.

$$8\,288\,608 < 2^{24} < 16\,777\,216 \quad (2)$$

The total chromosome dimension including 1500 genes:

$$24*1500= 36000 \text{ bits.} \quad (3)$$

It is quite a big value and operates such long bits strings that it is not optimal. The final conclusion is simple – using binary code for matrix weight coefficients representation is not adequate. In contrast, taking the real value matrix weight representation is the appropriate decision. Chromosome contains 1500 figures in 50 rows and 30 columns $w_{ij} \in R$ – the set of real figures. From the calculation point of view, the coding and decoding step is omitted in each iteration. The algorithm works faster.

3.2. Evaluation function

The main role of the evaluation function, target function or fitness function is to represent the requirements the population should adopt to meet. It is the base for the selection step and it boost improvements. As it is standard in an ANN teaching process, the minimum of the mean square error (MSE) is used (1). It should be as minimal as possible. However, in an EA the maximum of the target function is required to use the proportional selection. The simplest method is to change the minimum to the maximum,

$$\max\{-\Phi(x)] = \min[\Phi(x)] \tag{4}$$

But this simple method cannot guarantee that all $-\Phi(x) \geq 0$, which is required by the proportional selection. A better way is to add constant number C to all fitness values, achieving:

$$C - \Phi(x) \geq 0 \tag{5}$$

The problem is that it is hard to select a proper C value. If C is small, it cannot guarantee that $C - \Phi(x) \geq 0$ for all individuals. For an ANN teaching target function, the maximum of fitness value in the current population is calculated, which guarantees that:

$$\Phi_{\max}(\mathrm{x}) - \Phi(x) \geq 0 \tag{6}$$

In this way, the minimum optimization problem could be changed to:

$$\max\{\Phi_{\max}(\mathrm{x}) - \Phi(x)\} \tag{7}$$

3.3. Population dimensionality

The role of population is to represent a possible solution. In an ANN, the population is a multi-set of matrices and forms the unit of evolution. In almost all EA applications the population size is constant and does not change during the iteration process. The selection operators (two-step, parent selection and survival selection) work on the population level. But, in contrast, the variation operators (crossover and mutation) act on parent individuals. A population should have an adequate property. The main parameter is known as diversity and measures

the number of different solutions. Other statistical measures, such as variation and entropy, are used. If the *popsize* population is big, usually diversity is better, but an algorithm needs more time to evaluate the target functions.

3.4. Parent selection mechanism

Selection process imitates natural selection by granting fitter individuals higher opportunity to be selected into the crossover process. Individual "i" in the current population has a fitness value Φ_i. Fitter individuals have more chances to be selected and a relative fitness value of individuals is calculated:

$$p_i = \frac{\Phi_i}{\sum_{i=1}^{i=popsize} \Phi_i} \tag{8}$$

where: $i = 1 \ldots, popsize$.

In the program selection, an algorithm is simulated by the roulette and this process is known as roulette wheel selection (RWS) (Fig.4.).

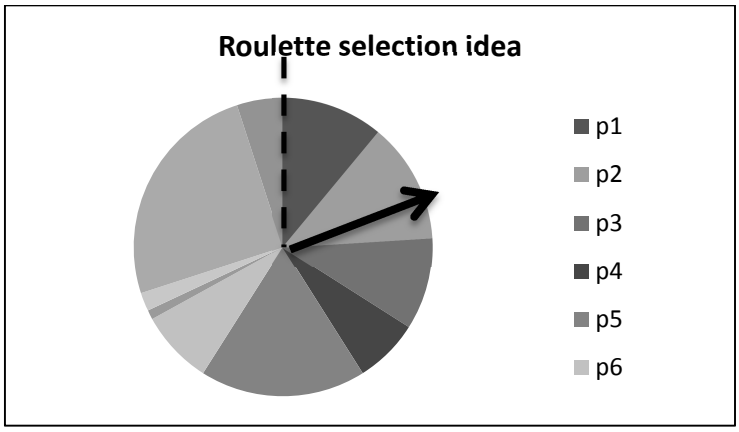

Figure 4. Roulette rotation process for 7 individuals.

In this way, some individuals in the population will be selected more than once and some will never be selected. But, from time to time, the fitter individuals could not be selected by RWS. This process is known as *selection bias*. To avoid this for an ANN, *stochastic universal sampling* (SUS) is used, as suggested by Baker [2]. RWS has one arrow. If the arrow stops at area "i", an individual "i" is selected. RWS is carried out in a serial way.

SUS has *popsize* uniformly distributed arrows (Fig. 5.). The angle between them is $2\pi/popsize$. If any arrow "i" stops at area 'j", "j" individuals are selected. SUS works in a parallel way.

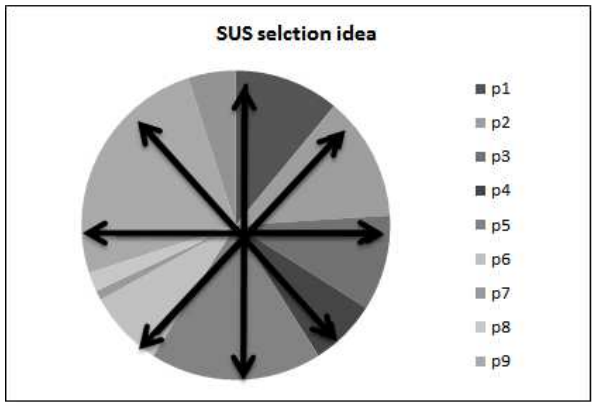

Figure 5. Stochastic Universal Sampling for 9 individuals

SUS could perform proportional selection without bias.

3.5. Recombination or crossover operator

In a standard GA with binary representation, one- or multi-point crossover is implemented. For an EA with chromosomes in real number representation different algorithms are used. For an ANN training process, Arithmetic and Blend Crossover are implemented. Chromosomes are presented by matrices. For

$$W1^1 = \left(w_{ij}^1\right) and\ W1^2 = \left(w_{ij}^2\right) for\ i = 1,2,\ldots..N_1; j = 0,1,2\ldots\ldots N_0 \quad (9)$$

The first offspring could be expressed by (10):

$$WP^1 = \alpha \cdot W1^1 + (1-\alpha) \cdot W1^2 \quad (10)$$

For the second offspring, taking $\beta = 1 - \alpha$, achieving (11)

$$WP^2 = \beta \cdot W1^1 + (1-\beta) \cdot W1^2 \quad (11)$$

Where α is uniformly distributed $\alpha \sim U(0,1)$. This algorithm generates two offspring, which are a linear combination of their parents. From an ANN point of view, this solution is not satisfying. The solution space should be penetrated in a more complex form. In literature the Blend Crossover (BLX) has been described and it has better statistical characteristics [2]. For

$$W1^1 = \left(w_{ij}^1\right) and\ W1^2 = \left(w_{ij}^2\right) for\ i = 1,2,\ldots..N_1; j = 0,1,2\ldots\ldots N_0 \quad (12)$$

the offspring gene is achieved by a non-linear algorithm, using uniformly distributed random number in the range (a, b), defined by the developer.

$$wp_{ij}^1 = rand\left(\left(w_{ij}^1 - \alpha \cdot \left(w_{ij}^2 - w_{ij}^1\right)\right), \left(\left(w_{ij}^2 + \alpha \cdot \left(w_{ij}^2 - w_{ij}^1\right)\right)\right)\right) \quad (13)$$

3.6. Mutation operator

In practice, the weight coefficients values are limited. Therefore, the mutation process has to be limited as well. Using mutation probability parameter p_m, selected from the whole population, a gene could be modified by Uniform Mutation.

$$w_{ij}^{p} = rand\ (a, b) \tag{14}$$

Where *a*, *b* are the upper and lower domain value, respectively. In this algorithm one can define the domain parameters in an arbitrary way. A better result is achieved using Normal Mutation. For a normal distributed randomly, the possible of the mutant value will be in the range

$$(w_{ij}^{p} - 3 \cdot \sigma\ ;\ w_{ij}^{p} + 3 \cdot \sigma) \tag{15}$$

Where σ is the standard deviation. Mutation is calculated by formula (16).

$$w_{ij}^{p}(n+1) = \ w_{ij}^{p}(n) + \sigma \cdot N(0,1) \tag{16}$$

4. Numerical example

A simple classification problem was investigated as an example. In the two-dimensional space the set of point is divided only into two subclasses (Fig. 6.). One layer ANN includes 2 input neurons and 2 output neurons. The activation function is non-linear with threshold characteristic. Selection probability $p_s = 0.7$, mutation probability $p_m = 0.01$ and $\sigma = 0.1$.

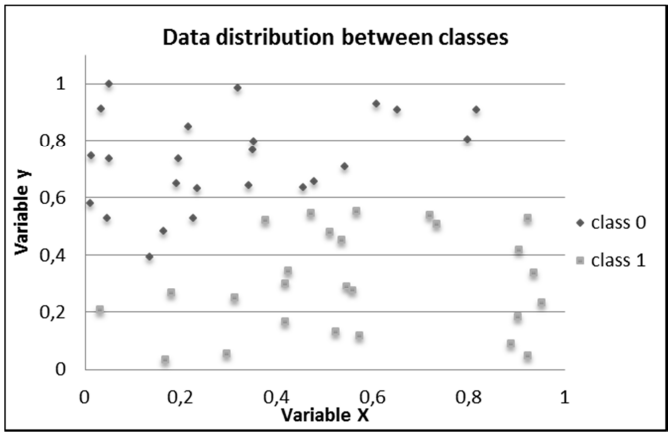

Figure 6. Input data structure divided into two subclasses

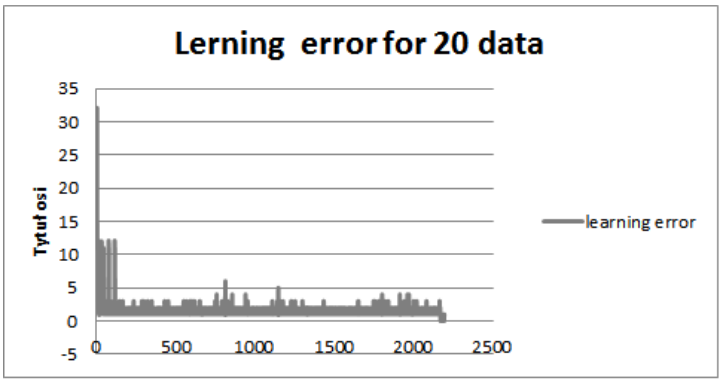

Figure 7. Learning process characterized by errors

In the EA 50 individuals are defined and all genes are initialized by the uniform value U(−1; +1). In Fig. 7 one can see that an average error achieves quite a big value. This means the fitness functions are distributed into the entire solution space. To achieve the minimum error an algorithm needs less than 2500 iterations only. It is connected with the threshold function.

5. Conclusion

An ANN calculates the output value Y using all the neurons in a layer, which are working in a parallel way. The modern computers with multiprocessors and programming languages with - have the tools to use ANNs in a wide area. An EA, using real figures value in genes, chromosome and population representation as the basic for its structure, can use the same computers and programming features. So, the connection of these two tools increases the final calculation power and speed. The article, in a very abbreviated form, describes all the main features and elements of an EA structure. In the calculation example SUS selection process, BLX crossover method and Normal Mutation algorithms were used. As the replacement mechanism the simplest algorithm was used, too. The *popsize* individuals (parents) generate the same offspring number, so the population is constant during the iteration process $\mu = \lambda$. Using threshold activation function, an algorithm needs less iteration to achieve the final target function value. Using a standard sigmoid function, a backpropagation algorithm needs more iteration to achieve the saturation value. In the evolution process there are two important issues: population diversity and selective pressure. These factors are related: an increase in the selective pressure decreases the diversity of the population, and *vice versa*. Strong selective pressure supports the premature convergence. Analyzing the average error in Fig.7. one can see that algorithm holds on the balance between population diversity and selective pressure.

REFERENCES

[1] A.E. Eiben, J.E. Smith: *Introduction to Evolutionary Computing*, Second Edition, Springer 2003, 2015.

[2] Michalewicz Z.: *Genetic Algorithm + Data Structure = Evolutionary Programs*, Springer – Verlag Berlin Haidelberg 1996.

[3] Montana DJ, Davis L,: "Training Feedforward Neural Network Using Genetic Algorithms. Proceedings of the 1989 International Join Conference on Artificial Intelligence", Morgan Kaufmann Publishers, San Mateo, CA, 1989.

[4] David E. Goldberg: *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison- Veslay Publishing Company, Inc. 1989.

[5] Xinjie Yu, Mitsuo Gen: *Introduction to Evolutionary Algorithm*, Springer London 2010.