Agent-Based Neuro-Evolution Algorithm

Rafał Dreżewski, Krzysztof Cetnarowicz, Grzegorz Dziuban, Szymon Martynuska, and Aleksander Byrski

AGH University of Science and Technology, Department of Computer Science, Kraków, Poland
drezew@agh.edu.pl

Abstract. Neural networks are nowadays widely used across many different areas, both enterprise and science related. Regardless of the field in which they are applied, optimization of their structure is usually needed. During the last decades many methods for this procedure have been proposed, among them techniques based on evolutionary algorithms. In this paper a new algorithm for optimization of neural networks architecture based on multi-agent evolutionary approach is proposed. Also results of preliminary experiments aimed at comparing the proposed technique to already existing ones are presented.

Key words: agent-based evolutionary algorithms, neural networks, prediction

1 Introduction

In the still growing range of numeric problems that require the use of large amounts of computational power, a significant area is taken by those, which are unsolvable or too costly in terms of resources for classical approaches. Among them there are tasks related to prediction, economical trends, weather forecasting, image recognition or exploring large data warehouses in search for hidden patterns.

Amid widely applied solutions for such challenging problems, artificial neural networks (ANN [7]) are, without a doubt, one of the most important. What gives an artificial neural network its advantage over classical approaches in the case of numerical problems is its distinctive ability to learn and adapt, which is loosely based on the way an actual neural network works.

No matter what approach to constructing neural networks we assume, the main issue in the procedure is to teach the NN to process input data in desired way. The whole point of creating a neural network is to avoid forcing user to find the equations for calculating the problem and interaction should be reduced to minimum [10].

For any problem defined, which can be solved using neural networks, we can point out unlimited number of different network structures that can provide us with valid output. These layouts vary greatly in complexity and computational power/time needed to reach the solution. Due to a huge number of possible network configurations, manual process of refining the neural network architecture requires many experiments and may not necessarily lead to the best solution. That is why several methods of automatic optimization of neural network architecture were invented.

Among the most known algorithms for automatic pruning are Magnitude Based Pruning, Optimal Brain Surgeon, Optimal Brain Damage and Optimal Brain Surgeon with Nodes [9]. From the construction procedures we can highlight the following methods: Cascade Correlation [17] and Flex Net [13].

Each of the methods mentioned above can be considered a "classical" neural network optimization technique. The main advantage of such methods is undoubtedly their relative simplicity. Unfortunately, as the size of networks grow, their convergence and overall performance declines greatly. Good NN optimization algorithm must have two major characteristic—complexity growing very slowly with the increase of network architecture and an ability to search for the solution very effectively in both wide, and narrow space.

In this paper we propose a new agent-based approach. We will also present results of preliminary experiments aimed at comparing proposed agent-based approach to "classical" evolutionary algorithm used for neural network optimization.

2 Evolutionary Neural Network Optimization

Evolutionary neural networks (ENN) are an alternative approach to neural network optimization, in which the search for a desirable neural architecture is made by an evolutionary algorithm [12, 19]. It is a very flexible solution, but it requires some time to work properly. This method can be characterized as a "black-box" approach (compare [15]) in which we ignore internal structure, or semantics, of created solution, but we can evaluate its correctness and quality. Similar algorithms have proven to be especially helpful in the case of complex problems, e.g. NP-complete class, for which classical approaches give no satisfactory results [20].

The term "evolutionary algorithm" is a very wide one and describes a group of different techniques. One of them are co-evolutionary algorithms in which fitness of an individual is computed on the basis of its interactions with other individuals present in the population.

In order to encode the structure of the network into a chromosome generally two approaches are used—direct encoding and indirect encoding. In direct encoding longer chromosomes are needed, and indirect encoding suffers from noisy fitness evaluation [19].

During the evolution process not only connections of NN can be evolved. The same process can operate on weights, but it is much easier to evolve the structure of connections, leaving the search for the values of weights to the network training algorithm.

NEAT System belongs to a class of evolutionary algorithms, which change weights and the structure of neural network [16].

NeuroAGE was designed for weight and network structure evolution controlling complex control systems [6].

Another approach to the neuro-evolution was proposed in [14]. In this paper a Symbiotic Adaptive Neuroevolution System (SANE) for solving controlling problems using neural networks was used. Instead of evolving the whole network, SANE carries out operations on populations of nodes in the hidden layer. Unlike the previous described methods, SANE focuses only on evolution of neuron weights, working on predetermined topology.

Evolution of neural network architecture can be performed in multi-agent system. Taking advantage of autonomy, parallel actions and other agency features [18] the system can become a versatile alternative for classical sequential algorithms. Considering prediction of time series [1], when the signal to be predicted is very complicated (e.g. when different trends change over time or temporal variations in relationships between particular sequences are present) the idea of a multi-agent predicting system may be introduced, in which each agent may perform an analysis of incoming data and give predictions [2]. In the case of classification systems, many agent can treat the processed data in a different way, sometimes repeating themselves, sometimes applying completely different points of view (different transforms, preprocessing algorithms etc.)

3 Proposed Multi-Agent Approach to Neuro-Evolution

Both NeuroAGE and NEAT system alike present classical approach to neuro-evolution problem. They each focus on developing topology and adapting weights of connections through evolution of networks treated as a coherent whole. In the SANE project it was proved, that better results can be achieved using separated optimization of individual parts of the given problem, by treating neurons as independent, cooperating, and the same time competing, entities. As a result, the population automatically preserves its diversity and is resistant to the premature convergence.

Another problem is the size of genotype growing in response to more complex network architectures. Genotype of a single individual must store information about subsequent connections. Again, in this scenario, SANE algorithm performs better, because genotypes of schemes and neurons have fixed size.

Proposed Multi-Agent Distributed SANE System (MADNESS) is an extension of ideas proposed in SANE system. MADNESS utilizes similar symbiotic approach to evolution of individual neurons but realizes it using agent-based system. Unlike its predecessor, system has the ability to modify topology of networks. However, maintaining two populations was abandoned in favor of free transfer of neurons between tested networks.

3.1 Distributed Agent Model

In MADNESS neurons were implemented as agents independent from each other and placed inside an environment providing resources needed for survival and access to necessary information. Each agent is given by the environment a certain amount of resource ("life energy") at the start of his life. When this energy reaches zero, the individual dies and is removed from population. The environment checks periodically the quality of all networks and chooses the best neurons, which are given a certain amount of energy from the pool.

System is based on distributed model, thus enabling parallel existence of several independent environments (Fig. 1). A central node exists in this model, responsible for control of the whole experiment and processing nodes. Each node creates an individual, independent instance of environment, and initializes it with a random population of agents.

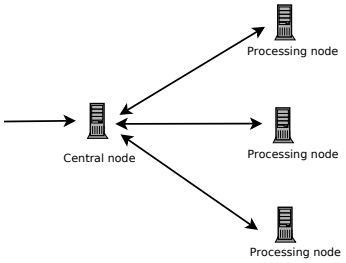


Fig. 1: Proposed system architecture

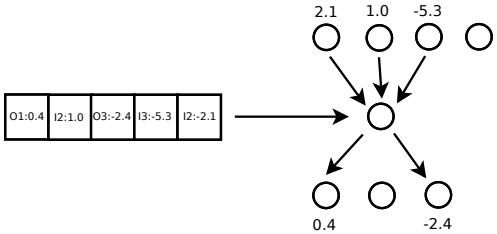


Fig. 2: A hidden layer neuron and its genotype

During the evolution central node controls and synchronizes the work of computing nodes. It also enables the transfer of neurons between environments. Distributing the processing on multiple independent environments significantly improves the algorithm performance. Thus, looking through a larger solution area is possible and the risk of clustering around a local minimum is reduced. An individual from one population, after migration and transferring to another one, has a significant possibility of boosting evolution in his new environment.

3.2 Evolutionary Operations

Genotype of a neuron is organized similarly to the one in SANE system. It is a sequence of 40-bit definitions of connections with neurons from input and output layers. Every definition begins with eight bits identifying neuron on the other side of connection. This eight bits can be simply treated as an unsigned integer from the range [0–255]. If its value is smaller than 128 then target neuron belongs to input layer, otherwise it comes from output layer. The result of modulo operation coming from division of this value and the number of neurons in target layer is the position of the neuron. After the first eight bits the next 32 come, which encode a floating point number serving as weight. An example genotype and a corresponding phenotype is presented in Fig. 2.

Mutation is done with the probability 0.001 and as an effect reverses a bit value. This operation is performed for every individual in each cycle.

For the neuron, that signals the readiness for reproduction and has enough energy, a partner is chosen from remaining individuals, which also are ready for the reproduction.

The process itself is performed, similarly as in the SANE system, through one-point division of both parents' genotypes. However, copying one of the parents was completely abandoned. Instead of this, individuals created from joined parts of genotypes are only created. Newly created offspring is given a certain portion of life energy transferred from the parents.

3.3 Individuals Life Cycle

Each of environments is at the beginning initialized with an identical number of individuals, divided randomly between the same number of networks. Number of networks maintained in a given environment is constant for the duration of simulation. Individuals posses randomly generated genotypes of configurable length that has to be a multiple of number 40, and one chosen by user from few predefined activation functions. To each of them manager of resources allocates equal starting amount of life energy. Environment is then ready to start the simulation.

Computations proceeds as follows. First, each neuron undergoes the mutation according to the rules described above. Next, quality of all networks is measured. Values from an experimental set are provided to their input layers and their output is compared with expected result. On this basis their adaptation factor—fitness function value is evaluated. Individuals with the best qualities receive energy from resource manager from available pool and the others lose adequate amount of energy that is returned to the energy pool.

In the notification phase every individual decides about undertaking one of the predefined actions. The desire to perform an action is reported to the environment, which at the end performs all requests, making the impression that all actions were done at the same time.

Migration involves changing environment by a neuron that requested it. From such an individual all energy is taken away. For the purpose of maintaining good performance of the system, all migrating individuals are buffered and sent to the central node together, at the end of the computation cycle. The central node redirects them to the environment possessing the smallest number of individuals.

Changing the network operation causes one neuron to be transferred between hidden layers of two networks. There are multiple strategies for designating target network—strategy of the smallest network, in which neuron is assigned to the network with the least neuron count in hidden layer, strategy of best network, where target network becomes the one with the biggest fitness function value, strategy of worst network and random strategy, which is simply a random choice from all available networks. Probability of undertaking an action of changing network is changed dynamically according to energy gains in the last part of evaluation phase. If a neuron was given some energy, then probability is decreased, and if it was taken from him, probability increases. Change of probability value takes place in accordance to logarithmic function.

Swapping places operation is done by exchanging networks between two neurons connected by parent-child relation. It can be activated only by an individual that already has some descendants. Its result is a transfer of child node to the hidden layer of parent and vice versa. This is dictated by the fact that the progeny is, as a product of crossover, potentially a better individual than his parent.

The last but maybe most important, action is reproduction. As it should be performed only by the individuals with a very good fitness value, represented by amount of life energy accumulated, the minimal threshold of energy needed for reproduction should be adequately high. For every individual that is ready for reproduction a partner is chosen. Thus formed couples are subjected to crossover operation, performed according to rules described above, resulting in creating two new individuals per pair. Energetic costs are paid only by neuron which initialized reproduction—this energy is distributed between children.

Computations end when every environment executes a predefined number of steps. The whole process remains constantly under control of the central node.

3.4 SANE Implementation

In order to verify the functionality of MADNESS system it was necessary to confront its results with those obtained from its predecessor—the SANE system. However, for the comparison to be reliable it was necessary to maintain a certain level of similarity in the construction of both systems, with particular emphasis on technology used for their implementation.

All utilized SANE algorithms were implemented to, as possible, strictly imitate the original project [14]. This section focuses on modifications which were done in comparison to prototype and aspects that were not discussed in the original document.

The size of a single neuron genotype was modified. Taking into account the inner representation of floating-point numbers in the used programming language, the number of genes representing weights of connections was increased from 16 to 32. Because of the way in which target neuron is encoded, it is possible to place two or more connections to the same endpoint in one genotype. In such a case they are replaced in phenotype by one connection with weight which value is the sum of all substituted weights. Genetic operations (mutation and crossover) are executed exactly as in the case of regular neuron.

Similarly to MADNESS system, activation function used by neurons from hidden layer and output layer can be configured. Basic SANE module architecture is presented in Fig. 3. The value of fitness function for neuron is calculated as the sum of fitness values from five best network schemes which this particular neuron co-creates.

Together with the neuron population, also the scheme population is maintained. Every individual that belongs to it is built according to a particular prototype. All schemes are subjects to identical, two-phase mutation. Crossover is the effect of one-point division of parent genotypes, but there is no copying of complete genotype from one parent—both descendants are the result of swapping genes. The way of choosing schemes for reproduction and the operation of selecting individuals for next generation are the same as in the case of neurons.

The algorithm works as follows. From all the schemes neural networks are constructed, and to each of them input signal from the test set for the given problem is provided. Basing on the results from calculations, the fitness values for individuals from scheme population are calculated. Having all the networks evaluated, neurons can be given their respective fitness value. Next, a neuron reproduction phase is executed, in which parents are selected and assigned to pairs. From this couples new individuals are

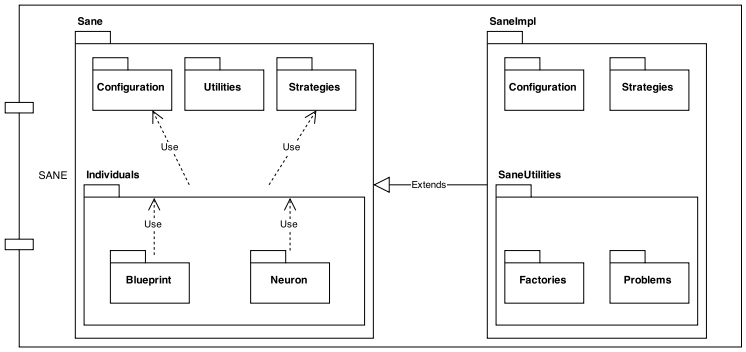


Fig. 3: Sane module architecture

created. From current population worst individuals are removed and all new neurons join the group, founding new generation. At the end, selected genotypes are subjected to mutation. This phase is repeated for network scheme population. The whole procedure is done in cycles until the stop condition (executing a predefined number of steps or reaching by one of the individuals the necessary fitness level) is fulfilled.

4 Experimental Results

System was subjected to series of experiments. A set of problems solvable by neural networks was selected and for each of them the quality of results obtained from SANE and MADNESS modules were compared. Additionally, a number of strategies for allocation of neurons in networks were tested:

1. Steady/Uniform—neurons were added to smallest network in population, keeping a balanced distribution.
2. Random.
3. To worst—neurons added to the network with worst fitness.
4. To best—neurons added to the network with best fitness.

Each of the mentioned experiments was performed multiple times, which enabled the possibility of simple parameters evolution. Only results with the best fitness were taken into account during modules evaluation.

Presented preliminary experimental results have been obtained for multi-layered perceptrons [8] applied to the following problems:

- classification of simple bitmap image—the network had 9 neurons in the input layer and had to classify 4 patterns,
- prediction of a popular chaotic benchmark Mackey-Glass time series [11]—170 neurons in the input layer, one neuron in the output layer.

The parameters of the system for the problem of bitmap classification are given in Table 1.

Table 1: The values of parameters (classification)

	Prob.	**Min. energy**	**Energ. cost**
Death	N/A	N/A	N/A
Addition	0.5	5	1
Removal	0.01	N/A	N/A
Inheritance	0.1	N/A	N/A
Reproduction	1	10	6
Mutation	1	0	0

The other parameters of the system were as follows: network population size was 50, hidden layer size was 10, size of neuron population was 50, size of genotype was 440, mutation probability was 0.001, percent of awarded individuals was 30, total amount of resources was 300, initial amount of resource was 5 and penalty value was 1.

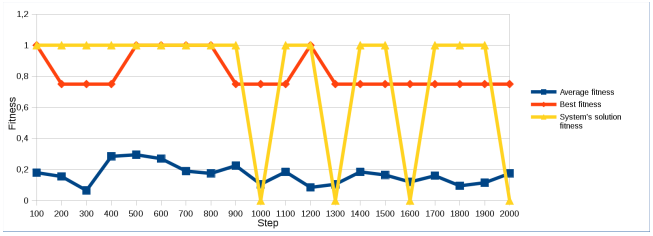


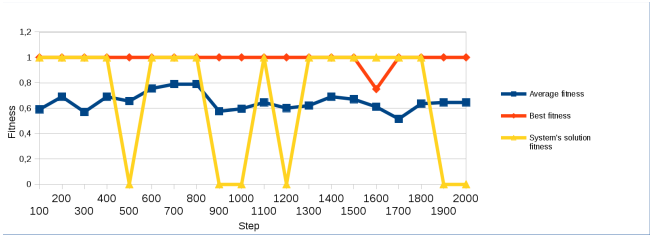
Fig. 4: SANE bitmap classification results

Fig. 4 shows the results of the reference SANE system applied to bitmap classification. The results of MADNESS system obtained for different strategies of neuron assignment are shown in Fig. 5a, 5b and 5c. It seems, that the proposed MADNESS system produced slightly worse results than referenced SANE systems for the problem of simple bitmap classification.

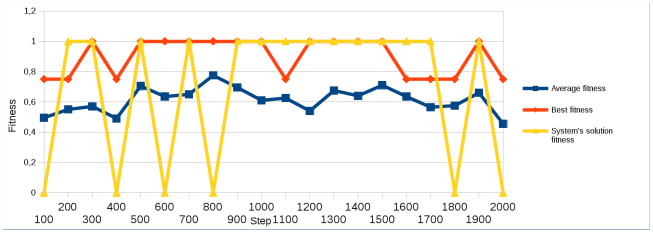
The parameters of the system for Mackey-Glass time series prediction problem are given in Table 2. The presented results were obtained for prediction range 20.

Table 2: The values of parameters (prediction)

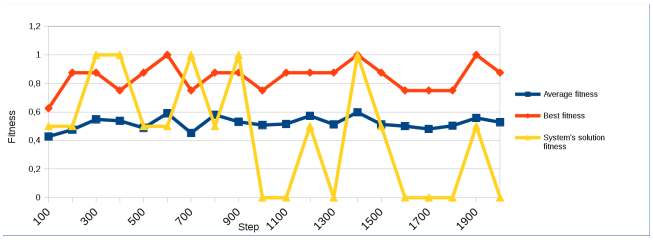
	Prob.	**Min. energy**	**Energ. cost**
Death	N/A	N/A	N/A
Addition	0.2	5	1
Removal	0.01	N/A	N/A
Inheritance	0.1	N/A	N/A
Reproduction	1	10	6
Mutation	1	0	0



(a) Uniform strategy



(b) “To the worst” strategy



(c) Random strategy

Fig. 5: MADNESS bitmap classification results

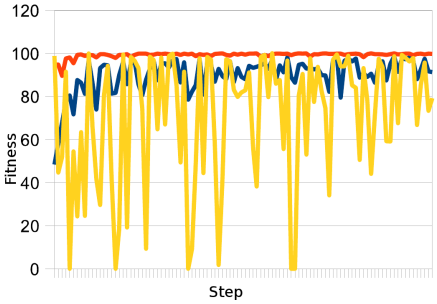
The other parameters for the system were as follows: network population size was 100, hidden layer size was 10, size of neuron population was 100, size of genotype was 1000, mutation probability was 0.001, percent of awarded individuals was 30, total amount of resources was 700, initial amount of resource was 5 and penalty value was 1.

Mackey-Glass series used in experiments is an approximate solution of the following differential equation:

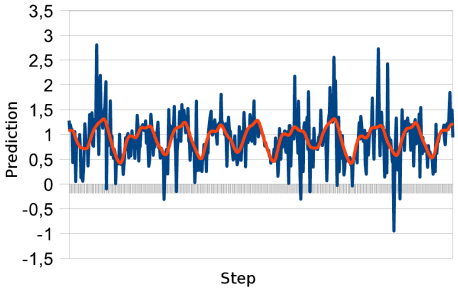
$$\frac{dx}{dt} = \beta \frac{x_\tau}{1 + x_\tau^n} - \gamma x, \gamma, \beta, n > 0 \tag{1}$$

with the following values of parameters: $\beta = 0.1$, $\gamma = 0.2$, $n = 10$, $\tau = 17$.

Fig. 6a shows the results of prediction obtained with SANE system (blue—average fitness, red—best fitness and yellow—system’s solution fitness), while in Fig. 6b the actual prediction that may be observed (blue—original time series, red—predicted time series). In the similar way, the prediction results obtained with MADNESS system are

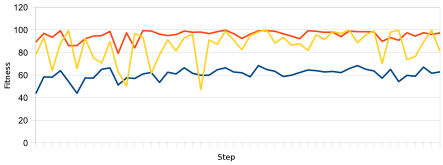


(a) Prediction results

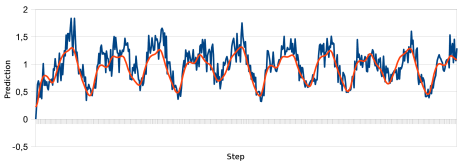


(b) Original and predicted time series

Fig. 6: SANE: Mackey-Glass series prediction results



(a) Prediction results



(b) Original and predicted time series

Fig. 7: MADNESS: Mackey-Glass series prediction results

presented in Fig. 7a and 7b. The prediction results are more or less equivalent in the means of prediction accuracy, however it is easy to see, that the diversity of solutions obtained with the use of MADNESS system is much lower than in the case of SANE system (compare Fig. 6a and 7a).

4.1 Efficiency of SANE and MADNESS

Promising results were obtained when examining efficiency of the two examined systems. Features of MADNESS agent-based approach (lack of global control, varying number of neural individuals) allowed to significantly decrease the computation time (see Table 3).

Table 3: Computation time (in ms/100 iterations) for different number of individuals

	50	100	200	500
SANE	266	590	780	1780
MADNESS	98	210	390	130

The number of neural individuals was the same in the beginning of computation, but it could be decreased in the case of MADNESS system, hence the possibility of

speeding-up the computation in an intelligent way. The presented results have been obtained on PC dual-core computer.

5 Concluding Remarks

In the passed years in which neural network based solutions were utilized, few distinctive groups of optimization algorithms emerged. The first wave—classical iterative algorithms had the advantage of being fairly simple and straightforward, but they soon proved to be far too limited for further use. Vacant spot for new optimization techniques was soon to be filled by genetic algorithms. These kind of methods were more flexible and provided many workarounds for problems previously categorized as unsolvable.

Nonetheless, their maximum potential was also reached in time. Seeing the more and more obvious limitations of GA, some of new implementations of optimization techniques started to be based on multi-agent approach.

In this paper the new agent-based evolutionary approach was presented. Preliminary results are very promising, especially for bitmap classification problem. Average, and for some cases best, fitness is significantly higher than when using pure SANE algorithm. Computation time needed to utilize the system is within reasonable boundaries but of course further experiments and analysis will be needed.

Future research will include performance improvements of the proposed agent-based algorithm. Also new techniques based on the general agent-based neuro-evolution approach would be proposed—for example different variants of agent-based co-evolution, such as cooperative [5] and competitive [4] approach, as well as techniques based on sexual selection mechanism [3].

Acknowledgments. This research was partially supported by Polish Ministry of Science and Higher Education under AGH University of Science and Technology, Faculty of Computer Science, Electronics and Telecommunications statutory project.

References

1. Byrski, A., Kisiel-Dorohinicki, M., Nawarecki, E.: Agent-based evolution of neural network architecture. In: Hamza, M. (ed.) Proc. of the IASTED Int. Symp.: Applied Informatics. IASTED/ACTA Press (2002)
2. Cetnarowicz, K., Kisiel-Dorohinicki, M., Nawarecki, E.: The application of evolution process in multi-agent world (MAW) to the prediction system. In: Tokoro, M. (ed.) Proc. of the 2nd Int. Conf. on Multi-Agent Systems (ICMAS'96). AAAI Press (1996)
3. Dreżewski, R., Cetnarowicz, K.: Sexual selection mechanism for agent-based evolutionary computation. In: Shi, Y., van Albada, G.D., Dongarra, J., Sloot, P.M.A. (eds.) Computational Science - ICCS 2007, 7th International Conference, Beijing, China, May 27 - 30, 2007, Proceedings, Part II. LNCS, vol. 4488, pp. 920–927. Springer-Verlag, Berlin, Heidelberg (2007)
4. Dreżewski, R., Siwik, L.: Multi-objective optimization technique based on co-evolutionary interactions in multi-agent system. In: M. Giacobini, et al. (ed.) Applications of Evolutinary Computing, EvoWorkshops 2007: EvoCoMnet, EvoFIN, EvoIASP,EvoINTERACTION, EvoMUSART, EvoSTOC and EvoTransLog, Valencia, Spain, April11-13, 2007, Proceedings. LNCS, vol. 4448, pp. 179–188. Springer-Verlag, Berlin, Heidelberg (2007)

5. Drężewski, R., Siwik, L.: Agent-based co-operative co-evolutionary algorithm for multi-objective optimization. In: Rutkowski, L., Tadeusiewicz, R., Zadeh, L.A., Zurada, J.M. (eds.) Artificial Intelligence and Soft Computing – ICAISC 2008, 9th International Conference, Zakopane, Poland, June 22-26, 2008, Proceedings. LNCS, vol. 5097, pp. 388–397. Springer-Verlag, Berlin, Heidelberg (2008)
6. Dürr, P., Mattiussi, C., Floreano, D.: Neuroevolution with analog genetic encoding. In: Parallel Problem Solving from Nature — PPSN IX, Procedings. pp. 671–680 (2006)
7. Floreano, D., Mattiussi, C.: Bio-inspired artificial intelligence. MIT Press (2008)
8. Haykin, S.: Neural Networks: A Comprehensive Foundation. Prentice Hall (1999)
9. Kavzoglu, T., Vieira, C.A.O.: An analysis of artificial neural network pruning algorithms in relation to land cover classification accuracy. In: Proc. of the Remote Sensing Society Student Conference. pp. 53–58. Oxford University Press (1998)
10. Kriesel, D.: A brief introduction to neural networks (2007), `http://www.dkriesel.com`
11. Mackey, M., Glass, L.: Oscillation and chaos in physiological control systems. Science 197, 287–289 (1977)
12. Mitchell, M.: An Introduction to Genetic Algorithms. MIT Press (1998)
13. Mohraz, K., Protzel, P.: Flexnet a flexible neural network construction algorithm. In: Proc. of the 4th European Symposium on Artifical Neural Networks (ESANN '96). pp. 111–116 (1996)
14. Moriarty, D.E., Miikkulainen, R.: Forming neural networks through efficient and adaptive coevolution. Evolutionary Computation 5, 373–399 (1997)
15. Precup, R.E., David, R.C., Petriu, E.M., Preitl, S., Paul, A.S.: Gravitational search algorithm-based tuning of fuzzy control systems with a reduced parametric sensitivity. In: Soft Computing in Industrial Applications, pp. 141–150. Springer (2011)
16. Stanley, K.O., Miikkulainen, R.: Evolving neural networks through augmenting topologies. Evolutionary Computation 10(2), 99–127 (2002)
17. T. Ragg, H.B., Landsberg, H.: A comparative study of neural network optimization techniques. In: 13th International Conference On Machine Learning: Workshop Proceedings On Evolutionary Computing And Machine Learning. pp. 111–118. Springer (1997)
18. Wooldridge, M.: Introduction to Multiagent Systems. John Wiley & Sons, Inc., New York, NY, USA (2001)
19. Yao, X., Liu, Y.: Evolving artificial neural networks through evolutionary programming. In: L. J. Fogel, et al. (ed.) Evolutionary Programming V: Proc. of the 5th Annual Conf. on Evolutionary Programming. MIT Press (1996)
20. Zăvoianu, A.C., Bramerdorfer, G., Lughofer, E., Silber, S., Amrhein, W., Klement, E.P.: Hybridization of multi-objective evolutionary algorithms and artificial neural networks for optimizing the performance of electrical drives. Engineering Applications of Artificial Intelligence 26(8), 1781–1794 (2013)