Adaptive Control of Neural Network Synthesis

Pavel Vařacha[1]

[1]Tomas Bata University in Zlín, Faculty of Applied Informatics, Zlín, Czech Republic

Abstract. Neural Network Synthesis is a method based on based on Analytic Programming and asynchronous implementation of Self-Organising Migration Algorithm. This synthesis woks as an algorithm capable of creating and learning and artificial neural networks as well as optimizing their structures and connections. This paper introduce an idea of adaptive control of Self-Organising Migration Algorithm based on complexity of processed neural network structure. Such approach already recorded several successful application in modelling and simulation.

1. Introduction

Neural Network Synthesis (ANN synthesis) is an algorithm capable of creating and learning and artificial neural networks as well as optimizing their structures and connections. To describe this method in all details would significantly exceed a possible extend of this paper. Nevertheless ANN synthesis mechanisms are very well elaborated in papers [1] and [2]. An interested reader is respectfully asked to study this previous publications as are referred at the end of the paper.

ANN synthesis is based on Analytic Programming (AP) and asynchronous implementation of Self-Organizing Migration Algorithm (SOMA). Important facts about AP which are necessary for basic understanding of an idea proposed in the paper as an adaptive individual handling are described in chapters 2. And 3. For better understanding of SOMA, please, refer paper [3] or [4].

This paper explores a possibility to make it more effective by adaptive control of SOMA parameters. The main idea is an intelligent control of the ANN synthesis based on complexity of processed neural network structure.

2. Analytic Programming

The main principle (core) of AP is based on a discrete set handling (DSH) (Fig. 1). DSH shows itself as a universal interface between the EA and the symbolically solved problem. This is why AP can be used almost by any evolutionary algorithm. In the case of ANN synthesis the algorithm is SOMA.

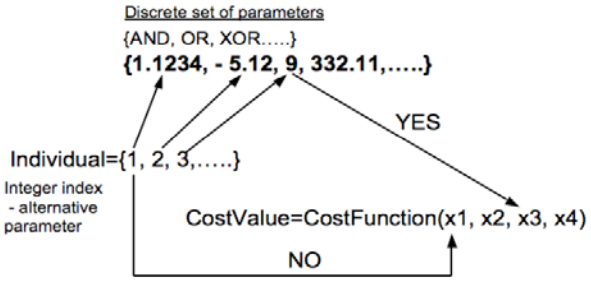

Figure 1. DSH principle

Briefly stated, in AP, individuals consist of non-numerical expressions (operators, functions,…) which are represented within the evolutionary process by their integer indexes. Each index then serves as a pointer into the set of expressions and AP uses it to synthesize the resulting function-program for the Cost Function evaluation.

=All simple functions and operators are in the so called General Function Set (GFS) divided into groups according to the number of arguments which can be inserted during the evolutionary process to create subsets GFS3, GFS2...GFS0.

Table 1. Example of GFS and its subsets

GFS Degree	Contains
GFSall	f(x1, x2, x3), +, -, *, /, Power, Abs, Round, Sin, Cos, t, K, τ, 1, 2
GFS3	f(x1, x2, x3)
GFS2	+, -, *, /, Power
GFS1	Abs, Round, Sin, Cos
GFS0	t, K, τ, 1, 2

* Corresponding author: varacha@fai.utb.cz

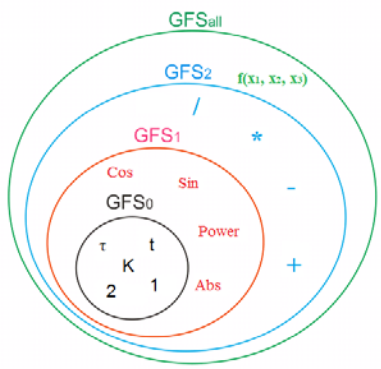

Figure 2. GFS subsets hierarchy

The functionality of AP can be seen in the specific example in Fig. 3:

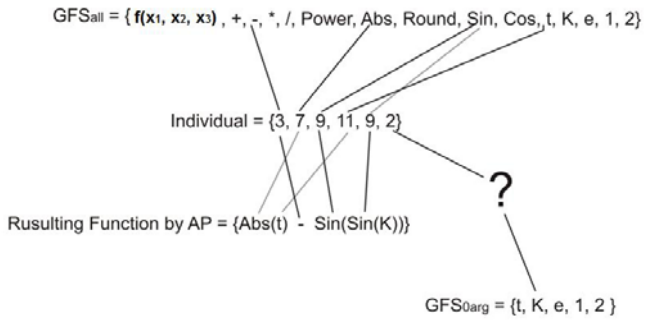

Figure.3. Main principles of AP

The individual consists of 6 arguments (indices, pointers to GFS). The first index is 3, meaning that it is taken from the set of functions GFS_{all}. The function *minus* has two arguments; therefore indexes 7 and 9 are arguments of *minus*.

$$6 + 7 \quad (1)$$

Index 7 is then replaced by *Abs* and index 9 by *Sin*.

$$Abs + Sin \quad (2)$$

Abs and *Sin* are one-argument functions. Then, index 9 follows index 11, which is replaced by *t*.

$$Abs(t) + Sin \quad (3)$$

Sin is also a one-argument function. Then, after index 11, the individual takes index 9, which is replaced by *Sin* and this *Sin* becomes an argument of the previous *Sin*.

$$Sin(Tan) + Sin(Sin(\quad (4)$$

The last index is 2, but in this case there is the function *Plus*. *Plus* needs two arguments to work properly. AP will not allow this, as there is not any other free pointer to be used as the argument. Instead of *Plus*, AP will jump into the subspace, in this case directly to the GFS_{0arg}. In the GFS_{0arg} it finds the second element, which is *K*. And by doing so, we get (5).

$$Abs(t) + Sin(Sin(K)) \quad (5)$$

The number of pointers actually used from an individual before the synthesized expression is closed is called *depth*. This example is based on the relevant and previously published work in.

3. Neural Network Synthesis

The previous chapter described basic concepts of AP. Such concepts can be easily employed to synthetize different ANN structures as can be seen from Fig. 4. This process of structural synthesis as well as ANN learning is well described for example in [3] and [4].

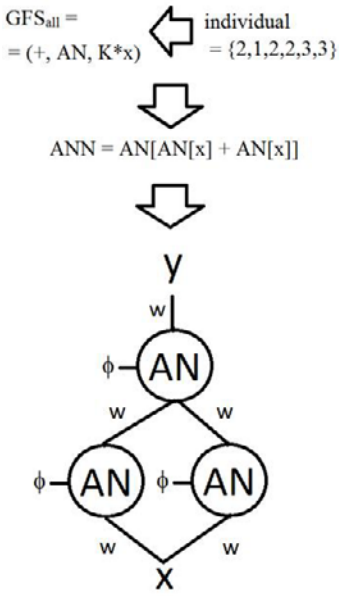

Figure 4. An Example of ANN synthesis

4. Experiment Designed for Adaptive Control

Processes described in chapters 2. and 3. have to be optimized by a specific evolutionary algorithms. In case of ANN synthesis it basically means to try variously complex ANN structures (individuals in AP) to improve them in evolutionary way. In place of such evolutionary algorithm ANN synthesis commonly employ SOMA. An algorithm which is well described in [3] and [4]. However SOMA is not anyhow adaptive to coop with various lengths of individuals in AP. SOMA simply treat all such individuals as an individual of constant length (typically 100). For this paper one control parameter of SOMA was chosen to improve this behavior.

An adaptive control proposed in this paper aim to influence control parameter of SOMA named PRT to improve its ability to deal with different lengths of individuals. For each individual SOMA will used different setting of PRT bases on length of a solved individual itself.

Commonly, SOMA is set on PRT = 0.1. In contradiction this paper proposes adaptive strategy so PRT will differ from individual to individual PRT = 1 / depth.

In order to statistically evaluate this new adaptive handling approach the function approximation problem was chosen as an aim of the experiment. The function (6) proposed by [5] as an appropriate approximation benchmark was chosen to be approximated by the ANN.

$$y = x_i^5 - 2\,x_i^3 + x_i \quad (6)$$
$$\text{where } x_i \text{ is in } \langle -1, \text{by the step } 0.04, 1 \rangle$$

Fig. 5 (automatically generated by ANN synthesis software) shows an example of synthetized ANN approximating (6). The difference between the ANN and

(7) is depicted as a red area which could be minimized by the process of synthesis. An goal of ANN synthesis in this case can be simply described as founding smallest possible ANN able to approximate given function. Simplicity of such function is purely purposeful here as the main goal is not test ability of ANN synthesis to solve complicated tasks but to test how the method improve while applying proposed adaptive strategy.

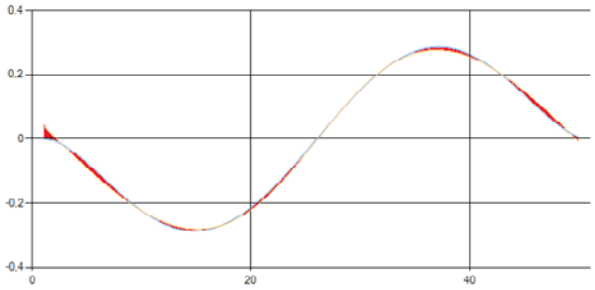

Figure 5. Approximation of (7) by synthetized ANN

AP was executed 100 times (physically on 8 cores of the Super Micro Server) to produce an ANN with the RMSD < 0.005. The main intention was to find such an ANN which met this condition and which simultaneously used as few AN as possible. All SOMA's control parameters required for repetition of this experiment are described in following tables. The setting of Asynchronous SOMA used as the EA for AP can be seen in Table 1. and SOMA setting used for ANN learning in Table 2.

Table 1. Setting of SOMA used as EA for AP

Number of Individuals	48
Individual Parameters	100
Low	0
High	3
PathLength	3
Step	0,11
PRT	Based on **experiment**
Divergence	0.01
Period	1

Table 2. Setting of SOMA used to optimize Kn

Number of Individuals	number ofKn* 0.5 (at least 10)
Individual Parameters	100
Low	-10
High	10
PathLength	3
Step	0,11
PRT	$1/K_n$
Divergence	0.01
Period	6

Based on experiment setting PRT which SOMA used to optimize Kn is set either conservatively to PRT = 0.1 or adaptively PRT = 1 / depth.

5. Results

The control of SOMA parameters consists in replacement of the static PRT value by the value which inversely depends on depth of a currently operated individual.

Table 3. Static PRT vs. adaptive control

	PRT = 1/ depth	PRT = 0.1
Average time needed for synthesis	194 s	373s
Average number of used AN	9	13

A total of 1,189,870 evaluations of AP individual fitness were completed during 100 AP executions while the PRT was set to 0.1 and the separate SOMA run was performed for all of them to set their Kn value. Without the adaptive PRT, AP was able to find an optimal ANN in only 1 case in comparison with 4 successful cases in the original experiment.

Conclusion

ANN synthesis already recorded several successful application considering practical casers of modelling and simulation [6] – [9]. It was also applied on large set of widely recognized benchmark functions [10], [11] with respect to the function approximation, prediction and problems. This results vindicate efforts for its further development.

Obtained results of experiment considered in this paper proves an ability of proposed adaptive control to further improve ANN synthesis. It can significantly improve ANN synthesis ability to overshadow concurrent method of ANN optimization as are [12] – [17].

The method of adaptive individual handling proved in this paper are going to be applied on a practical industrial example and comparison of the improved ANN synthesis is to be subjected by a future study.

Acknowledgements

This work was supported by the Ministry of Education, Youth and Sports of the Czech Republic within the National Sustainability Programme project No. LO1303 (MSMT-7778/2014).

References

1. Zelinka I., Vařacha P., Oplatková Z., Volná, E. *Structural Synthesis of Neural Network by Means of Analytic Programmig*. In 12th International Conference on Soft Computing. Czech Republic, VUT Brno, p. 25-30 (2006)
2. Vařacha P. *Neural Network Synthesis Dealing with Classification Problem*. In Recent Researches in Automatic Control. Montreux : WSEAS Press, p. 377-382 (2011)
3. Šenkerik R., Zelinka I., Davendra D., Oplatkova Z. *Utilization of SOMA and differential evolution for robust stabilization of chaotic Logistic equation*, Computers & Mathematics with Applications, Volume 60, Issue 4, Pages 1026-1037, ISSN 0898-1221, doi 10.1016/j.camwa.2010.03.059. (2010)
4. Šenkeřík R. Oplatkova Z., Zelinka I, Davendra D., *Synthesis of feedback controller for three selected chaotic systems by means of evolutionary techniques: Analytic programming*, Mathematical and Computer Modelling, ISSN 0895-7177, 10.1016/j.mcm.2011.05.030 (Available online 27 May 2011)
5. Vařacha P., Zelinka I. *Analytic Programming Powered by Distributed Self-Organizing Migrating Algorithm Application*. In IEEE Proceedings 7th International Conference Computer Information Systems and Industrial Management Applications. Ostrava : IEEE Computer Society, p. 99-100 (2008)
6. Prechelt L., *Proben1—A Set of Neural Network Benchmark Problems and Benchmarking Rules*, Universität Karlsruhe, Germany (1994)
7. Mangarianm O.L., Wolberg W.H., *Cancer diagnosis via linear programming*, SIAM News, **Volume 23**, Number 5, , p. 1-18 (1990)
8. Král E., Dolinay V., Vašek L., Vařacha P. *Usage of PSO Algorithm for Parameters Identification of District Heating Network Simulation Model*. In 14th WSEAS International Conference on Systems. Latest Trands on Systems.Volume II, Rhodes, WSEAS Press (GR) . p. 657-659 (2010)
9. CHRAMCOV, Bronislav. *Identification of time series model of heat demand using Mathematica environment*. In Recent Researches in Automatic Control. Montreux : WSEAS Press, s. 346-351 (2011)
10. Zelinka I., *Studies in Fuzziness and Soft Computing*, New York : Springer-Verlag, (2004)
11. Koza J. R., *Genetic Programming*, MIT Press, ISBN 0-262-11189-6 (1998)
12. Jui-Yu W., *MIMO CMAC neural network classifier for solving classification problems*, Applied Soft Computing, **Volume 11**, Issue 2, The Impact of Soft Computing for the Progress of Artificial Intelligence, p. 2326-2333 (2011)
13. Falco D.I., Cioppa E., Tarantino, *Discovering interesting classification rules with genetic programming*, Applied Soft Computing 1, p. 257–269 (2002)
14. Brameier M., Banzhaf W., *A comparison of linear genetic programming and neural networks in medical data mining*, IEEE Transactions on Evolutionary
15. Turner et al., *Grammatical Evolution of Neural Networks for Discovering Epistasis among Quantitative Trait Loci Evolutionary Computation*, *Machine Learning and Data Mining in Bioinformatics Book Series Title*: Lecture Notes in Computer Science Publisher: Springer Berlin / Heidelberg, p: 86 – 97 (2010)
16. Vonk E., Jain L.C., Johnson R.P., *Automatic Generation of Neural Network Architecture Using Evolutionary Computation*, Advances in Fuzzy Systems – Applications and Theory, **Volume 14**, World Science, ISBN: 981-02-3106-7 (1997)
17. Koza J. R. et al. *Genetic Programming III*; *Darwinian Invention and problem Solving,* Morgan Kaufmann Publisher, (1999)