Distributed Self-Organizing Migrating Algorithm and Evolutionary Scanning

Pavel Varacha and Ivan Zelinka
Department of Applied Informatics
Tomas Bata University in Zlin
Nad Stranemi 4511, Zlin, 760 05, Czech Republic
E-mail: {varacha,zelinka}@fai.utb.cz

KEYWORDS

neural network, symbolic regression, analytic programming, evolutionary searching, evolutionary scanning, evolutionary algorithms, distributed computing

ABSTRACT

This paper presents an idea of new algorithm combining advantages of evolutionary algorithm and simple distributed computing to perform tasks which required many re-runs of the same program. Computing time is shorted due to elementary distribution within a number of common computers via the Internet. Progressive .NET Framework technology allowing this algorithm to run effectively and examples of possible usage are also described.

The algorithm deals with a problem of synthesis of the artificial neural networks using the evolutional scanning method. The basic task to be solved is to create a symbolic regression algorithm on principles of analytic programming, which will be capable of performing a convenient neural network synthesis. The main motivation here is the computerization of such synthesis and discovering so far unknown solutions.

INTRODUCTION

Distributed Self-Organizing Migrating Algorithm (DISOMA) is a highly effective evolutionary optimization heuristic combining eminent computational performance with attainable technical requirements. DISOMA is derived from the Self-Organizing Migrating Algorithm (SOMA) (Zelinka 2002,2004) which has proven to be very useful for optimization of many hard-solvable problems. However there exist some tasks that cannot be solved by a small band of SOMA executes, for example, a task that is deduced from the model containing a number of stochastic based variables. In such cases huge set of SOMA cycles has to be run and final solution is interpolated through all obtained possibilities via statistic procedures. This technique can be extremely demanding in terms of computing time. The mainframes capable of computing such exacting assignments are not always available. DISOMA reduces such demand by distributing separate SOMA executes to bigger group of independent computers. The Internet connection and .NET Framework presence are the only conditions required and that each computer is to be part of the process. This paper introduces DISOMA as a new method of functional optimization, bringing a utile tool for the users who can not get access to expensive mainframes but have at their disposal large group of personal computers, for example lab computers. Various DISOMA aspects are discussed including installation, structure, methods of remote control and communication, operation system requirements and examples of use. The paper concludes by explaining the importance of DISOMA for connected research of Analytic Programming (Zelinka, Oplatkova 2003) and also for research connected with SOMA itself.

METHODS

Combination of many times proved SOMA and modern .NET Framework architecture is what makes DISOMA such powerful. Main techniques used to construct DISOMA are described in this chapter.

SOMA

SOMA is an effectual artificial intelligence method of functional optimization. Together with Genetic Algorithms, Simulated Annealing and Differential Evolution, SOMA can be treated as an evolutionary algorithm. Set of hypothetic solutions is as a group of individuals moving towards Leader through their environment (N dimensional hyper-plane) to reach the best living condition (functional minimum). Owing to this quality SOMA is able to work out the best (or very good) solution for many problems which are unsolvable by rigorous mathematical techniques. Following table of parameters has to be set SOMA can be run.

Proceedings 22nd European Conference on Modelling and Simulation ©ECMS Loucas S. Louca, Yiorgos Chrysanthou, Zuzana Oplatková, Khalid Al-Begain (Editors)
ISBN: 978-0-9553018-5-8 / ISBN: 978-0-9553018-6-5 (CD)

Table 1. SOMA parameter

Parametr name:	Description:
CostFunction	function defining the problem it the way of founding minimal value
PathLength	how far an individual goes on its way toward Leader
Step	how many times individual stops during its path
PRT	degree of randomness during individual moves
PopSize	number of individuals within the population
Migrations	how many times population will be migrating toward the Leader
MinDiv	terminal vertical distance between the best and the worst individual

Windows Services

Windows Services are applications that can be automatically started when the operating system boots. They can run without having an interactive user logged on the system. Any Functionality can be built in the service such as scanning for files to do backup or a virus check, or starting a .NET Remoting server, for example. (Robinson 2004)

.NET Remoting

Many applications are not standalone applications that are running on a single system, but they use some network communication technologies to invoke methods on a remote server. This is what .NET Remoting is good for. .NET Remoting can be used for accessing objects in another application domain. Instead of sending data, it can invoke methods across the network across the network. (Robinson 2004)

DISOMA

DISOMA works on ordinary client - server architecture however unconventionally there are many servers and only one client. Within DISOMA structure every server is called “Solver” and the client is called “Scheduler”. DISOMA itself can be defined as a set of one Scheduler and altering figure of Solvers. Large number of Solvers communicating with Scheduler (more than 10 recommended) is crucial for DISOMA capability to solve more extensive tasks. Main attributes of Solver and Scheduler are described further.

Solver

Solver is designed as Windows Service. This conception brings basically automated method how to install Solver via Windows Service Installer. Every computer running on Windows XP or Windows 2000 can be easily equipped by one instance of Solver. After installation Solver is running together with Windows no matter if any user is logged in or not. Every solver communicates with Scheduler only and knows nothing about possible existence of other Solvers. Computer with Solver instance has to be connected with Internet to allowed Solver participating on DISOMA.

Solver contains two main parts, communication routines and SOMA. After Windows is switched on Solver sends message to Scheduler that it is ready to participate on calculations and what is its IP address. Than this message is sent periodically till Windows are not switched off. If an order comes from Scheduler, Solver starts computing one instance of SOMA. After an adequate solution is found its coordinate are send back to Scheduler and Solver is ready for another task. If the solution was not found within specified time an error warning is sent to Scheduler and calculation is stopped.

Scheduler

Scheduler is a simple WinForm application connected to SQL database via ADO .NET technology. Scheduler itself can be run without installation on computer equipped by .NET Framework however Microsoft SQL Server has to be present in order to help Scheduler administrating SQL database. Also Scheduler position within the Internet is limited due to its exact IP address with has been specified firmly within Solvers code.

Scheduler has to handle thee main databases - defined tasks, found solutions and connected Solvers. The user creates “Defined tasks database“ manually. This definition can be originated directly via SQL server or by the help of Scheduler window. Every defined task has to contain vector of SOMA parameters and Cost Function (see Table 1) also number of repetition can be included. “Connected Solvers database” contains all Solvers connected to Scheduler and has to be periodically updated in order to remain relevant. Every included Solver assumes ready, calculating or error state. Scheduler sends tasks to Solvers which are ready - always one task (the oldest item from Defined task database) to one Solver. When Scheduler receives coordinates of successfully found solution form Solver the solution is stored into “Found solutions database” together with task parameters and solved task is deleted from Defined task database. If any Solver does not return awaited solution within specified time or stops responding its status is changed to error and its task is redirected to another ready Solver.

APLICATION

Class of problems which can be effectively solved by DISOMA includes problems which require solving large number of single SOMA tasks during evaluation

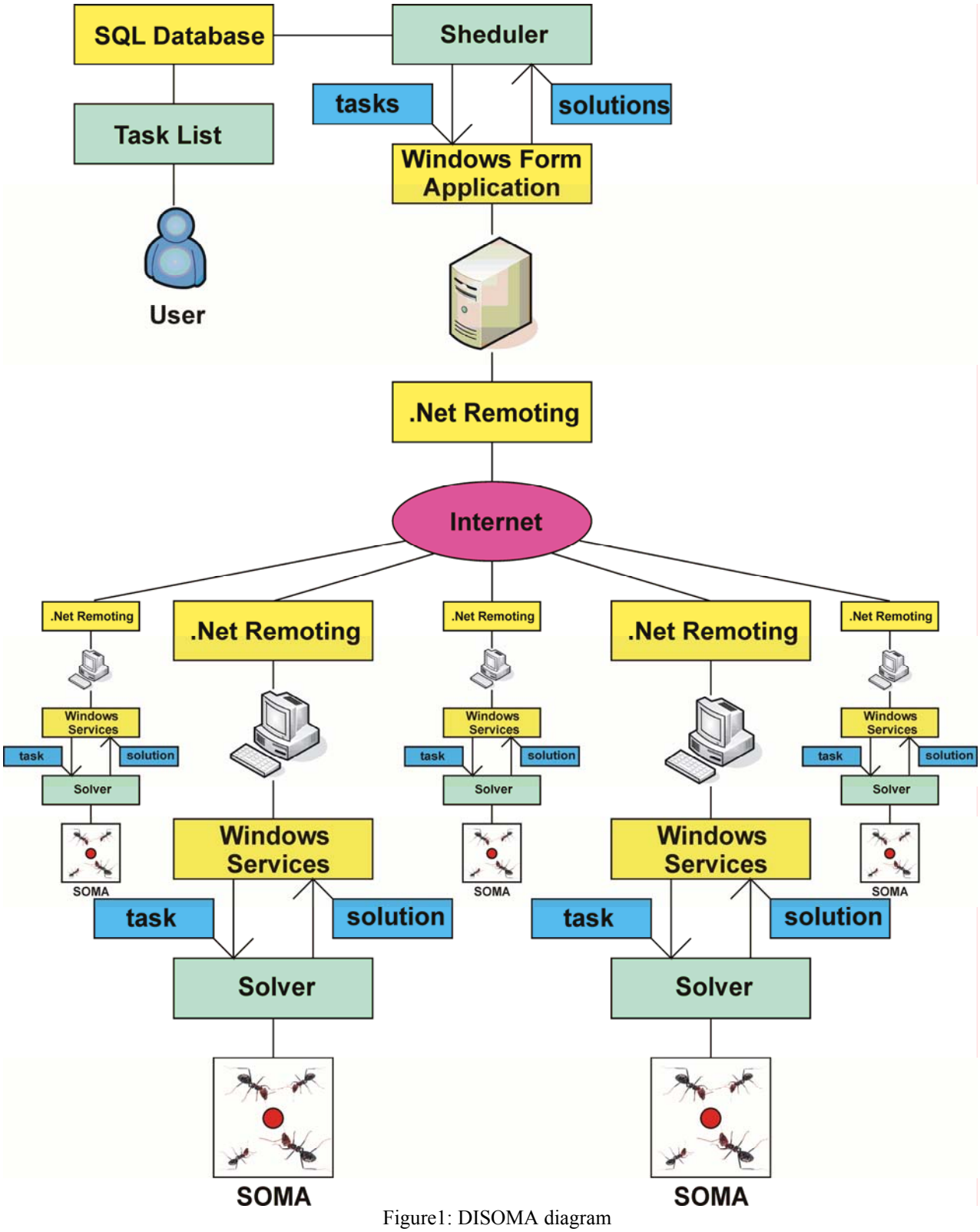


Figure1: DISOMA diagram

which make it ideal tool for Evolutionary Scanning. For more details see (Varacha, Zelinka 2007a).

Successfulness of SOMA sensitively depends on parameters setting (see Table 1). For statistic determination of ideal SOMA parameters many runs of SOMA have to be examines on large representative set of functions. (Zelinka 2004)

EVOLUTIONARY SCANNING

Clause: *Let there be a set of all neural networks with forward running propagation* $\boldsymbol{ANN_{all}} = \{ANN_1, ANN_2, ..., ANN_i, ...\}$ *and a set of all functions* $\boldsymbol{F_{all}} = \{\boldsymbol{f_1}, \boldsymbol{f_2}, ...,$

$\boldsymbol{f_k}$, ...}. *Then for each ANNi* $\in \boldsymbol{ANN_{all}}$ *exists a function* $\boldsymbol{f_k} \in \boldsymbol{F_{all}}$, *alternatively a set of functions* $\boldsymbol{F_k} \subset \boldsymbol{F_{all}}$ *such, that holds* $ANN_i \Leftrightarrow \boldsymbol{f_k}$, *alternatively* $ANN_i \Leftrightarrow \boldsymbol{F_k}$.

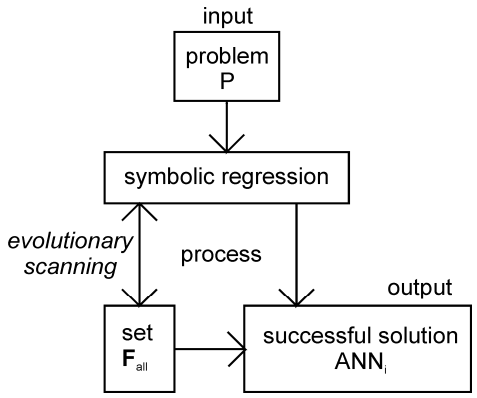
Kolmonogorov theorem (Bose and Liang 1996) further shows also validity of the inverse clause: *For every continuous function* $\boldsymbol{f_k} \in \boldsymbol{F_{all}}$ *exists* $ANN_i \in \boldsymbol{ANN_{all}}$ *such, that holds* $\boldsymbol{f_k} \Leftrightarrow ANNi$.

Task: Design an algorithm, which will by the means of the symbolic regression methods, evolutionary scan a set $\mathbf{F_{all}}$ in order to find:

a) $\mathbf{f_k} \Leftrightarrow ANN_i$

b) $\mathbf{f_k}$, whose at least some subfunctions $\{\mathbf{f_1}, \mathbf{f_2}, ...\} \Leftrightarrow \{ANN_n, ANN_m, ...\}$

which solve the particular problem *P* with global error $E_T < \xi$, where ξ is the user defined biased tolerance threshold.



Figures 2: Principle of the Evolutionary scanning

ANALYTIC PROGRAMMING

Algorithm referred to as analytic programming (AP) proved itself by providing the solution to a variety of problems of the symbolic regression as: synthesis of trigonometric functions (Zelinka 2005a), polynomial functions (Zelinka and Oplatkova 2003), functions of boolean parity (Zelinka and Oplatkova 2004), functions of boolean symmetry (Zelinka at al. 2004), solution of differential equations (Zelinka 2002b) and optimization of the artificial ant tour (Zelinka and Oplatkova 2006). Those accomplishments have led to the motivation to utilise an AP also in the ANN synthesis.

AP has been built-up on the sets of functions, operators and so called terminals, which are mostly invariables or independent variables. These mathematical objects form a set, out of which the AP strives to synthesize a suitable solution – so called *General Function Set (***GFS**). Members of **GFS** are then the principal functions **gf**. (That is, we can easily understand the operators and terminals as functions.) **GFS** = {$\mathbf{gf}_1$, ... **gf***i*, ... **gf***n*}.

The basic principle of AP is based on the handling with Discrete Sets (*Discrete Set Handling*, DSH). DSH forms the interface between the Evolutionary Algorithm (*Evolutionary Algorithm*, EA) (Zelinka, 2004), a problem, which should be solved symbolically.

Thanks to that, almost any EA can be applied in the AP. In such a way the EA in effect works as an engine actuating the AP. From the EA nature further results to be rational to define the Cost Function (**CF**), whose value is necessary to be minimized.

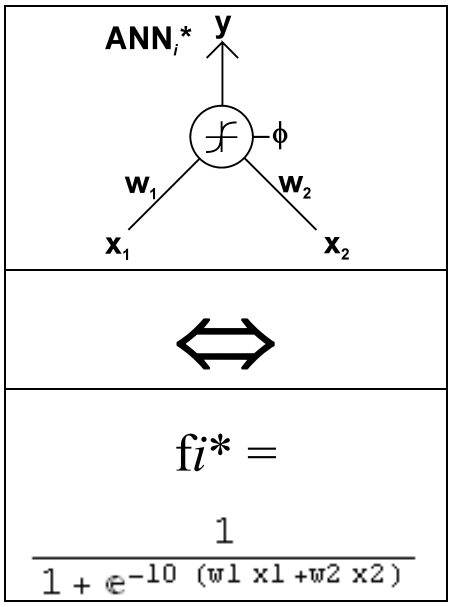
Note: *AP does not use as terminals particular, however, general constants K1 to Kn, whose particular value is determined by means of nonlinear interlacing as closely as possible near the assessment of the relevant* ***CF****.*

SYNTHESIS OF NEURAL NETWORKS

To accomplish by the means of an AP the successfully synthesize of the ANN, it is necessary at first to appropriately choose the elements of **GFS** and correctly define the **CF**, applicable for the problem being solved. To fulfill both those steps is, however, necessary to comprehend the relationship of equivalence between the ANN and functions – namely an AP itself does not understand the ANN conception, it is capable to synthesise only functions, eventually algorithms.

The relation between an ANN and mathematical functions

The connection between an ANN and mathematical functions is often neglected in the professional literature, at the same time each ANN can be expressed as to its equivalent class of functions **f***. The ANN synthesis process is thus a process of searching of a suitable **f* in** the set of class of functions **F**, which contains the classes accrued by combinations of (basic) base functions **b** of a set **B**. The constitution of elements of **B** then differs according to individual types of ANN. The process of ANN education can then be expressed as searching for convenient values of invariables, which differentiate particular functions **f** in **f*** from each other. The resulting function **f** is then equivalent to the searched for ANN.



Figures 3: Equivalence between ANNi* and fi*

That previous described feature of ANN together with aforementioned stated characteristics of an AP, makes from the AP an ideal algorithm for the ANN synthesis. The most important task to solve, is to select a suitable **B**, which will be used as a **GFS.**

Conception of used GFS

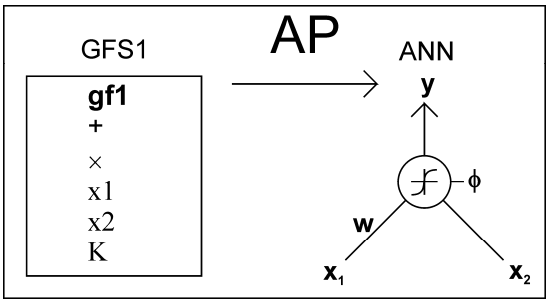
From the preceding text it is clear, that suitable selection of individual **gf** by using **GFS**, is an absolutely crucial question. As a fundamental base stone - function **gf1** – is synthesised ANN, and the choice then becomes **sigmoid** – the transfer function of two input arguments ***arg1***, ***arg2***:

$$gf1 = \frac{1}{1 + e^{-\lambda\ (arg1\ +\ arg2)}}$$

Sigmoid function is in the world of ANN the often used transfer function (Master 1993), which is derived from behavior of real neurons in human brain. For a parameter λ the invariable value 10 was selected, so that the sigmoid nears rather to the unipolar binary function, which is suitable for solutions of classification problems. Against the binary function it , however, has the advantage, that there exists a derivation of it in its whole interval and that it is possible to apply the algorithm dependent on such derivation of function, as it is , for example, just the nonlinear interlacing used in the AP, on the structures, which are thus being created.

Beyond **gf1** there must be an integral part of, as constituted by this research, **GFS** also input independent variables x1 and x2, which represent coordinates of individual points on the classified plane, and then invariables K (K1 to Kn), which shall figure the thresholds ϕ and weights w of the network. To be able to tie up properly the weights and thresholds to the network, it is necessary to add into **GFS** next two **gf,** + (plus) and × (times).

By that one finally arrives at **GFS1** = {**gf1**, +, ×, x1, x2, K}. From the point of view of the problem it handles about so-called **complete GFS,** consequently **GFS1** has by the means of AP guaranteed the potential of solving the ANN syntheses – for the set **B,** through combination of which rises already known solution ANN of the particular problem, is valid **B** ⊆ **GFS1** - so it comprises all elements, which build and compose the relevant **f**.



Figures 4: Principle of ANN synthesis from **GFS1**

Such definition of **GFS1** is an attractive one because an AP can, thanks to it synthesize not only classical neural structures, known from the rigorous theory of ANN, however, also the networks of pseudo-neural character, eventually quite non-neural functions. These properties from the world of neural functions shall, thanks to AP involuntarily combine with further mathematical transformations.

Effect of GFS1 on the synthesized ANN topology

The sub-functions can arise from the set **GFS1** under favorable conditions during cultivation of an individual in AP, which will compose the winner individual, and on which can be looked on as neurons in ANN. Examples of such arrangement of individual **gf** ∈ **GFS1** into convenient subfunctions will be introduced in future studies.

Definition of CF

The correct definition of **CF** is unique for every problem to be solved. **CF** should in itself namely include the whole testing set of the synthesized ANN. **CF** works in such cases on the basis of testing the ANN responses on particular elements of the testing set. What, however, remains always same is the chosen philosophy, under which the relevant **CF** is created.

As a fundamental model of CF1, the staircase conception **CF** was selected for the synthesis. Its operation is, that at the beginning of scoring the **CF** of particular individual (ANN) is valid **CF** = 0. Step by step all elements of the tested set are passed through. If

there shows at any tested element, that the response of ANN is divergent to the requirements of the testing set an increment of **CF** = **CF** + 1 is performed. By that way the CF1 can gather the discrete values {1,2, ..., n}, where n is number of elements in the testing set. CF1 thus calculates on how many elements of testing set replied ANN with the inadmissible reply.

By all of that it is assumed, that at the output of ANN there is inserted one more neuron, which standardizes the replies greater than 0.8 to 1 and replies less than 0.2 damps to 0.

CONCLUSION

Applied on described class of problems DISOMA can compensate expensive mainframes inside organizations which cannot afford such advanced accessories but manages number of standard computer. This appears to be reasonable way how to save unnecessary investments. Presently DISOMA goes trough the stadium of debugging however this algorithm is going to be expose on the Internet for a public use. During the progress many completive ideas, such as creating remote control of Scheduler via ASP .NET services, are proposing and waiting for integration into DISOMA. For more information see (Varacha, Zelinka 2007b).

ACKNOWLEDGMENTS

This work was supported by grant NO.MSM 7088352101 of the Ministry of Education of the Czech Republic and by grants of Grant Agency of Czech republic GACR 102/06/1132 and GACR 102/050271.

REFERENCE

BOSE, B.K., LIANG, P. 1996. *Neural Network Fundamentals with Graphs, Algorithms, and Aplications*. McGraw-Hill Series in Electrical and Computer Engineering, 478 p. ISBN 0-07 006618-3.

MASTER T., 1993, Practical Neural Networks Recipes in C++, London : Academi Press, 1993, 490 s. ISBN 0-12-479040-2.

ROBINSON, R., AT AL. 2004 *Professional C# 3rd Edition*, Wiley Publishing, Inc., Indianapolis , ISBN 0-7645-5759-9.

ZELINKA I. 2002a. Analytic programming by Means of Soma Algorithm. Mendel ’02, *In: Proc. 8th International Conference on Soft Computing Mendel’02*, Brno, Czech Republic, 93-101., ISBN 80-214-2135-5.

ZELINKA I. 2002b. Analytic programming by Means of Soma Algorithm. ICICIS’02, *First International Conference on Intelligent Computing and Information Systems*, Egypt, Cairo, ISBN 977-237-172-3

ZELINKA I., OPLATKOVA Z. 2003. Analytic programming – Comparative Study. *CIRAS’03, The second International Conference on Computational Intelligence, Robotics, and Autonomous Systems*, Singapore, ISSN 0219-6131.

ZELINKA I., OPLATKOVA Z. 2004. Boolean Parity Function Synthesis by Means of Arbitrarry Evolutionary Algorithms - Comparative Study, *In: 8th World Multicon-ference on Systemics, Cybernetics and Informatics (SCI 2004)*, Orlando, USA, in July 18-21.

ZELINKA I., OPLATKOVA Z., NOLLE L. 2004. Boolean Symmetry Function Synthesis by Means of Arbitrarry Evolutionary Algorithms - Comparative Study, *In: 18th European Simulation Multiconference (ESM 2004)*, Magdeburg, Germany, June 13-16, 2004, ISBN 3-936150-35-4.

ZELINKA I. 2004. *SOMA* - Self Organizing Migrating Algoritm, kap. 7, str. 33, in B.V. Batu, G. Onwubolu (eds), *New Optimization Techniques in Engineering*, Springer-Verlag.

ZELINKA I, OPLATKOVA 2006. Z. Investigation on Artificial Ant using Analytic Programming, *GECCO 2006*, Seatle, WA, USA

VARACHA P., ZELINKA I. 2007a. Distributed Self-Organizing Migrating Algorithm (DISOMA). *ICCC 2007, 8th International Carpathian Control Conference, Štrbkké Pleso*, Slovak Republic, ISBN 978-80-8073-805-1

VARACHA P., ZELINKA I. 2007b. Synthesis of artificial neural networks by the means of evolutionary scanning - preliminary study. *ECMS 2007, 21st European Conference on Modelling and Simulation*, Praha, Czech Republic, ISBN 978-0-9553018-2-7