INTERNATIONAL JOURNAL OF COMPUTERS AND COMMUNICATIONS
DOI: 10.46300/91013.2021.15.9 Volume 15, 2021

Analytic Programming – a Novel Tool for Synthesis of Controller for Chaotic Lozi Map

Roman Senkerik, Zuzana Kominkova Oplatkova and Michal Pluhacek
Faculty of Applied Informatics
Tomas Bata University in Zlin
T.G. Masaryka 5555, 760 01 Zlin, Czech Republic
{senkerik , kominkovaoplatkova , pluhacek}@fai.utb.cz

Abstract— **In this paper, it is presented a utilization of a novel tool for symbolic regression, which is analytic programming, for the purpose of the synthesis of a new feedback control law. This new synthesized chaotic controller secures the fully stabilization of selected discrete chaotic systems, which is the two-dimensional Lozi map. The paper consists of the descriptions of analytic programming as well as selected chaotic system, used heuristic and cost function design. For experimentation, Self-Organizing Migrating Algorithm (SOMA) and Differential evolution (DE) were used. Two selected experiments are detailed described.**

Keywords—Analytic Programming; Symbolic regression; Chaos control; Evolutionary algorithms; Lozi map

I. INTRODUCTION

During the recent years, usage of new intelligent systems in engineering, technology, modeling, computing and simulations has attracted the attention of researchers worldwide. The most current methods are mostly based on soft computing, which is a discipline tightly bound to computers, representing a set of methods of special algorithms, belonging to the artificial intelligence paradigm. The most popular of these methods are neural networks, evolutionary algorithms, fuzzy logic and tools for symbolic regression like genetic programming. Currently, evolutionary algorithms are known as a powerful set of tools for almost any difficult and complex optimization problem.

The interest about the interconnection between evolutionary techniques and control of chaotic systems is spread daily. First steps were done in [1] representing the utilization of differential evolution algorithm for the synchronization and control of chaotic systems. The papers [2], [3] were concerned to tune several parameters inside the original control technique for discrete chaotic systems. The evolutionary tuned control technique was based on Pyragas method: Extended delay feedback control – ETDAS [4]. Another example of interconnection between deterministic chaos and evolutionary algorithms represents the research focused on the embedding of chaotic dynamics into the evolutionary algorithms [5] - [7].

This paper shows a possibility how to generate the whole control law by means of analytic programming (AP) (not only to optimize several parameters) for the purpose of stabilization of the selected discrete chaotic system. The synthesis of control is inspired by the Pyragas's delayed feedback control technique [8], [9].

AP is a superstructure of EAs and is used for synthesis of analytic solution according to the required behaviour. Control law from the proposed system can be viewed as a symbolic structure, which can be synthesized according to the requirements for the stabilization of the chaotic system.

Firstly, AP is explained, and then a problem design is proposed. The next sections are focused on the description of used soft-computing tools and the design of cost function. Results and conclusion follow afterwards.

II. MOTIVATION

This work is focused on the expansion of AP application for synthesis of a whole robust control law instead of parameters tuning for existing and commonly used control technique to stabilize desired Unstable Periodic Orbits (UPO) of selected discrete chaotic system.

This work represents an extension of previous research [10], [11], with the application to the chaotic discrete Lozi map

In general, this research is concerned to stabilize Lozi map chaotic system at p-1 UPO, which is a stable state, utilizing the synthesized control law.

III. LOZI MAP

Lozi map is the selected example of chaotic systems, which represents the simple discrete two-dimensional chaotic map. The *x*, *y* plot of the Lozi map is depicted in Fig. 1. The map equations are given in (1). The parameters are: a = 1.7 and b = 0.5 as suggested in [12], [13]. The chaotic behavior of the uncontrolled Lozi map is depicted in Fig. 2.

$$\begin{aligned} X_{n+1} &= 1 - a\left|X_n\right| + bY_n \\ Y_{n+1} &= X_n \end{aligned} \tag{1}$$

This work was supported by European Regional Development Fund under the project CEBIA-Tech No. CZ.1.05/2.1.00/03.0089 and by Internal Grant Agency of Tomas Bata University under the project No. IGA/FAI/2013/012.

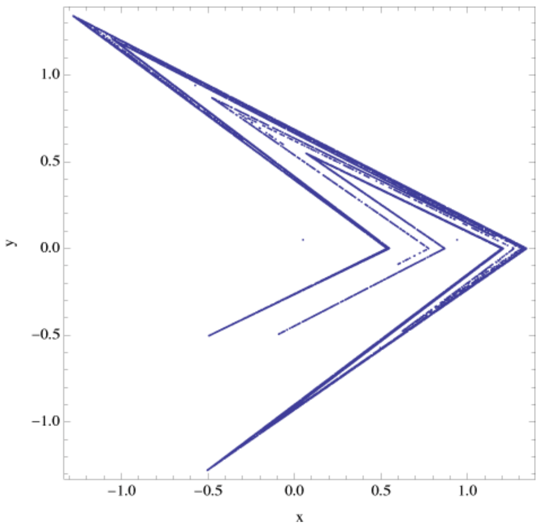

Fig. 1. *x*, *y* plot of the Lozi map

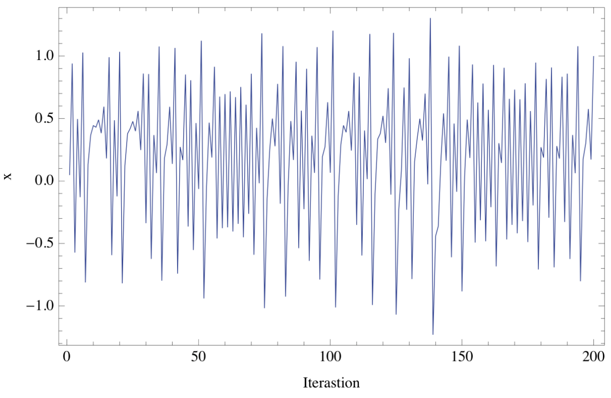

Fig. 2. Iterations of the uncontrolled Lozi map (variable *x*)

IV. Original Chaos Control Method

This work is focused on explanation of application of AP for synthesis of a whole control law instead of demanding tuning of any original method control law to stabilize desired Unstable Periodic Orbits (UPO). In this research desired UPO is only p-1 (the fixed point, which represents the stable state). Original Time-Delay-Auto-Synchronization (TDAS) delayed feedback control method was used in this research as an inspiration for synthesizing a new feedback control law by means of evolutionary techniques and for preparation of sets of basic functions and operators for AP.

The original control method – TDAS has form (2) and its discrete form is given in (3).

$$F(t) = K\left[x(t-\tau) - x(t)\right] \tag{2}$$

$$F_n = K\left(x_{n-m} - x_n\right) \tag{3}$$

Where: K is adjustable constant, F is the perturbation, τ_d is a time delay; and m is the period of m-periodic orbit to be stabilized. The perturbation F_n in equation (3) may have arbitrarily large value, which can cause diverging of the system. Therefore, F_n should have a value between $-F_{\max}$, $F_{\max}$. In this work a suitable $F_{\max}$ value was taken from the previous research.

V. Used Soft-Computing Tools

This section gives the brief overview and the description of used soft-computing tools. This research utilized the symbolic regression tool, which is analytic programming and two evolutionary algorithms: Self-Organizing Migrating Algorithm [14]; and Differential Evolution [15].

Future simulations expect a usage of soft computing GAHC algorithm (modification of HC12) [16] and a CUDA implementation of HC12 algorithm [17].

A. Analytic Programming

Basic principles of the AP were developed in 2001. Until that time only genetic programming (GP) and grammatical evolution (GE) had existed. GP uses genetic algorithms while AP can be used with any evolutionary algorithm, independently on individual representation. AP represents synthesis of analytical solution by means of evolutionary algorithms. Various applications of AP are described in [18] - [21].

The core of AP is based on a special set of mathematical objects and operations. The set of mathematical objects is set of functions, operators and so-called terminals (as well as in GP), which are usually constants or independent variables. This set of variables is usually mixed together and consists of functions with different number of arguments. Because of a variability of the content of this set, it is called here "general functional set" – GFS. The structure of GFS is created by subsets of functions according to the number of their arguments. For example GFS_{all} is a set of all functions, operators and terminals, GFS_{3arg} is a subset containing functions with only three arguments, GFS_{0arg} represents only terminals, etc. The subset structure presence in GFS is vitally important for AP. It is used to avoid synthesis of pathological programs, i.e. programs containing functions without arguments, etc. The content of GFS is dependent only on the user. Various functions and terminals can be mixed together [20].

The second part of the AP core is a sequence of mathematical operations, which are used for the program synthesis. These operations are used to transform an individual of a population into a suitable program. Mathematically stated, it is a mapping from an individual domain into a program domain. This mapping consists of two main parts. The first part is called discrete set handling (DSH) [21] and the second one stands for security procedures which do not allow synthesizing pathological programs. The method of DSH, when used, allows handling arbitrary objects including nonnumeric objects like linguistic terms {hot, cold, dark…}, logic terms (True, False) or other user defined functions. In the AP DSH is used to map an individual into GFS and together with security procedures creates the above mentioned mapping which transforms arbitrary individual into a program.

AP needs some evolutionary algorithm that consists of population of individuals for its run. Individuals in the population consist of integer parameters, i.e. an individual is an integer index pointing into GFS. The individual contains numbers which are indices into GFS. The detailed description is represented in [20].

AP exists in 3 versions – basic without constant estimation, AP_{nf} – estimation by means of nonlinear fitting package in Mathematica environment and AP_{meta} – constant estimation by means of another evolutionary algorithms; meta means meta-evolution.

B. *Self-Organizing Migrating Algorithm (SOMA)*

Self-Organizing Migrating Algorithm is a stochastic optimization algorithm that is modeled on the basis of social behavior of cooperating individuals [14]. It was chosen because it has been proven that the algorithm has the ability to converge towards the global optimum [14] and due to the successful applications together with AP [22], [23].

SOMA works on a population of candidate solutions in loops called *migration loops*. The population is initialized randomly distributed over the search space at the beginning of the search. In each loop, the population is evaluated and the solution with the highest fitness becomes the leader *L*. Apart from the leader, in one migration loop, all individuals will traverse the input space in the direction of the leader. Mutation, the random perturbation of individuals, is an important operation for evolutionary strategies (ES). It ensures the diversity amongst the individuals and it also provides the means to restore lost information in a population. Mutation is different in SOMA compared with other ES strategies. SOMA uses a parameter called PRT to achieve perturbation. This parameter has the same effect for SOMA as mutation has for genetic algorithms.

The novelty of this approach is that the PRT Vector is created before an individual starts its journey over the search space. The PRT Vector defines the final movement of an active individual in search space.

The randomly generated binary perturbation vector controls the allowed dimensions for an individual. If an element of the perturbation vector is set to zero, then the individual is not allowed to change its position in the corresponding dimension.

An individual will travel a certain distance (called the PathLength) towards the leader in *n* steps of defined length. If the PathLength is chosen to be greater than one, then the individual will overshoot the leader. This path is perturbed randomly.

C. *Differential evolution*

DE is a population-based optimization method that works on real-number-coded individuals [24] - [26]. DE is quite robust, fast, and effective, with global optimization ability. It does not require the objective function to be differentiable, and it works well even with noisy and time-dependent objective functions. Description of used DERand1Bin strategy is presented in (4). Please refer to [15], [26] for the description of all other strategies.

$$u_{i,G+1} = x_{r1,G} + F \cdot \left(x_{r2,G} - x_{r3,G}\right) \tag{4}$$

VI. Cost Function Design

The proposal of the basic cost function (CF) is in general based on the simplest CF, which could be used problem-free only for the stabilization of p-1 orbit. The idea was to minimize the area created by the difference between the required state and the real system output on the whole simulation interval – τ_i. This CF design is very convenient for the evolutionary searching process due to the relatively favorable CF surface. Nevertheless, this simple approach has one big disadvantage, which is the including of initial chaotic transient behavior of not stabilized system into the cost function value. As a result of this, the very tiny change of control method setting for extremely sensitive chaotic system causing very small change of CF value, can be suppressed by the above-mentioned including of initial chaotic transient behavior

But another universal cost function had to be used for stabilizing of extremely sensitive chaotic system and for having the possibility of adding penalization rules. It was synthesized from the simple CF and other terms were added.

This CF is in general based on searching for desired stabilized periodic orbit and thereafter calculation of the difference between desired and found actual periodic orbit on the short time interval - τ_s (40 iterations for higher order UPO) from the point, where the first minimal value of difference between desired and actual system output is found (i.e. floating window for minimization – see Fig. 3.).

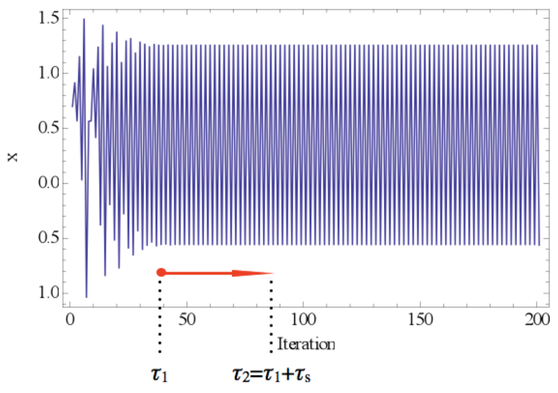

Fig. 3. "Floating window" for minimization

Such a design of universal CF should secure the successful stabilization of either p-1 orbit (stable state) or higher periodic orbits anywise phase shifted. Furthermore, due to CF values converging towards zero, this CF also allows the using of decision rules, avoiding very time demanding simulations. This rule stops EA immediately, when the first individual with good parameter structure is reached, thus the value of CF is lower then the acceptable (CF_{acc}) one. Based on the numerous experiments, typically CF_{acc} = 0.001 at time interval τ_s = 20 iterations, thus the difference between desired and actual output has the value of 0.0005 per iteration – i.e. successful stabilization for the used control technique. The CF_{Basic} has the form (5):

$$CF_{Basic} = pen_1 + \sum_{t=\tau 1}^{\tau 2} \left| TS_t - AS_t \right|, \tag{5}$$

where:

TS - target state, AS - actual state

τ_1 - the first min value of difference between TS and AS

τ_2 – the end of optimization interval (τ_1+ τ_s)

pen_1= 0 if τ_i - $\tau_2 \geq \tau_s$

pen_1= 10*(τ_i - τ_2) if τ_i - $\tau_2 < \tau_s$ (i.e. late stabilization).

VII. Results

Analytic Programming requires some EA for its run. In this paper, AP_{meta} version was used. Meta-evolutionary approach means usage of one main evolutionary algorithm for AP process and the second algorithm for coefficient estimation, thus to find optimal values of constants in the evolutionary synthesized control law.

SOMA algorithm was used for the main AP process and DE was used in the second evolutionary process. Settings of EA parameters for both processes given in Table 1 and Table 2 were based on performed numerous experiments with chaotic systems and simulations with AP_{meta}.

TABLE I. SOMA settings for AP

SOMA Parameter	Value
PathLength	3
Step	0.11
PRT	0.1
PopSize	50
Migrations	4
Max. CF Evaluations (CFE)	5345

TABLE II. DE settings for meta-evolution

DE Parameter	Value
PopSize	40
F	0.8
CR	0.8
Generations	150
Max. CF Evaluations (CFE)	6000

The data set for AP required only constants, operators like plus, minus, power and output values x_n and x_{n-1}. The set of elementary functions for AP was inspired in the original delayed feedback chaos control method TDAS (See section 4; (2) and (3)). Thus AP dataset consists only of simple functions (operators) with two arguments and functions with zero arguments, i.e. terminals (constants and system output values). Functions with one argument, e.g. Sin, Cos, etc.; were not required.

Basic set of elementary functions for AP:

GFS2arg= +, -, /, *, ^

GFS0arg= $data_{n-1}$ to $data_n$, K

Total number of cost function evaluations for AP was 5345, for the second EA it was 6000, together 32.07 millions per each simulation.

Following description of two selected experiments results contains illustrative examples of direct output from AP – synthesized control laws without coefficients estimated (6) and (8); further the notations with simplification after estimation by means of second algorithm DE (7) and (9), Table 3 with corresponding CF values and the average error value between actual and required system output, and finally Fig. 4 - 7 with simulation results.

TABLE III. Cost Function values and simple statistics

Experiment No.	CF Value	Avg. Error per iteration
Experiment 1	$6.2992 \cdot 10^{-15}$	$3.1496 \cdot 10^{-16}$
Experiment 2	$1.4567 \cdot 10^{-6}$	$7.2836 \cdot 10^{-8}$

A. Experiment 1

$$F_n = 2K_1 (-x_{n-1} - 2x_n)(x_{n-1} - x_n) x_n \tag{6}$$

$$F_n = 1.18253 (-x_{n-1} - 2x_n)(x_{n-1} - x_n) x_n \tag{7}$$

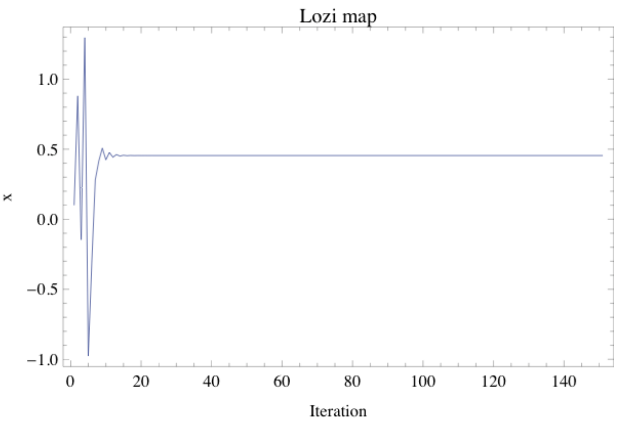

Fig. 4. Simulation results – Experiment 1, variable x of Lozi map

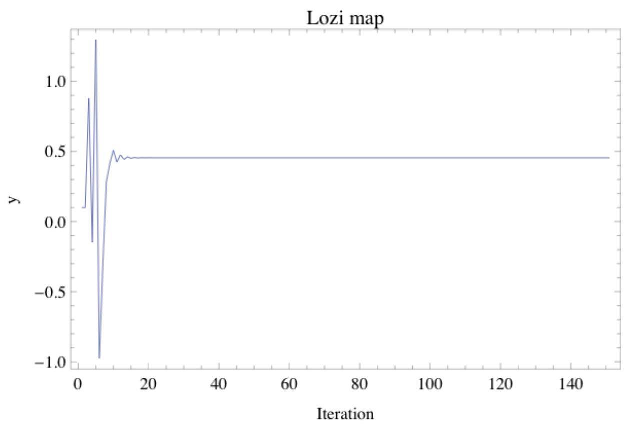

Fig. 5. Simulation results – Experiment 1, variable *y* of Lozi map

B. *Experiment 2*

$$F_n = K_1 x_{n-1} - 2(x_{n-1} + K_2)x_{n-1} + x_n x_{n-1} + x_{n-1} \quad (8)$$

$$F_n = 16.1492 x_{n-1} - 2(x_{n-1} + 7.8473)x_{n-1} + x_n x_{n-1} + x_{n-1} \quad (9)$$

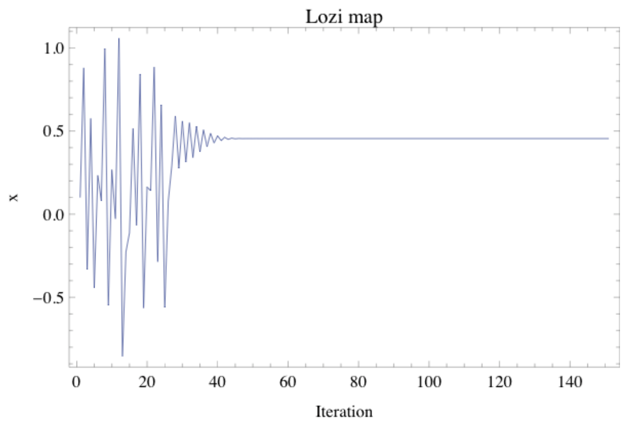

Fig. 6. Simulation results – Experiment 2, variable *x* of Lozi map

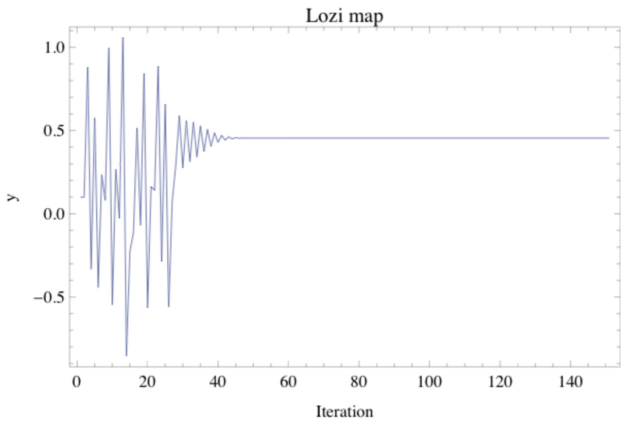

Fig. 7. Simulation results – Experiment 2, Variable *y* of Lozi map

VIII. Conclusions

This paper deals with a synthesis of a new universal robust control law by means of AP for stabilization of selected discrete chaotic system at fixed point. Two-dimensional Lozi map as the example of discrete chaotic systems were used in this research.

Obtained results reinforce the argument that AP is able to solve this kind of difficult problems and to produce a new robust synthesized control law in a symbolic way securing desired behaviour and precise stabilization of the selected chaotic systems.

Presented two simulation examples show two different results. Extremely precise stabilization and simple control law in the first case and not very precise and slow stabilization and relatively complex notation of chaotic controller in the second case. This fact lends weight to the argument, that AP is a powerful symbolic regression tool, which is able to strictly and precisely follow the rules given by cost function and synthesize any symbolic formula, in the case of this research – the feedback controller for chaotic system.

The future research will include the development of better cost functions, testing of different AP data sets, and performing of numerous simulations to obtain more results and produce better statistics, thus to confirm the robustness of this approach.

References

[1] B. Liu, L. Wang, Y.H. Jin, D.X. Huang and F. Tang, “Control and synchronization of chaotic systems by differential evolution algorithm”, Chaos, Solitons & Fractals, Volume 34, Issue 2, 2007, pp. 412-419, ISSN 0960-0779.

[2] I. Zelinka, R. Senkerik and E. Navratil, “Investigation on evolutionary optimization of chaos control”, Chaos, Solitons & Fractals, Volume 40, Issue 1, 2009, pp. 111-129.

[3] R. Senkerik, I. Zelinka, D. Davendra and Z. Oplatkova, “Utilization of SOMA and differential evolution for robust stabilization of chaotic Logistic equation”, Computers & Mathematics with Applications, Volume 60, Issue 4, 2010, pp. 1026-1037.

[4] K. Pyragas, “Control of chaos via extended delay feedback”, Physics Letters A, vol. 206, 1995, pp. 323-330.

[5] I. Aydin, M. Karakose and E. Akin, “Chaotic-based hybrid negative selection algorithm and its applications in fault and anomaly detection”, Expert Systems with Applications, Vol. 37, No. 7, 2010, pp. 5285–5294.

[6] D. Davendra, I. Zelinka and R. Senkerik, “Chaos driven evolutionary algorithms for the task of PID control”, Computers & Mathematics with Applications, Vol. 60, No. 4, 2010, 1088-1104, ISSN 0898-1221.

[7] M. Pluhacek, R. Senkerik, D. Davendra, Z. Kominkova Oplatkova and I. Zelinka, “On the behavior and performance of chaos driven PSO algorithm with inertia weight”, *Computers & Mathematics with Applications*, 2013, (article in press), ISSN 0898-1221, DOI 10.1016/j.camwa.2013.01.016.

[8] W. Just, “Principles of Time Delayed Feedback Control”, In: Schuster H.G., Handbook of Chaos Control, Wiley-Vch, 1999.

[9] K. Pyragas, “Continuous control of chaos by self-controlling feedback”, Physics Letters A, 170, 1992, pp. 421-428.

[10] R. Senkerik, Z. Oplatkova, I. Zelinka and D. Davendra, “Synthesis of feedback controller for three selected chaotic systems by means of evolutionary techniques: Analytic programming”, Mathematical and Computer Modelling, Vol. 57, No. 1 - 2, 2013, pp. 57 – 67, ISSN 0895-7177.

[11] Z. Kominkova Oplatkova, R. Senkerik, I. Zelinka and M. Pluhacek, “Analytic programming in the task of evolutionary synthesis of a controller for high order oscillations stabilization of discrete chaotic

systems", Computers & Mathematics with Applications, ISSN 0898-1221, 2013, (articele in press), DOI 10.1016/j.camwa.2013.02.008.

[12] R.C. Hilborn "Chaos and Nonlinear Dynamics: An Introduction for Scientists and Engineers", Oxford University Press, 2000, ISBN: 0-19-850723-2.

[13] J.C. Sprott, "Chaos and Time-Series Analysis", Oxford University Press, 2003.

[14] I. Zelinka, "SOMA – Self Organizing Migrating Algorithm", In: New Optimization Techniques in Engineering, (B.V. Babu, G. Onwubolu (eds)), Springer-Verlag, 2004, ISBN 3-540-20167X.

[15] K. Price, R.M. Storn and J. A. Lampinen, "Differential Evolution: A Practical Approach to Global Optimization", Springer, 2005.

[16] R: Matousek and E. Zampachova, "Promising GAHC and HC12 algorithms in global optimization tasks", Optimization Methods & Software, Vol. 26, No. 3, 2011, pp. 405-419. ISSN 1055-6788.

[17] R. Matousek, "HC12: The Principle of CUDA Implementation". In Proceedings of 16th International Conference On Soft Computing Mendel 2010, 2010, pp. 303-308. ISBN 978-80-214-4120- 0.

[18] I. Zelinka, Ch. Guanrong and S. Celikovsky, "Chaos Synthesis by Means of Evolutionary algorithms", International Journal of Bifurcation and Chaos, Vol. 18, No. 4, 2008, pp. 911–942

[19] Z. Oplatkova and I. Zelinka, "Investigation on Evolutionary Synthesis of Movement Commands", Modelling and Simulation in Engineering, Vol. 2009 (2009), Article ID 845080, 12 pages, Hindawi Publishing Corporation, ISSN: 1687-559.

[20] I. Zelinka, D. Davendra, R. Senkerik, R. Jasek and Z. Oplatkova, "Analytical Programming - a Novel Approach for Evolutionary Synthesis of Symbolic Structures", In Evolutionary Algorithms, Eisuke Kita (Ed.), InTech, 2011.

[21] B. Chramcov and P. Varacha, "Usage of the Evolutionary Designed Neural Network for Heat Demand Forecast". In: Proceedings of Nostradamus 2012: Modern Methods of Prediction, Modeling and Analysis of Nonlinear Systems, 2013, pp. 103-122. ISBN 978-3-642-33226-5.

[22] P. Varacha and R. Jasek, "ANN Synthesis for an Agglomeration Heating Power Consumption Approximation". In: Recent Researches in Automatic Control. Montreux : WSEAS Press, p. 239-244. ISBN 978-1-61804-004-6.

[23] P. Varacha and I. Zelinka, "Distributed Self-Organizing Migrating Algorithm Application and Evolutionary Scanning". In: Proceedings of the 22nd European Conference on Modelling and Simulation ECMS. 2008, p. 201-206. ISBN 0-9553018-5-8.

[24] J. Lampinen, I. Zelinka, "New Ideas in Optimization – Mechanical Engineering Design Optimization by Differential Evolution", Volume 1. London: McGraw-hill, 1999, 20 p., ISBN 007-709506-5

[25] K. Price, "An Introduction to Differential Evolution", In: (D. Corne, M. Dorigo and F. Glover, eds.) New Ideas in Optimization, London: McGraw-Hill, pp. 79–108, 1999.

[26] K. Price and R. Storn, "Differential evolution homepage", 2001, http://www.icsi.berkeley.edu/~storn/code.html, [Accessed 01/03/2013].