VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

MODERN TECHNIQUES OF SOFT COMPUTING AND THEIR SELECTED APPLICATIONS

MODERNÍ TECHNIKY SOFT COMPUTINGU A JEJICH VYBRANÉ APLIKACE

HABILITAČNÍ PRÁCE
HABILITATION THESIS

AUTOR PRÁCE
AUTHOR Ing. Zuzana KOMÍNKOVÁ OPLATKOVÁ, Ph.D.

BRNO 2012

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

HABILITAČNÍ PRÁCE

obor

Výpočetní technika a informatika

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

FACULTY OF INFORMATION TECHNOLOGY

téma

MODERN TECHNIQUES OF SOFT COMPUTING AND THEIR SELECTED APPLICATIONS

MODERNÍ TECHNIKY SOFT COMPUTINGU A JEJICH VYBRANÉ APLIKACE

autor

Ing. **Zuzana Komínková Oplatková**, Ph.D.

FAKULTA APLIKOVANÉ INFORMATIKY

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ

Brno 2012

Abstrakt

Předkládaná habilitační práce je zaměřena na problematiku z oblasti umělé inteligence, především z oblasti soft computingu. Tato oblast je v posledních letech velmi studována a zkoumána odborníky z různých oblastí. Nástroje soft computingu pomáhají ve všech oblastech lidského života k získání optimálních výsledků požadovaných zadání a řešené problematiky. Mezi tyto nástroje patří především neuronové sítě, evoluční algoritmy, fuzzy logika a nadstavba evolučních algoritmů tzv. symbolická regrese. Všechny uvedené prostředky umožňují řešit úlohy v oblastech jako je řízení procesů, diagnostika, image processing, operační výzkum, sledování finančních trhů a predikce vývoje burzovních kurzů a mnohé další.

Evoluční algoritmy jako genetické algoritmy, diferenciální evoluce, optimalizace rojení částic, samoorganizující se migrační algoritmus a další se v dnešní době používají k řešení velmi složitých a komplexních optimalizačních úloh. Rovněž se používají k hledání optimálních struktur tzv. symbolické regrese, kde k řešení používáme genetické programování, gramatickou evoluci či novější analytické programování. Nástroje soft computingu se neustále rozvíjejí a vznikají nové. Symbolickou regresí lze dokonce vytvářet i nové evoluční algoritmy nebo struktury neuronových sítí.

Neuronové sítě jsou oblastí, která se od 80. let minulého století vyvíjí velkou rychlostí pro své paralelní výpočetní, rozpoznávací a klasifikační možnosti. Lze je s velkým úspěchem využít i v oblasti tzv. dobývání znalostí a data miningu, tedy dolování dat a zjednodušování velmi složitých úloh.

Nosným tématem habilitační práce je metaevoluce, která znamená několik možností. V kontextu této práce je nejčastěji použita v případě použití dvou evolučních procesů použitých v symbolické regresi, jeden pro řízení symbolické regrese a druhý ve jejím vnitřním procesu. Vedle metaevoluce habilitační práce spojuje principy a aplikace evolučních algoritmů, symbolické regrese a neuronových sítí. Jednak jsou představeny aplikace využívající jednotlivé techniky a metody samostatně a jednak propojené aplikace – tedy sloučení více soft computingových technik, např. vytváření evolučního algoritmu pomocí symbolické regrese nebo vytvoření pseudo neuronové sítě pomocí symbolické regrese. Další představenou kombinovanou metodou využívající propojení schopností uvedených metod je metaevoluce pro

syntézu techniky řízení chaotických systémů. Zde jsou použity dva evoluční algoritmy, jeden pro řízení symbolické regrese a druhý ve vnitřním procesu symbolické regrese.
Hlavním cílem habilitační práce je ukázat vlastnosti a možnosti použití moderních soft computingových nástrojů jednak samostatně, ale také při jejich vzájemném propojení. Mezi vybranými aplikacemi jsou popsány detailněji syntéza optimální struktury řídící techniky pro systémy vykazující deterministický chaos a stegoanalýza pomocí neuronových sítí. Dále je v práci uvedena aplikace vytváření pseudo neuronové sítě pomocí symbolické regrese či okrajově evolučního algoritmu, na kterém je vysvětlený jiný metaevoluční přístup. Mimo výše popsané aplikace se také práce zaměřuje na problematiku úprav, vývoje a testování soft computingových metod.

Abstract

The presented habilitation thesis is focused on the issuess of the field of artificial intelligence, especially in the area of soft computing. This area has been studied and examined by experts in various fields in recent years. Soft computing tools help to obtain optimal results of required assignments and solved problems in all areas of human life. These tools are mainly neural networks, evolutionary algorithms, fuzzy logic and the superstructure of evolutionary algorithms called symbolic regression. All of these techniques enable us to solve problems in areas such as process control, diagnostics, image processing, operation research, monitoring of financial markets and the prediction of exchange rates and much more.
Evolutionary algorithms such as genetic algorithms (GA), differential evolution (DE), particle swarm optimization (PSO), self-organizing migrating algorithm (SOMA) and others are used to solve very complicated and complex optimization problems nowadays. They are also used to search for optimal structures called symbolic regression, where methods such as genetic programming, grammatical evolution or newer analytical programming are used. Soft computing tools are constantly developed and new ones are created. Symbolic regression can even create new evolutionary algorithms or neural network structures.

Neural networks is the area that has been developing for its high-speed parallel computing, recognition and classification capabilities since 1980s. They can be used with great success in the field of knowledge discovery and data mining, i.e. to simplify very complex tasks.

The principal theme of this habilitation thesis is metaevolution which stands for several possibilities. In the context of this thesis, the most often cases are two evolutionary processes used in symbolic regression, one for the control of symbolic regression and the other in the inner symbolic regression process. Beside metaevolution, the thesis combines the principles and applications of evolutionary algorithms, symbolic regression and artificial neural networks. Presented applications using various techniques and methods are in the thesis presented both separately and as connected applications, a merger of multiple soft computing techniques such as the synthesis of an evolutionary algorithm by means of symbolic regression or creating a pseudo neural network using symbolic regression. The other combined technique utilizing interconnected capabilities of mentioned methods is a metaevolution for the synthesis of chaotic system control technique.

The main aim of the habilitation thesis is to show the properties and possible applications of modern soft computing tools both separately and also in their mutual connection. Among the selected applications, the synthesis of the optimal structure of the control technique for systems exhibiting deterministic chaos and steganalysis by means of artificial neural networks are described in detail. Furthermore, the thesis mentions creating pseudo neural networks by means of symbolic regression and marginally the synthesis of an evolutionary algorithm for the explanation of another metaevolutionary approach. Apart from the above described applications, the thesis is also focused on adjustments, development and testing of soft computing methods.

Klíčová slova

Soft computing, metaevoluce, evoluční algoritmy, umělé neuronové sítě, optimalizace.

Keywords

Soft computing, metaevolution, evolutionary algorithms, artificial neural networks, optimization.

Citace

Komínková Oplatková Zuzana: Modern techniques of soft computing and their selected applications, habilitační práce, Brno, FIT VUT v Brně, 2012

Acknowledgements

I would like to express my warm thanks to:

- ❖ my *husband* Aleš for his love and support,
- ❖ *my colleagues and friends*,
- ❖ my *parents*.

Content

1 Introduction

The presented habilitation thesis is focused on the issues of the field of artificial intelligence, especially in the area of soft computing. This area has been studied and examined by experts in various fields in recent years. Soft computing tools help to obtain optimal results of required assignments and solved problems in all areas of human life. These tools are mainly neural networks, evolutionary algorithms, fuzzy logic and the superstructure of evolutionary algorithms called symbolic regression [111], [105], [45]. All of these techniques enable us to solve problems in areas such as process control, diagnostics, image processing, operation research, monitoring of financial markets and the prediction of exchange rates and much more.

Evolutionary algorithms are a group of algorithms that use their special operators as mutation, crossover and other to find an ideal solution. Possible candidates are defined by a cost function whose arguments are values of each solution. The best one is in the global extreme – maximum or minimum [105], [45].

These evolutionary algorithms have been known for decades and have lived through the advancement from weaker ones to more robust ones, which are used with success in a lot of tasks nowadays. Since their first appearance there is quite a long queue of representatives: genetic algorithms (GA) [45], [2], [12], differential evolution (DE) [2], [47], [68], self-organizing migrating algorithm (SOMA) [110], particle swarm optimization (PSO) [17], ant colony optimization [16], artificial immune system [21], and more are used to solve very complicated and complex optimization problems nowadays.

They are also used to search for optimal structures called symbolic regression, where methods such as genetic programming (GP) [44], [43], grammatical evolution (GE) [53], [52] or newer analytical programming (AP) [102], [108], [109], [104], [57], [107], [103] are used. Also, some other approaches to the field of symbolic regression can be found – either based only on evolutionary techniques or hybrid ones. Interesting investigations using symbolic regression were shown by Johnson [37] working on Artificial Immune Systems and Salustowicz in Probabilistic Incremental Program Evolution (PIPE) [78] which generates programs from an adaptive probability distribution over all possible programs. GADS is a forerunner of grammatical evolution which solves the approach to grammar [65], [64]. Also from evolutionary algorithms artificial immune systems evolved the artificial immune system programming for symbolic regression [37]. Approaches that differ in representation and grammar are described in gene expression programming [24], multiexpression programming [54], meta-modelling by symbolic regression and pareto simulated annealing [92]. To the group of hybrid approaches belong mainly numerical methods connected with evolutionary systems, e.g. [11]. One of novel techniques is the transplant evolution that is closely associated with the conceptual paradigm of AP and modified for GE. GE was also extended to include DE [99]. These techniques can produce a complex formula from basic functions according to required behaviour of a function in the case of a mathematical data set, of an electronic circuit, trajectory of robots, etc. Soft computing tools are constantly developed and new ones are created. Symbolic regression can even create new evolutionary algorithms [57] or neural network structures [107].

Neural networks is the area that has been developing for its high-speed parallel computing, recognition and classification capabilities since 1980s. They can be used with great success in the field of knowledge discovery and data mining, i.e. to simplify very complex tasks [31], [98], [30], [22].

All above-mentioned techniques and their principles are capable to be combined and to work in an optimal way for applications and solved tasks. The motivation for this habilitation thesis is to present examples using various techniques and methods, both separately and as connected applications that merge multiple soft computing techniques. The author of this thesis is motivated to study theoretical background of the soft computing methods and tries to adjust these useful techniques.

The principal theme of this habilitation thesis is metaevolution which stands for several possibilities – the tuning of algorithm parameters, tuning of algorithm operators, synthesis of another evolutionary algorithm. Therefore one of the presented tasks is the synthesis of a new optimization algorithm, evolutionary in principle. The used principle is metaevolution which means that a new algorithm of an evolutionary character is synthesized with another evolutionary algorithm and symbolic regression. Metaevolution does not mean only the described procedure; it also means searching for optimal values of parameters or settings of particular operators [57].

In the context of this thesis, another tack of metaevolution is used. It implements two evolutionary algorithms within symbolic regression, one for the control of symbolic regression and the other in an inner symbolic regression process. The described method is demonstrated on an example with a synthesis of chaotic system control technique.

The author is also interested in the area of deterministic chaotic systems and chaos theory which belongs to soft computing as well. This thesis describes not only the area of metaevolution but also the interconnection of soft computing techniques. From the area of developing and/or tuning of EA, one of the presented applications is aimed at using of deterministic chaos instead of a classical random generator in inner processes of evolutionary algorithms.

The last but not least interesting area for the author is artificial neural networks. This thesis therefore contains two practical applications of neural networks – steganalysis and optimal modelling of dynamic flight. Both were an inspiration for creating a pseudo neural network by means of metaevolutionary approach with symbolic regression.

The main aim of the habilitation thesis is to show the properties and possible applications of modern soft computing tools both separately and also in their mutual connection.

The structure of this thesis is following:

PART 1 – Theoretical background

Symbolic regression

- Genetic Programming
- Grammatical Evolution
- Analytic Programming

Evolutionary algorithms

- Self-organizing Migrating Algorithm (SOMA)
- Differential Evolution
- Particle Swarm Optimization (PSO)

Artificial Neural Networks (ANN)

Metaevolution

PART 2 – Selected applications

Selected applications – introduction

From the field of symbolic regression

- Metaevolution in the design of evolutionary algorithms
- Metaevolution for the synthesis of control law for deterministic chaotic systems
- Synthesis of Pseudo ANN – interdisciplinary connection between symbolic regression and ANN

From the field of ANN

- Steganalysis by means of ANN
- Optimal modelling of dynamic flight

From the field of evolutionary techniques

- Chaotic number generator in PSO – interdisciplinary connection between evolutionary techniques and deterministic chaotic systems

The thesis starts with the area of symbolic regression and three particular methods – GP was developed as the first technique of its kind. GE is another well-known method in this area. The last described technique – analytic programming – has been developed, tested and used for different tasks at the Institute of Information Technologies which was transformed to the Faculty of Applied Informatics at Tomas Bata University in Zlin in 2006.

Since symbolic regression comes from the principle of evolutionary algorithms, next part is focused on the describition of evolutionary algorithms used in the thesis. All three algorithms – differential evolution, self-organizing migrating algorithm and particle swarm optimization were proved in a lot of benchmark tests as suitable, relatively fast and capable of solving complex optimization tasks.

The following part deals with neural networks that are used in several presented applications, either in practical ones or in their synthesis by means of symbolic regression.

The last part of the theoretical background describes metaevolutionary approaches after which their application follows.

The first application was the first research study of the author in the area of metaevolution. Since then there have been some issues connected with one approach to metaevolution, the application of evolutionary algorithm synthesis is provided in the first place. After this research, another kind of metaevolution has been studied for a long period which is described in the second application – Metaevolution for the synthesis of control law for deterministic chaotic systems. This approach together with two practical applications of artificial neural networks inspired the author to work on the synthesis of Pseudo ANN by means of symbolic regression. The last application is dedicated to the interconnection between evolutionary computation and chaotic systems studied during the synthesis of control laws which led to better performance of evolutionary algorithms, here demonstrated on particle swarm optimization. The chaotic system is used in PSO as a pseudorandom number generator.

All applications discussed in this thesis are only a part of the author's research portfolio and it is supposed that the pontential of presented ideas will be developed further in the future.

PART 1

–

Theoretical Background

2 Symbolic Regression

In statistics, regression is a method of curve fitting, i.e. finding a curve that matches a series of data points and possibly also other constraints. It is done by means of a regression analysis. Two types of regression are used – linear and nonlinear, which depend on data sets. The final formula, which fits data as close as possible, is done using classic mathematical and statistical techniques [3], [10].

Symbolic regression in the context of this thesis means a synthesis of a final formula from basic simple functions (e.g. Fig. 2.1, Fig. 2.2). This procedure can be used for mathematical and also for non-mathematical fields.

This approach was firstly introduced by John Koza in genetic programming [44], [43], then in grammatical evolution [53], [52] by Conor Ryan and the technique used in the simulations performed for the publication purposes was developed by Ivan Zelinka in analytic programming [102], [108], [109], [104], [57], [107], [103].

2.1 Genetic Programming

Genetic programming was introduced at the end of the 1980's by John Koza [44], [43]. He suggested a modification of genetic algorithm and he named it genetic programming. In this concept a new population is bred not in a normal numerical way but in an analytical way. It means that the solution of such breeding is not values of parameters but a function itself.

According to genetic algorithms each value is similarly to nature called gene. Genes in GP are not represented by integers or real values, parameters in a chromosome string are functions themselves. In the simplest version there are variables, constants, basic arithmetical functions and elementary functions. From this group, a function, e.g. x*(1+x) can be made. This can be seen in a parse tree (Fig. 2.1), where the top is called the root of the tree.

Interpreting the parse tree is easy. During the run, the function x * (1 + x) is evaluated through this tree from the bottom to the top.

In GP, the operators of crossover and mutation are used as they are used in genetic algorithms [45], [2], [12]. But here the individual contains basic operators, not numerical values. Therefore whole parts of the tree are changed in the case of mutation (Fig. 2.2) or crossover (Fig. 2.3).

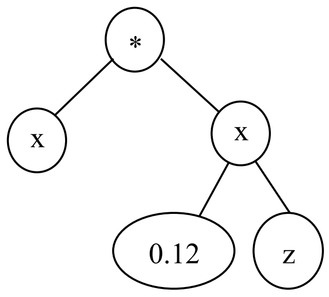


Fig. 2.1: A parse tree

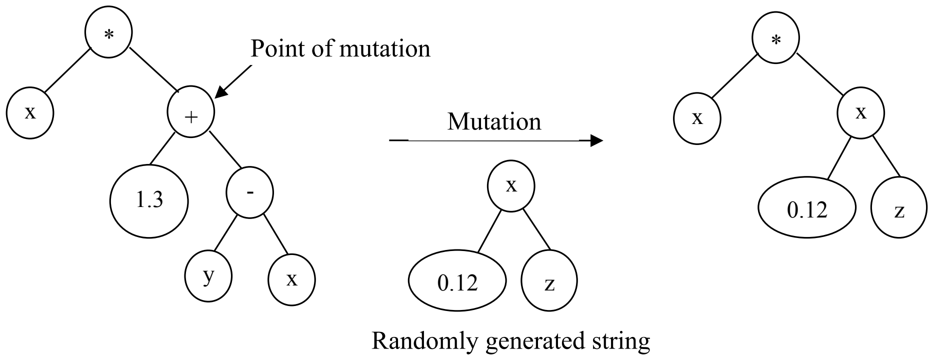


Fig. 2.2: Mutation in Genetic Programming

Another approach to GP is enforcing dimensional constraints through formal grammar. It restricts GP search space to dimensionally admissible laws [44].

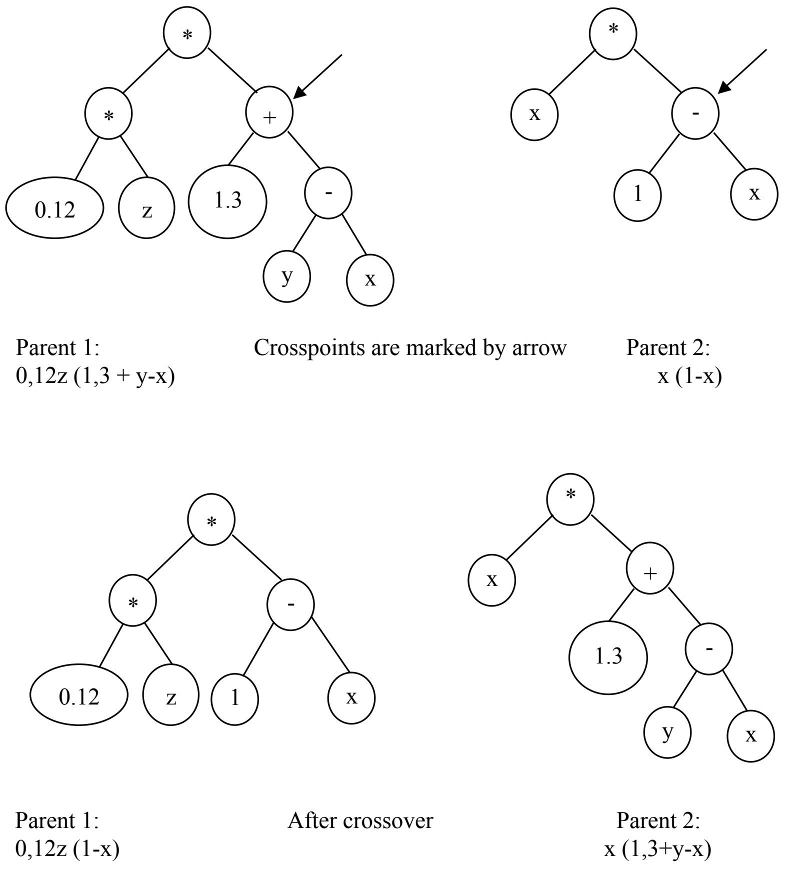


Fig. 2.3: Crossover in Genetic Programming

2.2 Grammatical Evolution

Grammatical evolution (GE) is another tool for doing symbolic regression by means of computers. The advantage of this tool, compared to GP, is that GE can evolve complete programs in an arbitrary programming language [53], [52] using a variable – length binary string. It uses Backus Naur Form grammar definition for mapping process to a program. GE performs the whole process on a variable – length binary strings. A mapping process is employed to generate programs in any language by using the binary strings to select production rules in the Backus Naur Form (BNF) grammar definition. The result is the construction of a syntactically correct program from a binary string that can then be evaluated by a fitness function.

Variable-length binary string genomes are used with each codon representing an integer value where codons are consecutive groups of 8 bits in order to make the genetic code degenerate. The integer values are used in a mapping function to select an appropriate production rule from the BNF definition. The numbers generated always represent one of the rules that can be used at that time.

Below is an example of a BNF definition, where N is a set of nonterminals and T is a set of terminals.

N ={expr, op, pre_op, var}

T = {Sin, + , - , / , * , X , 1.0}}

and can be represented as:

A)	<expr> : :	=	<expr> <op> <expr>	(0)
		|	(<expr> <op> <expr>)	(1)
		|	<pre-op> (<expr>)	(2)
		|	<var>	(3)

B)	<op> : :	=	+	(0)
		|	-	(1)
		|	/	(2)
		|	*	(3)

C)	<pre-op> : :	=	Sin

D)	<var> : :	=	X	(0)
		|	1.0	(1)

In Table 2.1, the numbers of possibilities for each rule are given. The mapping starts with reading codons of 8 bits [53] to generate a corresponding integer value from which an appropriate production rule is selected by using the mapping function (2.1).

Table 2.1: The number of choices available from each production rule

RULE TYPE	**CHOICES**
A	4
B	4
C	1
D	2

Rule = (Codon integer value)
MOD
(Number of choices for the current non-terminal) (2.1)

Fig. 2.4 shows an example of an individual with content of integer values generated from 8 bit binary number (codon).

220	40	16	203	101	53	202	203	102	55	220	202	19

.....

130	37	202	203	32	39	202	203	102

Fig. 2.4: An example of an individual for GE

The first codon is 220. If we apply eq. (2.1) we obtain value 0. That means we use rule A with its terminal 0. It represents an inscription A.0. Our program looks like

<expr><op><expr>

Then we continue with the left-most non-terminal which is <expr>. We take the second codon from the individual and apply the formula (2.1) again, i.e. 40 MOD 4. We obtain 0. <expr> is replaced by <expr><op><expr>. The result is following

<expr><op><expr><op><expr>

The next step starts again with <expr>. For the third time by reading codon we obtain the rule A.0.

<expr><op><expr><op><expr><op><expr>

Now the left-most <expr> is determined by codon with value 203 which gives the rule A.3 after applying the formula (2.1), thus non-terminal <var>.

<var><op><expr><op><expr><op><expr>

The next codon will then determine the value of var; there are 2 possibilities. 101 MOD 2 gives then rule D.1 which has the value 1.0. Our expression then results in

1.0 <op><expr><op><expr><op><expr>

Next codon will then determine what <op> will become. We have 53 MOD 4 which is equal to 1. The first terminal in <op> means the operator minus. The next <expr> has to be expanded by the codon 202 that is 202 MOD 4 = 2. We get following

1.0 - <pre-op>(<expr>)<op><expr><op><expr>

Because <pre-op> has only one possibility, we obtain

1.0 – Sin (<expr>)<op><expr><op><expr>

Then we can continue similarly as before until we end with this final formula.

1.0 – Sin(x)*Sin(x) - Sin(x)*Sin(x)

The program is finished when all non-terminals are replaced with terminals. If codons run out earlier, then they are used cyclically from the beginning. The above description is for mapping from codons to final formula in GE. During an evolutionary process, mutation and crossover operators are used as in genetic algorithms.

2.3 Analytic Programming

Basic principles of the AP were developed in 2001 [102], [108]. Until that time only genetic programming (GP) and grammatical evolution (GE) had existed. GP uses genetic algorithms (GA) while AP can be used with any EA, independently on an individual representation. To avoid any confusion, based on the nomenclature according to the used algorithm, the name - Analytic Programming was chosen, since AP represents the synthesis of analytical solution by means of EA. Various applications of AP are described in [102], [108], [109], [104], [57], [107], [103].

The core of AP is based on a special set of mathematical objects and operations. The set of mathematical objects is a 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 the variability of the content of this set, it is termed the "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. The hierarchy of GFS is depicted in Fig. 2.5. It is used to avoid the 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 [102].

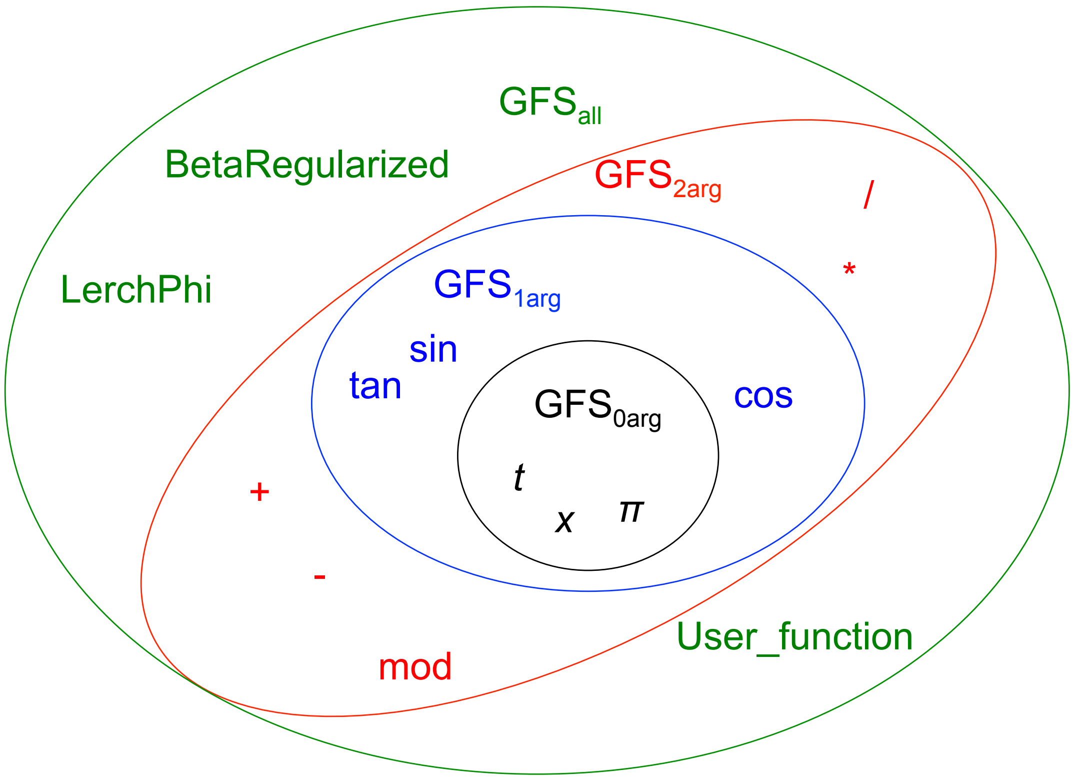


Fig. 2.5: Hierarchy in the GFS

The second part of the AP core is a sequence of mathematical operations used for the program synthesis. These operations are used to transform an individual of a population into a suitable program. Mathematically stated, it is mapping from an individual domain into a program domain. The mapping consists of two main parts. The first part is called Discrete Set Handling (DSH) (Fig. 2.6) [102] 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 such as 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 an arbitrary individual into a program.

AP needs some EA [102] that consists of a 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 creation of the program can be schematically observed in Fig. 2.7. The individual contains numbers which are indices for GFS.

Fig. 2.7 demonstrates an artificial example as to how a final function is created from an integer individual via Discrete Set Handling (DSH).

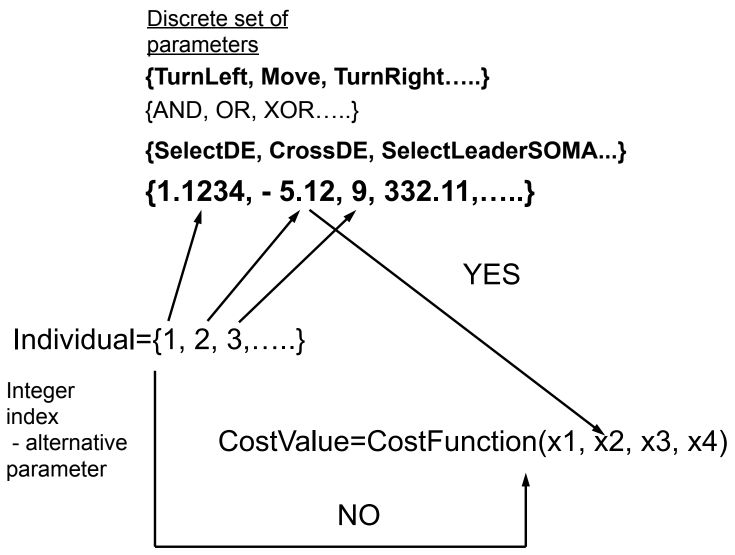

Discrete set of parameters
{TurnLeft, Move, TurnRight.....}
{AND, OR, XOR.....}
{SelectDE, CrossDE, SelectLeaderSOMA...}
{1.1234, - 5.12, 9, 332.11,.....}
YES
Individual={1, 2, 3,.....}
Integer index - alternative parameter
CostValue=CostFunction(x1, x2, x3, x4)
NO

Fig. 2.6: Discrete set handling

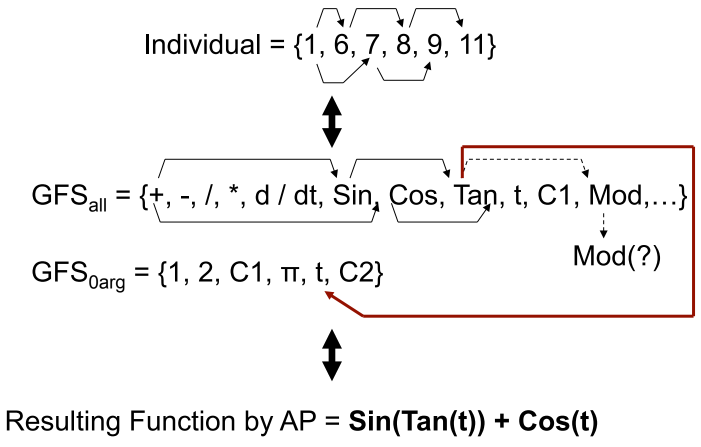

Individual = {1, 6, 7, 8, 9, 11}
GFS_all = {+, -, /, *, d / dt, Sin, Cos, Tan, t, C1, Mod,...}
Mod(?)
GFS_0arg = {1, 2, C1, π, t, C2}
Resulting Function by AP = Sin(Tan(t)) + Cos(t)

Fig. 2.7: The main principle of AP

The number 1 in the position of the first parameter means that the operator plus (+) from GFS_{all} is used (the end of the individual is far enough). Because the operator

+ must have at least two arguments, the next two index pointers 6 (sin from GFS) and 7 (cos from GFS) are dedicated to this operator as its arguments. The two functions, sin and cos, are one-argument functions, therefore the next unused pointers 8 (tan from GFS) and 9 (*t* from GFS) are dedicated to the sin and cos functions. As an argument of cos, the variable *t* is used, and this part of the resulting function is closed (*t* has zero arguments) in its AP development. The one-argument function tan remains, and there is one unused pointer 11, which stands for Mod in GFS_{all}. The modulo operator needs two arguments but the individual in the example has no other indices (pointers, arguments). In this case, it is necessary to employ security procedures and jump to the subset with GFS_{0arg}. The function tan is mapped on *t* from GFS_{0arg} which is on the 11th position, cyclically from the beginning.

2.3.1 AP Versions

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

AP_{basic} stands for the version where constant estimation is done in the same way as in genetic programming. If for example data approximation needs to estimate coefficients in the approximated polynomial or move the basic curve from the axes origin. In the AP_{basic} the user has to assign a set of constant values into GFS. It means a huge enlargement of the functional sets and deceleration of the evolutionary procedure. Therefore two other strategies were adopted - AP_{nf} and AP_{meta}.

These two versions of AP use the constant K which is indexed during the evolution (2.2). When K is needed, a proper index is assigned – K_1, K_2, ... K_n (2.3). Numeric values to indexed Ks are estimated (2.4) either via nonlinear fitting methods in the Mathematica environment (www.wolfram.com) - AP_{nf} or via the second evolutionary algorithm – AP_{meta}.

$$\frac{x^2+K}{\pi^K} \tag{2.2}$$

$$\frac{x^2 + K_1}{\pi^{K_2}} \tag{2.3}$$

$$\frac{x^2 + 3.156}{\pi^{90.78}} \tag{2.4}$$

AP$_{meta}$ is a time consuming process and the number of cost function evaluations, which is one of comparative factors, is usually very high. This fact is given by two evolutionary procedures (Fig. 2.8).

$$EA_{master} \Rightarrow program \Rightarrow K_{indexing} \Rightarrow EA_{slave} \Rightarrow K_{estimation} \Rightarrow final \cdot solution$$

Fig. 2.8: Schema of AP procedures

EA$_{master}$ is the main evolutionary algorithm for AP, EA$_{slave}$ is the second evolutionary algorithm inside AP. Thus the number of cost function evaluation (CFE) is given by (2.5).

$$CFE = EA_{master} * EA_{slave} \tag{2.5}$$

Despite this fact, some simulations cannot be done with nonlinear fitting methods in the Mathematica environment. The presented applications use the AP$_{meta}$ in most cases.

2.3.2 AP with Reinforced Evolution

Analytic programming is capable of reinforced evolution. During evolution, more or less appropriate individuals are synthesized. Some of these individuals are used to reinforce the evolution towards a better solution synthesis. The main idea of reinforcement is based on the addition of the just-synthesized and partly successful program into an initial set of terminals (GFS$_{0arg}$). Reinforcement is based on a user-defined deciding criterion. This criterion adds an individual into the initial set of terminals according to the value defined in the threshold. The threshold value is dependent on a cost value and, according to previous testing, the threshold is set up to a suitable initial value. Such a value means that the individual stands for a partial successful solution which should help increase the speed to find the final best solution. To avoid having a lot of partial solutions in GFS, only one individual is accepted for adding into GFS and the criterion value is decreased with each such step.

For example, if the threshold is set to 5, and the fitness of all individuals (programs in the population) is higher than 5, then the evolution is running on the initially defined GFS. When the cost value of the best individual in the current population is less than 5, then it is entirely added into the initial GFS and is marked as terminal. From this moment, the evolution is running on the enriched GFS containing the partially successful program. Due to this advantage, the evolution process is able to synthesize final solutions much faster than the AP without reinforcement. Simulations on different problems have repeatedly verified this fact.

It is quite similar to Automatically Defined Functions (ADF) for GP; however, the set of functions and terminals in GP can contain more than one ADF (which increases the complexity of the search space to the order of n!, at least theoretically). GP has to have checking procedures for critical situations (self calling...) and if arguments of this ADF are defined properly. This is not a problem of AP reinforcement, the added item belongs to terminals, i.e., no function, no arguments, no self calling, etc., and the cardinality of the initial GFS set increases only by one.

2.3.3 Similarities and Differences

Because analytic programming was partly inspired by genetic programming, some differences as well as similarities between AP, GP and grammatical evolution exist. Some of these are [102]:

- Synthesized programs (similarity): AP, as well as GP and GE, is able to do symbolic regression in a general point of view. It means that the output of AP is according to simulations, similar to programs from GP and GE.
- Functional set (similarity): AP_{basic} operates in principle on the same set of terminals and functions as GP or GE, while AP_{meta} or AP_{nf} use a universal constant K (difference), which is indexed after a program synthesis.

- Individual coding (difference): coding of an individual is different. AP uses an integer index instead of direct representation as is in canonical GP. GE uses the binary representation of an individual, which is consequently converted into integers for mapping into programs by means of BNF.
- Individual mapping (difference): AP uses DSH while GP in its canonical form uses direct representation in LISP and GE uses BNF.
- Constant handling (difference): GP uses a randomly generated subset of numbers - constants, GE utilizes user determined constants and AP uses only one constant K for AP_{meta} and AP_{nf}, which is estimated by another EA or by nonlinear fitting.
- Security procedures (difference): to guarantee the synthesis of non-pathological functions, procedures are used in AP that redirect the flow of mapping into subsets of a whole set of functions and terminals according to the distance from the end of the individual. If pathological function is synthesized in GP, synthesis is repeated. In the case of GE, when the end of an individual is reached, the mapping continues from the individual's beginning, which is not the case in AP. It is designed so that a non-pathological program is synthesized before the end of the individual is reached (no later than the end is reached).

3 Evolutionary Algorithms

Evolutionary algorithms are a group of algorithms suitable for optimization that use their special operators as mutation, crossover and others to find an ideal solution. Possible candidates are defined by a cost function whose arguments are values of each solution. The best one is in the global extreme – maximum or minimum [111], [105], [45], [2].

Different fields of human activities need to optimize countless cases of difficult tasks every day. Everybody wants to maximize profit and minimize cost. This means that optimizing is in every task of industry, transportation, medicine, everywhere. For these purposes, we need to have suitable tools that are able to solve very difficult and complicated problems. As previous years proved, the use of artificial intelligence and soft computing contribute to improvements in many activities.

This group covers a lot of different older and newer techniques that can be used independently or with symbolic regression. Two of several possible divisions (Fig. 3.1, Fig. 3.2) of evolutionary techniques might be as follows [111], [105]. For the purpose of this habilitation thesis, only three evolutionary algorithms, which were used for performed simulations, are described below.

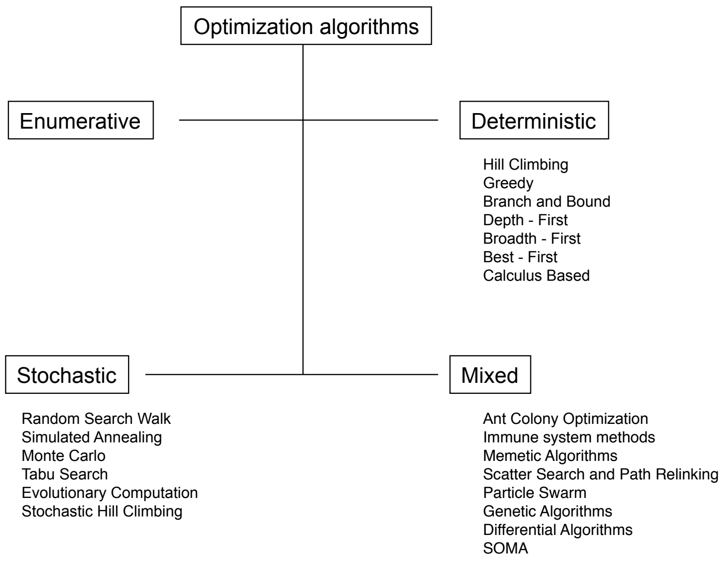


Fig. 3.1: The division of evolutionary algorithms – taken from [105]

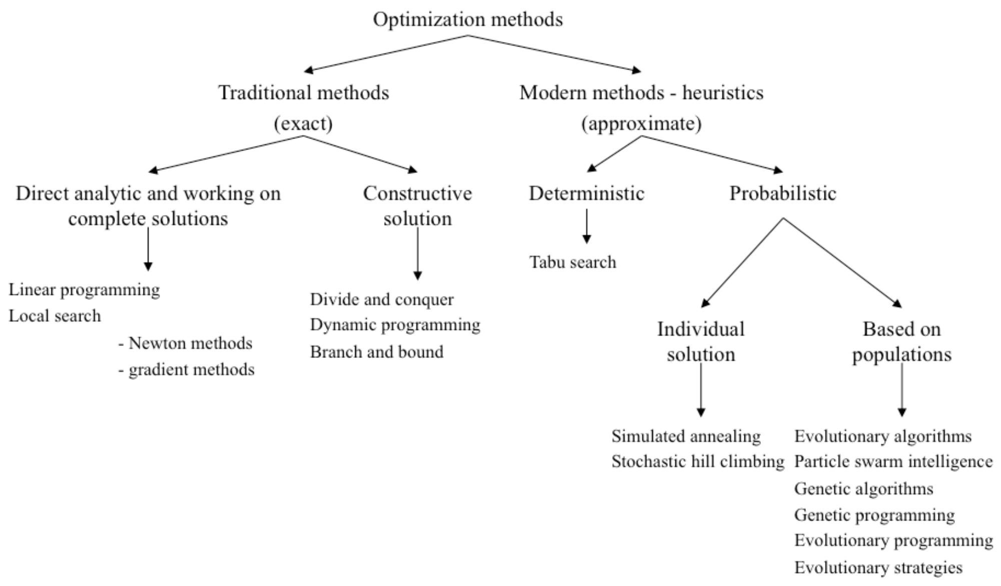


Fig. 3.2: Another possibile of division of evolutionary algorithms – taken from [105]

3.1 Self-Organizing Migrating Algorithm (SOMA)

SOMA works with groups of individuals (population) whose behavior can be described as a competitive–cooperative strategy [110]. The construction of a new population of individuals is not based on evolutionary principles (two parents produce an offspring) but on the behavior of a social group, e.g. a herd of animals looking for food. This algorithm can be classified as an algorithm of a social environment. To the same group of algorithms, sometimes called swarm intelligence, the Particle Swarm Optimization (PSO) algorithm can also belong. In the case of SOMA, there is no velocity vector as in PSO, only the position of individuals in the search space is changed during one generation, here called migration loop.

The rules are as follows: in every migration loop the best individual is chosen, i.e. individual with the minimum cost value, it is called the Leader. An active individual from the population moves in the direction towards the Leader in the search space. The movement consists of jumps determined by the Step parameter until the individual reaches the final position given by the PathLength parameter. For each step, the cost function for the actual position is evaluated and the best value is saved. At the end of the crossover, the position of the individual with the minimum cost value is chosen. If the cost value of the new position is better than the cost value of an individual from the old population, the new one appears in the new population. Otherwise the old one remains there. The main principle is depicted in Fig. 3.3, Fig. 3.4 and Fig. 3.5 and the crossover is described by the equation (3.1):

$$x_{i,j}^{ML+1} = x_{i,j,START}^{ML} + (x_{L,j}^{ML} - x_{i,j,START}^{ML}) * t * PRTVector_j \quad (3.1)$$

where:

$x_{i,j}^{ML+1}$ - value of i–individual's j–parameter, in step t in migration loop ML + 1,

$x_{i,j,START}^{ML}$ - value of i–individual's j-parameter, Start position in actual migration loop,

$x_{L,j}^{ML}$ - value of Leader's j– parameter in migration loop ML,

t - step $\in$ <0, by Step to, PathLength >,

PRTVector - vector of ones and zeros dependent on PRT. If a random number from the interval <0, 1> is less than PRT, then 1 is saved to PRTVector, otherwise it is 0.

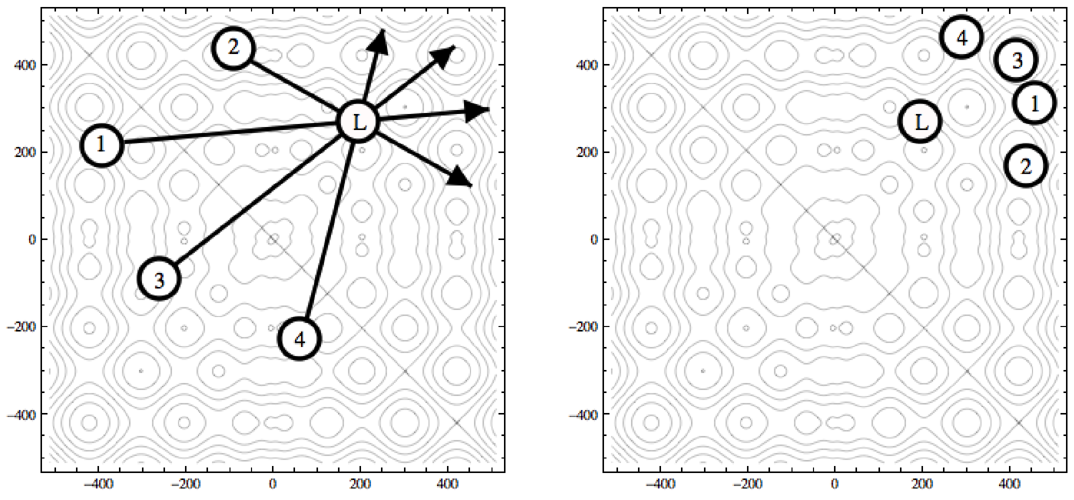


Fig. 3.3: The basic principle of SOMA

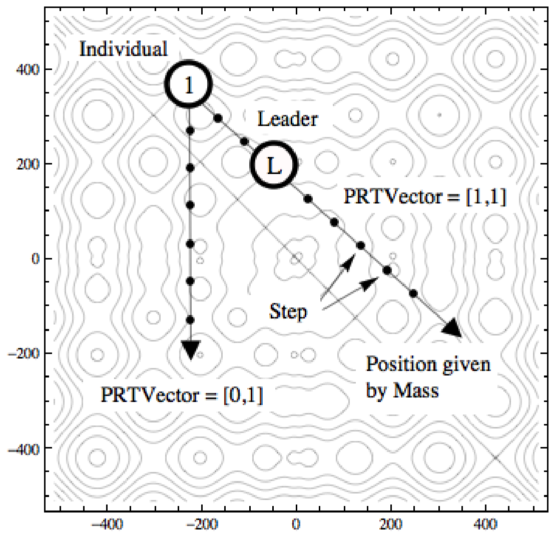
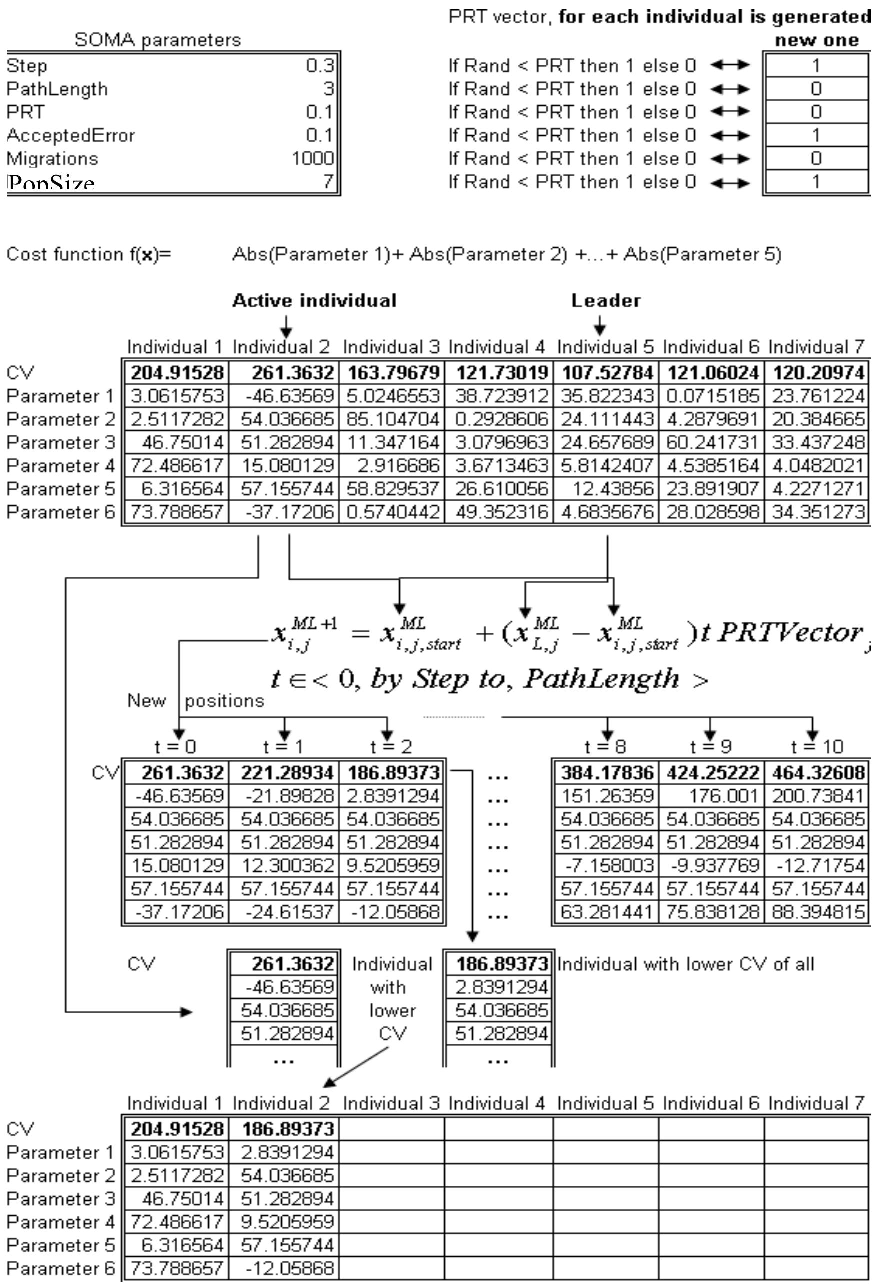


Fig. 3.4: The basic principle of crossover in SOMA, PathLength is replaced here by an older terminology Mass

There are four versions of SOMA – AllToOne, AllToOneRand, AllToAll, and AllToAllAdaptive. In this thesis, a version AllToOne is used despite the fact that AllToAll and AllToAllAdaptive can be much better in searching. They can search for a wider area of solutions and the possibility of finding the global optimum is then more probable. On the other hand, these two variations of SOMA need more time for the successful end of evolution. Therefore for simulations, less time-consuming computing of AllToOne was used in this thesis.



SOMA parameters

Step	0.3
PathLength	3
PRT	0.1
AcceptedError	0.1
Migrations	1000
PopSize	7

PRT vector, **for each individual is generated new one**

If Rand < PRT then 1 else 0 ⟷	1
If Rand < PRT then 1 else 0 ⟷	0
If Rand < PRT then 1 else 0 ⟷	0
If Rand < PRT then 1 else 0 ⟷	1
If Rand < PRT then 1 else 0 ⟷	0
If Rand < PRT then 1 else 0 ⟷	1

Cost function f(**x**)= Abs(Parameter 1)+ Abs(Parameter 2) +...+ Abs(Parameter 5)

Active individual (Individual 2) **Leader** (Individual 5)

	Individual 1	Individual 2	Individual 3	Individual 4	Individual 5	Individual 6	Individual 7
CV	**204.91528**	**261.3632**	**163.79679**	**121.73019**	**107.52784**	**121.06024**	**120.20974**
Parameter 1	3.0615753	-46.63569	5.0246553	38.723912	35.822343	0.0715185	23.761224
Parameter 2	2.5117282	54.036685	85.104704	0.2928606	24.111443	4.2879691	20.384665
Parameter 3	46.75014	51.282894	11.347164	3.0796963	24.657689	60.241731	33.437248
Parameter 4	72.486617	15.080129	2.916686	3.6713463	5.8142407	4.5385164	4.0482021
Parameter 5	6.316564	57.155744	58.829537	26.610056	12.43856	23.891907	4.2271271
Parameter 6	73.788657	-37.17206	0.5740442	49.352316	4.6835676	28.028598	34.351273

$$x_{i,j}^{ML+1} = x_{i,j,start}^{ML} + (x_{L,j}^{ML} - x_{i,j,start}^{ML})\, t\, PRTVector_j$$

$$t \in < 0,\ by\ Step\ to,\ PathLength >$$

New positions

	t = 0	t = 1	t = 2	…	t = 8	t = 9	t = 10
CV	**261.3632**	**221.28934**	**186.89373**	…	**384.17836**	**424.25222**	**464.32608**
	-46.63569	-21.89828	2.8391294	…	151.26359	176.001	200.73841
	54.036685	54.036685	54.036685	…	54.036685	54.036685	54.036685
	51.282894	51.282894	51.282894	…	51.282894	51.282894	51.282894
	15.080129	12.300362	9.5205959	…	-7.158003	-9.937769	-12.71754
	57.155744	57.155744	57.155744	…	57.155744	57.155744	57.155744
	-37.17206	-24.61537	-12.05868	…	63.281441	75.838128	88.394815

		Individual with lower CV		
CV	**261.3632**		**186.89373**	Individual with lower CV of all
	-46.63569		2.8391294	
	54.036685		54.036685	
	51.282894		51.282894	
	…		…	

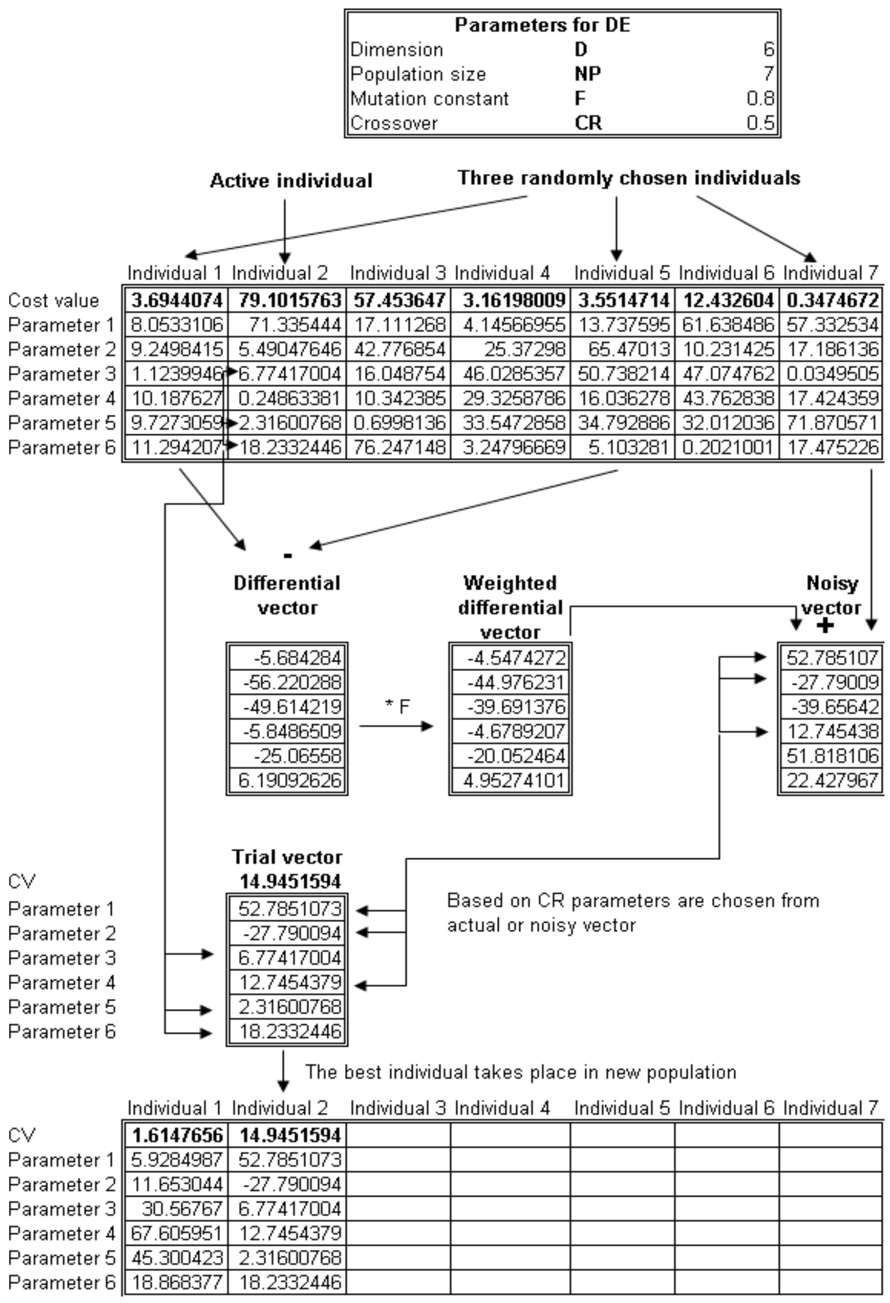
	Individual 1	Individual 2	Individual 3	Individual 4	Individual 5	Individual 6	Individual 7
CV	**204.91528**	**186.89373**					
Parameter 1	3.0615753	2.8391294					
Parameter 2	2.5117282	54.036685					
Parameter 3	46.75014	51.282894					
Parameter 4	72.486617	9.5205959					
Parameter 5	6.316564	57.155744					
Parameter 6	73.788657	-12.05868					

Fig. 3.5: SOMA example

3.2 Differential Evolution

DE is a population-based optimization method that works on real-number-coded individuals [17]. For each individual $\vec{x}_{i,G}$ in the current generation G, DE generates a new trial individual $\vec{x}'_{i,G}$ by adding the weighted difference between two randomly selected individuals $\vec{x}_{r1,G}$ and $\vec{x}_{r2,G}$ to a randomly selected third individual $\vec{x}_{r3,G}$. The resulting individual $\vec{x}'_{i,G}$ is crossed-over with the original individual $\vec{x}_{i,G}$. The fitness of the resulting individual, referred to as a perturbed vector $\vec{u}_{i,G+1}$, is then compared with the fitness of $\vec{x}_{i,G}$. If the fitness of $\vec{u}_{i,G+1}$ is greater than the fitness of $\vec{x}_{i,G}$, then $\vec{x}_{i,G}$ is replaced with $\vec{u}_{i,G+1}$; otherwise, $\vec{x}_{i,G}$ remains in the population as $\vec{x}_{i,G+1}$. 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. The example of DE is in Fig. 3.6. Please refer to (3.2) for notation of cross-over, and to [68] and [47] for the detailed description of used DERand1Bin strategy and all other DE strategies.

$$u_{i,G+1} = x_{r1,G} + F * \left(x_{r2,G} - x_{r3,G} \right) \tag{3.2}$$



Parameters for DE		
Dimension	**D**	6
Population size	**NP**	7
Mutation constant	**F**	0.8
Crossover	**CR**	0.5

	Individual 1	Individual 2	Individual 3	Individual 4	Individual 5	Individual 6	Individual 7
Cost value	**3.6944074**	**79.1015763**	**57.453647**	**3.16198009**	**3.5514714**	**12.432604**	**0.3474672**
Parameter 1	8.0533106	71.335444	17.111268	4.14566955	13.737595	61.638486	57.332534
Parameter 2	9.2498415	5.49047646	42.776854	25.37298	65.47013	10.231425	17.186136
Parameter 3	1.1239946	6.77417004	16.048754	46.0285357	50.738214	47.074762	0.0349505
Parameter 4	10.187627	0.24863381	10.342385	29.3258786	16.036278	43.762838	17.424359
Parameter 5	9.7273059	2.31600768	0.6998136	33.5472858	34.792886	32.012036	71.870571
Parameter 6	11.294207	18.2332446	76.247148	3.24796669	5.103281	0.2021001	17.475226

Differential vector	Weighted differential vector	Noisy vector
-5.684284	-4.5474272	52.785107
-56.220288	-44.976231	-27.79009
-49.614219	-39.691376	-39.65642
-5.8486509	-4.6789207	12.745438
-25.06558	-20.052464	51.818106
6.19092626	4.95274101	22.427967

	Trial vector
CV	**14.9451594**
Parameter 1	52.7851073
Parameter 2	-27.790094
Parameter 3	6.77417004
Parameter 4	12.7454379
Parameter 5	2.31600768
Parameter 6	18.2332446

Based on CR parameters are chosen from actual or noisy vector

The best individual takes place in new population

	Individual 1	Individual 2	Individual 3	Individual 4	Individual 5	Individual 6	Individual 7
CV	**1.6147656**	**14.9451594**					
Parameter 1	5.9284987	52.7851073					
Parameter 2	11.653044	-27.790094					
Parameter 3	30.56767	6.77417004					
Parameter 4	67.605951	12.7454379					
Parameter 5	45.300423	2.31600768					
Parameter 6	18.868377	18.2332446					

Fig. 3.6: A DE example

3.3 Particle Swarm Optimization

The PSO (Particle swarm optimization) algorithm is based on the natural behaviour of birds and fish and was firstly introduced by R. Eberhart and J. Kennedy in 1995 [17], [16]. As an alternative to genetic algorithms [12] and differential evolution [68], PSO proved itself to be able to find better solutions for many optimization problems. The term “swarm intelligence” [17], [16] refers to the capability of particle swarms to exhibit surprisingly intelligent behavior assuming that some form of communication (even very primitive) can occur among the swarm particles (individuals).

PSO is initialized by a population of randomly located particles. A velocity vector, which indicates the direction of individual movement in the next step, is generated to each individual. Then a value of cost function is calculated. The individual with the best value (usually minimum) saves its current position in the common memory of the population which means that individuals know where the best solution is located. The best value found in a population is called gBest. At the same time, every individual finds out if its current position is better than its previous position. If so, the new position is stored in its own memory and is referred to as pBest.

After gBest and pBest are found, the particle adjusts its velocity and changes the position according to the equations (3.3) and (3.4). The influence of velocity, pBest and gBest values is depicted in Fig. 3.7. Particles tend to go their own way or to return to their best position or to adaptively follow the particle with the best value in the population. The trends and real directions are visible in Fig. 3.7.

$$v_d(t+1) = v_d(t) + c_1 * rand * \left(pBest_{i,d} - x_{i,d}(t)\right) + c_2 * rand * \left(gBest_d - x_{i,d}(t)\right) \quad (3.3)$$

$$x_{i,d}(t+1) = x_{i,d}(t) + v_d \quad (3.4)$$

where

$v_d(t+1)$	– a velocity of the particle in the next step
$v_d(t)$	– a velocity of the particle in the current step
$x_{i,d}(t+1)$	– a position of the particle in the next step

$x_{i,d}(t)$	– a position of the particle in the current step
$pBest_{i,d}$	– the best existing position of the particle
$gBest_d$	– the best found position in the population
$rand$	– random number in the interval (0, 1)
c_1, c_2	– priority factors

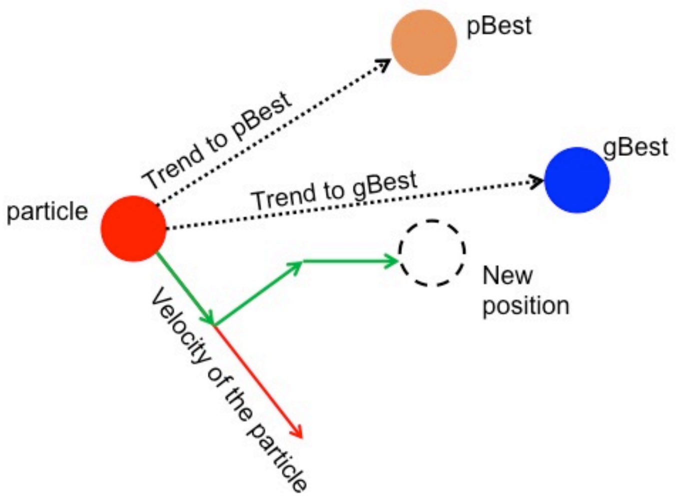


Fig. 3.7: A velocity, pBest and gBest values influence in PSO

Particle velocities are associated with the maximum speed V_{max}. If the velocity of the particle exceeds this rate, a new speed is generated or the velocity vector is reduced to the value of V_{max}. This measure is here because of the particle's tendency to sharply increase its speed. Particles reach the borders of the searched area quickly in that case. If a particle is outside permitted values its new position is generated.

The parameters of PSO:

Dimensions and permitted values are given by the optimized problem.

The number of particles is the size of the population.

Vmax sets a maximal value of the velocity up.

Priority factors c_1, c_2 partly influence the movement of particles. The priority factor c_1 gives preference to return to particle's best own position before to follow the best result of the population. On the contrary, the priority factor c_2 tends to shift particles to the best value of the population.

4 Artificial Neural Networks

Artificial neural networks (ANN) are tools of artificial intelligence developed in the first half of the 1940s. After Pitts – McCulloch model [31], [98], [30], [22] of neuron (Fig. 4.1) and Rosenblatt's first neural net, the perceptron, with a learning algorithm were published, Minsky and Papert caused the temporal abandoning of ANN because the perceptron was not able to solve nonlinear separable problems. Fortunately, in 1980s researches returned and the boom started [31], [98], [30], [22].

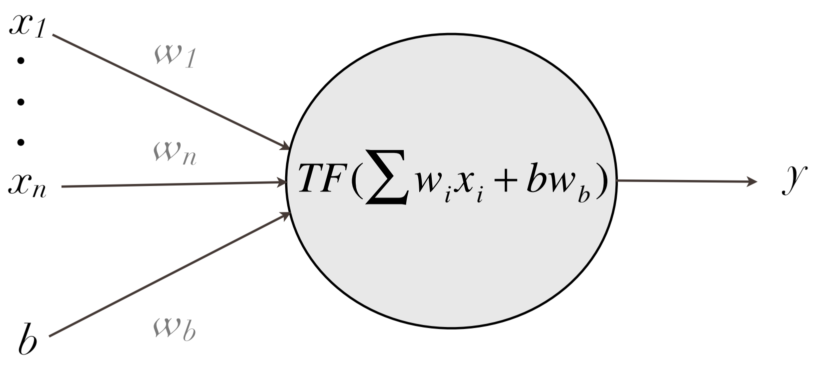


Fig. 4.1: A model of a neuron – TF (transfer function), x_1 - x_n (inputs to neural network), b – bias, w_1 – w_n, w_b – weights, y – output

Artificial neural networks are inspired by the biological neural nets and are used for complex and difficult tasks. The most often usage is the classification of objects because ANN are capable of generalization, hence the classification is natural for them. Some other possibilities are in pattern recognition, control, filtering of signals and also data approximation.

There are several types of artificial neural networks. They are mainly divided into supervised and unsupervised neural networks. Supervised neural nets need a training set with inputs and required outputs which help to train the neural network. Unsupervised neural networks work on different basis. They try to group items in a training set according to similar properties. The other difference is in settings of layers, neurons in layers, types of transfer functions etc. In the case of this thesis, the supervised artificial neural nets were used. Simulations were performed with a feedforward net with supervision. ANN needs a training set of known solutions to be trained. The neural network works so that suitable inputs in numbers have to be given to the input vector. These inputs are multiplied by weights which are adjusted during the training. In the neuron the sum of inputs multiplied by weights are transferred through a mathematical function such as sigmoid, saturated linear (Fig. 4.2), hyperbolic tangent, radial basis functions, etc. Therefore ANN can also be used for data approximation.

Feedforward nets have different training algorithms; the well-known are Backpropagation, Pruning algorithm, gradient methods, Levenberg-Marquardt [76] and others. In the performed simulations, the Levenberg-Marquardt algorithm was used.

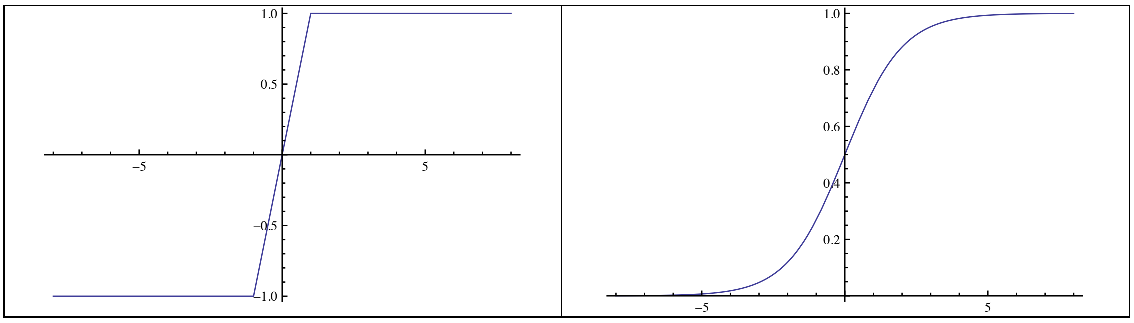


Fig. 4.2: A linear saturated function (left), Sigmoid function (right)

The single neuron units (Fig. 4.1) are connected to different structures to obtain ANN (e.g. Fig. 4.3 -

Fig. 4.6). These networks were designed for different tasks.

Fig. 4.6 shows a different schema of a two layer neural net where the last bottom neuron in the left input layer is bias equal to one.

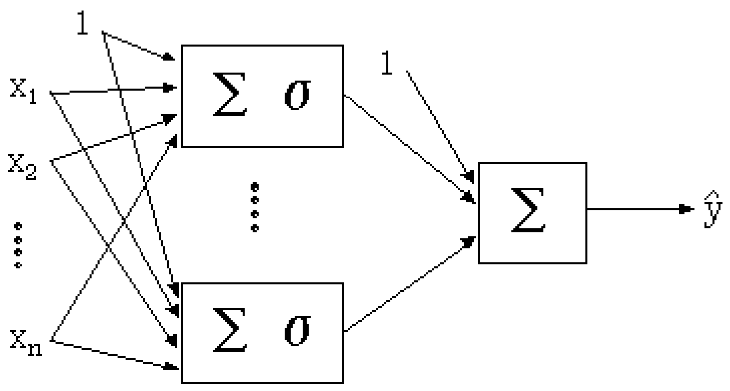


Fig. 4.3: One hidden layer neural net and one output, where $\sum\sigma = TF[\sum(w_i x_i + bw_b)]$ and in this case $\sum = TF[\sum(w_i x_i + bw_b)]$, where TF is for example logistic sigmoid.

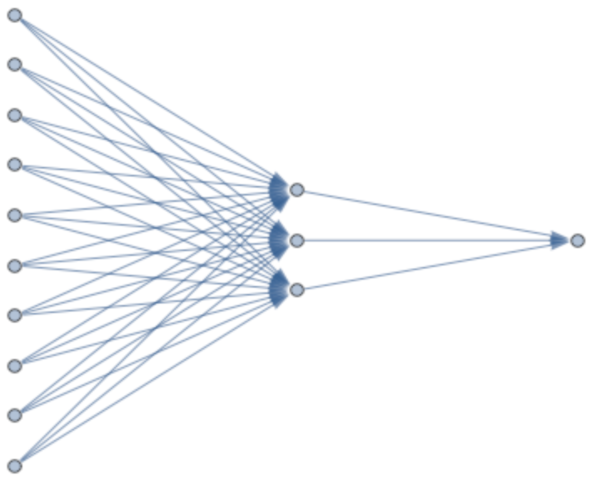


Fig. 4.4: One hidden layer neural net and one output, a different schema

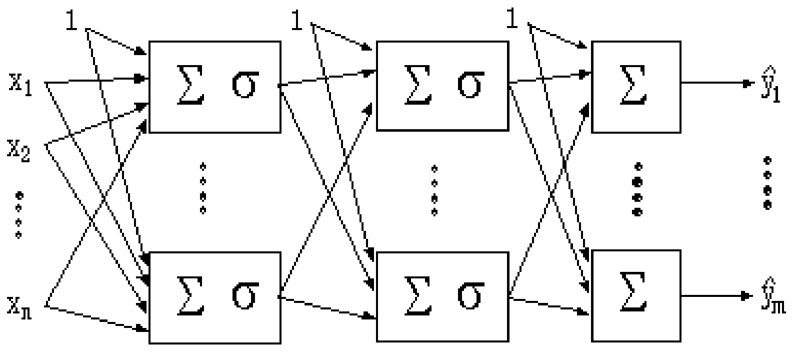


Fig. 4.5: Two hidden layer neural net, where $\sum\sigma = TF[\sum(w_i x_i + bw_b)]$ and in this case $\sum = TF[\sum(w_i x_i + bw_b)]$, where TF is for example a logistic sigmoid. These pictures are taken from Neural Networks Toolbox for Mathematica environment (www.wolfram.com) as this tool was used during the simulations. Names are also taken from this tool to avoid misunderstandings.

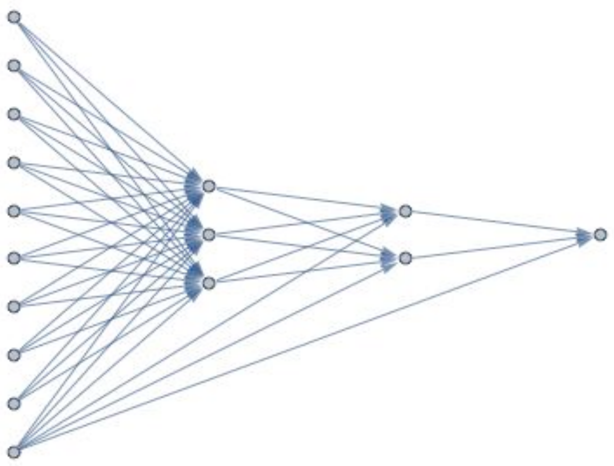


Fig. 4.6: Two hidden layer neural net, a different schema

5 Metaevolution

Metaevolution is one of the main topics that underpin the whole thesis. The range, which is covered by the technical term meta, is quite wide. Meta means going beyond the basic term. The evolutionary algorithms described earlier in this thesis are sometimes also called meta-heuristic [38]. According to [38] and [75], heuristic is defined as a technique which seeks or finds good solutions to a difficult model. Meta-heuristic goes beyond this to draw on ideas and concepts from another discipline to help solve the artificial system that is being modelled.

Generally, metavolution means evolution of evolution. Metaevolutionary techniques for optimization tasks belong to soft computing methods as well as evolutionary algorithms [2], [57] and symbolic regression [57].

Metaevolution means several approaches [57], [15], [18], [38], [42], [55], [62]. This thesis is focused mainly on the third described approach and partly the second one.

First attempts of researchers were in the usage of an evolutionary algorithm for tuning or controlling of another evolutionary technique [19], [14], [89]. During this process, usually the best types of evolutionary operators and settings of their parameters were evolutionarily selected. Tuning of parameters is the usage of the best-found values which will be set up at the initialization of the evolutionary algorithm and used with the same values for a whole process. Compared to this, parameter control adjusts the values during the evolutionary process. It is adapted either by means of a predetermined rule or some kind of self-adaptation [89]. The further performance was then tested and studied on given problems [42], [18].

Another approach is to let evolution create a structure and parameters of the used evolutionary operators such as selection crossover or mutation. Diosan and Oltean use Meta Genetic Programming [18] for the evolutionary design of evolutionary algorithms [15]. In the thesis, this metaevolutionary approach is described with the first application (chapter 7). Analytic programming is used here with symbolic regression principles to breed a completely new structure of the optimization algorithm of evolutionary character which comes from the basic selection of operators in the AP settings.

The last technique of metaevolution is discussed in several applications presented in this thesis. This is the estimation of coefficients in symbolic regression when two evolutionary algorithms help each other. One evolutionary algorithm drives the main process of symbolic regression, in this case analytic programming, and the second is used for the constant estimation. This meta approach of analytic programming has to be used when the constants are not possible to estimate in another manner because of the character of the problem. In data approximation tasks, there can be used a technique from non linear fitting package, which is adopted in Wolfram Mathematica environment, because the problem is designed so that the found constants (e.g. coefficients of polynoms) move the basic shape of the curve around the coordinate system. It is not possible to employ such a package in the case of the synthesis of control laws for chaotic systems or Pseudo ANN. These applications do not use the found result as a model which could be adjusted to some "measured" values in the sense of interpolation but the found solution is used further as a part of complex technique to find a quality of the solution and cost function estimation.

PART 2

–

Selected Applications

6 Selected Applications

The following applications are selected from the research area of the author. The first of the following chapters shows a technique from the field of symbolic regression, which is the main domain of the author. The chapter is focused on metaevolution for breeding of a new algorithm where one evolutionary method creates another one.

The second chapter describes the metaevolution for the interdisciplinary task of synthesis of control law for deterministic chaotic systems. This part of research area has been the main field of research of the author during the last three years.

The next section deals with metaevolution for synthesis of pseudo artificial neural networks. The research is at the beginning, still opened and the work in this field will also continue after the submission of this thesis.

As ANNs are also the area of the research, next two chapters are focused on real applications – the steganalysis and optimal modelling of airplane behaviour by means of ANN.

The author was participating also in the research that connects evolutionary algorithms and deterministic chaotic systems together to explore the influence on the evolutionary dynamics inside the PSO algorithm. Instead of a standard computer pseudorandom number generator, the PSO algorithm used a pseudorandom number generator based on selected chaotic systems for inner operators.

7 Metaevolution in Design of Evolutionary Algorithm

The objective was to try to create a new optimization algorithm, probably of evolutionary character, which could be robust and effective to optimize difficult problems in the world. This research has been started in the doctoral thesis of the author. In this habilitation thesis, it is briefly described to show differences in the metaevolution used in simulations performed after the submission of the doctoral thesis.

This is a metaevolutionary approach in context when evolutionary algorithm breeds another evolutionary algorithm [57]. According to previous approaches, metaevolution is determining the optimal evolutionary algorithm, the best types of evolutionary operators and their parameter setting for a given problem. It basically means that one evolutionary algorithm tunes another one [57]. But the approach used for the synthesis a new algorithm is different. The metaevolution is used on a higher level for creating a new algorithm completely not only for setting of its parameters [57].

The simulations used different operators of known evolutionary algorithms such as their mutation or crossover operators and found following notations for new algorithms (7.1) – (7.4):

SOMAATORandWithoutPRT(SOMAATORandWithPRT(SOMAATORandWithPRT(MutateDECurrentToBest(SelectSOMALeader)))) (7.1)

SOMAATOWithPRT(SOMAATOWithPRT(SOMAATORandWithPRT(CrossDEBin(SOMAATOWithPRT(SelectSOMARandLeader))))) (7.2)

CrossDEBin(SOMAATOWithPRT(MutateDECurrentToBest(SelectSOMALeader))) (7.3)

$$\text{SOMAATOWithPRT(SelectSOMALeader)} \tag{7.4}$$

This kind of metaevolution used analytic programming in its basic version. No coefficients were necessary to be estimated. All operators belong to GFS with one argument and the functions of selection are part of the GFS with zero arguments, ie. terminals.

The most difficult is the design of a cost function which represents suitability and quality of the solution. In the case of creating a new evolutionary algorithm, benchmarking on some test functions is necessary. During this research, two test functions were used for simulations and the cost function tested whether or not a found algorithm achieves the minimum in both test functions. The two benchmark functions were the Sphere model, 1st De Jong as an example of a unimodal function and Schwefel as an example of a multimodal function [105], [57] – Fig. 7.1 and Fig. 7.2.

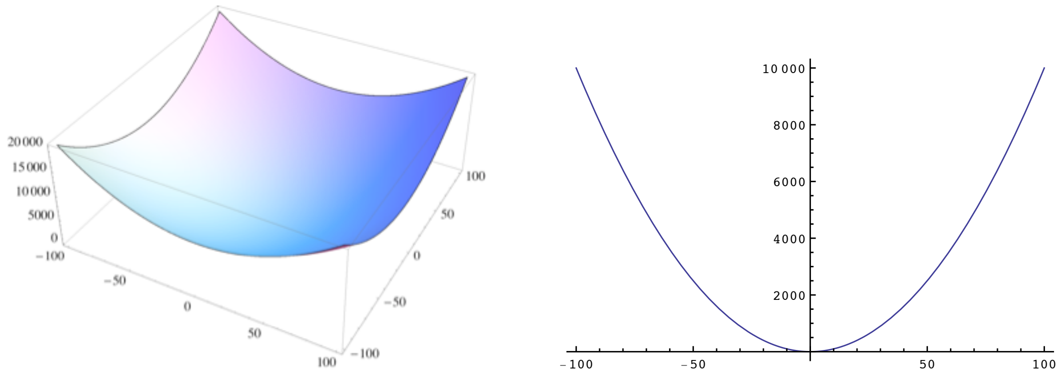


Fig. 7.1: The DeJong function – unimodal (left – 2 arguments and right – 1 argument used)

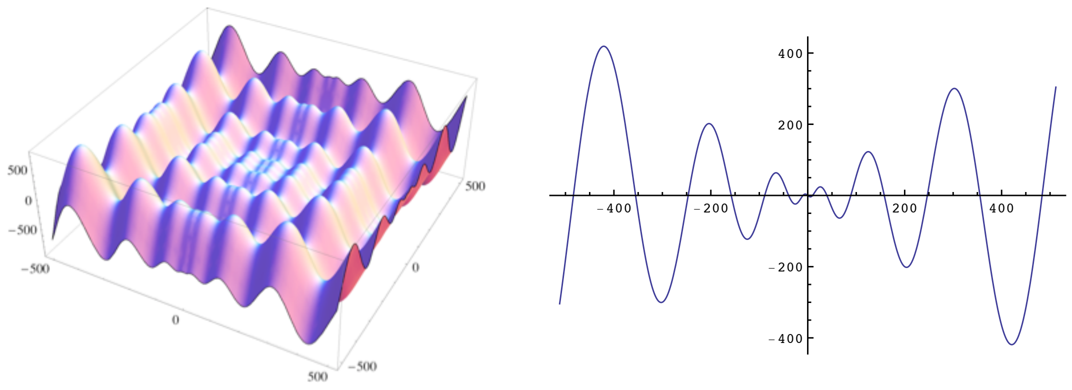


Fig. 7.2: The Schwefel function – multimodal (left – 2 arguments and right – 1 argument used)

The 1st De Jong and Schwefel functions are in the analytical way as can be seen in equations (7.5) and (7.6), where Dim means the number of arguments (dimension of the problem). No other condition was applied.

$$f(x) = \sum_{i=1}^{Dim} x_i^2 \tag{7.5}$$

$$f(x) = \sum_{i=1}^{Dim} -x_i \cdot \sin\left(\sqrt{|x_i|}\right) \tag{7.6}$$

The value of the cost function was designed so that firstly the generated algorithm is verified as to the ability to find the minimum on the easy unimodal function 1st De Jong. If the minimum is reached, the Schwefel function is tested. Then the cost value is the output from the Schwefel. If there is no successful result from the 1st De Jong, the output value is the absolute value of the 1st De Jong. The values of benchmark function minimum in different dimensions are known. For faster computation, the benchmark functions were used with 2 arguments. The future research expects a better design of the cost function including more benchmark functions and computations in a higher dimensional space.

After the results were obtained – the 4 above mentioned algorithms, more tests on other benchmark functions were performed to find out how effective the found evolutionary algorithms are. Here is only one table (Table 7.1) showing that algorithms competed not only between themselves but also in dimensions - 2D, 20 D and 100 D.

Table 7.1: The winner for each benchmark function

	Algorithm 1 (7.1)	Algorithm 2 (7.2)	Algorithm 3 (7.3)	SOMAATO (7.4)
2 D	1, 3, 4, 5, 6, 8, 10, 11, 12, 15, 16	5, 6, 7, 8, 10, 11, 12, 13, 15, 16	5, 6, 7, 8, 10, 11, 12, 13, 15, 16	2, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16
20 D	1, 3, 4, 7, 11, 14, 15	5, 6, 8, 10, 12, 13, 16	2	9
100 D	12, 13, 14	1, 2, 3, 5, 7, 8, 15, 16,		4, 6, 9, 10, 11

The numbers are for each benchmark problem as follows: 1 - 1st De Jong function, 2 - 2nd De Jong function, 3 - 3rd De Jong function, 4 - 4th De Jong function, 5 - Rastrigin function, 6 - Schwefel function, 7 - Griewangk function, 8 - Sine Envelope Sine Wave function, 9 - Stretched V sine wave function - Ackley, 10 - Ackley test function, 11 - Ackley function, 12 - Egg Holder function, 13 - Rana function, 14 - Pathological function, 15 - Michalewicz function, 16 - Master’s cosine wave function.

If the same number appears in more cells on the same row it means that algorithms finished in the same cost value. For more details, please refer to [57].

8 Metaevolution for Synthesis of Control Law for Chaotic Systems

8.1 Introduction

The interest in the interconnection between evolutionary techniques and the control of chaotic systems is spread nowadays. First steps were done in [106], [86], [81] where the control law was based on the Pyragas method: Extended delay feedback control – ETDAS [73]. These papers were focused on how to tune several parameters inside the control technique for a chaotic system. Compared to previous research, this chapter shows a possibility of how to generate the whole control law (not only to optimize several parameters) for the purpose of stabilization of a chaotic system. The synthesis of control is inspired by the Pyragas's delayed feedback control technique [39], [72]. Unlike the original OGY (Ott – Grebogi – York) control method [63], it can be simply considered as a targeting and stabilizing algorithm together in one package [46]. Another great advantage of the Pyragas method for evolutionary computation is the amount of accessible control parameters which can be easily tuned by means of evolutionary algorithms (EA). Apart from soft-computing (artificial intelligence) methods, following methods of mathematical optimization are commonly used: the simplex method (linear programming) [50], quadratic programming [49], the branch and bound method [97] and NEH Heuristic. The simplex method or linear programing is used generally for real-time and simple optimizations, mostly with unimodal cost functions, whereas the branch and bound method has proved highly successful for permutative constrained problems. NEH heuristic was developed for the optimizations of scheduling problems. Previous experiments with the connection of optimization problems and chaotic systems proved the difficulty of this task due to the highly nonlinear and erratic cost function surfaces, thus common mathematical optimization techniques could not be utilized.

In this research, instead of evolutionary algorithms (EA) utilization [40], analytic programming (AP) is used. The control law from the proposed system can be viewed as a symbolic structure that can be synthesized according to the requirements for the stabilization of a chaotic system. The advantage is that it is not necessary to have some "preliminary" control law and to estimate its parameters only. This system will generate the whole structure of the law even with suitable parameter values.

This research is focused on the research expansion and usage of analytic programming for the synthesis of a whole control law instead of parameters tuning for existing and commonly used control law method that is used to stabilize desired Unstable Periodic Orbits (UPO) of chaotic systems. The research presented in this chapter is focused on the stabilization of p-1 UPO – a fixed point (stable state) and higher periodic orbits p-2 UPO (oscillation between two points) and p-4 UPO. Two approaches were adopted for the stabilization – a simple evolutionary approach with the cost function utilizing the position of the desired UPO and the blackbox evolutionary approach utilizing special cost functions, thus without the knowledge of the exact UPO position in the chaotic attractor. This means that EA is used to find the best control parameter set up based only on the demanded type of chaotic system behavior and not based on the position of UPO.

This research has been already published in book chapters, conference and journal papers, e.g. [61], [55], [83], [84], [85], [87].

In this demonstration, analytic programming with meta version was used. Metaevolution here means the usage of one evolutionary algorithm for main the AP process and the second algorithm for coefficient estimation, as was explained above. The SOMA algorithm was used for the main AP process and DE was used in the second evolutionary process.

8.2 Methodology for Control by Means of AP

The methodology used in these simulations is divided into two parts – control laws for p-1 orbit (stable state) and higher periodic orbits p-2 and p-4 UPO (oscillation between two, respectively 4 values). In the first case, the inspiration for preparation of sets of basic functions and operators for AP was the simpler TDAS control method (8.1) and its discrete form given in (8.2).

$$F(t) = K[x(t-\tau) - x(t)] \tag{8.1}$$

$$F_n = K(x_{n-m} - x_n) \tag{8.2}$$

This means that only current output value x_n and one previous x_{n-1} were used in the set of basic functions together with constants, operators such as plus, minus and power.

The latter case was inspired by the method ETDAS due to the recursive attributes of the delay equation *S* utilizing previous states of the system. Therefore the data set for AP was expanded and covers a longer system output history.
The original control method – ETDAS has the form (8.3).

$$F(t) = K\left[(1-R)S(t-\tau_d) - x(t)\right]$$

$$S(t) = x(t) + RS(t-\tau_d) \tag{8.3}$$

Where: *K* and *R* are adjustable constants, *F* is the perturbation; *S* is given by the delay equation utilizing previous states of the system and τ_d is a time delay. The original control method – ETDAS in the discrete form has the form (8.4).

$$F_n = K\left[(1-R)S_{n-m} - x_n\right]$$

$$S_n = x_n + RS_{n-m} \tag{8.4}$$

Where: *m* is the period of *m*-periodic orbit to be stabilized. The perturbation F_n in equations (8.4) may have arbitrarily large value which can cause the diverging of the system. Therefore, F_n should have a value between $-F_{\max}$, $F_{\max}$. In this thesis a suitable $F_{\max}$ value was taken from the previous research [88].

8.3 Settings

The novelty of the meta-evolutionary approach represents the synthesis of a feedback control law F_n (perturbation) inspired by the original ETDAS or TDAS control method. The perturbation is the feedback to the system which helps to stabilize it.

Therefore the basic set of elementary functions for AP was selected as follows. The main items are several previous values (data) to current value in the controlled chaotic system.

GFS_{2arg} = +, -, /, *, ^

GFS_{0arg} = $data_{n-1}$ to $data_n$, K (for p-1 orbit)

GFS_{0arg} = $data_{n-9}$ to $data_n$, K (for p-2 orbit).

GFS_{0arg} = $data_{n-11}$ to $data_n$, K (for p-4 orbit).

The following tables (Table 8.1, Table 8.2) contain the settings of evolutionary algorithms for AP, the main procedure and also the meta approach algorithm.

Table 8.1: Parameters setting for SOMA used as the main algorithm in the meta-evolutionary approach.

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

Table 8.2: Parameters setting up for DE used as the second algorithm in the meta-evolutionary approach.

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

8.4 Cost Function Design

The examples of the results show two approaches to the cost function design – simple and blackbox mode.

The first proposal for the cost function comes from the simplest Cost Function (CF). 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.

Because of the stabilization of an extremely sensitive chaotic system, another universal cost function with the possibility of adding penalization rules had to be used. It was synthesized from the simple CF and other terms were added. In this case, it is not possible to use the simple rule of minimizing the area created by the difference between the required and actual state on the whole simulation interval – τ_i, due to many serious reasons, for example: including of initial chaotic transient into the final CF value or degrading of the possible best solution by phase shift of higher periodic orbit, which represents the oscillations between several values.

This CF is in general based on searching for a desired stabilized periodic orbit and thereafter calculation of the difference between the desired and found actual periodic orbit on the short time interval - τ_s (20 iterations) from the point where the first minimal value of the difference between the desired and actual system output is found. Such a design of CF should secure the successful stabilization of either p-1 orbit (stable state) or a higher periodic orbit anywise phase shifted. The CF_{Basic} has the form (8.5).

$$CF_{Basic} = pen_1 + \sum_{t=\tau 1}^{\tau 2} |TS_t - AS_t|, \tag{8.5}$$

where:

TS - target state, AS - actual state

τ_1 - the first min value of the difference between TS and AS, τ_2 – the end of the optimization interval ($\tau_1 + \tau_s$)

$pen_1 = 0$ if $\tau_i - \tau_2 \geq \tau_s$; $pen_1 = 10*(\tau_i - \tau_2)$ if $\tau_i - \tau_2 < \tau_s$ (i.e. late stabilization).

The second type of the cost function (CF2) used in the simulations for the stabilizing of the chaotic system was in the "blackbox mode", ie. without the exact numerical value of the target state. In this case, it is not possible to use the simple rule of minimizing the area created by the difference between the required and actual state on the whole simulation interval – τ or its arbitrary part.

This approach is based on searching for periodic orbits in a chaotic attractor and stabilizing the system on these periodic orbits by means of applying the optimal feedback perturbation F_n. It means that this new CF did not take any numerical target state into consideration but the selected target behavior of system. Therefore, this kind of CF is based on the search for optimal feedback perturbation F_n securing the stabilization on any type of the selected UPO (p-1 orbit – stable state, p-2 orbit – oscillating between two values etc.). The slight disadvantage of this approach is that for each UPO (i.e. different behavior) a different CF is needed.

The results in this thesis show only one case with the blackbox mode, only for demonstration. The other systems with the blackbox mode were published at conferences or in journals.

The proposal of CF_2 used in the case of p-2 orbit is based on the following simple rule. The iteration $y(n)$ and $y(n+2)$ must have the same value. But this rule is also valid for the case of – p-1 orbit, where in discrete systems, the iteration $y(n)$ and $y(n+1)$ of the output value must be the same. Thus another condition had to be added. It says that in the case of p-2 orbit there must be a difference between the *n* and *n+1* output iteration. Considering the fact of minimizing the CF, the value of this condition had to be rewritten into this suitable form (8.6)

$$\frac{1}{|y(n+1)-y(n)|+c} \tag{8.6}$$

where: c – small constant 1.10^{-16} which was added to prevent the evolutionary optimization from crashing because of division by zero which the suboptimal solution stabilized at p-1 orbit returns. The CF_2 has the form (8.7).

$$CF_2 = p1 + \sum_{t=0}^{\tau} |y(n+2)-y(n)| + \frac{1}{|y(n+1)-y(n)|+c} \tag{8.7}$$

where: p1 = penalization. In the proposed CF_2, penalization has to be included because it should avoid finding solutions where the stabilization on saturation boundary values {0, 1} or oscillation between them (i.e. artificial p-2 orbit) occurs. This penalization was calculated as the sum of the number of iterations, where the system output reaches the saturation boundary value.

8.5 Selected Chaotic Systems Used in Simulations

8.5.1 Logistic Equation

The Logistic equation (Logistic map) is a one-dimensional discrete-time example of how complex chaotic behaviour (8.8) can arise from a very simple non-linear dynamical equation. This chaotic system was introduced and popularized by the biologist Robert May [48]. It was originally introduced as a demographic model as a typical predator–prey relationship. The chaotic behaviour can be observed by varying the parameter *r*. At r = 3.57 is the beginning of chaos. At r > 3.57, the system exhibits chaotic behaviour. The example of this behaviour is depicted in the bifurcation diagram – Fig. 8.1.

$$x_{n+1} = rx_n(1 - x_n) \tag{8.8}$$

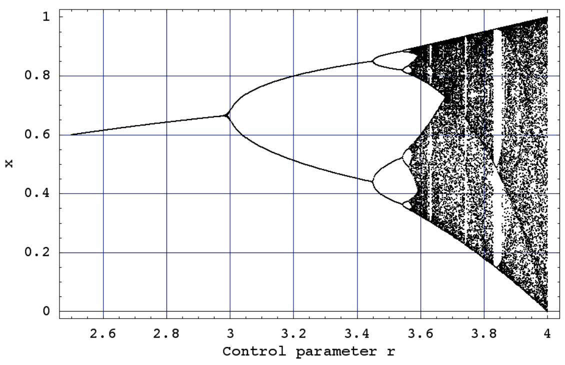


Fig. 8.1: The bifurcation diagram of the Logistic equation

8.5.2 Hénon Chaotic System

The second chosen example of a chaotic system was the two dimensional Hénon map in the form (8.9).

$$\begin{aligned} x_{n+1} &= a - x_n^2 + by_n \\ y_{n+1} &= x_n \end{aligned} \tag{8.9}$$

This is a model invented with a mathematical motivation to investigate chaos. The Hénon map is a discrete-time dynamical system which was introduced as a simplified model of the Poincaré map for the Lorenz system. It is one of the most studied examples of dynamical systems that exhibit chaotic behavior. The map depends on two parameters, *a* and *b*, which for the canonical Hénon map have values of $a = 1.4$ and $b = 0.3$. For these canonical values the Hénon map is chaotic [33].

The example of this chaotic behavior can be clearly seen in the bifurcation diagram – Fig. 8.2, which was created by plotting of a variable *x* as a function of one control parameter for the fixed second parameter.

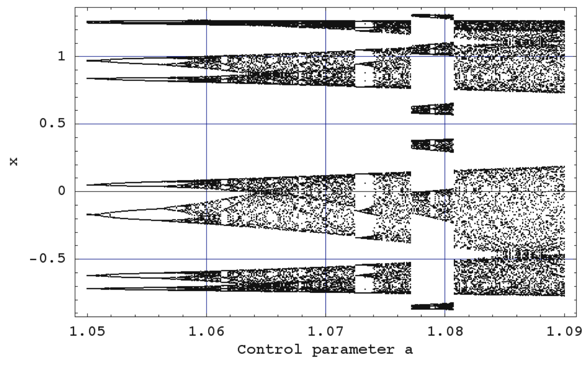


Fig. 8.2: The bifurcation diagram of the Hénon Map

8.5.3 Synthesized Chaotic System

The other selected example of chaotic systems was a synthesized system (8.10) introduced in [103]. The experiments published in [103] were made for the purpose of synthesizing various chaotic systems by means of analytic programming. The presented approach of the synthesis of a whole control law suppresses the fact that appeared in the previous research [88], that some of the synthesized systems are barely controllable.

$$x_{n+1} = \frac{A\left(2A - 2x_n^2 - 3x_n\left(A - x_n + Ax_n\right)\right)}{-A + x_n - x_n^2} \tag{8.10}$$

This system exhibits chaotic behavior for the control parameter A in the ranges <0.1, 0.13> and <0.8, 1.2> (see Fig. 8.3, Fig. 8.4).

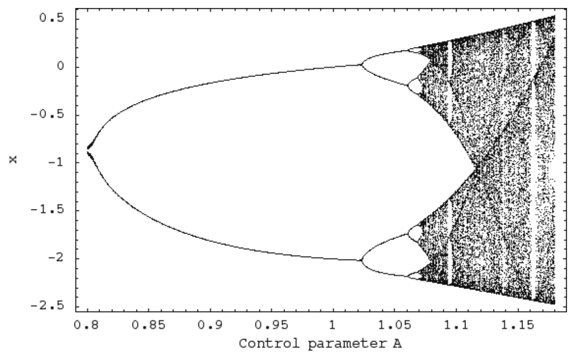


Fig. 8.3: The bifurcation diagram for A = <0.8, 1.2>

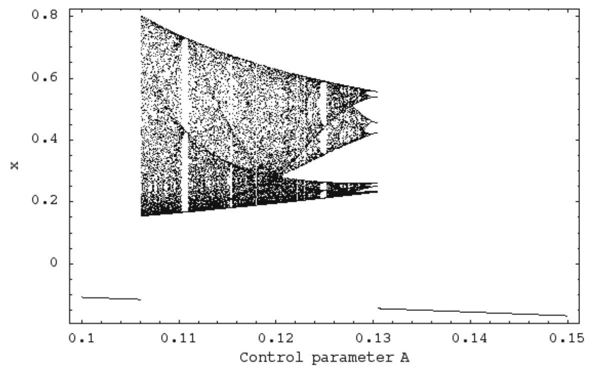


Fig. 8.4: The bifurcation diagram for A = <0.1, 0.15>

8.5.4 Lozi Map

The Lozi map is a complex nonlinear discrete two-dimensional chaotic map. The map equations are given in (8.11) and (8.12) and in the bifurcation diagram in Fig. 8.5. The parameters used in this thesis are: a = 1.7 and b = 0.5 as suggested in [91].

$$X_{n+1} = 1 - a|X_n| + bY_n \tag{8.11}$$

$$Y_{n+1} = X_n \tag{8.12}$$

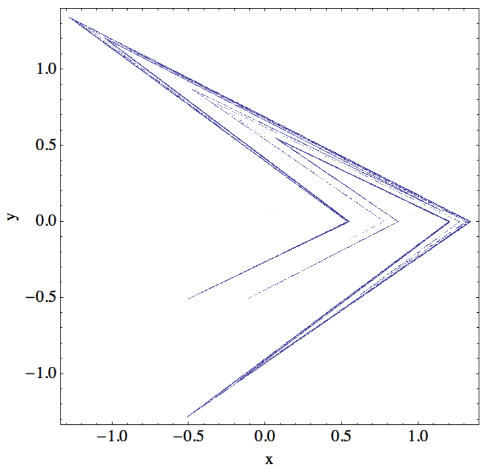


Fig. 8.5: The bifurcation diagram for the Lozi map

8.5.5 Burger's Map

The Burger's map is a simple two-dimensional discrete chaotic system. The map equations are given in (8.13) and (8.14) and in the bifurcation diagram in Fig. 8.6. This map uses parameters *a* = 0.75 and *b* = 1.75 as suggested in [91].

$$X_{n+1} = aX_n + Y_n^2 \tag{8.13}$$

$$Y_{n+1} = bY_n + X_nY_n \tag{8.14}$$

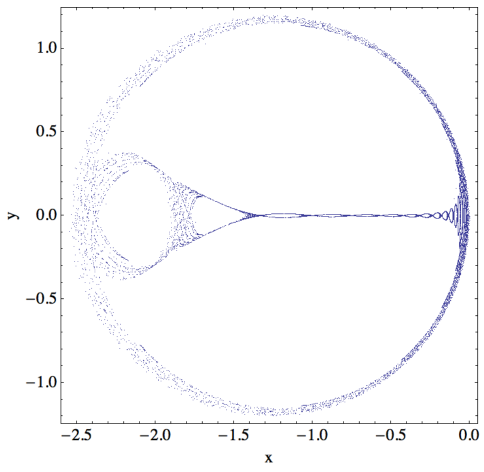


Fig. 8.6: The bifurcation diagram for the Burger's map

8.5.6 Delayed Logistic Equation

The Delayed logistic equation is a simple two-dimensional discrete system. It is a two dimensional extension of the logistic equation [48]. The map equations are given in (8.15) and (8.16) and in the bifurcation diagram in Fig. 8.7. The parameter used in this thesis is *A* = 2.27 [91].

$$X_{n+1} = AX_n\left(1 - Y_n\right) \tag{8.15}$$

$$Y_{n+1} = X_n \tag{8.16}$$

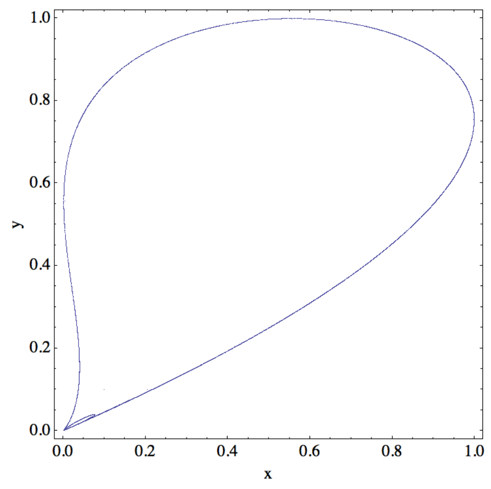


Fig. 8.7: The bifurcation diagram for the Delayed logistic equation

8.5.7 Cubic map

The Cubic map is a simple one-dimensional discrete system very similar to the most known and studied logistic equation [48]. The map equation is given in (8.17) and in the bifurcation diagram in Fig. 8.8. The parameter used in this thesis is *A*= 3.0 as it was also suggested in [91].

$$X_{n+1} = AX_n\left(1 - X_n^2\right) \tag{8.17}$$

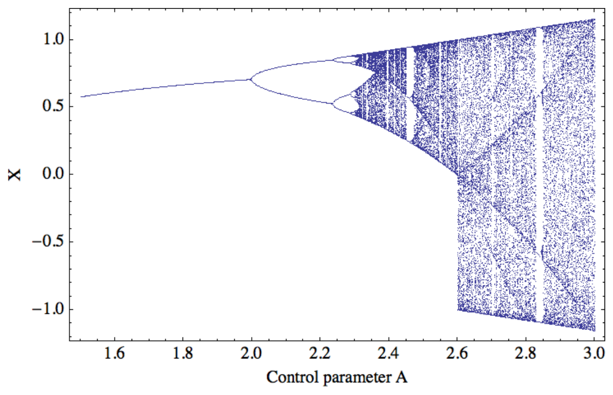


Fig. 8.8: The bifurcation diagram for the Cubic map

8.6 Results for p-1 Orbit – Stable State

The examples of new synthesized feedback control laws F_n (perturbation) for the controlled logistic equation (8.18), the Hénon map (8.19), the evolutionary synthesized system (8.20), the Lozi map (8.21), the Burger's map (8.22), the Delayed logistic equation (8.23) and the Cubic map (8.24):

$$x_{n+1} = rx_n\left(1 - x_n\right) + F_n \tag{8.18}$$

$$x_{n+1} = a - x_n^2 + by_n + F_n \tag{8.19}$$

$$x_{n+1} = \frac{A\left(2A - 2x_n^2 - 3x_n\left(A - x_n + Ax_n\right)\right)}{-A + x_n - x_n^2} + F_n \tag{8.20}$$

$$X_{n+1} = 1 - a\left|X_n\right| + bY_n + F_n \tag{8.21}$$

$$X_{n+1} = aX_n + Y_n^2 + F_n \tag{8.22}$$

$$X_{n+1} = AX_n\left(1 - Y_n\right) + F_n \tag{8.23}$$

$$X_{n+1} = AX_n\left(1 - Y_n\right) + F_n \tag{8.24}$$

which were inspired by the original TDAS control method (8.2) are given in Table 8.4. The values for p-1 UPO (a fixed point) of unperturbed chaotic systems based on the mathematical analysis of the system are depicted in Table 8.3.

Table 8.3: The values for p-1 UPO (a fixed point)

Chaotic system	Value of p-1 UPO of unperturbed system
Logistic equation	$x_F = 0.73842$
Hénon map	$x_F = 0.8$
Evolutionary synthesized system	$x_F = -1.0772$
Lozi map	$x_F = 0.4545$
Burger's map	$x_F = 0.74999$
Delayed logistic equation	$x_F = 0.55947$
Cubic map	$x_F = -0.8165$

Simulation outputs are depicted for all selected chaotic systems in Fig. 8.9. The last four systems are not studied and used for the confirmation of the research methodology so often. Therefore only one example is provided.

The identical minimal final CF value very close to zero for all selected examples gives weight to the argument, that AP is able to synthesize various types of control laws, securing the precise and fast stabilization with machine numerical precision on the p-1 unstable periodic orbit of a real chaotic system.

In the case of the logistic equation, one interesting phenomenon occurred. AP has found the notation of original TDAS [73] which was the inspiration for creating the basic data sets for AP. For comparison, please refer to the first line in Table 8.4. and the notation of the TDAS method in (8.2), where *K* is the gain constant for the logistic equation, the recommended value is around -0.5.

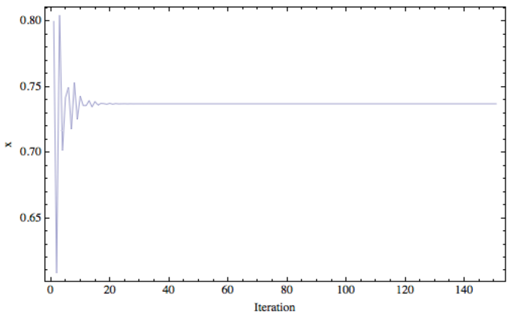
In the case of the Hénon map, the stabilization on a real chaotic UPO was very precise; the only difference across all simulation results was the speed of stabilization. Nevertheless this quality parameter was not included in the CF this time.

The results for the complicated evolutionary synthesized chaotic system give weight to the argument that AP is able to synthesize various new control laws securing very quick and full 100% stabilization even for artificially synthesized systems.

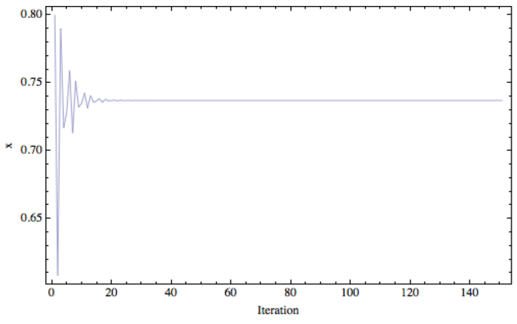
Table 8.4: The simulation results for chaotic systems and stabilization at p-1 UPO

Nr.	Control Law with estimated coefficients	CF Value	UPO Value	Figure
Logistic equation				
1	$F_n = -0.527311(x_{n-1} - x_n)$	$6.6613.10^{-16}$	0.73842	Fig. 8.9 a)
2	$F_n = (x_{n-1} - x_n)(0.352456 - x_{n-1}^{1-x_{n-1}})$	$6.6613.10^{-16}$	0.73842	Fig. 8.9 b)
Hénon map				
1	$F_n = \frac{(x_{n-1} - x_n - 1.62925)(x_{n-1}x_n - x_n^2)}{2x_{n-1}}$	$1.3323.10^{-15}$	0.8	Fig. 8.9 c)
2	$F_n = -\frac{0.781971(x_{n-1}x_n - x_n^2)}{x_{n-1}}$	$1.3323.10^{-15}$	0.8	Fig. 8.9 d)
Evolutionary synthesized system				
1	$F_n = \frac{0.215597(x_{n-1} - x_n)(x_{n-1} + x_n)}{x_{n-1}^2}$	0	-1.0772	Fig. 8.9 e)
2	$F_n = \frac{x_{n-1}}{x_n - \frac{3.40013}{(x_{n-1} - x_n)x_n^2(2x_n - x_{n-1})}}$	0	-1.0772	Fig. 8.9 f)
Lozi map				
1	$F_n = \frac{(x_{n-1} - x_n)}{(2x_{n-1} - 3.8934)x_n}$	$6.2992.10^{-15}$	0.4545	Fig. 8.9 g)
Burger's map				
1	$F_n = -0.01294(x_n + 20.7686(x_{n-1} + x_n) + 39.8298)$	0	0.74999	Fig. 8.9 h)
Delayed logistic equation				
1	$F_n = -\frac{x_n(x_n - x_{n-1})}{x_n^2 + 2x_n - 0.851425}$	0	0.55947	Fig. 8.9 i)
Cubic map				
1	$F_n = 2x_n - \frac{14.0697}{x_n - 7.79938}$	$4.3087.10^{-10}$	-0.8165	Fig. 8.9 j)

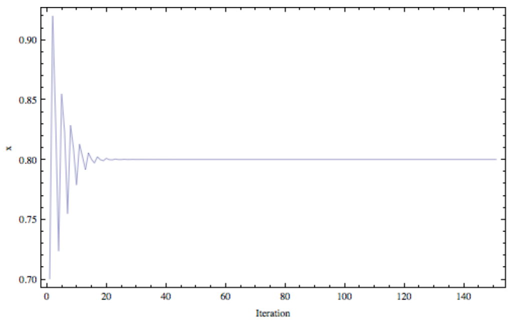
a)


x
0.80
0.75
0.70
0.65
0
20
40
60
80
100
120
140
Iteration

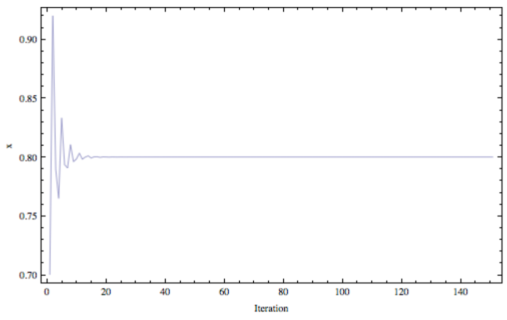
b)


x
0.80
0.75
0.70
0.65
0
20
40
60
80
100
120
140
Iteration

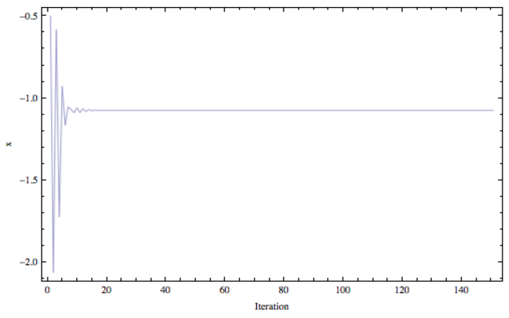
c)


x
0.90
0.85
0.80
0.75
0.70
0
20
40
60
80
100
120
140
Iteration

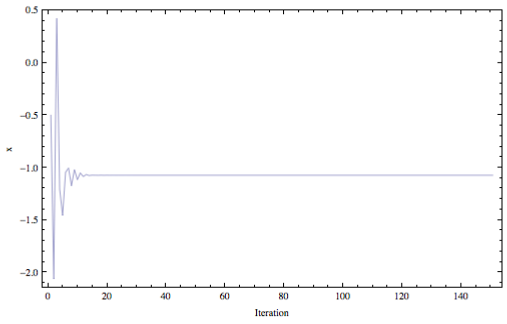
d)


x
0.90
0.85
0.80
0.75
0.70
0
20
40
60
80
100
120
140
Iteration

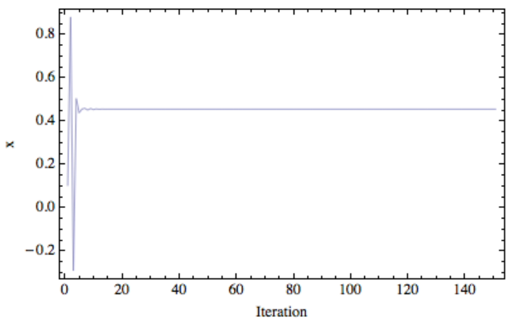
e)


x
−0.5
−1.0
−1.5
−2.0
0
20
40
60
80
100
120
140
Iteration

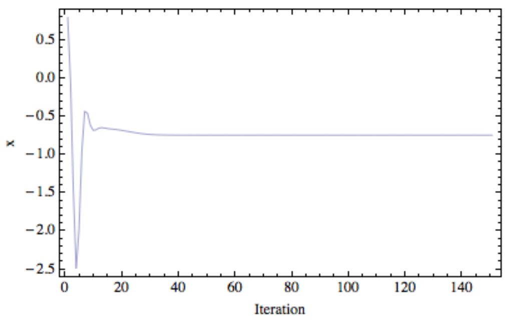
f)


x
0.5
0.0
−0.5
−1.0
−1.5
−2.0
0
20
40
60
80
100
120
140
Iteration

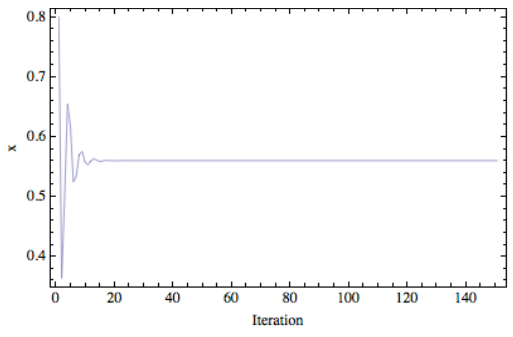
g)


x
0.8
0.6
0.4
0.2
0.0
−0.2
0
20
40
60
80
100
120
140
Iteration

h)


x
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−2.5
0
20
40
60
80
100
120
140
Iteration

i)



j)

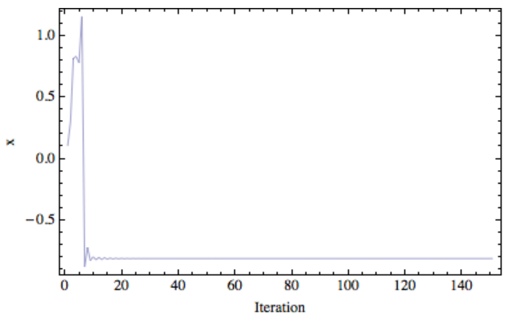


Fig. 8.9: Examples of results – the stabilization of chaotic systems by means of control laws given in Table 8.4 – a), b) – the logistic equation, c), d) – the Hénon map, e), f) - the evolutionary synthesized system, g)- the Lozi map, h) – the Burger's map, i) – the Delayed logistic equation, j) – the Cubic map

8.7 Results for p-2 Orbit – Oscillation between Two Points

The simulations for p-2 were carried in both modes – the simple cost function and blackbox mode. The simple cost function was used for the Hénon map and the blackbox mode for the logistic equation and evolutionary synthesized systems. The perturbed system notations can be found in (8.18) – (8.20). The synthesized control laws for the three selected systems are provided in Table 8.5. The values for p-2 UPO (oscillation between two points) of unperturbed chaotic systems based on the mathematical analysis of the system are depicted in Table 8.6. The relevant figures of the simulation output are given in Fig. 8.10.

Both approaches were able to find the control laws that secure the fast stabilization for p-2 orbit.

An interesting phenomenon was discovered for systems with the blackbox mode of the cost function. The synthesized control laws are able to stabilize the chaotic system on optional artificial periodic orbits as can be seen in Table 8.5 and Fig. 8.10. This is caused by the fact that there was no information about the exact position of p-2 orbit in the chaotic attractor transferred into the evolutionary process, and the cost function was designed to operate on the basis of the selection of desired system behaviour.

Table 8.5: The simulation results for chaotic systems and stabilization at p-2 UPO

Nr.	Control Law with estimated coefficients	CF Value	UPO Value	Figure
Logistic equation				
1	$F_n = -\frac{x_{n-6} - 50.0535}{x_{n-7} + 47.367} - x_{n-3}$	198.68	0.98 – 0.44	Fig. 8.10 a)
2	$F_n = x_{n-1}{}^{19.464}$	149.06	0.94 – 0.21	Fig. 8.10 b)
Hénon map				
1	$F_n = 0.523744(x_{n-1} + x_n - 0.700001)$	$1.39845.10^{-5}$	-0.56 – 1.26	Fig. 8.10 c)
2	$F_n = 0.20375x_{n-8}(x_{n-4} - x_{n-3} + 1.87967)x_{n-3}(x_{n-2} - x_n)$	$1.70211.10^{-5}$	-0.56 – 1.26	Fig. 8.10 d)
Evolutionary synthesized system				
1	$F_n = -\left(-\frac{1.86885}{x_n} + x_{n-7} + x_{n-2}\right)x_n(-0.0874 - x_n)$	50.4868	-2.45 – 0.12	Fig. 8.10 e)
2	$F_n = -(x_{n-8} + x_{n-6} + x_{n-4} + 0.873148)x_n - 0.266463$	50.613	-2.45 – 0.12	Fig. 8.10 f)

Table 8.6: The values for p-2 UPO (oscillation between two points)

Chaotic system	Values of p-2 UPO of unperturbed system
Logistic equation	x_F = 0.37 and 0.89
Hénon map	x_F = -0.56 and 1.26
Evolutionary synthesized system	x_F = -2.03 and 0.12

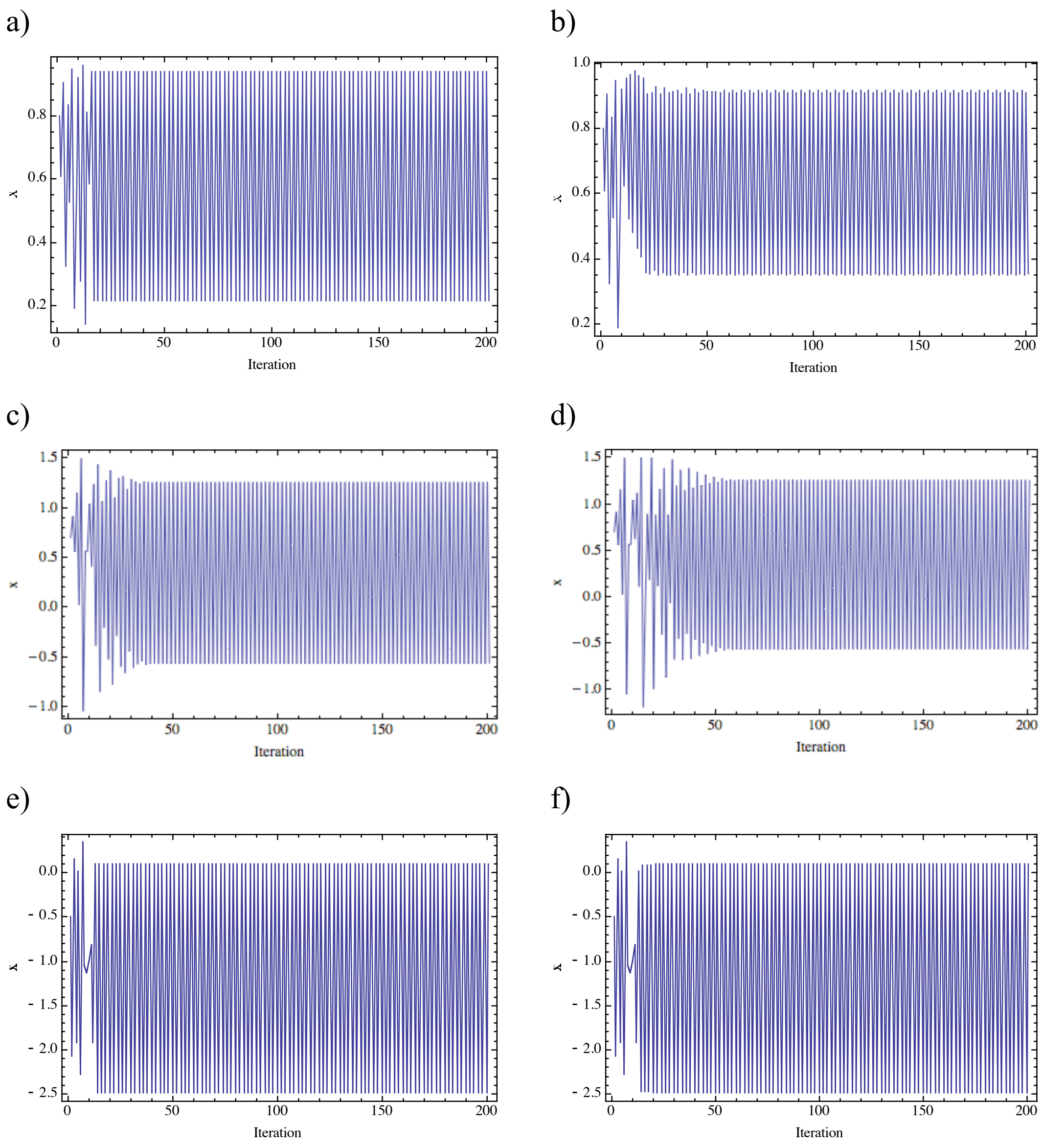


Fig. 8.10: Examples of results – the stabilization of chaotic systems by means of control laws given in Table 8.5 – a), b) – the logistic equation, c), d) – the Hénon map, e), f) - the evolutionary synthesized system

Most of common control methods were developed only for stabilization on real UPO with low energy costs. The question of energy costs and more precise stabilization will be included into the future research together with the development of better cost functions, a different AP data set, and performing of numerous simulations to obtain more results and produce better statistics, thus to confirm the robustness of this approach.

8.8 Results for p-4 Orbit – Oscillation between Four Points

Last simulations were carried out also for p-4 UPO when an oscillation between four points appears. The perturbed system notations for the logistic equation and the Hénon map can be found in (8.18) – (8.19). The synthesized control laws for the two selected systems are provided in Table 8.7. The values for p-4 UPO (oscillation between four points) of unperturbed chaotic systems based on the mathematical analysis of the system are depicted in Table 8.8. The tests were performed under the simple cost function, not in the blackbox mode. The relevant figures of the simulation output are given in Fig. 8.11.

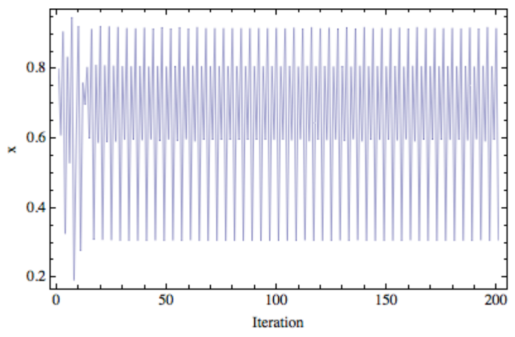
Table 8.7: Simulation results for chaotic systems and stabilization at p-4 UPO

Nr.	Control Law with estimated coefficients	CF Value	UPO Value	Figure
Logistic equation				
1	$F_n = \left(x_{n-5}^{x_{n-2}} + x_{n-2}\right) x_{n-2}^{\left(\frac{1}{x_n^{35.5805}} - x_{n-11}\right)}$	0.1139	0.30, 0.80, 0.6, 0.91	Fig. 8.11 a)
2	$F_n = x_{n-2}^{x_n^{34.6383}}$	0.1007	0.3, 0.8, 0.6, 0.9	Fig. 8.11 b)
Hénon map				
1	$F_n = -0.527409 x_{n-8} x_{n-7} x_{n-3} \left(x_{n-4} - x_n\right)$	0.0984	0.13, 1.45, -0.86, 0.89	Fig. 8.11 c)
2	$F_n = \frac{x_{n-7} x_{n-6} \left(10.0667 + x_n\right)}{59.4863 + \frac{-0.0174 x_{n-6} - 52.191}{\frac{x_n 5.9706}{x_{n-3}} + 39.4742 + x_{n-2}}}$	0.7095	0.13, 1.45, -0.86, 0.89	Fig. 8.11 d)

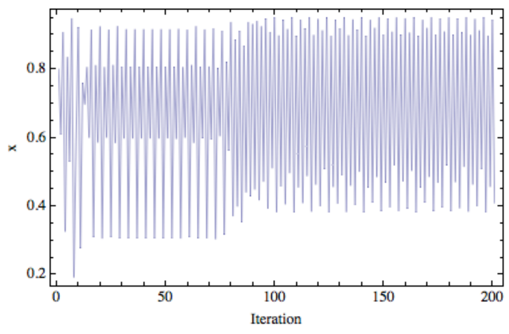
Table 8.8: The values for p-4 UPO (oscillation between four points)

Chaotic system	Values of p-2 UPO of unperturbed system
Logistic equation	x_F = 0.3038, 0.8037, 0.5995 and 0.9124
Hénon map	x_F = 0.139, 1.4495, -0.8594 and 0.8962

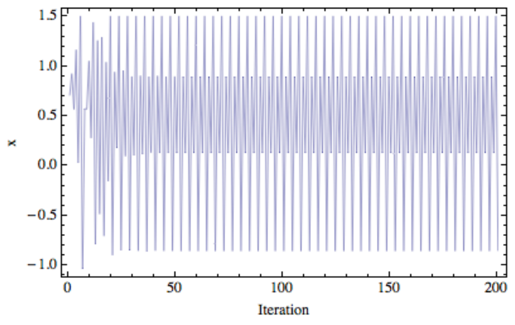
a)



b)



c)



d)

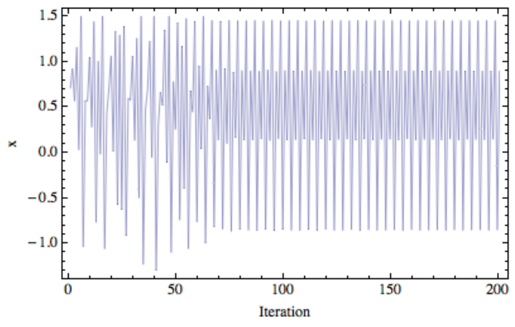


Fig. 8.11: Examples of results – stabilization of chaotic systems by means of control laws given in Table 8.7 – a), b) – the logistic equation, c), d) – the Hénon map

The presented simulation examples show two different results. First group has low CF values indicating precise but unfortunately slow stabilization and sometimes only temporary, together with a simple control law. The second group promises not very precise (as the higher CF values denote) but very fast stabilization and relatively complex notation of the chaotic controller. This phenomenon is caused by the design of CF which was borrowed from the research focused on the simpler cases. Satisfactory results were obtained for example for p-2 orbit.

8.9 Synthesis of Control Laws for Chaotic Systems – Conclusion

The area presented in the chapter 8 introduces the usage of analytic programming for the optimization of stabilization of selected chaotic systems.

Obtained results show that synthesized control laws provided better results than the original control method which served as an inspiration. This fact reinforces the argument that AP is able to solve these difficult problems and to produce a new synthesized control law in a symbolic way securing desired behavior of a chaotic system. Precise and fast stabilization gives weight to the argument that AP is a powerful symbolic regression tool which is able to strictly and precisely follow the rules given by the cost function and synthesize any symbolic formula. In the case of this research, it means to synthesize some kind of a feedback controller for a chaotic system.

The research never ends. The question of energy costs and more precise stabilization will be included into the future research together with the development of better cost functions, a different AP data set, and performing of numerous simulations to obtain more results and produce better statistics, thus to confirm the robustness of this approach.

Presented data and statistical comparison can be summarized as follows:

All simulations were performed at least 50x to obtain statistics. All cases have found the solution for the stabilization of chaotic systems. As presented results show, some cases give only temporary stabilization and some even stabilize the system on artificial UPOs. The number of cost function evaluations for 32 millions per one simulation means that the consumed time is really high. The future research will be supposed to search for time efficiency and decreasing of the simulation time.

The other points compare the design of cost functions. The simple evolutionary approach is easy to implement, it is very fast and gives satisfactory results. But the quality of results is restricted by the limitations of the mathematical formulas, control laws, models etc., for which the parameters are tuned by EA.

The Blackbox approach brings the advantage of avoidance of the mathematical analysis of chaotic systems but in this case an interesting phenomenon was discovered – stabilization on artificial UPOs. Since there was no information about the exact position of orbits in the chaotic attractor transferred into the evolutionary process and the cost function was designed to operate in the blackbox mode, the evolution found satisfactory behaviour but not on the precise values.

Nevertheless the proposed blackbox mode approach is very advantageous and simple to implement in the case of an unknown chaotic system or chaotic oscillations in any system because of its ability to control the chaotic system or oscillations without any previous demanding mathematical analysis, ie. without the knowledge of the exact UPOs position. It can be used as a powerful tool to promptly check the controllability of any new discrete chaotic system.

9 Synthesis of Pseudo Artificial Neural Networks

9.1 Pseudo ANN - Introduction

The interest in classification by means of some automatic process has been enlarged with the development of artificial neural networks (ANN). They can be used also for many other possible applications such as pattern recognition, prediction, control, signal filtering, approximation, etc. All artificial neural networks are based on a relation between inputs and output(s) that utilize mathematical transfer functions and optimized weights from a training process. The setting-up of layers, number of neurons in layers, estimating of suitable values of weights is a demanding procedure. On that account, pseudo neural networks that represent the novelty approach using symbolic regression with evolutionary computation is proposed in this chapter.

The evolutionary techniques have been recently commonly used for the synthesis of artificial neural networks but in a different manner than is presented here. One possibility is the usage of evolutionary algorithms for the optimization of weights to obtain a ANN training process with a small or no training error result. Some other approaches represent the special ways of encoding the structure of the ANN either into the individuals of evolutionary algorithms or into the tools such as Genetic Programming. But all of these methods still work with the classical terminology and separation of ANN to neurons and their transfer functions [23].

The proposed technique uses symbolic regression and is similar to the synthesis of the analytical form of the mathematical model between input(s) and output(s) in a training set used in neural networks. Therefore it is called Pseudo Neural Networks [62]. The proposed technique synthesizes the structure without a prior knowledge of transfer functions and inner potentials. It synthesizes the relation between inputs and output of a training set items used in neural networks so that the items of each group are correctly classified according to the rules for the cost function value.

The example of the relation between two inputs and one output can be shown in the mathematical form (9.1). It represents the case of only one neuron and a logistic sigmoid function as a transfer function.

$$y = \frac{1}{1 + e^{-(x_1 w_1 + x_2 w_2)}} \tag{9.1}$$

where y – output

x_1, x_2 – inputs

w_1, w_2 – weights.

The aim of the proposed technique is to find a similar relation to (9.1). This relation is completely synthesized by evolutionary symbolic regression – analytic programming.

9.2 Pseudo ANN - Problem Design

The classification tools are usually tested on an XOR problem (Fig. 9.1). This is the example of a non-linear separable problem, i.e. there is not possible to put a straight line (linear function) as a border between two classes – red and green dots. This chapter presents only the simulations for an XOR problem for two dimensions (two inputs) as shown in Fig. 9.1.

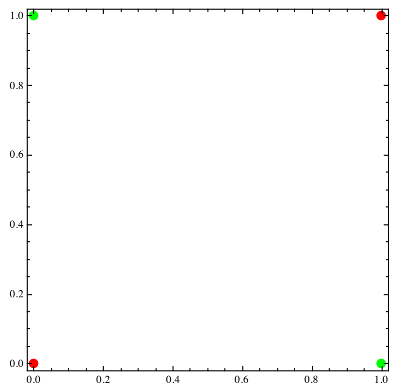


Fig. 9.1: An example of an XOR problem, first class - red dots in left bottom and right upper corner, second class - green dots in left upper and right bottom corner

This classification problem is binary. Therefore the idea from the binary transfer function was adopted into the rules for the cost function value. The result value from the synthesized notation is 1 if the value is higher than 0.5, respectively 0 in the opposite case. This is necessary to be able to evaluate the error between required values and obtained values.

The required values are following:

Red dots	–	input {0,0}	output 1
		input {1,1}	output 1
Green dots	–	input {0,1}	output 0
		input {1,0}	output 0

In the case of the performed simulations, 10^{-6} was used instead of absolute zero. During the synthesis of pseudo ANN this approach shows better results then in the case of absolute zero. Such values are sometimes recommended also for the usage of classical artificial neural networks to avoid e.g. division by zero.

9.3 Pseudo ANN - Results

For performed simulations, AP_{meta} version was used. The meta approach uses two evolutionary algorithms, one for the main AP process, here the SOMA algorithm and also DE, and the second for tuning parameters, here only DE was used. The settings of EA parameters for both processes were based on numerous performed experiments with AP_{meta} (Table 9.1 and Table 9.2).

Table 9.1: SOMA settings for AP

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

Table 9.2: DE settings for AP and meta-evolution

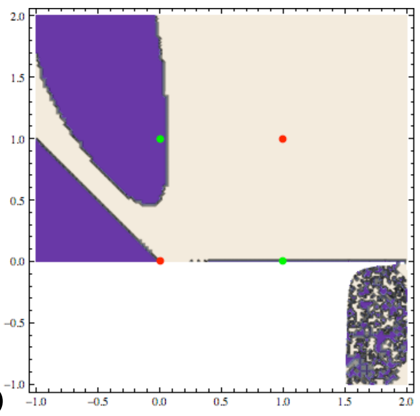
PopSize	40
F	0.8
CR	0.8
Generations	150
Max. CF Evaluations (CFE)	6000

The basic set of elementary functions for AP was inspired by the items contained in artificial neural nets:

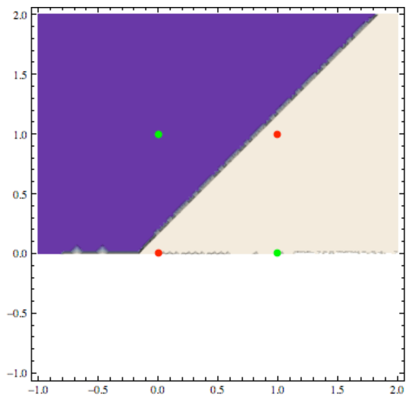
GFS2arg= +, -, /, *, ^, exp

GFS0arg= x_1, x_2, K

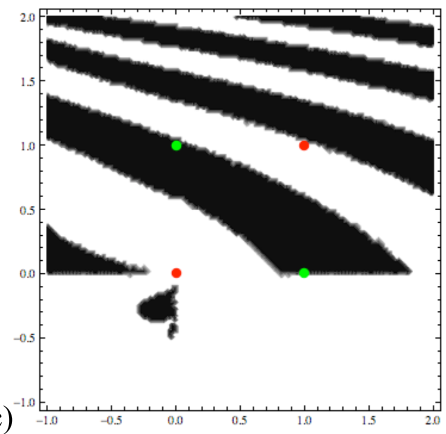
The performed simulations were successful and the following figure (Fig. 9.2) shows the found borders between the classes.



a)



b)



c)

Fig. 9.2: Examples of solutions

The following notation (9.2) is the solution for Fig. 9.2 c).

$$y = \exp\left(\begin{array}{l} 967.7328 + x1 + x1^{x2} + x2 + 2.836.10^{-193^{-687.33578 + \exp(x2) + x1}} * \\ * \exp\left(\exp\left(x1^{x2}\right)\right)\left(-x2\right)^{-687.336 + \exp(x2) + x1} x2 \end{array}\right) \tag{9.2}$$

where y – output

$x1, x2$ – inputs

All found solutions were successful. However it is necessary to discuss some critical points. It seems that green dots {1, 0} in Fig. 9.2 a) and b) belong to the other class. The violet colour is not visible much in that part but the green dot is still in the violet part. On the contrary, the red dot {0,0} seems to be in the violet part but they are on the edge. Fig. 9.2 c) shows clear groups but the stripes are not suitable for classification into the two groups. All these observed phenomena will be taken into consideration during the future testing and cost value rules development.

9.4 Pseudo ANN - Conclusion

This chapter dealt with a novel approach – pseudo neural networks. Within this approach, the classical optimization of the structure or weights was not performed. The proposed technique is based on symbolic regression with evolutionary computation. It synthesizes a whole structure in a symbolic form without a prior knowledge of the ANN structure or transfer functions. It means that the relation between inputs and output(s) is synthesized. As can be seen in the result section, such approach is promising. For further tests, some observed critical points have to be taken into consideration. Future plans will be focused on the better cost function design and also on performing numerous simulations with more difficult tasks than the presented one.

10 Steganalysis by Means of ANN

10.1 Motivation behind ANN Usage for Steganalysis

Steganalysis means the techniques used to discover the covert communication in transferred data files. The very basic test, which is called the visual attack, uses human senses such as sight for discovering irregularities in a represented medium. Such test is limited by human individuality. Two people always have different sensitivity to an examined object. Another steganalysis tool, called the structural attack, is similar to the visual attack. It is computer based and focuses on discovering irregularities in the data structure of a cover medium. Every computer data file has its own characteristic structure. Embedding a message will leave a trace in such a structure. The difference between a stego file (with hidden content) and a cover file (an empty file without a message) is given by the quality of a steganographic tool. The statistical attack has more scientific approach than two above mentioned one and it is more complicated. In general, statistics is used for determining the level of randomness, entropy of the redundant data or colour frequencies occurrence in stego files.

The statistical steganalysis has been deeply described by many researchers, e.g. by Niel Provos and Peter Honeyman [71] or [70]. Andreas Westfeld together with Andreas Pfitzmann introduced their Chi-square statistical attack [100]. Jessica Fridrich and her teams published many research papers on the JPEG steganalysis [27], [26], [28] on conventional mathematical–statistical basis. There were more people working on various steganalysis techniques. All above mentioned people have been the most dedicated to this field.

The techniques described in the above mentioned papers have been powerful and functional. They have only one disadvantage. They suffer from the false positive classification. The reason is simple, the steganography classification is a mathematically complicated process and input steganograms (files with a message inside) are strongly diversified.

The approach to the steganalysis proposed in the research in the thesis is based on artificial intelligence, mainly artificial neural networks (ANNs). ANNs are known as a strong tool for solving difficult classification tasks. ANNs have been successfully implemented in many other projects focused on classification.

A big challenge for classification based on artificial intelligence was to deal with the double compression of JPEG – a file that was the main source of false positive classification. This research is focused on pin-pointing the stego image classification by a new sampling methodology and the reduction of false positive classification by means of a trained ANN classifier on pairs of cover-stego samples. The research was published for example in [58], [41], [35].

10.2 Steganalysis - Introduction

With the spread of computers into human lives the need for security has arisen. The field that covers the development of the impossibility of secret message decoding is called Cryptography [29]. The other method connected with security hides transmitted information because of the distraction of attention from messages which contain very sensitive data. This method is called Steganography [7], [77], [51]. Hiding information is both useful and dangerous. Therefore it is important to develop tools and methods for a forensic analysis to prevent the abuse of hiding methods for criminal purposes.

Steganography is the art of hiding communication by embedding secret messages into innocent file content, mainly into multimedia files. The carrier files in steganography are called the “cover images”, while files with hidden information embedded by some steganography technique are called the “stego files”.

Steganography can be misused. Unwanted leaking of “know-how” or other confidential content is in the first place. An example of such a process can be described as follows: imagine a company with employees and secret information, e.g. a database with secret data that is located on a database server accessible from an employee’s terminal. If an employee decides to steal the confidential data and uses a regular email to do so (as seen in Fig. 10.1) his/her action is revealed almost immediately because he/she has a monitored services email account. When the internal data is saved into a regular email and sent by the department email account to a home computer then the email is checked.

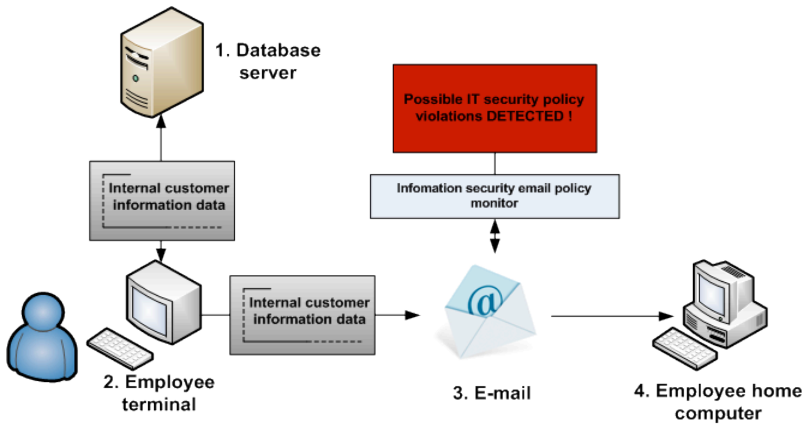

1. Database server
Possible IT security policy violations DETECTED !
Internal customer information data
Infomation security email policy monitor
Internal customer information data
2. Employee terminal
3. E-mail
4. Employee home computer

Fig. 10.1: Message transport through a plain text email

There is an email monitor between the terminal and the email gateway which scans all-outgoing emails for viruses as well as its body and attachments for any internal business information. In the described case, the security monitor detects that the email attachment contains sensitive data. The security department is immediately informed about this incident and the employee is charged for the information fraud.

In Fig. 10.2 a similar scenario is shown. If the employee from this case decides to steal confidential data from the employer's database it is not a difficult task with the use of a steganography tool. Steganography helps with the secure transfer of secret messages compared to cryptography, which is strong in the usage of the key for coding of messages. Steganography codes a message into images, a video file or data stream. If a human eye sees a picture with steganographic content, it would not recognize the secret message inside. This is the main aim – to hide information itself.

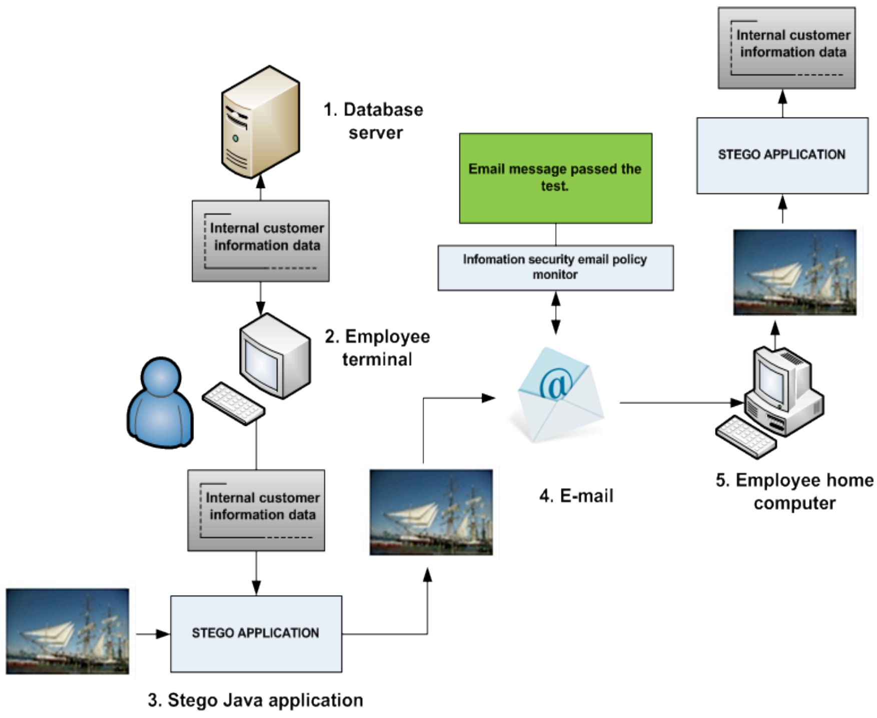

Internal customer information data
1. Database server
Email message passed the test.
STEGO APPLICATION
Internal customer information data
Infomation security email policy monitor
2. Employee terminal
Internal customer information data
4. E-mail
5. Employee home computer
STEGO APPLICATION
3. Stego Java application

Fig. 10.2: Message transport with steganography

The whole second scenario with a steganography tool is very easy. The employee can use e.g. a Java application downloaded from the Internet because of the company rules and forbidden instalations of any application on the employee's terminal or computer and Java is consider to be used for multi platforms. Then he/she has to prepare images in the JPEG format and use a steganography Java application that embeds a text file containing internal business information into the image files. After that the images are sent to his/her personal email account. The outgoing server does not recognize any danger of an information leak.

The main goal of steganography is to not attract any attention. Therefore it is necessary to have a method for its detection because it is vulnerable. The research deals with this particular phenomenon – the method of detection by means of Artificial Neural Networks (ANN) [31], [98], [30], [22].

The steganalysis techniques employ different ways of detection such as statistical methods, searching for specific signature of a steganography method [51]. Also the methods of artificial intelligence (AI) were used. The research field within AI is connected with a support vector machine (SVM) [74], [13]. Some researches use ANN as the case of this research. SVM and ANN are similar tools but SVM is usually used for lineary separable problems. The approach in the thesis is different in the design of training sets. This thesis does not use pixel differences or joint features of discrete wavelet transform and polynomial fitting errors or reversible data [77], [74], [13]. The approach uses Huffman coding of bit word lengths extracted from discrete cosine transformation coefficients.

10.3 ANN Training Sets

Training sets are necessary for the correct running of artificial neural nets. Within all experiments supervised, ANNs were used. In this case, each item of a training set has an output value which says if the image is with or without any hidden content.

The used training set consists of numbers obtained from Huffman coding [8]. Huffman coding was applied to adjustments and modifications of the basic 2183 images that were acquired for testing purposes from three digital cameras (Sony DSC-P93, Olympus SP550UZ, Pentax K10D) in fine or superfine quality. The lowest image resolution for this basic group is more then 2560x1600, the average picture resolution is 3529x2458 pixels and the maximum picture resolution is 3872x2592 pixels with the average file size of 2616.6 kB and the maximum file size of 4403.2 kB. The images from the basic set were resized to several sizes as described below.

10.3.1 Cover Images

Cover samples, which are images without any hidden information, were created by resizing the original digital images with the Linux tool ImageMagick [36] into different file resolutions estimated by their common appearance on the Internet. The entire image pool contains almost 22 000 images.

The list of all image resolutions used for the test group:

800x600, 1024x768, 1280x1024, 1440x900, 1680x1050, 1920x1440, 2560x1600 and one special group containing original files with resolution higher then 2560x1600 pixels.

10.3.2 Stego Images

All samples from the cover image pool were used for Outguess, Steghide and the PQ algorithm. Due to the problems with the F5 java implementation, the input cover file pool was reduced only to images up to the maximum resolution of 1680 x 1050 pixels in this case.

A secret message and an encryption password were generated by the Linux pseudorandom number generator that collects environmental noise from device drivers and other sources into the entropy pool. The amount of hidden information was set up by the measurement of common length of short messages.

The list of all message lengths used for the stego test samples:

5, 10, 15, 30, 75, 150, 300 and 600 Bytes.

10.3.3 Huffman Coding

Huffman coding was designed by David Huffman in 1952. This method takes symbols represented e.g. by values of discrete cosine transformation (which is one of the methods how to present information such as colour, brightness etc. in pictures), and codes it into a changeable length code so that according to statistics the shortest bit representation is assigned to the symbols with the most frequent appearance [8]. It has two very important properties – it is a code with minimal length and prefix code that means that it can be decoded uniquely. On the other hand, the disadvantage is that we have to know the appearance of each symbol a priori. In the case of pictures, it is possible to work with estimation, which will be edited during the compression. Fig. 10.3, Fig. 10.4, Table 10.1 and Table 10.2 show the differences between the cover and stego images in DC or AC (direct or alternating part) class. The pictures show the number of each bit word in the image.

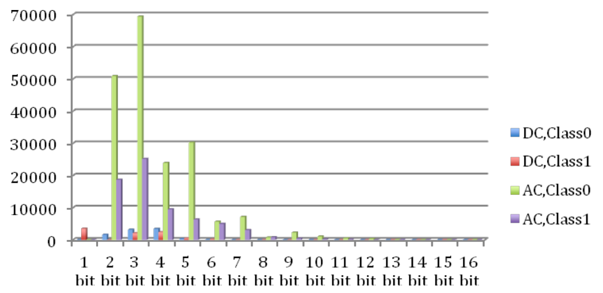


Fig. 10.3: Huffman coding histogram – cover image (clear pictures)

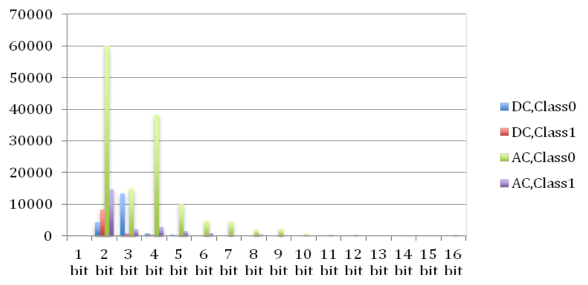


Fig. 10.4: Huffman coding histogram – stego image (coded picture)

Table 10.1: Huffman coding histogram – cover image

Length of the word [bits]	DC, Class 0	DC, Class 1	AC, Class 0	AC, Class 1
1	0	3504	0	0
2	1623	0	50871	18704
3	3178	2060	69370	25155
4	3435	2371	23902	9522
5	342	527	30216	6311
6	86	170	5642	4968
7	0	31	7102	3032
8	0	1	771	805
9	0	0	2285	425
10	0	0	1022	204
11	0	0	522	115
12	0	0	345	40
13	0	0	74	49
14	0	0	20	8
15	0	0	0	6
16	0	0	50	13

Table 10.2: Huffman coding histogram – stego image

Length of the word [bits]	DC, Class 0	DC, Class 1	AC, Class 0	AC, Class 1
1	0	0	0	0
2	4433	8312	59998	14595
3	13283	730	14904	2224
4	906	343	38276	2755
5	444	181	10142	1444
6	86	10	4742	925
7	0	0	4680	89
8	0	0	1943	428
9	0	0	2149	77
10	0	0	667	42
11	0	0	444	12
12	0	0	316	0
13	0	0	0	0
14	0	0	0	1
15	0	0	73	0
16	0	0	477	0

For the concept of the main principle, please refer to the following picture (Fig. 10.5). Each bit word can stand as a brick in the wall. It is possible to get two equally big walls but each of them will be assembled from different bricks and brick sizes. These two walls are of the same size but with a different structure (a different set of bricks, some bricks appear more often then others). By the same analogy, the differences in cover and stego files can be percieved. The objective is to compare the different bit word length and different sizes of bricks in the walls for cover and images affected by steganography.

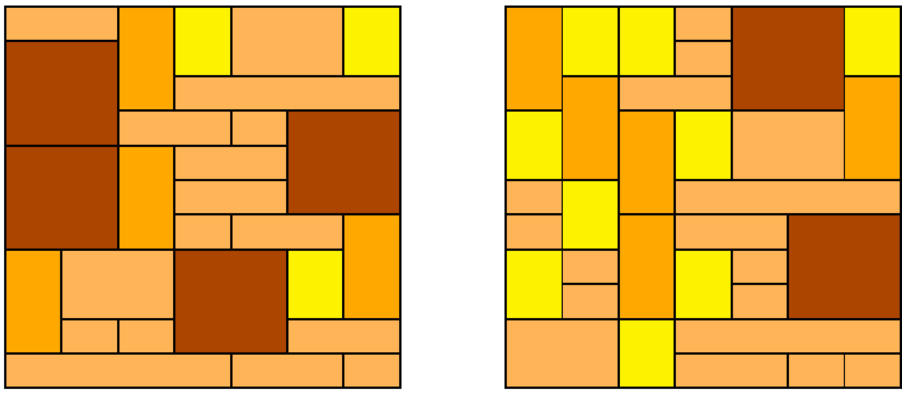


Fig. 10.5: Illustration of Huffman coding histogram – left) cover image, right) stego image

The main goal of steganography is to not attract attention. Stego images appear as ordinary pictures taken by a digital camera. But there are significant changes in the structure of the stego images. The changes in the JPEG structure are relevant and used in the case of the presented research for correct training of an artificial neural network.

10.3.4 Examples of Training Set Items

Values obtained from Huffman coding were transferred into a training set, i.e. all four columns from each table (Table 10.1 and Table 10.2) were joined to create a training vector. An example follows.

{0,2178,49642,11918,7758,3614,2113,1328,181,0,0,0,0,0,0,0,0,37824,17026,9608,6323,4486,2771,692,2,0,0,0,0,0,0,0,0,1184565,266816,406818,225770,85887,84320,39638,27400,14811,6889,1516,0,0,105,5231,0,295156,155514,135282,76214,48989,12495,16659,9154,3609,1601,94,0,868,625,208}

The vector contains 64 values – real numbers of bit words which represent inputs into ANN. The ANN output neuron is only one and has the output value 0 or 1 (sigmoid function) or -1 and 1 (saturated linear function). The threshold for determining the output class is in the middle of the interval – 0.5 in the case of 0 and 1 or 0 in the latter case.

To use numbers in the interval <0,1> instead of the real quantities turned out as an inappropriate way. There are too big differences between each position. On that account, small numbers are not visible if expressed as a percentage in the interval <0,1>. They would approximate zero. The result is that the input information into the neural network is misrepresented.

10.4 Brief Outlook of Steganography Tools

10.4.1 Outguess

OutGuess is a universal steganography tool which is able to insert hidden information into redundant bits of input data [90]. The type of input data is not important for OutGuess at all because this software uses specific drivers for specific graphic formats that extract redundant bits and writes these bits back after they are changed. The version, which was used for the simulations, is able to work with the JPEG and PNG formats. JPEG pictures were used in this thesis. OutGuess is available under the Berkeley Software Distribution (BSD) license. OutGuess is hard to detect by means of statistics calculation based on the frequency analysis. The results of the statistical analysis are not able to reveal steganography content because OutGuess finds out the maximal length of the message before the picture is inserted. This causes that the resulting image is not changed from the point of view of frequency analysis as was described in [69].

10.4.2 Steghide

Steghide is steganography software that is able to hide data in various kinds of image and audio files. The colour - respectively sample-frequencies are not changed thus make the embedding resistant to the first-order statistical tests. Steghide uses a graph theory approach to steganography. The embedding algorithm works roughly as follows: At first, secret data is compressed and encrypted. Then a sequence of pixel positions in the cover file is created based on a pseudorandom number generator initialized with the passphrase (the secret data will be embedded in the pixels at these positions). The positions that do not need to be changed (because they have already contained the correct value by chance) are sorted out. Then a graph-theory matching algorithm finds pairs of positions so that exchanging their values has the effect to embedding of the corresponding part of the secret data. If the algorithm cannot find any more such pairs, all exchanges are actually performed. The pixels at the remaining positions (the positions that are not the part of such a pair) are also modified to contain the embedded data (but this is done by overwriting them, not by exchanging them with other pixels). The fact that (most of) the embedding is done by exchanging pixel values implies that the first-order statistics (i.e. how many times a colour occurs in the picture) is not changed. For audio files the algorithm is the same, except for that audio samples are used instead of pixels. The default encryption algorithm is the Rijndael with a key size of 128 bits (which is AES - the advanced encryption standard) in the cipher block-chaining mode [32].

10.4.3 F5 Algorithm (CipherAWT)

The F5 steganographic algorithm was introduced by German researchers Pfitzmann and Westfeld in 2001 [101]. The goal of their research was to develop concepts and a practical embedding method for JPEG images that would provide high steganographic capacity without sacrificing security. Guided by their χ2 attack, they challenged the paradigm of replacing bits of information in the cover-image with the secret message while proposing a different paradigm of incrementing image components to embed message bits. Instead of replacing the least significant bits (LSBs) of quantized discrete cosine transform (DCT) coefficients with the message bits, the absolute value of the coefficient is decreased by one. The F5 authors argue that this type of embedding cannot be detected using their χ2 statistical attack.

The F5 algorithm embeds message bits into randomly chosen DCT coefficients and employs matrix embedding that minimizes the necessary number of changes to embed a message of certain length.

The F5 algorithm modifies the histogram of DCT coefficients, but some crucial characteristics of the histogram are preserved, such as its monotonicity and monotonicity of increments. The F5 algorithm cannot be detected using the χ2 attack because the embedding is not based on bit-replacement or exchanging any fixed pairs of values [27].

10.4.4 PQ Algorithm

Perturbed quantization (PQ) steganography [25] is a quite successful data hiding approach which current steganalysis methods fail to work for [90]. In other words, PQ does not leave any traces in the form that the current steganalysis methods can catch. However, linear dependency between image rows and/or columns in the spatial domain is affected by PQ embedding due to random modifications on discrete cosine transform (DCT) coefficients' parities during data hiding.

In PQ steganography, the cover object is applied an information reducing operation that involves quantization such as lossy compression, resizing, or A/D conversion before data embedding. The quantization is perturbed according to a random key for data embedding, therefore called "perturbed quantization". PQ steganography, which uses JPEG compression for information reducing operation, is different from their DCT based counterparts. Since message bits are encoded by changing DCT parities after quantization, the cover image can be thought of just as a recompressed input image. To achieve high embedding rates, recompression is realized by doubling the input quantization table with the assumption that recompression of cover JPEG images does not draw any suspicion because of its wide usage in digital photography [25]. Since the original cover image is recompressed via embedding operation, its compressed version should be considered as "stego" instead of the original image.

10.5 Results

All experiments were performed with a supervised feed forward net, which uses the Levenberg-Marquardt training algorithm.

The testing of the proposed approach was performed with different settings of neurons in one hidden layer net (number from 1 to 20) and 9 combinations of transfer functions (logistic sigmoid, saturated linear and hyperbolic tangent for hidden layer and output neuron). The tests were carried out for each stego algorithm individually.

Experiments with only one general neural network for all stego algorithms and even their detection have not been successful yet.

The whole data set was divided into training and testing sets. For training 14400 cover items and 135000 stego images were used. The exact number of testing items is written in Table 10.3 for all testing stego algorithms.

All simulations used sixty four input neurons (obtained from Huffman coding) and one output neuron that classifies the training item into the class of cover or stego images. The following table (Table 10.3) shows the best results from the performed experiments. These tables contain information about the number of cover and stego images and misclassified items (= an item should contain stego content and ANN output was a group of cover images and viceversa). The last two rows represent the total error in the whole set of cover and stego images.

Table 10.3: Results of testing success for four steganographic tools

Type of algorithm	**OutGuess**	**Steghide**	**F5 algorithm**	**PQ algorithm**
Nr. of hidden neurons	12	1	15	1
type of function in inner layer	logistic sigmoid	logistic sigmoid	saturated linear	logistic sigmoid
type of function in output layer	saturated linear	saturated linear	hyperbolic tangent	saturated linear
Cover total	5246	5246	9746	22711
Cover errors	2	32	8	1
Cover % error	0.0381	0.61	0.0821	0.0044
Cover % success	99.9619	99.39	99.9179	99.9956
Stego total	135 891	142 772	77 314	126 599
Stego errors	0	3248	24	733
Stego % error	0	2.275	0.031	0.6106
Stego % success	100	97.725	99.969	99.3894
Total % error	0.0014	2.216	0.0368	0.5184
Total % success	99.9986	97.784	99.9632	99.4816

Based on the presented results, it can be stated that the total error in experiments was under 1 % for all algorithms except for the Steghide algorithm. Compared to the earlier research [34], [60], [59] where the total error was almost zero, the testing set consists of more items with more message payload etc. This probably caused the worsening of the results. As the ANN output is a mathematical function – mathematical dependency of inputs, it is possible to extract it into the equation. This kind of equation can be used in stego detector software without the knowledge of ANN structures. Such a mathematical equation needs only suitable inputs, which are extracted from Huffman coding, and the output is obtained. The notations of these equations are very complex they are therefore not presented here.

10.6 Steganalysis by Means of ANN - Conclusion

This research introduces a method of steganalysis by means of neural networks. The novelty is in the training set design. The training set consists of 64 inputs obtained from Huffman coding extracted from discrete cosine transformation coefficients and counting of bit words of the same lengths. During the simulations files with embedded message by means of 4 steganographic algorithms – OutGuess, Steghide, the PQ and F5 algorithm were tested. ANNs were able to detect the cover and stego groups with less than 1 % error. The exception was the case of Steghide where the error was around 2 %. According to the presented results, the proposed technique was successful.

The optimization of consumed time in the broad area of computations is very important and required in all fields. The research in the area of steganalysis by means of ANN covers also datamining techniques used in the simulations for less time consuming training. This was done through the ANN structure optimization. The input dimension was reduced from sixty four to three neurons. Therefore also number of weights, which were necessary to be trained, was reduced and the ANN training phase was not so time consuming.

The aim for the future is to develop a steganalysis detector. The detector might be part of the outgoing email servers. This device will not decode the message itself, since this research is focused only on the recognition of the use of steganography tools (encoder) on analysed JPEG pictures.

11 Optimal Modelling of Dynamic Flight

11.1 Modelling of Flight - Introduction

This chapter is focused on how to solve the approximation of a real dynamic system by a suitable analytic solution. There is a dynamic flight model which uses several classes of large sets of aerodynamic lift, drag, speed, force, balance and mass data [20], [96], [95], [94], [93]. The aircraft company and industry is able to work with differential equations to obtain data. However, this takes a lot of time to obtain some results. Therefore a need to find an analytical solution has arisen. Because the input data and the "response", ie. output data, are known, there are possibilities how to find a suitable dependence between this data. One of the suitable techniques are artificial neural networks (ANN) [31], [98], [30], [22].

11.2 Aircraft and Parameters

The following picture (Fig. 11.1) shows an example of an aircraft and its controls and their influence on manoeuvrability and stability.

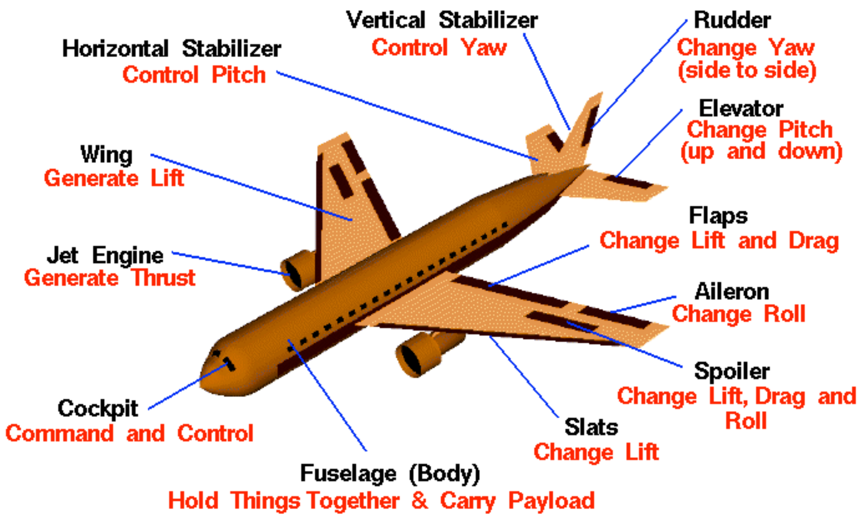


Fig. 11.1: A model of an aircraft

The model, which is considered here, is an airplane dynamic behavioral mockup describing the aircraft body movements as reaction to flight conditions (altitude, mass, speed) and control positions. The important parts are:

Ailerons which control the roll of an airplane.

Elevators control the pitch of an airplane, ie. aircraft nose up or down.

Rudder controls the yaw, ie. side to side motion.

Roll is a banking turn of an airplane. Ailerons together with rudder cause the flight-heading change.

Pitch is the up and down motion of an airplane. To climb, the elevators have to move up, which pushes the nose up too.

Yaw is the side-to-side motion of an airplane. To cause an airplane to yaw to the right, the rudder is deployed to the right. This pushes the tail to the left and the nose to the right [20], [96], [95], [94], [93].

For the purposes of this simulation, following flight conditions such as altitude, mass, speed, side wing were "frozen" as follows. For the future, these parameters will be considered as input values too.

- Mass [kg] : 3000.000
- Speed TAS [m/s]: 200.000
- Altitude [m]: 3000.000
- Side Wind [m/s] : 0.000

The rest of the monitored variables were used as inputs to the neural network:

- PLA gas lever [1]
- aileron deflection [rad]
- elevator deflection [rad]
- rudder deflection [rad]
- angle of attack [rad]
- beta slip angle [rad]
- gamma roll angle [rad].

And in the case of modelling, we were interested only in three outputs out of the seven:

- alpha derivative – pitch

- beta derivative – yaw
- acceleration [m/s^2].

There is numerical computational software in C++ language which can produce the output data on the basis of given input data. It utilizes several databases of design data for lift, drag, engines thrust, centre of gravity and aerodynamic centre equations. Based on the input, the model calculates a row of polynomials to approximate the intermediate values. Then the approximated inputs are inserted into system of nonlinear differential equations and integrated with the modified Runge-Kutta. The integration in the complex plane has been replaced with the quaternion's method of calculation. But the computation is quite slow because of numerous derivation procedures inside the programme. Thus neural networks were used to produce an analytic formula which can be easily and quickly recalculated. This advantage of speed is absolutely necessary for example for autopilot systems in the aircraft industry.

11.3 Settings and Aim of Simulations

During the simulations one hidden layer net was used with 4 neurons in the hidden layer with a sigmoid transfer function. An output transfer function was used linear function. The feedforward ANN with the Levenberg-Marquardt training algorithm was used.

The seven inputs into the neural networks were as follows (in brackets the notation in the Fig. 11.2): PLA gas lever [l], aileron deflection [rad] (AIL deflection), elevator deflection [rad] (ELE deflection), rudder deflection [rad] (RUD deflection), angle of attack [rad] (ATT angle), beta slip angle [rad] (SID slip angle) and gamma roll angle [rad] (BNK angle).

The required outputs were only three: alpha derivative – pitch, beta derivative – yaw, acceleration [m/s^2].

From the C++ language time consuming model 76 samples were obtained with the above inputs and outputs which serve as the training set. The required dependency can be seen in (Fig. 11.2).

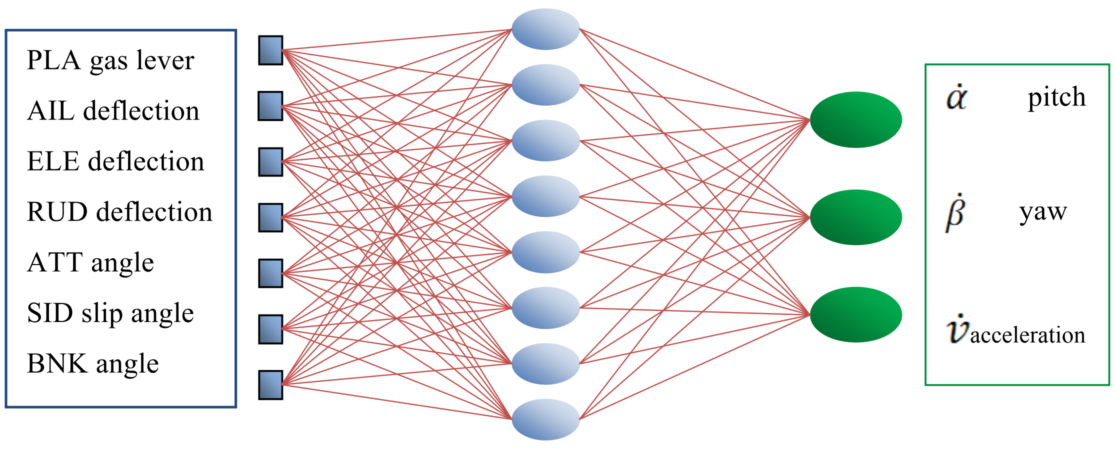


Fig. 11.2: A model of a neural network with one hidden layer and its “real” inputs and outputs

11.4 Modelling of Dynamic Flight - Results

For calculations 20 training iterations were set up. The following picture (Fig. 11.3) shows the evolution of root mean square error (RMSE) during the training. At the end of the training it was under 0.01 which is close to zero thus it can be stated that the neural network is trained well. Also the following graphs (Fig. 11.4 - Fig. 11.6) are the proof of this statement. They show the output data from each output neuron compared to the required ones. On the x axis the order of input data in the training set is shown and on the y axis is the corresponding output value for each neuron (output parameter).

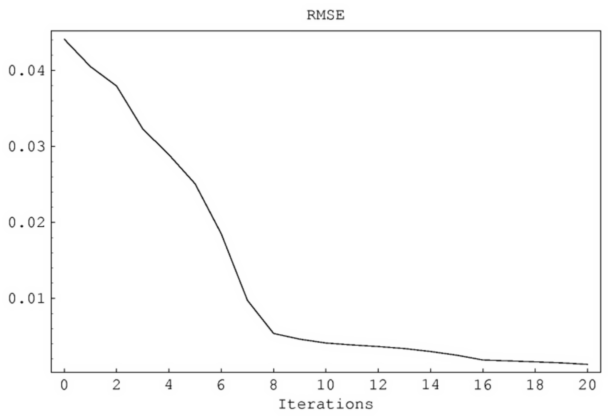


Fig. 11.3: RMSE dependent on training epochs (iterations)

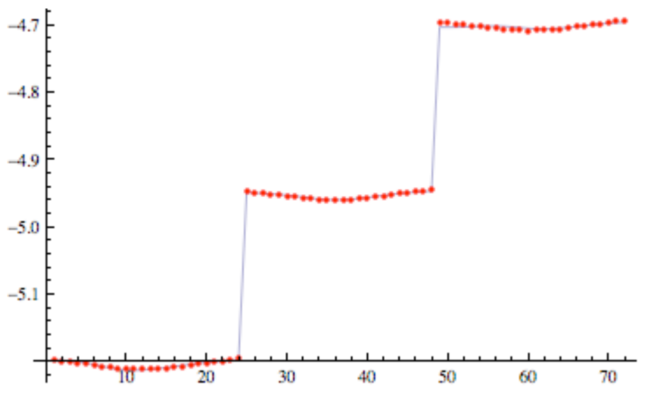


Fig. 11.4: The first output parameter – Alfa derivative – proportional to the pitching moment on the y axis dependent on the line in the training set, ● red dots = original data, ● blue line = fitted function

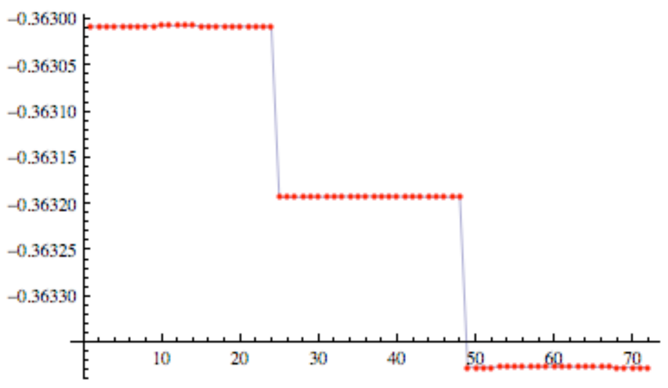


Fig. 11.5: The second output parameter – Beta derivative – proportional to the yawing moment on the y axis dependent on the line in the training set, ● red dots = original data, ● blue line = fitted function

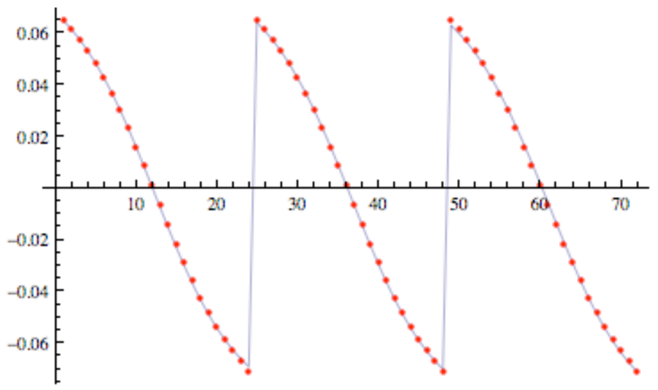


Fig. 11.6: The third output parameter – Velocity derivative – corresponding to the acceleration on the y axis dependent on the line in the training set, ● red dots = original data, ● blue line = fitted function

As can be seen from the above figures the approximation was very precise. Therefore there are the notations (11.1) – (11.3) for each output parameter (alpha derivative – pitch, beta derivative – yaw, velocity derivative - acceleration) which are based on the 7 inputs: aa - PLA gas lever [l], bb - aileron deflection [rad], cc - elevator deflection [rad], dd - rudder deflection [rad], ee - angle of attack [rad], ff - beta slip angle [rad] and gg - gamma roll angle [rad].

alpha derivative =

$$-4.39214-\frac{0.0260493}{1+e^{-2.9622+3.28284\,aa-14.174\,bb-13.0753\,cc-15.7637\,dd-10.0301\,ee+8.68923\,ff-18.1451\,gg}}-\frac{0.0486773}{1+e^{-0.0904559-0.256359\,aa+5.37631\,bb+9.99259\,cc+25.95\,dd+6.17531\,ee+6.45433\,ff-10.7697\,gg}}-\frac{1.02526}{1+e^{-2.00116+3.57053\,aa-0.0170792\,bb-0.820911\,cc+8.97494\,dd+11.5405\,ee-10.5349\,ff-2.98128\,gg}}-\frac{0.0114783}{1+e^{0.280221-6.21559\,aa-11.8399\,bb-62.5105\,cc-25.9518\,dd-34.0904\,ee-40.285\,ff+8.15001\,gg}} \qquad (11.1)$$

beta derivative =

$$-0.363574+\frac{2.10265\times10^{-6}}{1+e^{-2.9622+3.28284\,aa-14.174\,bb-13.0753\,cc-15.7637\,dd-10.0301\,ee+8.68923\,ff-18.1451\,gg}}+\frac{6.03757\times10^{-6}}{1+e^{-0.0904559-0.256359\,aa+5.37631\,bb+9.99259\,cc+25.95\,dd+6.17531\,ee+6.45433\,ff-10.7697\,gg}}+\frac{0.000753962}{1+e^{-2.00116+3.57053\,aa-0.0170792\,bb-0.820911\,cc+8.97494\,dd+11.5405\,ee-10.5349\,ff-2.98128\,gg}}+\frac{9.04914\times10^{-7}}{1+e^{0.280221-6.21559\,aa-11.8399\,bb-62.5105\,cc-25.9518\,dd-34.0904\,ee-40.285\,ff+8.15001\,gg}} \qquad (11.2)$$

acceleration =

$$-0.0947686+\frac{0.00245839}{1+e^{-2.9622+3.28284\,aa-14.174\,bb-13.0753\,cc-15.7637\,dd-10.0301\,ee+8.68923\,ff-18.1451\,gg}}+\frac{0.172342}{1+e^{-0.0904559-0.256359\,aa+5.37631\,bb+9.99259\,cc+25.95\,dd+6.17531\,ee+6.45433\,ff-10.7697\,gg}}+\frac{0.00878507}{1+e^{-2.00116+3.57053\,aa-0.0170792\,bb-0.820911\,cc+8.97494\,dd+11.5405\,ee-10.5349\,ff-2.98128\,gg}}-\frac{0.00028283}{1+e^{0.280221-6.21559\,aa-11.8399\,bb-62.5105\,cc-25.9518\,dd-34.0904\,ee-40.285\,ff+8.15001\,gg}} \qquad (11.3)$$

11.5 Optimal Modelling of a Dynamic Flight - Conclusion

As this chapter shows neural networks can be used as an approximation tool in the search for the quick model computational analysis/simulation/response on such difficult problems that are represented with nonlinear systems of differential equations with partially discontinuous functional dependencies.

The future research will be focused on the modelling of the complex behaviour of the airplane, handling the obtained models themselves such as a deep analysis of the model behavior, optimization, real time control etc., where the speed achieved by the obtained analytical solution will assume high importance.

12 Chaotic Pseudorandom Number Generator in Algorithm PSO

This chapter deals with the use of a chaotic system as a pseudorandom number generator. The idea came from studying soft computing methods. More or less, all are inspired by nature. Therefore a question arises – will two combined methods inspired by nature produce better results? In this case, deterministic chaos and evolutionary computation are the two used methods. An evolutionary algorithm is used for optimization as usual and deterministic chaos is used as a pseudorandom number generator inside the algorithm when needed.

Recently some studies have indicated that using chaotic number generators may improve the performance of evolutionary optimization algorithms on such tasks as a PID controller design [9] or fuzzy modelling of an experimental thermal-vacuum system [1]. This study is focused on the investigation on the performance of the PSO algorithm with the implemented chaotic Lozi map as a pseudorandom number generator. The idea was published not only for PSO but also for DE [80], [80], [79], [67].

A chaos driven pseudorandom number generator is used in the main PSO formula (4.3) that determines new “velocity” and thus the position of each particle in the next generation (or migration cycle). The parameter Rand, i.e. a random number from the interval <0,1> is replaced with a chaotic generator (in this demonstration, by the use of the Lozi map – chapter 8.5.4) within the Chaos PSO algorithm. The generator selects the adapted value from each position of a huge array (from 500 000 to 1 000 000 values) generated by means of standard time iteration of the Lozi map. The chaotic values can be both positive and negative, thus the absolute value is then applied to all positions. To obtain a number from the interval <0,1> all values have to be divided by the highest value from the array. When a PSO works the index to the array is increased and an unused value on the indexed position is inserted instead of classical computer generated random number all the time.

12.1 Chaotic Pseudorandom Number Generator – Experiment Design

To compare the impact of using the Lozi map as a chaotic pseudorandom number generator, performance tests were performed for both PSO with chaotic and non-chaotic random number generator. The classic version of PSO with the inertia weight modification is labelled PSO Weight. The proposed novel PSO enhanced by the Lozi map with inertia weight is labelled PSO Lozi. As an algorithm for comparison the DERand1Bin strategy of differential evolution (DE) was selected.

Basic PSO control parameters were set based on previous experiments and literature [17], [16], [1] as follows:

Population size: 30, 50, 75, 100, 150, 200, 300, 400
Iterations / generations: 10 * dimension
w_{start}: 0.9
w_{end}: 0.4
Dimension: 2, 5, 10, 20, 40

The algorithms were tested on 4 different benchmark functions. For the statistical reasons, optimization for each dimension value was repeated 30 times.

12.2 Benchmark Functions

Following chapters contain the equations and the function minimums in the n-dimensional space where Dim means the number of arguments (dimension of the problem).

12.2.1 First De Jong

The First De Jong function is given by (12.1).

$$f(x)=\sum_{\mathrm{i=1}}^{\mathrm{Dim}} x_i^2 \tag{12.1}$$

Function minimum:

Position for E$_\mathrm{n}$: $(x_1,x_2...x_n)$ = E$_\mathrm{n}$: $(x_1,x_2...x_n)$ = (0,0,…, 0)

Value for E$_\mathrm{n}$: $y = 0$

12.2.2 Second De Jong

The Second De Jong function is given by (12.2).

$$f(x)=\sum_{i=1}^{Dim-1}\left(100\left(\mathrm{x}_i^2\text{-}\mathrm{x}_{i+1}\right)^2+\left(1\text{-}\mathrm{x}_i\right)^2\right) \tag{12.2}$$

Function minimum:

Position for E$_\mathrm{n}$: $(x_1,x_2...x_n)$ = (1,1,…,1)

Value for E$_\mathrm{n}$: $y = 0$

12.2.3 Rastrigin

The Rastrigin function is given by (12.3).

$$f(x)=10\sum_{i=1}^{Dim}\left(x_i^2-10\cos\left(2\pi x_i\right)\right) \tag{12.3}$$

Function minimum:

Position for E$_\mathrm{n}$: $(x_1,x_2...x_n)$ = (0,0,…,0)

Value for E$_\mathrm{n}$: $y = 0$

12.2.4 Schwefel

The Schwefel function is given by (12.4).

$$f(x)=\sum_{i=1}^{Dim}-\left(x_i\sin\left(\sqrt{|x_i|}\right)\right) \tag{12.4}$$

Function minimum:

Position for E_n: $(x_1,x_2...x_n)$ = (420.969, 420.969,…, 420.969)

Value for E_n: y = -418.983 * dimension

12.3 Chaotic Pseudorandom Number Generator - Results and Analysis

The results of experiments and brief commentary on these results are in this section. The Following tables (Table 12.1 - Table 12.8) contain the best, the worst and the median of obtained final results for all 30 runs of evolutionary algorithms. For the comparison of the algorithms, the best individual results are highlighted in bold in all tables. The results of the PSO algorithm are also compared with the performance of DE.

12.3.1 The First De Jong Function

The following tables (Table 12.1 and Table 12.2) contain the results for the 1st De Jong function. The proposed implementation of the chaotic Lozi map to the PSO algorithm seems to have improved the performance of the algorithm. The values for PSO Lozi (chaos number generator) are in all cases better than the values for PSO Weight (classic number generator). Furthermore those results are better or comparable with those of DE. However PSO seems to be less efficient than DE in solving higher dimension with the used setting.

Table 12.1: The results for the first De Jong function for Dim = 2, 5 and 10

	Dim = 2			Dim = 5			Dim = 10		
	PSO Weight	PSO Lozi	DE	PSO Weight	PSO Lozi	DE	PSO Weight	PSO Lozi	DE
The worst result	$2.38.10^{-03}$	**$1.60.10^{-04}$**	$2.37.10^{-04}$	$3.92.10^{-03}$	**$2.56.10^{-05}$**	$1.00.10^{-03}$	$3.75.10^{-02}$	$2.08.10^{-02}$	**$2.81.10^{-03}$**
The best result	$1.17.10^{-05}$	**$2.04.10^{-07}$**	$2.47.10^{-07}$	$1.65.10^{-05}$	**$2.33.10^{-07}$**	$7.43.10^{-05}$	$1.45.10^{-04}$	**$4.2.10^{-07}$**	$5.26.10^{-04}$
Median	$2.01.10^{-04}$	$2.35.10^{-05}$	**$2.31.10^{-05}$**	$1.74.10^{-04}$	**$2.81.10^{-06}$**	$1.9.10^{-04}$	$6.25.10^{-03}$	**$6.30.10^{-04}$**	$1.18.10^{-03}$

Table 12.2: The results for the first De Jong function for Dim = 20 and 40

	Dim = 20			Dim = 40		
	PSO Weight	PSO Lozi	DE	PSO Weight	PSO Lozi	DE
The worst result	1.62	1.95	**$4.01.10^{-02}$**	9.06	8.60	**1.68**
The best result	$8.4.10^{-03}$	$4.6.\ 10^{-02}$	**$1.04.10^{-02}$**	$2.40.10^{-01}$	**$2.43.10^{-03}$**	$6.57.10^{-01}$
Median	$4.78.10^{-02}$	$4.05.10^{-01}$	**$1.86.10^{-02}$**	4.49	4.25	**$9.97.10^{-01}$**

12.3.2 The Second De Jong Function

The results in the tables (Table 12.3 and Table 12.4) were obtained by optimizing the 2nd De Jong function. Almost similar trends as in the previous section (1st De Jong function) can be seen in the results with the exception of having worse performance of DE in comparison with PSO Lozi for the higher dimensions.

Table 12.3: Results for the second De Jong function for Dim = 2, 5 and 10

	Dim = 2			Dim = 5			Dim = 10		
	PSO Weight	PSO Lozi	DE	PSO Weight	PSO Lozi	DE	PSO Weight	PSO Lozi	DE
The worst result	$2.70.10^{-02}$	**$1.59.10^{-02}$**	$5.58.10^{-02}$	$1.64.10^{-01}$	**$1.15.10^{-02}$**	3.7	$1.92.10^{-01}$	**$1.70.10^{-02}$**	$1.90.10^{1}$
The best result	$6.30.10^{-05}$	**$2.91.10^{-05}$**	$2.06.10^{-04}$	$1.25.10^{-03}$	**$5.10.10^{-05}$**	$6.2.10^{1}$	$3.35.10^{-03}$	**$8.64.10^{-05}$**	7.7
Median	$3.44.10^{-03}$	**$8.4.10^{-04}$**	$1.09.10^{-02}$	$3.39.10^{-02}$	**$7.97.10^{-04}$**	2.1	$3.87.10^{-02}$	**$1.16.10^{-03}$**	$1.4.10^{1}$

Table 12.4: The results for the second De Jong function for Dim = 20 and 40

	Dim = 20			Dim = 40		
	PSO Weight	PSO Lozi	DE	PSO Weight	PSO Lozi	DE
The worst result	**2.60**$.10^{-01}$	$3.172.10^{-01}$	$5.2821.10^{1}$	1.0324	**1.3342**	$1.3381.10^{2}$
The best result	$5.29.10^{-03}$	**$7.8609.10^{-04}$**	$2.5292.10^{1}$	$7.9342.10^{-02}$	**$2.2454.10^{-02}$**	$5.0176.10^{1}$
Median	$8.01.10^{-02}$	**$3.679.10^{-02}$**	$3.7204.10^{1}$	$3.7495.10^{-01}$	**$2.1875.10^{-01}$**	$7.7725.10^{1}$

12.3.3 Rastrigin Function

In the case of the Rastrigin benchmark function there is no significant improvement in the PSO performance, however both PSO algorithms were significantly worse for Dim = 40 than those of DE (see Table 12.5 and Table 12.6).

Table 12.5: The results for the Rastrigin function for Dim = 2, 5 and 10

	Dim = 2			Dim = 5			Dim = 10		
	PSO Weight	PSO Lozi	DE	PSO Weight	PSO Lozi	DE	PSO Weight	PSO Lozi	DE
The worst result	2.4048	1.9933	**1.7853**	13.5637	9.9854	**9.6667**	**25.1741**	27.2665	26.7314
The best result	0.0032	**0.0015**	0.0056	1.0756	**0.0022**	0.1615	**0.2599**	3.0098	6.2952
Median	0.5399	**0.3790**	0.5707	6.5017	**3.0761**	4.0811	**9.5158**	10.4711	15.7813

Table 12.6: The results for the Rastrigin function for Dim = 20 and 40

	Dim = 20			Dim = 40		
	PSO Weight	PSO Lozi	DE	PSO Weight	PSO Lozi	DE
The worst result	74.9295	**68.5083**	75.9887	186.5780	192.4930	**155.9540**
The best result	27.4552	**17.2733**	18.7197	100.0440	47.8408	**38.8948**
Median	47.5319	**46.6510**	50.0244	152.0650	159.0090	**119.5870**

12.3.4 Schwefel Function

The presented results in Table 12.7 and Table 12.8 show increasing difference between the values of median for PSO Weight and PSO Lozi together with increasing dimension in favour of the Lozi map enhanced PSO. As in the previous case, both algorithms were surpassed by DE (most significantly in higher dimensions).

Table 12.7: The results for the Schwefel function for Dim = 2, 5 and 10

	dim = 2			dim = 5			dim = 10		
	PSO Weight	PSO Lozi	DE	PSO Weight	PSO Lozi	DE	PSO Weight	PSO Lozi	DE
The worst result	-673.39	-697.48	**-702.79**	-1191.59	-1262.01	**-1635.06**	-1915.02	-1980.15	**-3174.74**
The best result	-837.87	**-837.94**	-837.91	-1760.30	-1942.73	**-1964.83**	-2740.49	-3037.15	**-3562.67**
Median	**-820.35**	-819.43	-816.07	-1458.34	-1525.28	**-1842.63**	-2234.00	-2561.77	**-3372.21**

Table 12.8: Results for the Schwefel function for Dim = 20 and 40

	dim = 20			dim = 40		
	PSO Weight	PSO Lozi	DE	PSO Weight	PSO Lozi	DE
The worst result	-2886.88	-3133.19	**-5830.19**	-4839.42	-6112.29	**-9104.93**
The best result	-5372.33	-5722.76	**-6482.53**	-8400.45	-9935.40	**-11411.20**
Median	-3477.26	-4217.83	**-6358.48**	-6558.88	-7527.70	**-10308.10**

From the results presented in the tables (Table 12.1 - Table 12.8) it may be stated, that the majority of results obtained by the proposed PSO Lozi algorithm were better than the results of classic PSO Weight. The observed median of the final cost function values for all 30 runs was better in 17 of 20 experiments - 4 benchmark functions x 5 dimensions values (see Fig. 12.1).

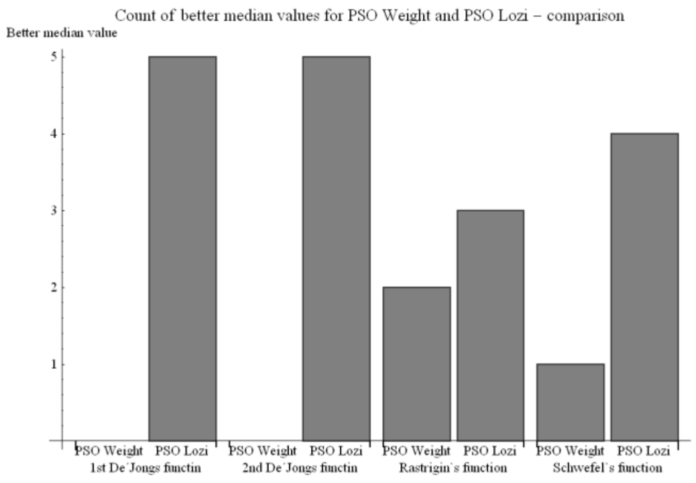


Fig. 12.1: PSO Weight and PSO Lozi results

On the y-axis, there is the number of dimensions where the better value was achieved.

The analysis of the results presented here shows an interesting phenomenon, that the performance of DE in comparison with both PSO algorithms is sometimes much better (especially in the case of the Schwefel function). Further, the rest of the chapter presents the investigation of this phenomenon. Based on the previous experience, the PSO algorithm is able to achieve better results for higher population size (NP). To prove this theory an experiment was set-up:

Population size: 30, 50, 75, 100 (Schwefel function only), 150, 200, 300, 400

Iterations / generations: 200

w_{start}: 0.9

w_{end}: 0.4

Dimension: 20

Benchmark functions: Schwefel, 1st De Jong

Results are shown in the following tables (Table 12.9 and Table 12.10).

Table 12.9: A mean value for 30 runs; the Schwefel function; Dim = 20; generations = 200

NP	30	50	75	100	150	200	300	400
PSO Weight	-3697.63	-3873.62	-4140.99	-4255.84	-4329.57	-4866.41	-5316.41	-5377.72
PSO Lozi	-4340.06	-4560.42	-5032.82	-5241.78	**-5801.99**	**-5998.05**	**-6174.55**	**-6225.63**
DE	**-6100.9**	**-5737.68**	**-5649.1**	**-5500.01**	-5635.55	-5651.5	-5673.33	-5651.23

Table 12.10: A mean value for 30 runs; the 1st De Jong function; Dim = 20; generations = 200

NP	150	200	300	400
PSO Weight	$7.33004.10^{-06}$	$3.4528.10^{-07}$	$3.7368.10^{-09}$	$2.60345.10^{-10}$
PSO Lozi	$2.7701.10^{-06}$	$\mathbf{4.1212.10^{-08}}$	$\mathbf{3.66273.10^{-11}}$	$\mathbf{1.0319.10^{-14}}$
DE	$\mathbf{4.25565.10^{-07}}$	$4.94665.10^{-07}$	$6.1166.10^{-07}$	$7.07218.10^{-07}$

From Table 12.9 and Table 12.10, it is clear that the increase in the size of population (NP) led to significant improvement in the performance of both PSO algorithms with inertia weight. Nevertheless the performance of DE showed the opposite trend. The presented results in this section support the claim, that using the Lozi map as a chaotic number generator could lead to the improvement of the performance of the PSO algorithm thus to achieve better or at least similar results when comparing PSO algorithms with another evolutionary algorithm - DE.

12.4 Chaotic Pseudorandom Number Generator – Conclusion

This chapter proposes and investigates the enhanced PSO algorithm with inertia weight and with a chaos number generator. The Lozi map was used as chaotic system for a number generator in the main formula of the PSO algorithm. Four different test functions were used to demonstrate the performance and behaviour of the proposed algorithm also in comparison with a classic non-chaotic version and one strategy of a differential evolution. The primary aim of this work was not to develop a new type of pseudorandom number generator, which should pass many statistical tests, but to try to combine natural chaotic dynamics and evolutionary algorithm inspired by nature to observe the performance.

Based on the presented results it can be stated that the Lozi map used as the number generator seems to have a significantly positive effect on the speed of convergence of the algorithm. The research with other chaotic maps and also with the differential evolution has been already done and all simulations give better results in the analysed benchmark functions. Furthermore, all obtained results point to the fact that they are very sensitive to the selection of the chaotic system that is used as a pseudorandom generator. Any change in the selection of a chaotic system or its parameter adjustment can cause a radical improvement of the evolutionary algorithm performance, however on the downside it can cause the worsening of observed parameters and subsequently the behavior of the algorithm as such.

Chaos driven evolutionary algorithms seems to be a promising area of research with many open questions to be answered. One of the questions can be the examination of the impact of chaotic system parameters on the generation of pseudorandom numbers, and thus its influence on the results obtained using a selected evolutionary algorithm. One of possible approaches to this issue is to use meta-evolution which will be done in the future.

13 Conclusion

The submitted thesis gives an overview of modern techniques of soft computing and its selected applications, which is the author interested in. The main thread is connected with a metaevolutionary approach in symbolic regression and their applications for benchmark or for real tasks.

Firstly, an introduction into several methods such as evolutionary algorithms (DE, SOMA, PSO), methods of symbolic regression (GP, GE, AP) and artificial neural networks is described. All these methods were used either separately or combined together for solving of complex tasks such as metaevolution for the synthesis of new optimization algorithms, metaevolutionary approach with AP for synthesis of a whole control law for deterministic chaotic systems, steganalysis by means of ANN, optimal modelling of a dynamic flight, synthesis of pseudo artificial neural networks and a chaotic generator used in evolutionary computation.

All these techniques and described applications can serve to other scientists as an inspiration for their work. The community connected with soft computing techniques is huge in the world but small in our country. Therefore, the author has to cooperate and discuss the results of her work with colleagues from abroad during conferences and visits or with invited lecturers within the Erasmus programme. The author has been accepted by the research community, which is documented by publication activities, by the best paper awards (see the two best paper awards in the Appendix 1 and list of author's publication in Appendix 2), by serving as a member of international committees of conferences and editorial boards of journals.

The discussed techniques and methodology in proposed applications will serve, hopefully, as an inspiration for experts in various fields. Soft computing tools, their combinations, their adjusted versions help to obtain optimal results of required assignments and solved problems in all areas of human life such as process control, diagnostics, image processing, operation research, medicine, monitoring of financial markets and the prediction of exchange rates and others.

The author proposed combinations of soft computing methods to produce better results and methodology for the usage of meta approach techniques with symbolic regression, evolutionary computation and artificial neural networks. The presented ideas and applications are only part of the author's research portfolio. The published techniques and applications within conference, journals and book chapters are transferred into lectures and laboratories/seminars of special courses focused on artificial intelligence that cover theoretical backround as well as examples of applications (not only) from the area of soft computing which allows students to have the latest news from this field.

The future plans of the author is to continue with the adjusting, development and combining of the described techniques for obtaining better results, optimizing the computational time and mainly optimizing the cost functions themselves. The design of the cost function is the crucial moment of all simulations and solving of complex tasks. The further plans will be to find and add suitable conditions to the cost functions which secure the smooth evolutionary process and provide the best results as possible.

References

[1] Araujo, E., Coelho, L.: Particle swarm approaches using Lozi map chaotic sequences to fuzzy modelling of an experimental thermal-vacuum system, Applied Soft Computing, v.8 n.4, p.1354-1364, September, 2008

[2] Back T., Fogel D. B., Michalewicz Z.: Handbook of evolutionary algorithms, Oxford University Press, 1997, ISBN 0750303921

[3] Chiang, C.L: Statistical methods of analysis, World Scientific, 2003, ISBN 981-238-310-7

[4] Coelho L.D., Mariani, V.C.: An efficient cultural self-organizing migrating strategy for economic dispatch optimization with valve-point effect, Energy Conversion And Management, Volume 51, Issue 12, pp 2580-2587, 2010, ISSN: 0196-8904

[5] Coelho L.D.: Self-Organizing Migrating Strategies Applied to Reliability-Redundancy Optimization of Systems, IEEE Transactions On Reliability, Volume 58, Issue 3, pp 501-510, 2009, ISSN: 0018-9529.

[6] Coelho L.D.: Self-organizing migration algorithm applied to machining allocation of clutch assembly, Mathematics and Computers in Simulation, Vol. 80, Issue 2, 2009, pp 427-435.

[7] Cole E., Krutz D. R.: Hiding Sight, United States of America: Wiley Publishing, Inc., 2003. 321 p. ISBN 0-471-44449-9.

[8] Cormen T. H., Leiserson Ch. E., Rivest R. L. and Stein C.: Introduction to Algorithms, Second Edition. MIT Press and McGraw-Hill, 2001. ISBN 0-262-03293-7. Section 16.3, pp. 385–392.

[9] Davendra D., Zelinka I., Senkerik R.: Chaos driven evolutionary algorithms for the task of PID control, Computers & Mathematics with Applications, Volume 60, Issue 4, 2010, pp 1088-1104, ISSN 0898-1221.

[10] David A. Freedman: Statistical Models: Theory and Practice, Cambridge University Press (2005)

[11] Davidson J.W., Savic D.A., Walters G.A.: Symbolic and numerical regression: experiments and applications, Informatics and Computer Science – An international Journal, Vol. 150, Elsevier, USA, 2003, pages 95 – 117, ISSN:0020-0255

[12] Dawis L.: Handbook of Genetic Algorithms, International Thomson Computer Press, 1996, ISBN 1850328250

[13] Der-Chyuan Lou, Chen-Hao Hu, Chao-Lung Chou, Chung-Cheng Chiu: Steganalysis of HMPD reversible data hiding scheme, Optics Communications, Volume 284, Issue 23, 1 November 2011, pp. 5406-5414, ISSN 0030-4018

[14] Deugo D., Ferguson D.: Evolution to the xtreme: Evolving evolutionary strategies using a meta-level approach, Proceedings of the 2004 IEEE congress on evolutionary computation, IEEE Press, Portland, Oregon, pp. 31–38, 2004

[15] Dioşan, L., Oltean, M.: Evolutionary design of evolutionary algorithms. Genetic Programming and Evolvable Machines, Vol. 10, Issue 3, p. 263-306, 2009

[16] Dorigo M.: Ant Colony Optimization and Swarm Intelligence, Springer, 2006, ISBN 3540226729

[17] Eberhart R., Kennedy J.: Swarm Intelligence (The Morgan Kaufmann Series in Artificial Intelligence), Morgan Kaufmann, 2001, ISBN 1558605959

[18] Edmonds, B.: Meta-genetic programming: Co-evolving the operators of variation, Elektrik, Vol. 9, Issue 1, pp. 13-29, 2001

[19] Eiben A.E., Michalewicz Z., Schoenauer M., Smith J.E.: Parameter control in evolutionary algorithms, pp. 19–46, Springer, 2007

[20] Etkin B., Lloyd D. R.: Dynamics of flight – Stability and Control. John Wiley & Sons, 1996

[21] Farmer J.D., Packard N., Perelson A.: The immune system, adaptation and machine learning, Physica D, vol. 2, pp. 187—204, 1986

[22] Fausett L. V.: Fundamentals of Neural Networs: Architectures, Algorithms and Applications, Prentice Hall, 1993, ISBN: 9780133341867

[23] Fekiac J., Zelinka I., Burguillo J. C.: A review of methods for encoding neural network topologies in evolutionary computation, ECMS 2011, Krakow, Poland, ISBN: 978-0-9564944-3-6

[24] Ferreira C.: Gene Expression Programming: Mathematical Modeling by an Artificial Intelligence, Springer, 2006, ISBN: 3540327967

[25] Fridrich J., Goljan M., Soukal D.: Perturbed quantization steganography, Multimedia Systems, Volume 11, Issue 2, pp 98-107, Springer – Verlag, 2005, ISSN: 0942-4962

[26] Fridrich, J., Goljan, M., and Hogea, D.: New Methodology for Breaking Steganographic Techniques for JPEGs. Submitted to SPIE: Electronic Imaging 2003, Security and Water-marking of Multimedia Contents. Santa Clara, California, 2003

[27] Fridrich, J., Goljan, M., and Hogea, D.: Steganalysis of JPEG Images: Breaking the F5 Algorithm, 5th Information Hiding Workshop, Noordwijkerhout, The Netherlands, Oct. 2002. URL: http://www.ws.binghamton.edu/fridrich/Research/f5.pdf. Last accessed: 2003-12-24.

[28] Fridrich, J.: Feature-Based Steganalysis for JPEG Images and Its Implications for Future Design of Steganographic Schemes. In Proceedings of Information Hiding, 2004, pp.67~81

[29] Goldwasser S., Bellare M.: Lecture Notes on Cryptography. Cambridge, Massachusetts [MIT]: [s.n.], 2001. 283 p.,online http://cseweb.ucsd.edu/~mihir/papers/gb.pdf

[30] Gurney K.: An Introduction to Neural Networks, CRC Press, 1997, ISBN: 1857285034

[31] Hertz J., Kogh A. and Palmer R. G.: Introduction to the Theory of Neural Computation, Addison – Wesley 1991

[32] Hetzl S.: Steghide (1) - Linux man page [online]. [cit. 2008-05-21]. available from WWW: <http://steghide.sourceforge.net/documentation/manpage.php>.

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

[34] Holoska J., Oplatkova Z., Senkerik R., Zelinka I.: Comparison Between Neural Network Steganalysis and Linear Classification Method Stegdetect, CIMSim 2010, Bali, Indonesie, IEEE, ISBN: 978-0-7695-4262-1

[35] Hološka, J., Oplatková, Z., Zelinka, I., Šenkeřík, R.: Steganografie jako hrozba úniku kritických obchodních informací a její detekce pomocí umělých neuronových sítí, FAMily media, s. r. o., Security magazín, Praha, 2010, 33-37, ISSN 1210-872

[36] ImageMagick - http://www.imagemagick.org/script/index.php

[37] Johnson Colin G., Artificial immune systems programming for symbolic regression, In C. Ryan, T. Soule, M. Keijzer, E. Tsang, R. Poli and E. Costa,

editors, Genetic Programming: 6th European Conference, LNCS 2610, p. 345-353, 2003, ISSN 0302-9743

[38] Jones D.F., Mirrazavi S.K., Tamiz M.: Multi-objective meta-heuristics: An overview of the current state-of-the-art, European Journal of Operational Research, Volume 137, Issue 1, 16 February 2002, Pages 1-9, ISSN 0377-2217.

[39] Just W.: Principles of Time Delayed Feedback Control, In: Schuster H.G., Handbook of Chaos Control, Wiley-Vch, 1999, ISBN 3-527-29436-8.

[40] Kalczynski P. J., Kamburowski J.: On the NEH heuristic for minimizing the makespan in permutation flow shops, Omega, vol. 35, 1, 2007, pp. 53-60.

[41] Kominkova Oplatkova Z., Holoska J., Senkerik R., Steganography content detection by means of feedforward neural network, International Journal of Innovative Computing and Applications, Vol. 5, No. 1, (accepted for publication 2012), ISSN 1751-648X.

[42] Kordík P., Koutník J., Drchal J., Kovářík O., Čepek M., Šnorek M.: Meta-learning approach to neural network optimization, Neural Networks, Vol. 23, Issue 4, p. 568-582, 2010, ISSN 0893-6080.

[43] Koza J. R. et al.: Genetic Programming III; Darwinian Invention and problem Solving, Morgan Kaufmann Publisher, 1999, ISBN 1-55860-543-6

[44] Koza J. R.: Genetic Programming, MIT Press, 1998, ISBN 0-262-11189-6

[45] Kvasnička V., Pospíchal J., Tiňo P.: Evolučné algoritmy, STU Bralislava, 2000, ISBN 80-227-1377-5

[46] Kwon O. J.: Targeting and Stabilizing Chaotic Trajectories in the Standard Map, Physics Letters A, vol. 258, 1999, pp. 229-236.

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

[48] May R.M.: Stability and Complexity in Model Ecosystems, Princeton University Press, 2001, ISBN: 0-691-08861-6.

[49] Murty K. G.: Linear complementarity, linear and nonlinear programming, Sigma Series in Applied Mathematics, Berlin: Heldermann Verlag, 1988, ISBN 3-88538-403-5.

[50] Murty K. G.: Linear programming, New York: John Wiley & Sons, 1983, ISBN 0-471-09725-X.

[51] Nissar A., Mir A.H.: Classification of steganalysis techniques: A study, Digital Signal Processing, Volume 20, Issue 6, December 2010, Pages 1758-1770, ISSN 1051-2004.

[52] O'Sullivan J., Ryan C.: An Investigation into the Use of Different Search Strategies with Grammatical Evolution, Proceedings of the 5th European Conference on Genetic Programming, p.268 - 277, 2002, Springer-Verlag London, UK, ISBN:3-540-43378-3

[53] O'Neill M., Ryan C.: Grammatical Evolution. Evolutionary Automatic Programming in an Arbitrary Language, Kluwer Academic Publishers, 2003, ISBN 1402074441

[54] Oltean M., Grosan C.: Evolving Evolutionary Algorithms using Multi Expression Programming, The 7th European Conference on Artificial Life, September 14-17, 2003, Dortmund, Edited by W. Banzhaf (et al), LNAI 2801, pp. 651-658, Springer-Verlag, Berlin, 2003

[55] Oplatkova Z., Senkerik R., Zelinka, I., Holoska, J.: Synthesis of Control Law for Chaotic Logistic Equation: Preliminary Study, IEEE 4th International Conference on Mathematical Modelling and Computer Simulation, 2010, pp. 65-70, ISBN-ISSN 978-0-7695-4062-7

[56] Oplatková Z., Zelinka I.: Investigation on Evolutionary Synthesis of Movement Commands, Modelling and Simulation in Engineering, Volume 2009, Article ID 845080, 12 pages, Hindawi Publishing Corporation, 2009, ISSN: 1687-559.

[57] Oplatkova Z.: Metaevolution: Synthesis of Optimization Algorithms by means of Symbolic Regression and Evolutionary Algorithms, Lambert Academic Publishing Saarbrücken, 2009, ISBN: 978-3-8383-1808-0

[58] Oplatkova, Z., Holoska J., Prochazka M., Senkerik, R., Jasek, R.: Optimization of Artificial Neural Network Structure in the Case of Steganalysis, In: Springer Series "Inteligent Systems" - "Handbook of Optimization", (Ivan Zelinka, Vaclav Snasel, Ajith Abraham(Eds.)), pp. 821 - 824, 2012, ISBN 978-3-642-30503-0

[59] Oplatkova, Z., Holoska, J., Zelinka, I., Senkerik, R.: Detection of Steganography Inserted by OutGuess and Steghide by means of Neural Networks, AMS2009 Asia

Modelling Symposium 2009, IEEE Computer Society, Piscataway, 2009, ISBN 978-0-7695-3648-4

[60] Oplatkova, Z., Holoska, J., Zelinka, I., Senkerik, R.: Steganography Detection by means of Neural Networks, IEEE Operations Center, Nineteenth International Workshop on Database and Expert Systems Applications, Piscataway, 2008, 571-576, ISBN 978-0-7695-3299-8

[61] Oplatková, Z., Šenkeřík, R., Bělašková, S., Zelinka, I.: Synthesis of Control Rule for Synthesized Chaotic System by means of Evolutionary Techniques, VUT v Brně, 16th International Conference on Soft computing, Brno, Czech Republic, 2010, 91-98, ISBN-ISSN 978-80-214-4120-0

[62] Oplatková, Z., Šenkeřík, R.: Classification with Pseudo Neural Networks Based On Evolutionary Symbolic Regression. In 2011 International Conference on P2P, Parallel, Grid, Cloud and Internet Compting. Piscataway: IEEE Operations Center, 2011, s. 396-401. ISBN 978-0-7695-4531-8.

[63] Ott E., Greboki C., Yorke J.A.: Controlling Chaos, Phys. Rev. Lett. Vol. 64, 1990, pp. 1196-1199.

[64] Paterson N., Livesey M.: Distinguishing genotype and phenotype in genetic programming, In Koza, Goldberg, Fogel & Riolo, eds. Late Breaking Papers at GP 1996, MIT Press, 1996, ISBN 0-18-201-031-7

[65] Paterson N.: Genetic Programming with context sensitive grammars, doctoral thesis, University of St. Andrews, 2003

[66] Pluhacek M., Budikova V., Senkerik R., Oplatkova Z., Zelinka I.: On The Performance Of Enhanced PSO algorithm With Lozi Chaotic Map – An Initial Study, In: Proceedings of the 18th International Conference on Soft Computing, MENDEL 2012, pp. 40 - 45, 2012, ISBN 978-80-214-4540-6.

[67] Pluhacek M., Budikova V., Senkerik R., Oplatkova Z., Zelinka I.: Extended Initial Study on the Performance of Enhanced PSO Algorithm with Lozi Chaotic Map, In Proceedings of Nostradamus 2012: International conference on prediction, modeling and analysis of complex systems, Springer Series: "Advances in Intelligent Systems and Computing", Vol. 192, 2012, pp. 167 – 178, ISBN: 978-3-642-33226-5.

[68] Price K., Storn R. M., Lampinen J. A.: Differential Evolution: A Practical Approach to Global Optimization, (Natural Computing Series), Springer; 1 edition. 2005

[69] Provos N.: Defending Against Statistical Steganalysis, 10th USENIX Security Symposium. Washington, DC, August 2001

[70] Provos, N. , Honeyman, P.: Hide and Seek: An Introduction to Steganography. IEEE Security & Privacy 1(3): 32-44 (2003)

[71] Provos, N., Honeyman, P.: Detecting Steganographic Content on the Internet. CITI Technical Report 01-11, 2001

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

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

[74] Qingzhong Liu, Andrew H. Sung, Mengyu Qiao, Zhongxue Chen, Ribeiro B.: An improved approach to steganalysis of JPEG images, Information Sciences, Volume 180, Issue 9, 1 May 2010, pp. 1643-1655, ISSN 0020-0255

[75] Reeves C.: Modern Heuristic Techniques for Combinatorial Problems, McGraw-Hill, New York, 1995

[76] Reitermanova Z.: Feedforward Neural Networks – Architecture Optimization and Knowledge Extraction, WDS'08 Proceedings of Contributed Papers, Part I, 159–164, 2008, MATFYZPRESS, ISBN 978-80-7378-065-4

[77] Sabeti V., Samavi S., Mahdavi M., Shirani S.: Steganalysis and payload estimation of embedding in pixel differences using neural networks, Pattern Recognition, Volume 43, Issue 1, January 2010, pp. 405-415, ISSN 0031-3203.

[78] Salustowicz R. P., Schmidhuber J.: Probabilistic Incremental Program Evolution, Evolutionary Computation, vol. 5, nr. 2, 1997, pp. 123 – 141, MIT Press, ISSN 1063-6560

[79] Senkerik R., Davendra D, Zelinka I., Pluhacek M., Oplatkova Z., An Investigation On The Differential EvolutionDriven By Selected Discrete Chaotic Systems, In: Proceedings of the 18th International Conference on Soft Computing, MENDEL 2012, pp. 157 - 162, 2012, ISBN 978-80-214-4540-6.

[80] Senkerik R., Davendra D., Zelinka I., Pluhacek M., Oplatkova Z., An Investigation on the Chaos Driven Differential Evolution: An Initial Study, In: Proceedings of the Fifth International Conference on Bioinspired Optimization Methods and Their Applications, BIOMA 2012, pp. 185 - 194, 2012, ISBN 978-961-264-043-9.

[81] Senkerik 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, 2010, pp. 1026-1037.

[82] Senkerik R., Zelinka I., Oplatkova Z.: Evolutionary Techniques for Deterministic Chaos Control, CISSE'08, In Proc. IETA 2008, International Joint Conferences on Computer, Information, and Systems Sciences, and Engineering, 5-13 December 2008, ISBN 978-90-481-3655-1.

[83] Senkerik, R., Oplatkova Z., Zelinka I., Davendra D., Jasek R.: Synthesis Of Feedback Controller For Chaotic Systems By Means Of Evolutionary Techniques, The Fourth Global Conference on Power Control and Optimization, 2010, pp. 1-7.

[84] Senkerik, R., Oplatkova, Z., Zelinka, I., Davendra, D., Jasek, R.: Application of Evolutionary Techniques for Optimization of Chaos Control – Introduction of Three Approaches, In: Springer Series "Inteligent Systems" - "Handbook of Optimization", (Ivan Zelinka, Vaclav Snasel, Ajith Abraham(Eds.)), p. 801-820, 2012, ISBN 978-3-642-30503-0..

[85] Senkerik, R., Oplatkova, Z., Zelinka, I., Davendra, D.: Synthesis of feedback controller for three selected chaotic systems by means of evolutionary techniques: Analytic programming, Elsevier Science Ltd., Mathematical and Computer Modelling, Oxford, 2011, ISSN 0895-7177, DOI: 10.1016/j.mcm.2011.05.030.

[86] Senkerik, R., Zelinka, I., Davendra, D., Oplatkova, Z: Evolutionary Design of Chaos Control in 1D, In. Zelinka I., Celikovski S., Richter H., Chen G.: Evolutionary Algorithms and Chaotic Systems, SpringerVerlag Berlin, 2010, pp.165 - 190, ISBN 9783642107061.

[87] Senkerik, R., Zelinka, I., Davendra, D., Oplatkova, Z.: Evolutionary Optimization Of Henon Map Control – A Blackbox Approach, International Journal of Operational Research, 2012, Vol. 13, No. 2, pp. 129 - 146.

[88] Senkerik, R., Zelinka, I., Oplatkova, Z.: Optimal Control of Evolutionary Synthesized Chaotic System, 15th International Conference on Soft Computing MENDEL 2009, 2009, pp. 220 – 227.

[89] Smith J., Fogarty T.: Operator and parameter adaptation in genetic algorithms, Soft Computing, Vol. 1, Issue 2, pp. 81–87, 1997

[90] Software OutGuess, www.outguess.org

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

[92] Stinstra E., Rennen G., Teeuwen G.: Meta-modelling by symbolic regression and Pareto Simulated Annealing, Tilburg University, Netherlands, nr. 2006-15, ISSN 0924-7815

[93] Tupý, J., Oplatková, Z., Zelinka, I.: Neural Differentiation in Modeling, Vysoké učení technické v Brně, Fakulta strojního inženýrství, MENDEL 2009 15th International Coference on Soft Computing, Brno, Czech Republic, 2009, 154-159, ISBN-ISSN 978-80-214-3675-6

[94] Tupy, J., Zelinka, I.: Evolucni algoritmy pri stabilizaci bezpilotnich prostredku (UAV). In Civilni bezpilotni systemy 2008. Praha : Odborná společnost letecká České republiky, 2008, s. D3.

[95] Tupy, J., Zelinka, I.: Evolutionary algorithms in aircraft trim optimization. In Nineteenth International Workshop on Database and Expert Systems Applications. Piscataway : IEEE Operations Center, 2008, s. 524-530.

[96] Tupy, J., Zelinka, I.: Search for equilibrium state flight. In CISSE 2008. University of Bridgeport, Conecticut, USA, University of Bridgeport, Bridgeport, CT 06604, USA, 2008, s. 1-15.

[97] Voutsinas T. G., Pappis C. P.: A branch and bound algorithm for single machine scheduling with deteriorating values of jobs, Mathematical and Computer Modelling, vol. 52, 1-2, 2010, pp 55-61.

[98] Wasserman P. D.: Neural Computing: Theory and Practice, Coriolis Group, 1980, ISBN: 0442207433

[99] Weisser, R., Osmera, P., Matousek, R.: Transplant evolution with modified schema of differential evolution: Optimization structure of controllers, International Conference on Soft Computing MENDEL, Brno, Czech Republic, 2010

[100] Westfeld, A., Pfitzmann, A.: Attacks on Steganographic Systems. In Proceedings of Infor-mation Hiding - Third International Workshop. Springer Verlag, September 1999.

[101] Westfeld, A.: High Capacity Despite Better Steganalysis (F5—A Steganographic Algorithm). In: Moskowitz, I.S. (eds.): Information Hiding. 4th International Workshop. Lecture Notes in Computer Science, Vol.2137. Springer-Verlag, Berlin Heidelberg New York (2001) 289— 302

[102] Zelinka et al.: Analytical Programming - a Novel Approach for Evolutionary Synthesis of Symbolic Structures, in Kita E.: Evolutionary Algorithms, InTech 2011, ISBN: 978-953-307-171-8

[103] Zelinka I., Chen G., Celikovsky S.: Chaos Synthesis by Means of Evolutionary Algorithms, International Journal of Bifurcation and Chaos, Vol 18, No 4, 2008, pp. 911 – 942, DOI: 10.1142/S021812740802077X

[104] Zelinka I., Oplatkova Z., Nolle L.: Boolean Symmetry Function Synthesis by Means of Arbitrary Evolutionary Algorithms-Comparative Study" International Journal of Simulation Systems, Science and Technology, Volume 6, Number 9, August 2005, pages 44 - 56, ISSN: 1473-8031.

[105] Zelinka I., Oplatková Z., Šeda M., Ošmera P., Včelař F.: Evoluční výpočetní techniky - principy a aplikace, BEN, Praha, 2008, 550 s., ISBN 80-7300-218-3

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

[107] Zelinka I., Varacha P., Oplatkova Z.: Evolutionary Synthesis of Neural Network, Mendel 2006 – 12th International Conference on Softcomputing, Brno, Czech Republic, 31 May – 2 June 2006, pp. 25 – 31, ISBN 80-214-3195-4

[108] Zelinka I.: Analytic Programming by Means of Soma Algorithm. ICICIS'02, First International Conference on Intelligent Computing and Information Systems, Egypt, Cairo, 2002, ISBN 977-237-172-3

[109] Zelinka I.: Analytic Programming by Means of Soma Algorithm. Mendel '02, In: Proc. 8th International Conference on Soft Computing Mendel'02, Brno, Czech Republic, 2002, 93-101., ISBN 80-214-2135-5

[110] Zelinka I.: SOMA – Self Organizing Migrating Algorithm, In: New Optimization Techniques in Engineering, (B.V. Babu, G. Onwubolu (eds)), chapter 7, 33, Springer-Verlag, 2004, ISBN 3-540-20167X

[111] Zelinka I.: Umělá inteligence v problémech globální optimalizace, BEN, Praha, 2002, ISBN 80-7300-069-5

List of Figures

List of Tables

List of Symbols and Abbreviations

AP	Analytic Programming
ANN	Artificial Neural Networks
CF	Cost Function
CFE	Cost Function Evaluations
CV	Cost value
DE	Differential Evolution
Dim	Dimensionality of given problem (number of arguments)
EA	Evolutionary Algorithms
GFS_{0arg}	Functions with 0 arguments in GFS, i.e. constants and variables
GFS_{1arg}	Functions with 1 argument in GFS (e.g. Sin, Cos, Tan..)
GFS_{2arg}	Functions with 2 arguments in GFS (e.g. +,-,/….)
GFS_{3arg}	Functions with 3 arguments in GFS
GFS	General Functional Space
GA	Genetic Algorithms
GP	Genetic Programming
GE	Grammatical Evolution
Population	Matrix NP x number of arguments of an individual
AP_{meta}	Metaevolutionary approach in Analytic Programming
PopSize	Number of individuals in population
PSO	Particle Swarm Intelligence
SOMA	Self-Organizing Migrating Algorithm
UPO	Unstable Periodic Orbit

Appendices

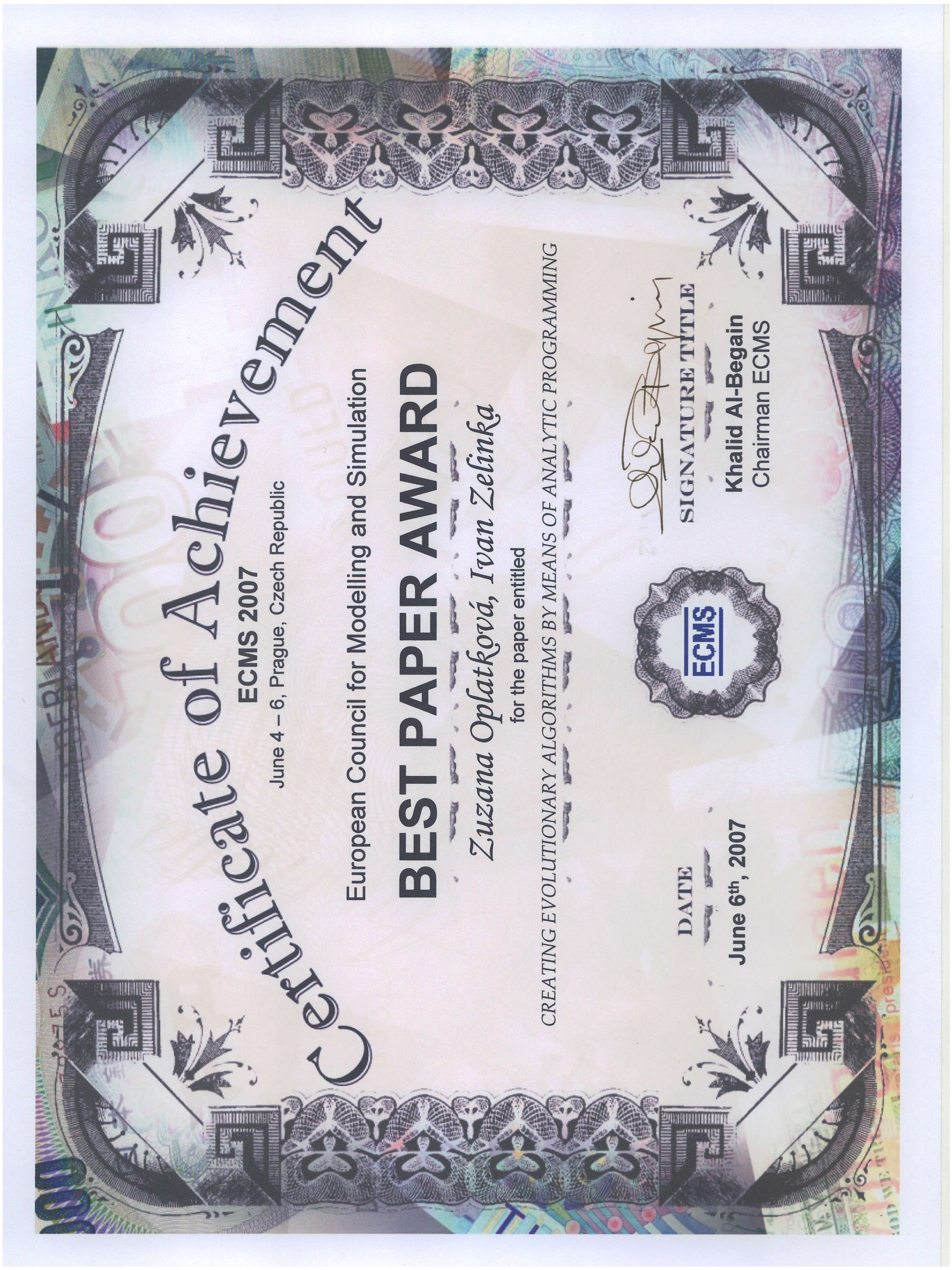
Appendix 1

Best paper award for:

Oplatková, Z., Zelinka, I.: Creating evolutionary algorithms by means of analytic programming - design of new cost function, European Council for Modelling and Simulation, ECMS 2007, Germany, 2007, 271-276, ISBN-ISSN 978-0-9553018-2-7

Best paper award for:

Šenkeřík, R., Zelinka, I., Davendra, D., Oplatková, Z.: Advanced Targeting Cost Function Design For Evolutionary Optimization Of Control Of Logistic Equation, University of Technology, The 3rd Global Conference on Power Control and Optimization, Sarawak, Malaysia, 2010, 1-6, ISBN-ISSN 978-983-44483-1-8



Certificate of Achievement

ECMS 2007

June 4 – 6, Prague, Czech Republic

European Council for Modelling and Simulation

BEST PAPER AWARD

Zuzana Oplatková, Ivan Zelinka

for the paper entitled

CREATING EVOLUTIONARY ALGORITHMS BY MEANS OF ANALYTIC PROGRAMMING

DATE

June 6th, 2007

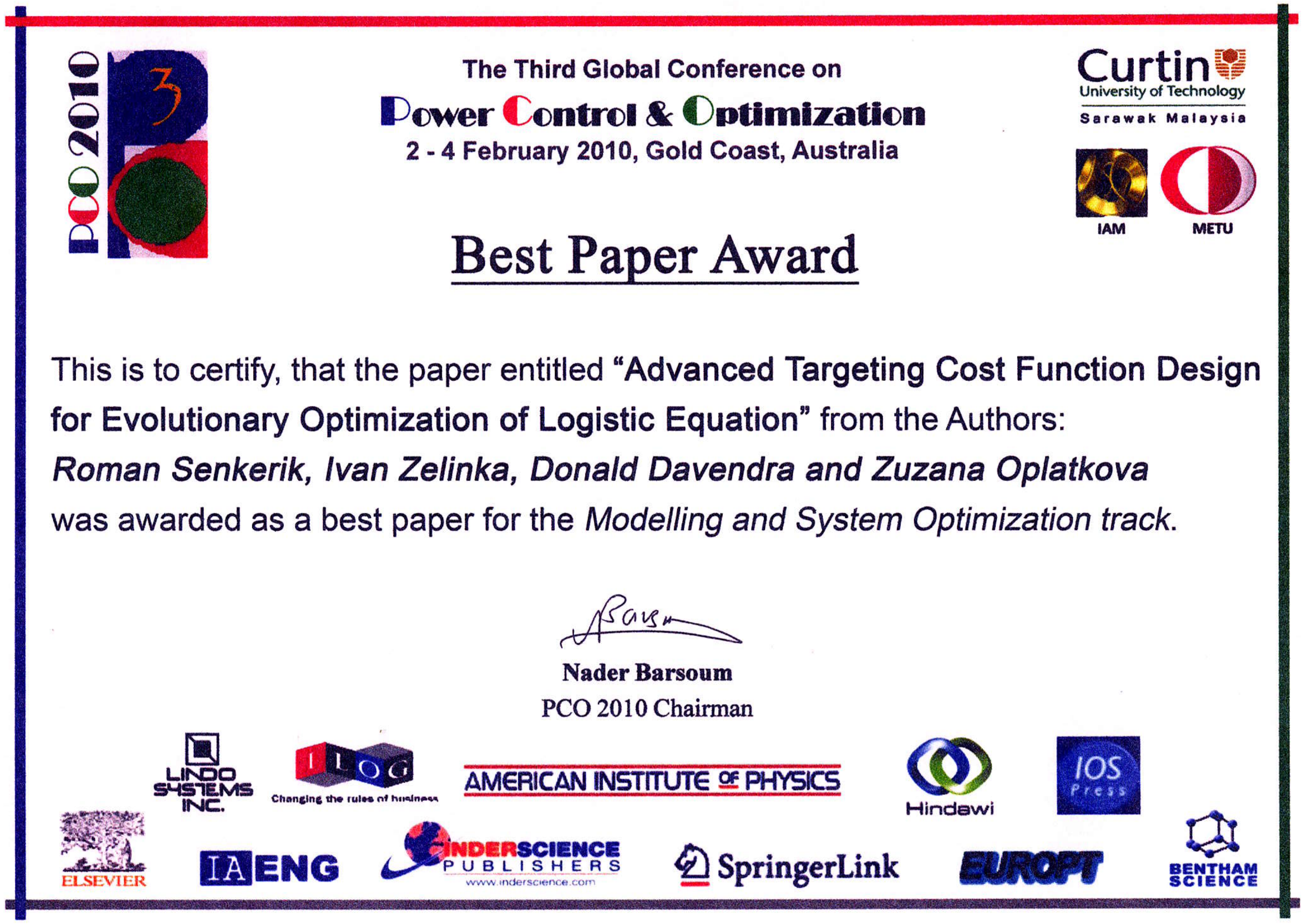
ECMS

[illegible]

SIGNATURE TITLE

Khalid Al-Begain

Chairman ECMS



PCO 2010

The Third Global Conference on

Power Control & Optimization

2 - 4 February 2010, Gold Coast, Australia

Curtin University of Technology Sarawak Malaysia

IAM METU

Best Paper Award

This is to certify, that the paper entitled **“Advanced Targeting Cost Function Design for Evolutionary Optimization of Logistic Equation”** from the Authors:

Roman Senkerik, Ivan Zelinka, Donald Davendra and Zuzana Oplatkova

was awarded as a best paper for the *Modelling and System Optimization track*.

[illegible]

Nader Barsoum

PCO 2010 Chairman

LINDO SYSTEMS INC. ILOG Changing the rules of business AMERICAN INSTITUTE OF PHYSICS Hindawi IOS Press

ELSEVIER IAENG INDERSCIENCE PUBLISHERS www.inderscience.com SpringerLink EUROPT BENTHAM SCIENCE

Appendix 2

List of publications - Zuzana Komínková Oplatková

status to 3rd December 2012

Overview of total number:

Textbooks: 2

Books: 3

Chapters in books: 8

Editor of books: 2

Journals with impact factor: 5

Journals: 13

Editor of konference proceedings: 2

Conference proceedings: 76

Overview of databases records:

ISI / WoS

Number of records: 33

Number of citations: 8

Number of citations without self-citations: 4

H-index: 2

SCOPUS

Number of records: 39

Number of citations: 38

Number of citations without self-citations: 18

H-index: 3

Google Scholar

Number of records: 79

Number of citations WITh self-citations: 256

H-index: 10

References in details:

Textbooks – education texts

1. Oplatková Z., Volná E.: Analytické programování, distanční opora, Ostravská univerzita, 2012
2. Zelinka I., Oplatková Z., Šenkeřík R.: Applications of artificial intelligence – czech edition, Aplikace umělé inteligence (aneb vybrané statě z evolučních algoritmů), Tomas Bata University in Zlin, Czech Republic, 2010, ISBN 978-80-7318-898-6.

Books

1. Hološka J., Komínková Oplatková Z.: Steganalysis by means of Artificial Neural Networks, Lambert Academic Publishing, Saarbrücken, 2012, ISBN 978-3-659-30172-8
2. Oplatková, Z.: Metaevolution: Synthesis of Optimization Algorithms by means of Symbolic Regression and Evolutionary Algorithms, Lambert Academic Publishing, Saarbrücken, 2009, ISBN 978-3-8383-1808-0.
3. Zelinka, I., Oplatková, Z., Ošmera, P., Šeda, M., Včelař, F.: Evoluční výpočetní techniky - principy a aplikace (Czech editon – Evolutionary computation techniques), BEN - technická literatura, Ben - technická literatura, Praha, 2008, ISBN 80-7300-218-3.

Chapter in books

1. Oplatkova, Z., Holoska J., Prochazka M., Senkerik, R., Jasek, R., Optimization of Artificial Neural Network Structure in the Case of Steganalysis , In: Springer Series "Inteligent Systems" - "Handbook of Optimization", (Ivan Zelinka, Vaclav Snasel, Ajith Abraham(Eds.)), pp. 821 - 824, 2012, ISBN 978-3-642-30503-0
2. Senkerik, R., Oplatkova, Z., Zelinka, I., Davendra, D., Jasek, R., Application of Evolutionary Techniques for Optimization of Chaos Control – Introduction of Three Approaches , In: Springer Series "Inteligent Systems" - "Handbook of Optimization", (Ivan Zelinka, Vaclav Snasel, Ajith Abraham(Eds.)), p. 801-820, 2012, ISBN 978-3-642-30503-0.
3. Senkerik, R., Davendra, D., Zelinka, I., Oplatkova, Z., Influence of Chaotic Dynamics on the Performance of Differential Evolution Algorithm, In: Springer Series " Emergence, Complexity and Computation", (Ivan Zelinka, Ali Sanayei, Hector Zenil, Otto E. Rössler (Eds.)), in press, ISSN 2194-7287.
4. Senkerik R., Oplatkova Z., Zelinka I., Davendra D., Jasek R., Application of Analytic Programming for Evolutionary Synthesis of Control Law - Introduction of Two Approaches , In: Springer Series "Studies in Computational Intelligence" - "Advances in Intelligent Modelling and Simulation: Simulation Tools and

Applications", (Aleksander Byrski, Zuzana Oplatkova, Marco Carvalho and Marek Kisiel Dorohinicki (Eds.)), pp. 253 – 268, 2012, ISBN 978-3-642-28887-6.
5. Volná E., Janošek M., Kocián V., Kotyrba M., Oplatková Z.: Robotics System – Applications, Control and Programming. In Dutta Ashish: Methodology for System Adaptation based on Characteristic Patterns. Intech – Open Access Publisher. 2012. ISBN 978-953-307-941-7.
6. Oplatková Z., Šenkeřík R.: Applications of Artificial Intelligence, In (Šilhavý, Radek; Šilhavý, Petr; Prokopová, Zdenka (Eds.)): Computer Science and Software Techniques in 2011, Šilhavý s. r. o., pp. 29-42, 2011, ISBN 978-80-904741-0-9.
7. Zelinka, I., Davendra, D., Šenkeřík, R., Jašek, R., Oplatková, Z.: Analytical Programming - a Novel Approach for Evolutionary Synthesis of Symbolic Structures. In Evolutionary Algorithms. Rijeka : InTech, 2011, pp. 149-176. ISBN 978-953-307-171-8.
8. Šenkeřík, R., Zelinka, I., Davendra, D., Oplatková, Z.: Evolutionary Design of Chaos Control in 1D in Zelinka I., Celikovski S., Richter H., Chen G.: Evolutionary Algorithms and Chaotic Systems, Springer-Verlag Berlin, Heidelberg, pp. 165-190, 2010, ISBN 978-3-642-10706-1

Editor of books

1. Byrski A., Oplatkova Z., Carvalho M. and Dorohinicki M. K. (Eds.): Springer Series "Studies in Computational Intelligence" - "Advances in Intelligent Modelling and Simulation: Simulation Tools and Applications", Springer, 2012, ISBN: 978-3-642-28887-6.
2. Computer Science and Software Techniques in 2011, ISBN 978-80-904741-0-9

Journals with impact factor

1. Kominkova Oplatkova, Z., Senkerik, R., Zelinka, I., Pluhacek, M., Analytic Programming in the task of Evolutionary Synthesis of Controller for High Order Oscillations Stabilization of Discrete Chaotic Systems, Computers & Mathematics with Applications, (Accepted for publication, 2012), ISSN 0898-1221, **IF: 1.747** (2011)
2. Pluhacek, M., Senkerik, R., Davendra, D., Kominkova Oplatkova, Z., On the Behaviour and Performance of Chaos Driven PSO Algorithm with Inertia Weight, Computers & Mathematics with Applications, (Accepted for publication, 2012), ISSN 0898-1221, **IF: 1.747** (2011)
3. Senkerik, 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, Vol. 57, No. 1 - 2, 2013, pp. 57 – 67, ISSN 0895-7177, DOI: 10.1016/j.mcm.2011.05.030, **IF: 1.346** (2011)
4. Senkerik, R., Oplatkova, Z., Zelinka, I., Davendra, D., Jasek, R.: Performance Comparison of Differential Evolution and SOMA on Chaos Control Optimization

Problems. International Journal of Bifurcation and Chaos. Vol. 22, No. 8, 2012, 16 p, ISSN: 0218-1274, DOI: 10.1142/S021812741230025X, **IF: 0.755** (2011)
5. Senkerik, R., Zelinka, I., Davendra, D., Oplatkova, Z.: Utilization of SOMA and Differential Evolution for Robust Stabilization of Chaotic Logistic Equation, Computers & Mathematics with Applications, Vol. 60, No. 4, 2010, 1026-1037, ISSN 0898-1221, DOI: 10.1016/j.camwa.2010.03.059, **IF: 1.472** (2010)

Journals

2012

1. Kominkova Oplatkova Z., Holoska J., Senkerik R., Steganography content detection by means of feedforward neural network, International Journal of Innovative Computing and Applications, Vol. 5, No. 1, 2013, ISSN 1751-648X.
2. Senkerik, R., Zelinka, I., Davendra, D., Oplatkova, Z. Evolutionary Optimization Of Henon Map Control – A Blackbox Approach, International Journal of Operational Research, 2012, Vol. 13, No. 2, pp. 129 – 146, ISSN: 1745-7645.
3. Senkerik R., Oplatkova Z., Zelinka I., Davendra D., Evolutionary Chaos Controller Synthesis for Stabilizing Chaotic Hénon Maps. Complex Systems, 2012, Vol. 20, No. 3, pp. 205-214, ISSN 0891-2513.

2011

4. Oplatková, Z., Zatloukal, J., Šenkeřík, R.: Software Mathematica with GUI for Applications with Artficial Neural Networks, Aplimat – Journal of Applied Informatics, 2011, Vol. 4., No. 2, pp. 435-442, ISSN 1337-6365.
5. Bělašková S., Oplatková Z.: Application of Software Mathematica in Mathematics, Aplimat – Journal of Applied Informatics, 2011, Vol. 4., No. 1, p. 335-340, ISSN 1337-6365

2010

6. Senkerik, R., Zelinka, I., Oplatkova, Z.: Evolutionary Techniques for Control of Discrete Chaotic Logistic Equation, Journal of Communication and Computer, Vol. 7, No. 2, pp. 70-77, ISSN 1548-7709.
7. Hološka, J., Oplatková, Z., Zelinka, I., Šenkeřík, R.: Steganografie jako hrozba úniku kritických obchodních informací a její detekce pomocí umělých neuronových sítí, FAMily media, s. r. o., Security magazín, Praha, pp. 33-37, 2010, ISSN 1210-8723

2009

8. Oplatková, Z., Zelinka, I.: Investigation on Evolutionary Synthesis of Movement Commands, Hindawi Publishing Corporation, Modelling and Simulation in Engineering, New York, 2009, N/A, ISSN 1687-5591
9. Hološka, J., Oplatková, Z., Zelinka, I., Šenkeřík, R.: Steganografie jako hrozba úniku kritických obchodních informací a její detekce pomocí umělých neuronových sítí, Fakulta aplikované informatiky, Univerzity Tomáše Bati ve Zlíně, Trilobit, Vol 1., Zlín, 2009, ISSN 1804-1795

2008

10. Oplatková, Z., Zelinka, I., Šenkeřík, R.: Santa Fe Trail for Artificial Ant by means of Analytic Programming and Evolutionary Computation, United Kingdom Simulation Society, International Journal of Simulation, Systems, Science and Technology, Nottingham, 2008, pp. 20 - 33, Vol. 9, No. 3, ISSN 1473-8031.
11. Zelinka, I., Senkerik, R., Oplatkova, Z.: Evoluční řízení a syntéza deterministického chaosu, Československý časopis pro fyziku, Vol. 58, No. 6, 2008, pp. 316-325, ISSN: 0009-0700.
12. Zelinka, I., Šenkeřík, R., Oplatková, Z.: Evolutionary identification of dynamical systems, United Kingdom Simulation Society, International Journal of Simulation, Systems, Science and Technology, Nottingham, 2008, pp. 47-59, Vol. 9, No. 3, ISSN 1473-8031

2005

13. Zelinka, I., Oplatková, Z., Nolle, L.: Analytic Programming - Symbolic Regression by Means of Arbitrary Evolutionary Algorithms, IJSST, International Journal of Simulation, Systems, Science and Technology, Great Britain, 2005, pp. 44-56, ISSN 1473-8031

Editor of Conference Proceedings

1. Zelinka, I., Oplatková, Z., Orsoni A.: 21st European Conference on Modelling and Simulation – Simulations in United Europe, 2007 held on Prague, Czech Republic, printed in Germany, ISBN 978-0-9553018-2-7
2. Louca L. S., Chrysanthou Y., Oplatková Z., Al-Begain K.: 22nd European Conference on Modelling and Simulation, 2008 held on Nicosia, Cyprus, printed in Germany, ISBN 0-9553018-5-8

Conference Proceedings

2012

1. Oplatkova Z., Holoska J., Senkerik R., Artificial Neural Networks for Detection Cover and Stego Images, In: Proceedings of the Fifth International Conference on Bioinspired Optimization Methods and Their Applications, BIOMA 2012, pp. 281 - 290, 2012, ISBN 978-961-264-043-9.
2. Oplatkova, Z., Senkerik, R., Zelinka I., Davendra D., Utilization Of Analytic Programming For The Stabilization Of High Order Oscillations Of Chaotic Hénon Map. In Proceedings 26th European Conference on Modelling and Simulation ECMS 2012, pp. 410-418. ISBN 978-0-9564944-4-3.
3. Oplatkova Z., Senkerik R.: Metaevolution with Analytic Programming, 2nd Computer Science On-line Conference in 2012, Silhavy s.r.o., Vsetin, p. 2-13, 2012, ISBN 978-80-904741-1-6

4. Senkerik, R., Oplatkova, Z., Davendra, D., Zelinka, I., Evolutionary and Meta-evolutionary Approach for the Optimization of Chaos Control, In Proceedings of ICAISC 2012, Swarm and Evolutionary Computation, Springer Series "Lecture Notes in Computer Science", Volume: 7269, 2012, pp. 350 – 358, ISBN: 978-3-642-29352-8.
5. Senkerik R., Davendra D., Zelinka I., Pluhacek M., Oplatkova Z., An Investigation on the Chaos Driven Differential Evolution: An Initial Study, In: Proceedings of the Fifth International Conference on Bioinspired Optimization Methods and Their Applications, BIOMA 2012, pp. 185 - 194, 2012, ISBN 978-961-264-043-9.
6. Senkerik R., Davendra D, Zelinka I., Pluhacek M., Oplatkova Z., An Investigation On The Differential EvolutionDriven By Selected Discrete Chaotic Systems, In: Proceedings of the 18th International Conference on Soft Computing, MENDEL 2012, pp. 157 - 162, 2012, ISBN 978-80-214-4540-6.
7. Senkerik R., Oplatkova Z., Zelinka I., Evolutionary Synthesis of Rules for Stabilization of Selected Discrete Chaotic Systems, In: Proceedings of the 18th International Conference on Soft Computing, MENDEL 2012, pp. 150 - 156, 2012, ISBN 978-80-214-4540-6.
8. Senkerik R., Davendra D., Zelinka I., Oplatkova Z., Pluhacek M., Optimization of the Batch Reactor by Means of Chaos Driven Differential Evolution, In Proceedings of SOCO 2012: Soft Computing Models in Industrial and Environmental Applications, Springer Series: "Advances in Intelligent Systems and Computing", Vol. 188, 2012, pp. 93 – 102, ISBN: 978-3-642-32922-7.
9. Senkerik R., Oplatkova Z., Zelinka I., Evolutionary Synthesis of Control Rules by means of Analytic Programming for the Purpose of High Order Oscillations Stabilization of Evolutionary Synthesized Chaotic System, In Proceedings of Nostradamus 2012: International conference on prediction, modeling and analysis of complex systems, Springer Series: "Advances in Intelligent Systems and Computing", Vol. 192, 2012, pp. 191 – 202, ISBN: 978-3-642-33226-5.
10. Senkerik R., Davendra D., Zelinka I., Oplatkova, Z., Influence of Chaotic Dynamics to the Performance of Evolutionary Algorithms – An initial study. In Proceedings of 10th International Conference on Numerical Analysis and Applied Mathematics ICNAAM 2012, AIP Conf. Proc. Vol. 1479, 2012, pp. 627-630, ISBN: 978-0-7354-1091-6.
11. Pluhacek M., Budikova V., Senkerik R., Oplatkova Z., Zelinka I., Extended Initial Study on the Performance of Enhanced PSO Algorithm with Lozi Chaotic Map, In Proceedings of Nostradamus 2012: International conference on prediction, modeling and analysis of complex systems, Springer Series: "Advances in Intelligent Systems and Computing", Vol. 192, 2012, pp. 167 – 178, ISBN: 978-3-642-33226-5.
12. Pluhacek M., Budikova V., Senkerik R., Oplatkova Z., Zelinka I., On The Performance Of Enhanced PSO algorithm With Lozi Chaotic Map – An Initial Study, In: Proceedings of the 18th International Conference on Soft Computing, MENDEL 2012, pp. 40 - 45, 2012, ISBN 978-80-214-4540-6.
13. Zelinka I., Skanderova L., Davendra D., Senkerik R., Oplatkova Z., Evolutionary Dynamics and Complex Networks, In: Proceedings of the 18th International Conference on Soft Computing, MENDEL 2012, pp. 88 - 93, 2012, ISBN 978-80-214-4540-6.

14. Zelinka I., Skanderova, L., Davendra D., Senkerik R., Oplatkova, Z., Controlling Complexity. In Proceedings of 10th International Conference on Numerical Analysis and Applied Mathematics ICNAAM 2012, AIP Conf. Proc. Vol. 1479, 2012, pp. 654-657, ISBN: 978-0-7354-1091-6.

2011

15. Oplatková, Z., Zatloukal, J., Šenkeřík, R.: Software Mathematica with GUI for Applications with Artficial Neural Networks, Aplimat 2011, STU, Bratislava, Slovak Republic, 2011, ISBN 978-80-89313-51-8.
16. Oplatkova Z., Senkerik R., Zelinka I., Jasek R.: Comparison of Two Cost Functions For Evolutionary Synthesis of Control Law for Higher Periodic Chaotic Logistic Equation. In 5th International Conference on Mathematical Modeling and Computer Simulation. IEEE. 2011. ISBN 978-0-7695-4414-4. p. 26-31.
17. Oplatková, Z., Šenkeřík, R.: Classification with Pseudo Neural Networks Based On Evolutionary Symbolic Regression. In 2011 International Conference on P2P, Parallel, Grid, Cloud and Internet Compting. Piscataway : IEEE Operations Center, 2011, s. 396-401. ISBN 978-0-7695-4531-8.
18. Oplatková, Z., Hološka, J., Zelinka, I., Šenkeřík, R., Jašek, R.: Steganalysis of PQ Algorithm by means of Neural Networks. In Proceedings 25th European Conference on Modelling and Simulation ECMS 2011. Krakov, Poland : ECMS, 2011, s. 446-451. ISBN 978-0-9564944-2-9.
19. Oplatková, Z., Šenkeřík, R., Zelinka, I., Davendra, D. Jašek, R.: Evolutionary Synthesis of Controller for Stabilization of Synthesized Chaotic System Oscillations. In MENDEL 2011 17th International Conference on Soft Computing. Brno: VUT Brno, Czech Republic, 2011, s. 73-79. ISBN 978-80-214-4302-0.
20. Kotyrba M., Oplatkova Z., Volna E., Senkerik R., Kocian V., Janosek M.: Time Series Pattern Recognition via SoftComputing. In International Conference on P2P, Parallel, Grid, Cloud and Internet Computing, Workshop SMECS 2011. IEEE. 2011. ISBN 978-0-7695-4531-8. p. 384-389.
21. Bělašková S., Oplatková Z.: Application of Software Mathematica in Mathematics, Aplimat 2011, STU, Bratislava, Slovak Republic, 2011, ISBN 978-80-89313-51-8
22. Prochazka M., Oplatkova Z., Holoska J., Gerlich V.: Optimization of Neural Network Inputs by Feature Selection Methods. In 25th European Conference on Modelling and Simulation. European Council for Modelling and Simulation. 2011. ISBN 978-0-9564944-2-9. p. 440 – 445.
23. Skála, R., Oplatková, Z., Šenkeřík, R.: Diagnostics of Machining Center by means of Neural Networks. In MENDEL 2011 17th International Conference on Soft Computing. Brno : VUT Brno, Czech Republic, 2011, s. 207-212. ISBN 978-80-214-4302-0.
24. Šenkeřík, R., Oplatková, Z., Zelinka, I., Jašek, R.: Evolutionary Synthesis Of Control Law For Higher Periodic Orbits Of Chaotic Logistic Equation. In Proceedings 25th European Conference on Modelling and Simulation ECMS 2011. Krakov, Poland, ECMS, 2011, s. 452-458. ISBN 978-0-9564944-2-9.

25. Šenkeřík, R., Oplatková, Z., Zelinka, I., Davendra, D., Jašek, R.: Synthesis Of Feedback Control Law For Stabilization Of Chaotic System Oscillations By Means Of Analytic Programming – Preliminary Study. In Proceedings of The Fifth Global Conference on Power Control and Optimization. Melville : American Institute of Physics, 2011, s. 1-6. ISBN 978-983-44483-4-9.
26. Šenkeřík, R., Oplatková, Z., Zelinka, I., Davendra, D., Jašek, R.: Synthesis Of Feedback Controller For Stabilization Of Chaotic Hénon Map Oscillations By Means Of Analytic Programming. In Proceedings of EMSS: The European Modelling & Simulation Symposium 2011. Genova : University of Genova, Italy, 2011, s. 593-597. ISBN 978-88-903724-4-5.
27. Šenkeřík, R., Oplatková, Z., Zelinka, I.: Investigation on Evolutionary Chaos Controller Synthesis for Hénon Map Stabilization. In International Conference on Numerical Analysis and Applied Mathematics 2011. Melville : American Institute of Physics, 2011, s. 1027-1030. ISBN 978-0-7354-0954-5.

2010

28. Oplatková, Z., Šenkeřík, R., Zelinka, I., Hološka, J.: Synthesis of Control Law for Chaotic Logistic Equation - Preliminary Study, IEEE, 4th International Conference on Mathematical Modelling and Computer Simulation, Pretoria, 2010, 65-70, ISBN-ISSN 978-0-7695-4062-7
29. Oplatková, Z., Šenkeřík, R., Bělašková, S., Zelinka, I.: Synthesis of Control Rule for Synthesized Chaotic System by means of Evolutionary Techniques, VUT v Brně, 16th International Conference on Soft computing, Brno, Czech Republic, 2010, 91-98, ISBN-ISSN 978-80-214-4120-0
30. Oplatková, Z., Šenkeřík, R., Zelinka, I., Hološka, J.: Synthesis of Control Law for Chaotic Henon System - Preliminary Study, ECMS, Simulation Meets Global Challenges, Germany, 2010, 277-282, ISBN-ISSN 978-0-9564944-0-5
31. Hološka, J., Oplatková, Z., Zelinka, I., Šenkeřík, R.: Comparison Between Neural Network Steganalysis And Linear Classification Method Stegdetect, IEEE, 2nd International Conference on Computational Intelligence, Modelling and Simulation, Pretoria, 2010, 15-20, ISBN-ISSN 978-0-7695-4262-1
32. Procházka, M., Oplatková, Z., Hološka, J.: Datamining Optimization of Steganalysis by means of Neural Network, Univerzita Tomáše Bati ve Zlíně, Fakulta aplikované informatiky, Internet, bezpečnost a konkurenceschopnost organizací. Řízení procesů a využití moderních teerminálových technologií, Zlín, Czech Republic, 2010, 319-324, ISBN-ISSN 978-83-61645-16-0
33. Peleteiro, A., Burguillo-Rial, J., Oplatková, Z., Zelinka, I.: EPMAS: Evolutionary Programming Multi-agent Systems, ECMS, Simulation Meets Global Challenges, Germany, 2010, 27-33, ISBN-ISSN 978-0-9564944-0-5
34. Šenkeřík, R., Zelinka, I., Davendra, D., Oplatková, Z.: Advanced Targeting Cost Function Design For Evolutionary Optimization Of Control Of Logistic Equation, University of Technology, The 3rd Global Conference on Power Control and Optimization, Sarawak, Malaysia, 2010, 1-6, ISBN-ISSN 978-983-44483-1-8
35. Šenkeřík, R., Oplatková, Z., Zelinka, I., Davendra, D., Jašek, R.: Synthesis Of Feedback Controller For Chaotic Systems By Means Of Evolutionary Techniques,

Curtin University, Proceeding of Fourth Global Conference on Power Control and Optimization, Sarawak, Malaysia, 2010, 1-7, ISBN-ISSN 978-983-44483-32
36. Šenkeřík, R., Davendra, D., Zelinka, I., Oplatková, Z.: Chaos Driven Differential Evolution in the Task of Chaos Control Optimization, IEEE Control Systems Society, 2010 IEEE World congress on Computational Intelligence, 2010, 1408-1415, ISBN-ISSN 978-1-4244-6910-9
37. Zelinka, I., Davendra, D., Snášel, V., Jašek, R., Šenkeřík, R., Oplatková, Z.: Preliminary Investigation on Relations Between Complex Networks and Evolutionary Algorithms Dynamics, IEEE, International Conference on Computer Information Systems and Industrial Management Applications, 2010. CISIM 2010., Krakow, Poland, 2010, 148-153, ISBN-ISSN 978-1-4244-5612-3
38. Šenkeřík, R., Zelinka, I., Oplatková, Z.: Evolutionary Techniques for Deterministic Chaos Control, in Iskander M., Kapila V., Karim M. A.: Technological Developments in Education and Automation, Springer Berlín, 2010, 391-396, ISBN 978-90-481-3655-1

2009

39. Oplatková, Z., Hološka, J., Zelinka, I., Šenkeřík, R.: Odhalování steganografie pomocí neuronových sítí, Univerzita Tomáše Bati ve Zlíně, Internet, competitiveness and Organisational Security in Knowledge Society, Zlín, Czech Republic, 2009, ISBN-ISSN 978-80-7318-828-3
40. Oplatková, Z., Hološka, J., Zelinka, I., Šenkeřík, R.: Detection of Stegoimages by F5 Programme by means of Neural Networks, VUT Brno, 15th International Conference on Soft Computing MENDEL 2009, Brno, Czech Republic, 2009, 186-191, ISBN-ISSN 978-80-214-3884-2
41. Oplatková, Z., Hološka, J., Zelinka, I., Šenkeřík, R.: Detection of Steganography Inserted by OutGuess and Steghide by means of Neural Networks, IEEE Operations Center, AMS2009 Asia Modelling Symposium 2009, Piscataway, 2009, 11-17, ISBN-ISSN 978-0-7695-3648-4
42. Tupý, J., Oplatková, Z., Zelinka, I.: Neural Differentiation in Modeling, Vysoké učení technické v Brně, Fakulta strojního inženýrství, MENDEL 2009 15th International Coference on Soft Computing, Brno, Czech Republic, 2009, 154-159, ISBN-ISSN 978-80-214-3675-6
43. Zelinka, I., Šenkeřík, R., Oplatková, Z., Davendra, D.: Evolutionary Identification of Chaotic System, International Federation of Automatic Control, 2nd IFAC Conference on Analysis and Control of Chaotic Systems, Soul, Korea, 2009, 1-8, ISBN-ISSN 1474-6670
44. Šenkeřík, R., Zelinka, I., Oplatková, Z.: Evoluční techniky v řízení deterministického chaosu, Univerzita Tomáše Bati ve Zlíně, Internet, competitiveness and Organisational Security in Knowledge Society, Zlín, Czech Republic, 2009, 1-6, ISBN-ISSN 978-80-7318-828-3
45. Šenkeřík, R., Zelinka, I., Oplatková, Z.: Comparison of Evolutionary Algorithms in the Task of Chaos Control Optimization ? Extended study: Complex Targeting CF, IEEE Control Systems Society, 2009 IEEE Congress on Evolutionary Computation, Reseach Publishing Service - Chennai, 2009, 2825 - 2832, ISBN-ISSN 978-1-4244-2959-2

46. Šenkeřík, R., Zelinka, I., Oplatková, Z., Davendra, D.: Evolutionary Synthesis and Control of Chaotic Systems, International Federation of Automatic Control, 2nd IFAC Conference on Analysis and Control of Chaotic Systems, Soul, 2009, 1-6, ISBN-ISSN 1474-6670
47. Šenkeřík, R., Zelinka, I., Oplatková, Z.: Optimal Control of Evolutionary Synthesized Chaotic System, VUT Brno, 15th International Conference on Soft Computing MENDEL 2009, Brno, Czech Republic, 2009, 220 - 227, ISBN-ISSN 978-80-214-3884-2
48. Šenkeřík, R., Zelinka, I., Oplatková, Z.: Design of Advanced Targeting Cost Function for Evolutionary Optimization of Chaos Control, ECMS Sp., 23rd European Conference on Modelling and Simulation, Madrid, Spain, 2009, 122 - 128, ISBN-ISSN 978-0-9553018-8-9
49. Šenkeřík, R., Zelinka, I., Oplatková, Z.: Evolutionary Blackbox Control of Logistic Equation, EUCA, European Control Conference 2009, San Ramon, 2009, 2652 - 2657, ISBN-ISSN 978-963-311-369-1

2008

50. Oplatková, Z., Zelinka, I.: Applications of Symbolic Regression with Evolutionary Computation, University of Bielsko-Biala, European Workshop on Intelligent computational Methods and Applied Mathematics, Bielsko-Biala, Poland, 2008, ISBN-ISSN N
51. Oplatková, Z., Zelinka, I.: Higher Dimensional Cost Function for Synthesis of Evolutionary Algorithms by means of Symbolic Regression, IEEE Operations Center, Second Asia International Conference on Modelling and Simulation, Piscataway, 2008, 486-492, ISBN-ISSN 0-7695-2799-X
52. Oplatková, Z., Hološka, J., Zelinka, I., Šenkeřík, R.: Detection of Steganography Content Inserted by Steghide by means of Neural Networks, Vysoké učení technické v Brně, Fakulta strojního inženýrství, MENDEL 2008 14th International Coference on Soft Computing, Brno, Czech Republic, 2008, 166-171, ISBN-ISSN 978-80-214-3675-6
53. Oplatková, Z., Hološka, J., Zelinka, I., Šenkeřík, R.: Steganography Detection by means of Neural Networks, IEEE Operations Center, Nineteenth International Workshop on Database and Expert Systems Applications, Piscataway, 2008, 571-576, ISBN-ISSN 978-0-7695-3299-8
54. Zelinka, I., Oplatková, Z., Šenkeřík, R.: Evolutionary Synthesis of Complex Structures, IEEE Operations Center, Nineteenth International Workshop on Database and Expert Systems Applications, Piscataway, 2008, 571-576, ISBN-ISSN 978-0-7695-3299-8
55. Zelinka, I., Oplatková, Z., Šenkeřík, R.: Evolutionary Scanning and Neural Network Optimization, IEEE Operations Center, Nineteenth International Workshop on Database and Expert Systems Applications, Piscataway, 2008, 571-576, ISBN-ISSN 978-0-7695-3299-8
56. Šenkeřík, R., Zelinka, I., Oplatková, Z.: Performance Comparison of Evolutionary Algorithms in the Task of Optimization of Chaos Control, IEEE Operations Center, Nineteenth International Workshop on Database and Expert

Systems Applications, IEEE, Piscataway, 2008, 514 - 518, ISBN-ISSN 978-0-7695-3299-8
57. Šenkeřík, R., Zelinka, I., Oplatková, Z.: Evolutionary Techniques for Deterministic Chaos Control, IEEE Computer Society, International Joint Conferences on computer, Information, and Systems Sciences, and Engineering, Bridgeport, USA, 2008, 500 - 505
58. Prokopová, Z., Oplatková, Z.: Simulation and Robust Control of Continuous Time Circulating Reactor, ECMS, 22nd European Conference on Modelling and Simulation, Nicosia, Cyprus, 2008, 519-524, ISBN-ISSN 978-0-9553018-5-8

2007

59. Oplatková, Z., Zelinka, I.: Santa Fe Trail for Artificial Ant with Analytic Programming and Three Evolutionary Algorithms, IEEE Computer Society, AMS 2007, Phuket, Thailand, 2007, 334-339, ISBN-ISSN 0-7695-2845-7
60. Oplatková, Z., Zelinka, I.: Creating evolutionary algorithms by means of analytic programming - design of new cost function, European Council for Modelling and Simulation, ECMS 2007, Germany, 2007, 271-276, ISBN-ISSN 978-0-9553018-2-7
61. Oplatková, Z., Zelinka, I.: Symbolic regression and evolutionary computation in setting an optimal trajectory for a robot, IEEE Computer Society, workshop ETID 2007 in DEXA 2007, Germany, 2007, 168-172, ISBN-ISSN 978-0-7695-2932-5
62. Oplatková, Z., Zelinka, I.: Learning of robots via symbolic regression and evolutionary computation, Vysoká škola báňská - Technická univerzita, WETDAP 2007 - Znalosti 2007, Ostrava, Czech Republic, 2007, 27-34, ISBN-ISSN 978-80-248-1332-5

2006

63. Oplatková, Z., Zelinka, I.: Santa Fe Trail for Artificial Ant with Simulating Annealing – Preliminary Study, European Council for Modelling and Simulation, 20th European Conference on Modelling and Simulation, Germany, 2006, 56-61, ISBN-ISSN 0-9553018-0-7
64. Oplatková, Z., Zelinka, I.: Investigation on Artificial Ant using Analytic Programming, The Association for Computing Machinery, Genetic and Evolutionary Computation Conference, USA, 2006, 949-950, ISBN-ISSN 1-59593-186-4
65. Oplatková, Z., Zelinka, I.: Setting an Optimal Trajectory by means of Analytic Programming, Faculty of Mechanical Engineering in Zenica, 10th International Research/Expert Conference & Trends in the Development of Machinery and Associated Technology, TMT 2006, Zenica, Bosna a Hercegovina, 2006, 673-676, ISBN-ISSN 9958-617-30-7
66. Oplatková, Z., Zelinka, I.: Creating Evolutionary Algorithms by means of Analytic Programming – Preliminary Study, FSI, VUT Brno, 12th International Conference on Soft Computing, Czech Republic, 2006, 19-24, ISBN-ISSN 80-214-3195-4
67. Zelinka, I., Vařacha, P., Oplatková, Z., Volná, E.: Structural Synthesis of Neural Network by means of Analytic Programming, FSI, VUT Brno, 12th

International Conference on Soft Computing, Czech Republic, 2006, 25-30, ISBN-ISSN 80-214-3195-4
68. Zelinka, I., Vojtěšek, J., Oplatková, Z.: Simulation Study of the CSTR Reactor for Control Purposes, European Council for Modelling and Simulation, 20th European Conference on Modelling and Simulation, Germany, 2006, 479-482, ISBN-ISSN 0-9553018-0-7

2005

69. Oplatková, Z.: Optimal Trajectory of Robots Using Symbolic Regression, IAC, IAC 2005, International Congress, Fukuoka, Japan, 2005, CDROM, ISBN-ISSN
70. Oplatková, Z., Zelinka, I.: Investigation on Shannon - Kotelnik Theorem Impact on SOMA Algorithm Performance, ESM, European Simulation Multiconference 2005, Riga, Latvia, 2005, 66-71, ISBN-ISSN 1-84233-112-4

2004

71. Oplatková, Z.: Analytic Programming - Boolean Symmetry Problems by Means of Evolutionary Algorithms, Ústav informatiky SAV, Ústav informatiky, Bratislava, Slovak Republic, 2004, 38-41, ISBN-ISSN 80-969202-0-0
72. Zelinka, I., Oplatková, Z., Včelař, F.: Logical Function Synthesis by Means of Arbitrarry Evolutionary Algorithms-Comparative Study, VUT FSI, 10 th International Conference on Soft Computing, Brno, Czech Republic, 2004, 77-82, ISBN-ISSN 80-214-2676-4
73. Zelinka, I., Oplatková, Z.: Boolean Parity Function Synthesis by Means of Arbitrary Evoutionary Algorithms - Comparative Study, SCI 2004, SCI 2004, Orlando, USA, 2004, 231-236, ISBN-ISSN 980-6560-13-2
74. Zelinka, I., Oplatková, Z., Nolle, L.: Boolean Symmetry Function Synthesis by means of Arbitrary Evolutionary Algorithms – Comparative Study, Society for Modeling and Simulation International, European Simlulation Multiconference, Magdeburg, Germany, 2004, 143-148, ISBN-ISSN 3-936150-35-4
75. Zelinka, I., Oplatková, Z., Včelař, F.: Symbolic Regression by means of Arbitrary Evolutionary Algorithms – Comparative Study, Technical University in Brno, Czech Republic, Mendel 2004 – International conference on Softcomputing, Brno, 2004, 77-82, ISBN-ISSN 80-214-2676-4

2003

76. Zelinka I., Oplatková Z.: Analytic Programming – Comparative Study, The second International Conference on Computational Intelligence Robotics and Autonomous Systems CIRAS, 2003, Singapore, ISSN 0219-6131