BOOLEAN SYMMETRY FUNCTION SYNTHESIS BY MEANS OF ARBITRARY EVOLUTIONARY ALGORITHMS - COMPARATIVE STUDY

Ivan Zelinka, Zuzana Oplakova
Institute of Control Processing and Information Technologies
Faculty of Technology
Tomas Bata University in Zlin
Mostni 5139 Zlin, Czech Republic
{zelinka,oplatkova}@ft.utb.cz

Lars Nolle
School of Computing and Mathematics
The Nottingham Trent University
Burton Street
Nottingham, NG1 4BU, UK
Lars.nolle@ntu.ac.uk

KEYWORDS
symbolic regression, genetic programming, grammar evolution, analytic programming, optimisation, SOMA

ABSTRACT

This contribution introduces analytical programming, a novel method that allows solving various problems from the symbolic regression domain. Symbolic regression was firstly proposed by J. R. Koza in his genetic programming and by C. Ryan for grammatical evolution. This contribution explains the main principles of analytic programming, and demonstrates its ability to synthesise suitable solutions, called programs. It is then compared with genetic programming and grammatical evolution. This comparative study is concerned with three Boolean k-symmetry problems from Koza's genetic programming domain, which are solved by means of analytical programming. Here, two evolutionary algorithms are used with analytical programming: differential evolution and self-organizing migrating algorithm.

INTRODUCTION

The term *symbolic regression* (SR) represents a process in which measured data is fitted by a suitable mathematical formula like $x^2 + C$, $\sin(x)+1/e^x$, etc. Amongst mathematicians, this process is quite well known and can be used when data of an unknown process can be obtained. For long time, SR was only the domain of humans but for the few last decades it has also become the domain of computers. Today there are two methods, which can be used for SR by means of computers. The first one is called genetic programming (GP) (Koza 1998), (Koza et al 1999) and the second one is grammatical evolution (O'Neill and Ryan 2002), (Ryan et al.1998).

The idea of how to solve various problems using SR by the means of evolutionary algorithms (EA) was introduced by from John Koza who used genetic algorithms (GA) for GP. Genetic programming is basically a symbolic regression, which is done by using evolutionary algorithms instead of a human being. The ability to solve really hard problems was proved many times, and hence, GP today performs so well that it can be applied, e.g. to synthesise highly sophisticated electronic circuits (Koza et al. 2003).

In the last decade of the 20th century, a novel method for SR was developed by C.Ryan, which is called grammatical evolution (GE). Grammatical evolution can be regarded as an unfolding of GP, because of some common principles, which are the same for both algorithms. One important characteristics of GE is, for example, the fact that GE can be implemented in any arbitrary computer language compared with GP, which is usually done in LISP. In contrast to other evolutionary algorithms, GE was used only with a few search strategies, with a binary representation of the populations (O'Sullivan and Ryan, 2002). Another interesting investigation using symbolic regression was carried out by (Johnson) working on Artificial Immune Systems.

In this paper, a novel method is presented which was developed, called analytical programming (AP) (Zelinka 2002 a), (Zelinka 2002 b), (Zelinka and Oplatkova, 2003) and (Zelinka and Oplatkova, 2004). AP is also a tool for symbolic regression, based on different principles compared to GP and GE. The important principles of AP, together with tests and a comparison with GP, are documented in this contribution.

ANALYTIC PROGRAMMING

Term *analytic programming* was coined by the authors of this article as a matter of simplicity: Because it is possible to use almost any evolutionary algorithm for AP, each EA used for the new approach would add its name to the emerging algorithm, e.g. SOMA programming, DE programming, SA programming etc. This clearly would be confusing and complicated. Analytic programming indicates the use of an EA for analytic solutions synthesis (i.e. symbolic regression). That is the main reason for choosing the term 'analytic programming'.

Analytic programming was inspired by the methods of variations in Hilbert spaces and by GP. The principles of AP are somewhere between these two philosophies: From GP stems the idea of the evolutionary creation of symbolic solutions, whereas the ideas of functional spaces and the building of resulting function by means of search process (usually done by numerical methods like the Ritz or Galerkin method) is adopted from Hilbert spaces. AP is based, as well as GP, on a set of functions, operators and so-called terminals, which are usually constants or independent variables, for example:

- functions: Sin, Tan, Tanh, And, Or
- operators: +, -, *, /, dt,…
- terminals: 2.73, 3.14, t,…

All these 'mathematical' objects create a set from which AP tries to synthesise an appropriate solution. The main principle of AP is based on discrete set handling, proposed in (Zelinka 2004) (see Figure 1 and Figure 2). Discrete set handling itself can be seen as a universal interface between EA and the problem to be solved symbolically. That is why AP can be carried out using almost any evolutionary algorithm. Analytical programming, together with a few basic examples, is for example discussed in more detail in (Zelinka and Oplatkova, 2003).

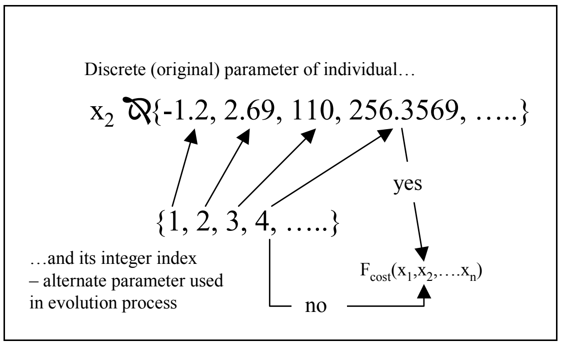
Discrete (original) parameter of individual…
x2 ∈ {-1.2, 2.69, 110, 256.3569, …..}
yes
{1, 2, 3, 4, …..}
…and its integer index – alternate parameter used in evolution process
Fcost(x1,x2,….xn)
no

Figure 1: Discrete set handling.

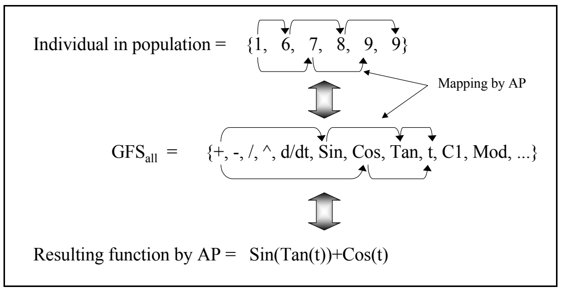
Individual in population = {1, 6, 7, 8, 9, 9}
Mapping by AP
GFSall = {+, -, /, ^, d/dt, Sin, Cos, Tan, t, C1, Mod, ...}
Resulting function by AP = Sin(Tan(t))+Cos(t)

Figure 2: Main principle of AP.

Briefly said, in AP, individual consist of non-numerical expressions (operators, functions,…) as described above, which are in the evolutionary process represented by their integer indexes (Figure 1 and 2). This index then serves like a pointer into the set of expressions and AP uses it to synthesise the resulting function-program for cost function evaluation (Zelinka 2004). Analytic programming exists so far in three versions. All three versions use for program synthesis the same sets of functions, terminals, etc., as Koza uses in GP (Koza 1998), (Koza et al 1999). The second version (AP_{meta}, lets call first version AP_{basic}) is modified in the sense of constant estimation. For example, Koza uses for the so-called sextic problem (Koza 1998) randomly generated constants, whereas AP here use only one, called K which is inserted into a formula at various places by the evolutionary process. When a program is synthesised, then all K's are indexed so that K_1, K_2, …, K_n, are in the formula obtained, and then all K_n are estimated using a second evolutionary algorithm. Because EA (slave) "works under" EA (master, i.e. EA_{master} ► program ► K indexing ► EA_{slave} ► estimation of K_n) then this version is called AP with metaevolution - AP_{meta}. Because this version is quite time consuming, AP_{meta} was modified to the third version, which differ from the second one in the estimation of K. This is done by using a suitable method for non-linear fitting (AP_{nf}). This method has shown the most promising performance when unknown constants are present, results of some comparative simulations can be found in (Zelinka and Oplatkova, 2003). For the simulations described here, AP_{basic} was used.

PROBLEM DESIGN

Problem selection

The class of booleans k-symmetry problems was chosen for this comparative study, based on case studies reported in (Zelinka and Oplatkova 2003) and (Zelinka and Oplatkova 2004), namely 3-symmetry, 4-symmetry and 5-symmetry problems. In general, boolean symmetry problems mean that if input values to a system are symmetric, then the output is True. If the input is not symmetrically then the output is False. The number of all possible inputs (combinations) is 8 (3-symmetry), 16 (4-symmetry) and 32 (5-symmetry). An example of the truth table for 3-symmetry problems is given in Table 1. For truth tables for other problems solved by AP see `www.ft.utb.cz/people/zelinka/ap`.

Table 1: Truth table for Boolean 3-symmetry problem according to (Koza 1998)

Input 1	Input 2	Input 3	Output
True	True	True	True
True	True	False	False
True	False	True	True
False	True	True	False
True	False	False	False
False	True	False	True
False	False	True	False
False	False	False	True

All simulations were based on a set of logic functions And, Nand, Or, Nor, and the needed number of inputs A, B, C, ...

The Fitness Function

The fitness (cost function) has been calculated using the Hamming distance between truth table output and synthesised candidate program (1). The theoretical maximum value (the worst solution of all) of this cost function is 8 for a 3-symmetry problem, 16 for a 4-symmetry problem, and 32 in case of 5–symmetry problems. The minimal value (the best solution) is 0 for all k-symmetry problems. The aim of all the simulations was to find the best solution, i.e. a solution that returns the cost value 0. For numerical calculations, False and True were replaced by 0 and 1.

$$f_{\cos t} = \sum_{i=1}^{2^n} \left| TT_i - P_i \right|$$

TT_i - i th output of truth table (1)

P_i - i th output of synthesised program

Optimisation Algorithm Used

For the experiments described here, stochastic optimisation algorithms, such as Differential Evolution (DE) (Price 1999) and SelfOrganizing Migrating Algorithm (SOMA) (Zelinka 2004), had been used. Alternative algorithms, like Genetic Algorithms (GA) and Simulated Annealing (SA), are now in process, and results are hoped to be presented soon. For an exact description of the algorithms use see (Price 1999) for DE and (Zelinka 2004) for SOMA.

EXPERIMENTAL RESULTS

Both algorithms (SOMA, DE) have been applied 50 times in order to find the optimum of all boolean k-symmetry problems. The primary aim of this comparative study is not to show which algorithm is better and worst, but to show that AP can be really used for different problems of symbolic regression by different EAs (based also on previous comparative studies and case studies (Zelinka 2002 a), (Zelinka 2002 b), (Zelinka and Oplatkova, 2003) and (Zelinka and Oplatkova, 2004)).
The control parameter settings have been found empirically and are given in Table 2 (SOMA) and Table 3 (DE). The main criterion for this setting was to keep the same setting of parameters as much as possible and of course the same number of cost function evaluations as well as population size (parameter PopSize for SOMA, NP for DE).
Outputs of all simulations are depicted in Figures and numerically reported in Tables 4 - 9. Figures 3, 4, 6, 7, 9 and 10 show results of all 50 simulations for each k-symmetry problem. Figures 5, 8 and 11 show a mutual comparison of algorithm performance in the point of view of the number of cost function evaluations. Length of synthesised programs is in tables marked as ‘leaf count’ (LC) – leafs are the elements of a program, i.e. And, Nor, Or, Input_A, etc.
Examples of typical solutions synthesised by both algorithms is represented by formulas (2) for a 3-symmetry problem, (3) for a 4-symmetry problem, and (4) for a 5-symmetry problem.

(2)

$$((A\overline{\vee}B)\overline{\wedge}(C\wedge B))\overline{\wedge}((C\overline{\vee}A)\overline{\vee}C\wedge C\wedge A)\vee(((C\wedge A)\overline{\wedge}(C\wedge A))\overline{\wedge}$$
$$(C\wedge A\wedge A))\overline{\wedge}((C\wedge A\vee C\wedge A)\overline{\wedge}A)$$

(3)

$$(D\overline{\wedge}D\overline{\vee}(B\overline{\wedge}(D\wedge C\wedge B\wedge A\vee\neg C\wedge\neg B\wedge D\wedge A\vee\neg D\wedge\neg A\wedge C\wedge B)\vee$$
$$D))\wedge C\vee((((D\wedge C\wedge B\wedge A\vee\neg C\wedge\neg B\wedge D\wedge A\vee\neg D\wedge\neg A\wedge C\wedge B)\wedge$$
$$B)\overline{\wedge}(D\wedge C\wedge B\wedge A\vee\neg C\wedge\neg B\wedge D\wedge A\vee\neg D\wedge\neg A\wedge C\wedge B))\overline{\wedge}(B\vee$$
$$C))\wedge((A\overline{\vee}D)\vee(D\wedge C\wedge B\wedge A\vee\neg C\wedge\neg B\wedge D\wedge A\vee\neg D\wedge\neg A\wedge C\wedge$$
$$B)\wedge(D\wedge C\wedge B\wedge A\vee\neg C\wedge\neg B\wedge D\wedge A\vee\neg D\wedge\neg A\wedge C\wedge B))$$

(4)

$$((B\vee E)\overline{\wedge}D\overline{\vee}((A\overline{\vee}A)\vee(E\wedge D\wedge B\wedge A\vee E\wedge D\wedge C\wedge A\vee\neg B\wedge E\wedge$$
$$C\wedge A\vee\neg D\wedge\neg B\wedge E\wedge A\vee\neg E\wedge\neg A\wedge D\wedge B\vee\neg E\wedge\neg D\wedge\neg B\wedge$$
$$\neg A)\wedge B))\overline{\vee}((D\overline{\wedge}(E\wedge D\wedge B\wedge A\vee E\wedge D\wedge C\wedge A\vee\neg B\wedge E\wedge C\wedge A\vee$$
$$\neg D\wedge\neg B\wedge E\wedge A\vee\neg E\wedge\neg A\wedge D\wedge B\vee\neg E\wedge\neg D\wedge\neg B\wedge\neg A))\wedge$$
$$(B\vee B)\vee(E\overline{\wedge}B)\wedge((E\wedge D\wedge B\wedge A\vee E\wedge D\wedge C\wedge A\vee\neg B\wedge E\wedge C\wedge A\vee$$
$$\neg D\wedge\neg B\wedge E\wedge A\vee\neg E\wedge\neg A\wedge D\wedge B\vee\neg E\wedge\neg D\wedge\neg B\wedge\neg A)\overline{\vee}B))$$

Table 2: SOMA setting for Boolean k-symmetry problems, k=3,4,5

	3	4	5
PathLength	3	3	3
Step	0.11	0.11	0.3
PRT	0.1	0.1	0.1
PopSize	300	300	300
Migrations	30	30	85
MinDiv	-0.1	-0.1	-0.1
Individual Length	30	30	30

Table 3: DE setting for Boolean k-symmetry problems, k=3,4,5

	3	4	5
NP	300	300	300
F	0.8	0.8	0.8
CR	0.2	0.2	0.2
Generations	800	800	2000
Individual Length	30	30	30

Table 4: Boolean 3-symmetry problem by SOMA

	Cost Function Evaluations	Leaf Count
Minimum	202	31
Average	4769	81
Maximum	14991	165

Table 5: Boolean 3-symmetry problem by DE

	Cost Function Evaluations	Leaf Count
Minimum	79	28
Average	2439	83
Maximum	8130	142

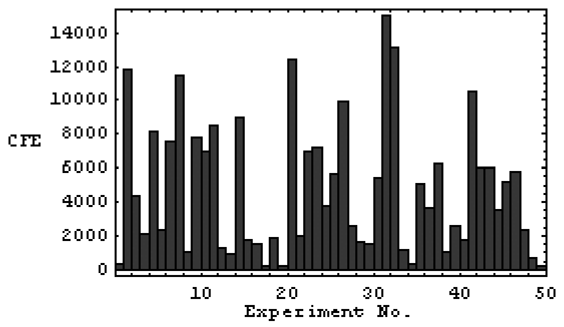

Figure 3: 3-symmetry by SOMA for 50 successful hits out of 50.

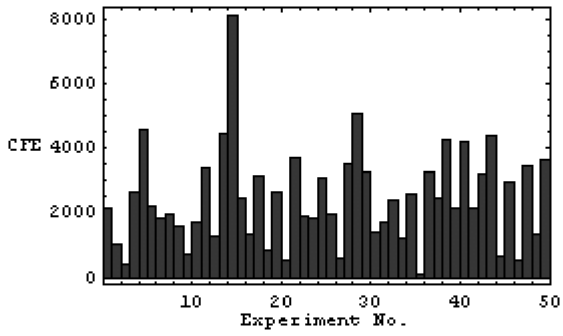

Figure 4: 3-symmetry by DE for 50 successful hits out of 50.

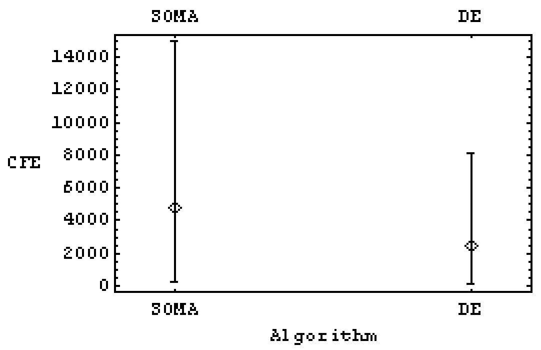

Figure 5: Algorithm performance of 3-symmetry by SOMA and DE.

Table 6: Boolean 4-symmetry problem by SOMA

	Cost Function Evaluations	Leaf Count
Minimum	23427	81
Average	84872	130
Maximum	185732	230

Table 7: Boolean 4-symmetry problem by DE

	Cost Function Evaluations	Leaf Count
Minimum	13696	50
Average	47790	133
Maximum	114709	206

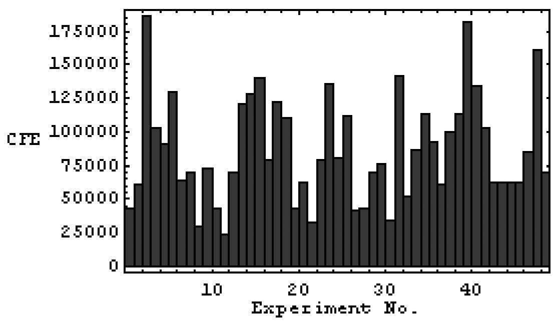

Figure 6: 4-symmetry by SOMA for 49 successful hits out of 50.

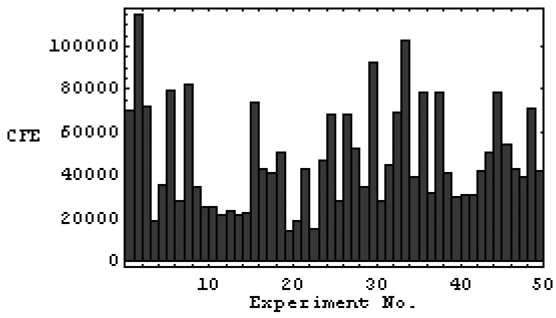

Figure 7: 4-symmetry by DE for 50 successful hits out of 50.

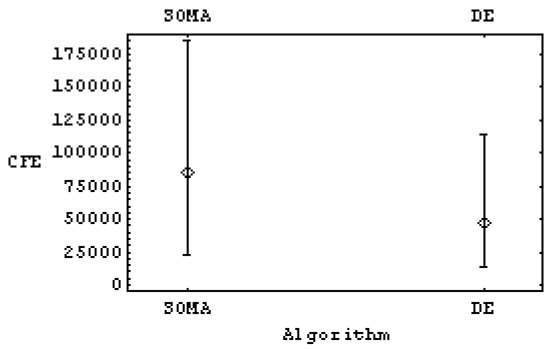

Figure 8: Algorithm performance of 4-symmetry by SOMA and DE

Table 8: Boolean 5-symmetry problem by SOMA

	Cost Function Evaluations	Leaf Count
Minimum	40029	48
Average	119903	170
Maximum	225972	284

Table 9: Boolean 5-symmetry problem by DE

	Cost Function Evaluations	Leaf Count
Minimum	58273	70
Average	172084	176
Maximum	363432	369

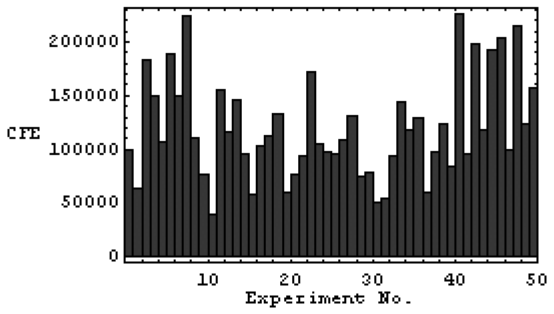

Figure 9: 5-symmetry by SOMA for 50 successful hits out of 50.

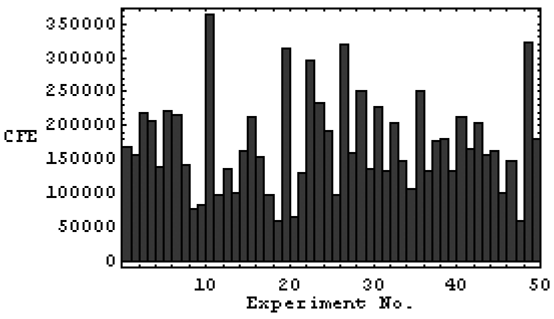

Figure 10: 5-symmetry by DE for 50 successful hits out of 50.

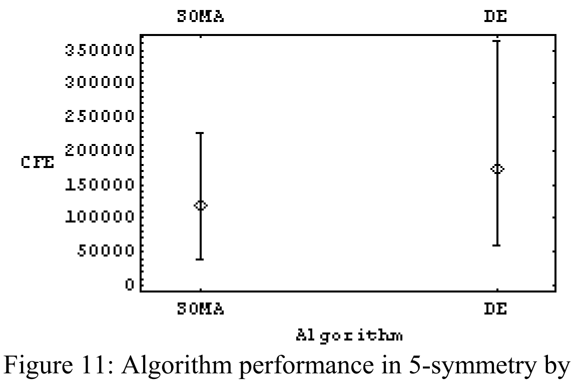

Figure 11: Algorithm performance in 5-symmetry by SOMA and DE

CONCLUSIONS

The method of analytic programming described here is relatively simple, easy to implement and easy to use. Based on its principles and its universality (it was tested with 4 evolutionary algorithms – SA, GA, SOMA and DE) it can be stated that AP is a metha algorithm rather than an algorithm itself.

The main aim of this paper was to show how various boolean k-symmetry problems were solved in the past (using GP), and how they can be solved by means of evolutionary algorithms applied in AP. Analytic programming was used here in three basic comparative simulations. Each comparative simulation was 50 times repeated and all 450 results (50 simulations for each algorithm and for each problem) were used to create graphs and tables for AP performance evaluation.

For the comparative study two algorithms were used - DE (Price 1999) and SOMA (Zelinka 2004). Using a wide variety of optimisation algorithm, i.e. with different structure and their different ability to locate global extreme, were chosen to prove that AP can be regarded as an equivalent to GP, and that it can be implemented using arbitrary evolutionary algorithms. As a conclusion the following statements are presented:

1. **Reduction of cost function evaluation.** During the simulations described here, a significant low number of cost function evaluations, needed to reach the optimal solution, were observed. While for GP usually 600000 cost function evaluations were needed, as reported in (Koza 1998) for 5-symmetry, AP usually needed 40029 – 363432 evaluations (see Table 8 and 9).
2. **Reduction of population size.** In all simulations only 300 individuals were used. When comparing the population size (4000 and 16000), used in GP as mentioned in (Koza 1998), then AP uses population with 133 - 533 times less individuals. This is probably another reason for the low number of cost function evaluations (see previous point).
3. **Reached results.** Based on results reported in Tables 2 – 9 and Figures 3 - 11 it can be stated that all simulations give satisfactory results and thus AP is capable of solving this class of problems.
4. **Mutual comparison.** When comparing both algorithms, then it is visible that both algorithms give good results. Parameter setting for both algorithms were based on heuristically approach and thus there is a possibility that better settings can be found there.
5. **Universality.** AP was used to solve differential equations (Zelinka 2002 b), trigonometrically data fitting (Zelinka 2002 a), four polynomial problems from (Koza 1998) (Sextic, Quintic,

Sinus Three, Sinus Four) by four EAs in (Zelinka and Oplatkova, 2003) and Boolean even-k-parity functions synthesis (Zelinka and Oplatkova, 2004). Together with the results for Boolean k-symmetry functions reported here it can be stated that AP is a universal method for symbolic regression by means of arbitrary EAs.

Future research is one of the key activities in the frame of AP. According to all results obtained during time it is planned that the main activities would be focused expanding of this comparative study for genetic algorithms and simulated annealing.

ACKNOWLEDGEMENT

This work was supported by grant No. MSM 26500014 of the Ministry of Education of the Czech Republic and by grants of the Grant Agency of the Czech Republic GACR 102/03/0070 and GACR 102/02/0204.

REFERENCES

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

Koza J. R., M. A. Keane, M. J. Streeter, 2003, Evolving Inventions, Scientific American, February 2003, p. 40-47, ISSN 0036-8733

Koza J.R. 1998, Genetic Programming II, MIT Press, ISBN 0-262-11189-6, 1998

Koza J.R.,Bennet F.H., Andre D., Keane M., 1999, Genetic Programming III, Morgan Kaufnamm pub., ISBN 1-55860-543-6, 1999

O'Neill M. and Ryan C. 2002, Grammatical Evolution. Evolutionary Automatic Programming in an Arbitrary Language. Kluwer Academic Publishers, ISBN 1402074441

O'Sullivan John, Conor Ryan, 2002, 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

Price K. 1999, An Introduction to Differential Evolution, in New Ideas in Optimization, D. Corne, M. Dorigo and F. Glover, Eds., s. 79–108, McGraw-Hill, London, UK, 1999. ISBN 007-709506-5

Ryan C., Collins J.J., O'Neill 1998, M. Grammatical Evolution: Evolving Programs for an Arbitrary Language. Lecture Notes in Computer Science 1391. First European Workshop on Genetic Programming 1998

Zelinka I., 2002 a, 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

Zelinka I., 2002 b, 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

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

Zelinka I., Oplatkova Z., 2004, Boolean Parity Function Synthesis by Means of Arbitrary Evolutionary Algorithms - Comparative Study", In: 8th World Multiconference on Systemics, Cybernetics and Informatics (SCI 2004), Orlando, USA, in July 18-21, 2004, in print

Zelinka Ivan, 2004, SOMA – Self Organizing Migrating Algorithm",Chapter 7, 33 p. in: B.V. Babu, G. Onwubolu (eds), New Optimization Techniques in Engineering, Springer-Verlag, ISBN 3-540-20167X

AUTHOR BIOGRAPHIES

IVAN ZELINKA was born in Czech Republic, and went to the Technical University of Brno, where he studied technical cybernetics and obtained his degree in 1995. He obtained Ph.D. degree in technical cybernetics in 2001 at Tomas Bata University in Zlin. Now he is senior lecturer (artificial intelligence, theory of information) and head of department. His e-mail address is: zelinka@ft.utb.cz and his Web-page can be found at http://www.ft.utb.cz/people/zelinka.

ZUZANA OPLATKOVA was born in Czech Republic, and went to the Tomas Bata University in Zlin, where she studied technical cybernetics and obtained her degree in 2003. She is now Ph.D. student. Her e-mail address is : oplatkova@ft.utb.cz and her Web-page can be found at www.zuzkaoplatkova.webz.cz

LARS NOLLE graduated from the University of Applied Science and Arts in Hanover in 1995 with a degree in Computer Science and Electronics. After receiving his PhD in Applied Computational Intelligence from The Open University, he worked as a System Engineer for EDS. He returned to The Open University as a Research Fellow in 2000. He joined The Nottingham Trent University as a Senior Lecturer in Computing in February 2002. His research interests include: applied computational intelligence, distributed systems, expert systems, optimisation and control of technical processes.