Comparative Evaluation of Algorithms for Automatic Construction of Nonlinear Models Based on Metaheuristic Programming with Gene Expression

Oleg Monakhov [1] and Emilia Monakhova [1]

[1] *Institute of Computational Mathematics and Mathematical Geophysics, SB RAS, pr. Lavrentiev, 6, Novosibirsk, 630090, Russia*

Abstract

A new algorithm of metaheuristic programming with gene expression based on various nature-inspired algorithms as a search engine for solving the problem of automatic synthesis of nonlinear models in an analytical form is proposed. We show that the proposed algorithm is superior to the standard gene expression programming algorithm.

Keywords

Metaheuristic programming, gene expression programming, nature-inspired algorithms, automatic construction of models

1. Introduction

The problem of generating and optimization of nonlinear models in the form of mathematical expressions using given sets of variables, basic functions, operations and experimental data is considered. It is necessary to find a mathematical expression *f* based on given sets of input *Y* and output *Z* experimental data, i.e. it is necessary to find a function $Z = f(Y)$ to represent the dependence of *Z* on *Y* with minimal error. This problem is also called symbolic regression or system identification.

The search for the expression *f* is performed based on the specified terminal set of variables and constants $T_1 = \{x_i, c_i\}$ and the specified set of basic functions and operations $F_1 = \{f_i \mid i \geq 1\}$. These sets are used to automatically generate analytical expressions (formulas) that represent the model. Let the objective function (fitness function) *FF* calculates the sum of the squares of the deviations of the given reference values Z_i from the output of the expression *f* for certain subsets input data of expression Y_i , $1 \leq i \leq N$:

$$FF = \sum_{i=1}^{N} (f(Y_i) - Z_i)^2$$

where *N* is the number of experimental data. For the automatically generated analytical expressions it is to determine the minimum of the residual: $min_{f \in D(F_1,T_1)} FF(f)$ where $D(F_1, T_1)$ is a set of expressions defined by a set of variables and basic functions.

Genetic Programming (GP) is one of the known approaches to this problem. They are focused primarily on automatic synthesis of programs based on training data using for search a genetic algorithm based on chromosomes with tree structures that, upon interpretation, represent some expressions. Gene Expression Programming (GEP) [1] is a member of the genetic programming family that is widely used in knowledge discovery and discovery of functions. GEP has a linear chromosome containing terminal and function characters in a fixed length string called a K-expression. K-expression can be translated into an individual phenotype as an expression tree using breadth-first traversal. The GEP search engine

Russian Advances in Fuzzy Systems and Soft Computing: Selected Contributions to the 10th International Conference «Integrated Models and Soft Computing in Artificial Intelligence» (IMSC-2021), May 17–20, 2021, Kolomna, Russian Federation
EMAIL: monakhov@rav.sscc.ru (A. 1)
ORCID:

is a genetic algorithm that works on the chromosomes of GEP to create new individuals through crossover, mutation and transposition operations.

In this work, we use a new approach of metaheuristic programming (MP) with gene expression for the synthesis of nonlinear models, which is based on the use of: (1) the genotype of nonlinear models is represented by a simple vector of real numbers (we do not use complex structures such as trees, networks or programs traditional for genetic programming), (2) new algorithms for transforming these vectors (genotype) into phenotype (in *K*-expressions, as in GEP, to represent a nonlinear model), (3) almost any metaheuristic, nature-inspired algorithm of the continuous global optimization [2, 3, 4] with simple search operators (rather than specialized operators for more complex structures) for search of optimal model, (4) a method of multi-variant coding of several solutions in one genotype.

Note that the MP method is a generalization and further development of the method of the multivariant evolutionary synthesis (MVES), proposed by the author [5]. The MP method is (1) more unified and versatile for use with many metaheuristic (bio- and nature-inspired) algorithms, and not only uses one discrete genetic algorithm (GA) or only one differential evolution (DE) as the MVES method, (2) uses a vector of real numbers as chromosome to represent phenotype with gene expressions, rather than in the form of a sequence of instructions, as in the MVES method, and, accordingly, (3) uses other operations to decode chromosomes.

In next section we describe the unified MP with gene expression algorithm with using fourteen different nature-inspired algorithms for automatic generation and optimization of nonlinear models. Then we compare our approach with standard gene expression programming and with the application of different nature-inspired algorithms and show that the MP algorithm has a higher efficiency of evolutionary search for many cases of used nature-inspired algorithms.

2. Metaheuristic programming with gene expression for synthesis of nonlinear models

Metaheuristic programming with gene expression for the synthesis of nonlinear models is based on modeling of nature-inspired processes in a population of individuals [2, 3, 4]. Individuals in the population are chromosomes, namely, vectors of real numbers that encode mathematical expressions and each chromosome in this algorithm defines a set of expressions (formulas) arising from it after decoding. The algorithm begins with the creation of an initial population by a random operator, then for each individual the fitness function are calculated by decoding the genotype (chromosome) into a phenotype (expression). For next generation, new individuals are created using migration operators (in the case of a standard genetic algorithm, migration operators are selection, mutation and crossover). The purpose of the migration operators is to move to the extremum of the objective function.

Metaheuristic programming algorithm presented in this paper is based on following main stages.

1. Initial population is randomly generated from vectors (chromosomes) with components in the real interval [0, 1].
2. The fitness of each individual of the population is computed after the decoding procedure of the genotype into a phenotype
3. A next generation of the population is created using migration operators of the nature-inspired algorithm (specified for each nature-inspired algorithm) to move individuals in the population to the extremum of the objective function
4. Item Repeating steps 2 and 3 until a solution is found that satisfies the specified criteria, or until the maximum number of generations is reached

In classic GEP [1] chromosomes (genes) are consisting of a head and a tail. The head consists of symbols representing functions (elements from the basic function set *F*) and terminals (elements from the terminal set *T*), whereas the tail consists of only terminals. The length of the head *h* is chosen for each problem, but the length of the tail *t* is evaluated by the equation $t = h\,(n - 1) + 1$, where *n* is the number of arguments of the function with the most arguments. Chromosomes of gene expression programming have all the same size, but encode the different mathematical expression. For example, for an expression $f(x, a) = (x + 2)/\sqrt{(ax - 5)}$ with the expression tree shown on Figure 1 we have the chromosome *{/+Qx2-*5 ax3axx4aa}*, based on standard breadth-first traversal of the expression

tree: $f(x,a) = (x+2)/\sqrt{(ax-5)}$ → *{/+Qx2-*5 ax3axx4aa}*, with head: *{/+Qx2-*5}*, tail: *{ax3axx4aa}*, where *Q* is square root.

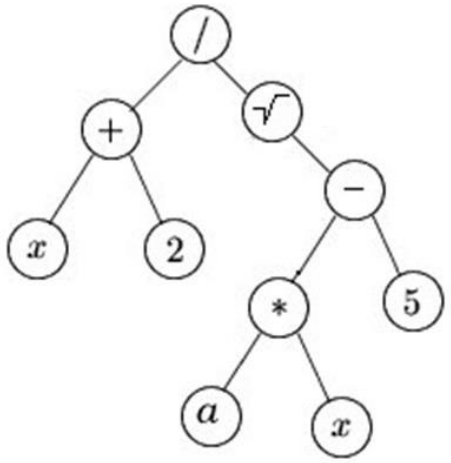

Figure 1: Expression tree for $f(x,a) = (x+2)/\sqrt{(ax-5)}$

In the proposed metaheuristic programming with gene expression to decode the chromosomes (vector of real-valued numbers) an unified approach is suggested for various nature-inspired algorithms. A vector of real-valued numbers (a genotype) is divided into three groups of elements (for head, tail and (optional part) constants) $(\{m_{1j}\},\{m_{2j}\},\{m_{3j}\})$, $0 < m_{ij} < 1$; $|\{m_{1j}\}| = h$; $|\{m_{2j}\}| = t$; $|\{m_{3j}\}| = t$. Each such group is interpreted with different alphabets, which are calculated by the following formulas:

$$h_j = \lfloor m_{1j} * (|F| + |T|) \rfloor;\ t_j = \lfloor (m_{2j} * |T|) \rfloor;\ c_j = \lfloor (m_{3j} * |C|) \rfloor, \quad (1)$$

where $|F|$ is the number of the basic functions, $|T|$ is the number of the variables and constants, $|C|$ is the number of the constants. For example, for an expression $f(x,a) = (x+2)/\sqrt{(ax-5)}$ we have the following two steps of decoding the expression: *[0,32;0,11;0,53;0,7;0,94; ... ;0,8]* → *{/+Qx2-*5 ax3axx4aa}* → $f(x,a) = (x+2)/\sqrt{(ax-5)}$. Note, (1) a linear (vector) representation of a chromosome with the constant size prevents the effect of unreasonable growth of expressions (bloat), and (2) simple operations (1) in decoding a genotype into a phenotype for the interpretation of a chromosome preserve syntactic correctness of the expressions when applying random migration operations of the nature-inspired algorithm.

Thus, decoding a real-valued chromosome results in the function being represented as an interpreted expression. But we used a multi-variant algorithm to decode an expression into multiple subexpressions. In this multi-variant algorithm, we start reading from the first character of the head part of the chromosome to build the first expression, which is completely similar to the classic GEP. We then continue reading the second character of the head to get the second expression. And repeat the reading of the head symbols to the end of the head. In contrast to the standard GEP, this method allows, to simultaneously evaluate a several of expressions and can reduce the time to find an optimal solution. In this case the evaluation of a certain part of the chromosome with the minimum value of the fitness function from the obtained set of chromosomes is selected as the evaluation of the whole chromosome. Note that for a given chromosome *K*, there are many expressions for *E*; the objective function value is calculated for each expression, and the suitability of the chromosome is determined as matching the best expression encoded by that chromosome. For example, for an expression $f(x,a) = (x+2)/\sqrt{(ax-5)}$ we have the following multi-variant decoding the expression: (1) *{/+Qx2-*5 ax3axx4aa}* → $(x+2)/\sqrt{(ax-5)}$, (2) *{+Qx2-*5 ax3axx4aa}* → $x+\sqrt{2}$, (3) *{Qx2-*5 ax3axx4aa}* → $\sqrt{x}$, (4) *{x2-*5 ax3axx4aa}* → *x*, (5) *{2-*5 ax3axx4aa}* → *2*, (6) *{-*5 ax3axx4aa}* → $ax-5$, (7) *{*5 ax3axx4aa}* → *5a*, (8) *{5 ax3axx4aa}* → *5*.

Thus, this algorithm encodes several solutions in a chromosome, unlike the classic GEP algorithm, which encode one solution in a chromosome. Also, unlike the algorithms from [5], this algorithm uses

an unified approach for various nature-inspired algorithms [2, 3, 4] as their evolution engine to find optimal models.

3. Experimental Results

We investigate the efficiency of the metaheuristic programming with gene expression (mpge) using of fourteen different nature-inspired algorithms [2, 3, 4] (Artificial Bee Colony (ABC), Continuous Ant Colony Optimization (ACOR), Bees Algorithm (BA), Biogeography-based Optimization (BBO), Covariance Matrix Adaptation Evolution Strategy (CMA-ES), differential evolution (DE) [6], Firefly Algorithm (FA), genetic algorithm (GA), Harmony Search (HS), Imperialist Competitive Algorithm (ICA), Invasive Weed Optimization (IWO), Particle Swarm Optimization (PSO), Simulated Annealing (SA), Teaching-Learning-based Optimization (TLB)) for the problem of generation and optimization of analytical functions and parameters of models on the basis of the experimental data, variables and basic functions. We used the following ten test functions (see Table 1).

Table 1

Test functions

Function	Function Definition	Variables	Range
f1	$f1(x) = x^4 + x^3 + x^2 + x$	1	[1,10]
f2	$f2(x) = sin(x^2 + x^4)$	1	[1,10]
f3	$f3(x) = sin(exp(sin(exp(sin(x)))))$	1	[1,10]
f4	$f4(x) = sin(x^3) + e^x$	1	[1,10]
f5	$f5(x) = x^5 - 2x^3 + x$	1	[1,10]
f6	$f6(x) = sin(x) + sin(x^2 + x)$	1	[1,10]
f7	$f7(x) = sin(x^2)\, exp(x) - 1$	1	[1,10]
f8	$f8(x, y) = 2sin(x)exp(y)$	2	[1,10]
f9	$f9(x, y) = sin(x) + sin(y^2)$	2	[1,10]
f10	$f10(x, y) = x^3 + y * x^2 + y$	2	[1,10]

The values of each function at 10 random points in the range [1,10] were used in the experiments. The set of basic functions for the test functions is $F = \{+, -, *, sin, exp\}$, the variables are for f1- f7 are $T = \{y\}$, for f8 - f10 are $T = \{y, z\}$.

The following parameters of the compared algorithms are applied: a crossover probability of 0.80, a mutation probability of 0.15, a population size of 200, and a maximum number of generations of 200. The length of the chromosomes is 40; for the GEP we use a population size of 200, and a maximum number of generations of 200 (for classGEP1) and a population size of 1200, and a maximum number of generations of 600 (for classGEP2). For each algorithm and each test function, experiments were performed 50 times and the results were averaged. In the experiments, the algorithm's parameters were not optimized separately and we used the values of the parameters of the algorithms recommended by their authors in the cited books and articles for a correct comparison of efficiency.

Metaheuristic programming using fourteen different natural algorithms was implemented using the resources of the Information Computing Center (ICC) of the Novosibirsk National Research State University (NSU) on Intel Xeon X5670 processors. We also compare MP, based on nature-inspired algorithms implemented in MATLAB, with the classic implementation of the gene expression programming algorithm (GEP) in MATLAB.

We used the probability (frequency) of success, that is, the probability that the algorithm found an expression that coincides precisely with the test function as one of the main indicators for measuring the effectiveness of evolutionary synthesis algorithms. This is the ratio of the number of successful experiments, when the algorithm found the correct expression, to the total number of experiments with given parameters. Let $\bar{p}$ is the average success rate for each algorithm over a set of test functions (see for $\bar{p}$ Figure 2). It can be seen that the probability of success in MP is higher than that of classGEP1 in 11 cases (in the following order: mpgeABC > mpgeTLB > mpgeFA > mpgeDE > mpgeBBO > mpgeHS

> mpgeACO > classGEP2 > mpgePSO > mpgeGA > mpgeCMA > classGEP1) and the probability of success in MP is higher than classGEP2 in 6 cases. Comparison of all fourteen different nature-inspired algorithms by the average success rate in descending order is shown in Figure 2. The probability of success for MP with gene expression is in most cases higher than for the classical GEP algorithm, due to the ability of MP to represent several expressions on one chromosome, resulting in a higher chance of finding a solution.

Let $\bar{t}$ - average execution time of each algorithm for a set of test functions, i.e., the average execution time of an algorithm before the algorithm finds (synthesizes) a given expression or executes the required number of generations. Comparison of all fourteen different natural algorithms by average execution time $\bar{t}$ in ascending order is shown in Figure 3. This result shows that MP has less time to find a solution than classGEP2 in all cases, and less time to find a solution than classGEP1 in 3 cases (in the following order: mpgeHS < mpgeACO < mpgePSO < classGEP1), but the algorithm classGEP1 has a low average success rate. For example, mpgeTLB has an average success rate of 60% higher than classGEP2 and 40% lower average execution time than classGEP2. In the case of classGEP1, mpgeTLB has an average success rate of 134% higher and only 11% higher than the average execution time than classGEP1. MP with gene expression has less time to find a solution in many cases than for the classical GEP algorithm, which is explained by the simple structure of data and operations in MP.

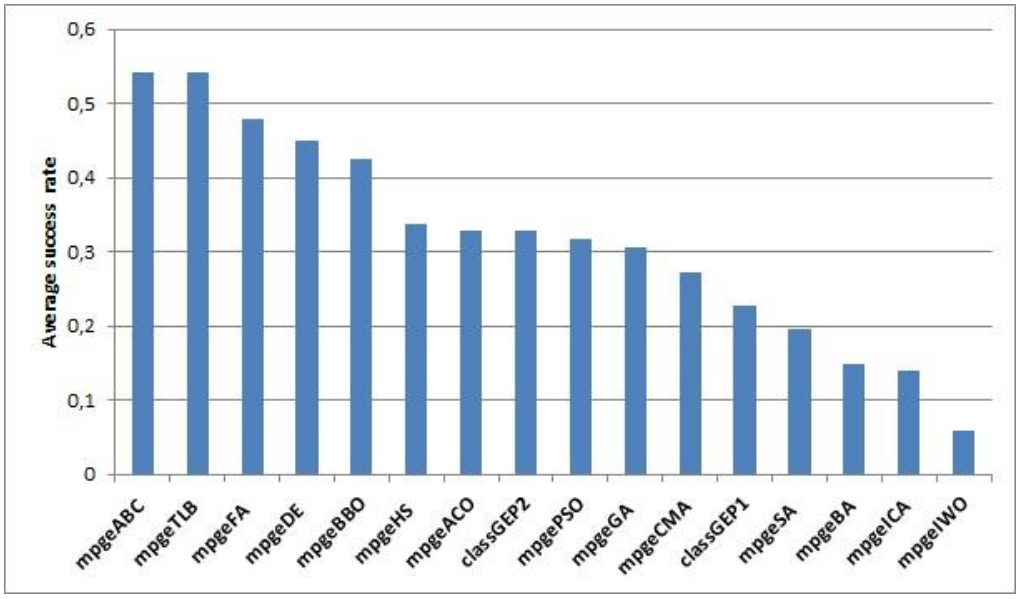

Figure 2: Average rate of success for each algorithm

4. Conclusion

We considered a new approach of metaheuristic programming with gene expression for solving the problem of generation and optimization of nonlinear models (represented by mathematical expressions and programs) based on given basic functions, experimental data and variables. To synthesize such models, metaheuristic programming has been developed as a unified approach for using almost any (fourteen in this article) different algorithms inspired by nature. This approach uses a linear (vector real) representation of a chromosome, simple operations to decode a genotype to a phenotype to interpret a chromosome as a mathematical expression, a multivariate method to represent a set of models (an expression) using a single chromosome. This approach prevents the effect of unreasonable growth of expressions (bloat), and preserve syntactic correctness of the expressions when applying random migration operations of the nature-inspired algorithm. The proposed MP approach with gene expression was implemented using fourteen different algorithms inspired by nature, and these algorithms were compared with each other and with a standard gene expression programming algorithm. We did not identify an absolute leader among the algorithms under consideration by two criteria (by the time of finding a solution and by the probability of finding a given function) at once, but we can choose the

most suitable algorithm based on a compromise solution. Experiments show that the proposed approach is in many cases superior to the GEP algorithm and we can always choose the most suitable MP algorithm, better than the GEP algorithm, based on the preferred criterion either by the time of finding a solution, or by the probability of finding a function (model), or by both criteria simultaneously.

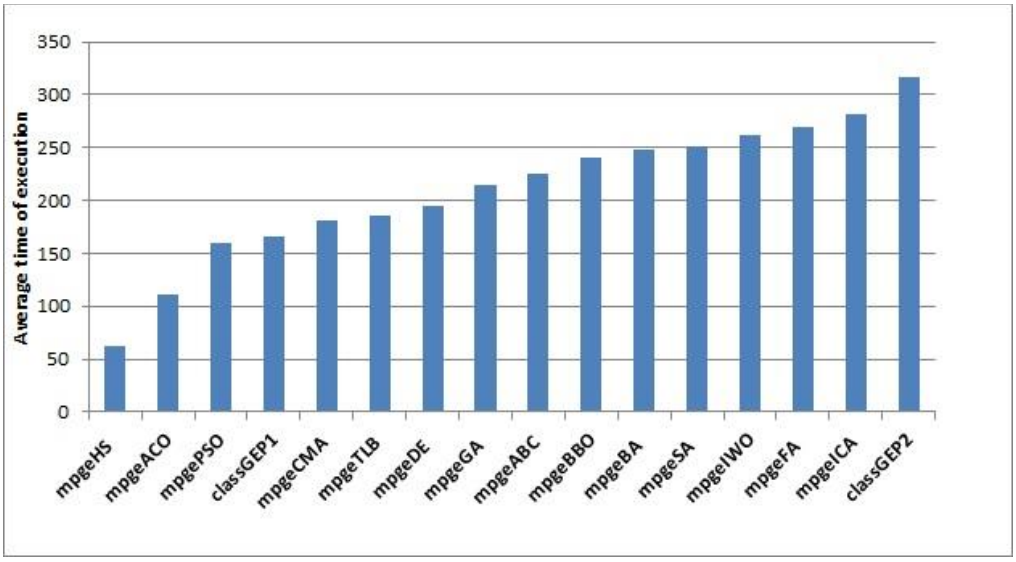

Figure 3: Average time of execution for each algorithm

5. Acknowledgements

This work was carried out under state contract with ICMMG SB RAS N 0251-2021-0005.

6. References

[1] C. Ferreira, Gene Expression Programming: Mathematical Modeling by an Artificial Intelligence, (2nd ed.), Springer-Verlag, Heidelberg, 2006.

[2] Ke-Lin Du, M. N. S. Swamy, Search and Optimization by Metaheuristics. Techniques and Algorithms Inspired by Nature, Birkhauser, Basel, 2016.

[3] E. Cuevas, A Rodríguez, Metaheuristic Computation with MATLAB, CRC Press, 2021.

[4] M. O. Okwu, L. K. Tartibu, Metaheuristic Optimization: Nature-Inspired Algorithms Swarm and Computational Intelligence, Theory and Applications, Springer International Publishing, 2021.

[5] O. G. Monakhov, Differential Evolution for Multi-Variant Evolutionary Synthesisof Nonlinear Models, in: Proceedings of 2018 XIV International Scientific-Technical Conference on Actual Problems of Electronic Instrument Engineering (APEIE2018), IEEE Press, 2018, pp. 487–491. doi:10.1109/APEIE.2018.8545984.

[6] O. G. Monakhov, E. A. Monakhova, M. Pant, Application of differential evolution algorithm for optimization of strategies based on financial time series, Numerical Analysis and Applications, 9 (2016) 150–158. doi:10.1134/S1995423916020063.