ISSN: 1803-3814 (Printed), 2571-3701 (Online)
https://doi.org/10.13164/mendel.2019.1.079

HYBRID SYMBOLIC REGRESSION WITH THE BISON SEEKER ALGORITHM

Jan Merta

Department of Process Control, University of Pardubice, Czech Republic
jan.merta@upce.cz

Abstract

This paper focuses on the use of the Bison Seeker Algorithm (BSA) in a hybrid genetic programming approach for the supervised machine learning method called symbolic regression. While the basic version of symbolic regression optimizes both the model structure and its parameters, the hybrid version can use genetic programming to find the model structure. Consequently, local learning is used to tune model parameters. Such tuning of parameters represents the lifetime adaptation of individuals. This paper aims to compare the basic version of symbolic regression and hybrid version with the lifetime adaptation of individuals via the Bison Seeker Algorithm. Author also investigates the influence of the Bison Seeker Algorithm on the rate of evolution in the search for function, which fits the given input-output data. The results of the current study support the fact that the local algorithm accelerates evolution, even with a few iterations of a Bison Seeker Algorithm with small populations.

Keywords: *genetic programming, symbolic regression, hybrid methods, local learning, bison seeker algorithm.*

Received: 10 May 2019
Accepted: 13 June 2019
Published: 24 June 2019

1 Introduction

Symbolic regression with genetic programming is a supervised machine learning method which searches for a mathematical model of a given dataset. The basic version of the method optimizes both the model structure and its parameters at the same time by evolution. There are also hybrid versions that try to find the best parameter settings for the given model structure using a variety of local optimization methods [1-4]. This extension represents a lifetime adaptation of an individual in the population and can, under the right circumstances, speed up the evolution of the correct mathematical model. However, it also has its disadvantages in the form of additional costs.

This paper wants to explore the Bison Seeker Algorithm's ability to speed up the search of a function that describes the given input-output data by local optimization of numeric leaf values. The Bison Seeker Algorithm belongs to a family of swarm intelligence algorithms and explores the search space by mimicking behaviour patterns of bison herds.

The first part of the paper focuses on the use of genetic programming for symbolic regression purposes. The second part describes different approaches to hybrid evolutionary learning. The third part deals with our use of hybrid method for symbolic regression problem and describes performed experiments. Finally, we added an evaluation of our experiments and a summary of the achieved results and acquired knowledge.

2 Background

2.1 Genetic Programming

Genetic programming was introduced by Stanford computer scientist John R. Koza. This method is an extension of genetic algorithms that can develop programs through evolution. Program structures are represented by the chromosomes in the form of syntactic trees. The genetic alphabet consists of a set of non-terminals (functions) and terminals (constants, variables, random numbers) [5, 6].

Genetic programming has a similar mechanism to genetic algorithms. First, an initial population of randomly generated syntactic trees is created. Then the individuals are evaluated using the fitness function. The fitness function must be evaluated for all inputs and corresponding outputs. Subsequently, individuals go through the selection and the reproduction process according to their fitness values. Genetic operators include a variety of crossover methods, such as a subtree crossover (random subtrees exchange between individuals), and various kinds of mutations, such as subtree mutation (randomly selects the tree vertex and replaces its subtree with a new randomly generated subtree). Thanks to varying chromosome lengths, trees can grow

quickly and generate overcomplicated solutions. This phenomenon is called bloat [5, 6]. This behaviour can be suppressed by determining the maximum size of the tree and by pruning large trees [6].

2.2 Symbolic Regression

Symbolic regression with genetic programming is a supervised machine learning method for finding a mathematical model that matches a given dataset by evolution. The syntactic tree corresponds to the mathematical equation. The set of non-terminals consists of mathematical operations (+, -, *, /, ...) and functions (sin, cos, …), while the set of terminals contains constants (1, 2, 3.05, π, ...) and variables (x, y, distance, …). Fitness is often defined as the difference (sum of squared residuals) between outputs of the model and the desired outputs [6].

More on genetic programming and symbolic regression can be found in [5] and [6]. The next part focuses on hybrid evolutionary techniques.

2.3 Hybrid Methods

The hybrid genetic algorithm combines the global search of an evolutionary approach at the population level with local search at the individual level. Local learning represents a lifetime adaptation of individuals in the population and moves them towards the optimum [7, 8, 9, 10]. Hybrid algorithms often distinguish between genotype and phenotype. Learning affects the phenotype of an individual and his final fitness [7, 9]. Lifetime learning smooths the fitness landscape and simplifies evolution [10]. The two main forms of a hybrid genetic search include Lamarckian evolution and Baldwin effect [7].

In Lamarckian evolution, offspring inherit learned behaviour through reproduction directly from parent genes. Lamarckism needs inverse mapping from the phenotype (and the environment) back to the genotype [10, 11]. In our case, this means that the constants learned during local learning are passed onto the offspring in the next generation by reverse gene transcription. Lamarckian evolution is not biologically accurate and Whitley in paper [7] argues that this approach distorts the population and is not compatible with Holland's Schema Theorem.

The Baldwin effect is purely Darwinian, and the knowledge learned during life, affects the individual's fitness. Baldwinian adaptation affects the behaviour of individuals indirectly. Good learners and individuals who are closer to the optimum have greater fitness, and therefore have more offspring on average. The Baldwin effect gives the chance for good (but not great) genes to resist accidental exclusion during the selection, and these genes remain in the population. In our case, this means that learned constants are not inherited in the next generation, and local learning only increases the chances of preserving models with promising structures [12].

When using a subtree crossover, it happens that the constants working in the parents no longer work well in their offspring and must be re-optimized. On this count, the transcription of constants seems unnecessary. However, if elitism is used to directly copy the best individuals to the next generation (without crossover and mutation), re-learning the correct constants in the Baldwinian approach appears to be a waste of computational time [13].

Local learning can accelerate evolution under certain conditions [9, 10]. It can improve variation in the population [14], which can improve the rate of evolution. It not only has benefits, but also additional costs [10]. Learning is expensive, it costs CPU time and sometimes when the learning is too strong, it can reduce the rate of evolution.

2.4 Literature Review

The hybrid versions of GP symbolic regression try, for a given structure, to optimize its constants with local learning methods. One of the techniques is an adaptive program called STROGANOFF [1, 2], which uses multiple regression analysis for constant tuning. In article [3] the gradient ascent was tested and in paper [4] authors tried simulated annealing. Topchy et al. in the paper [13] used a few iterations of gradient descent. In the paper [15] authors showed possibilities of using the Particle Swarm Optimization method. Raidl and Gunther in the paper [16] introduced a robust variant of the method of least squares by identifying and extending all top-level terms and multiplying them with locally optimized factors. Another paper [17] describes GPA-ES hybrid algorithm, which uses genetic programming for solution structure development and Evolutionary Strategy (ES) for parameters identification.

2.5 Swarm Intelligence

Particle Swarm Optimization is a population based stochastic optimization technique for continuous optimization of nonlinear functions (without the need of calculating the derivative). This technique mimics the

flight of a flock of birds in search of food, or the schooling behaviour of fish. The behaviour of an individual is based on its current velocity, its historical best position (individual component), and the historical best global position of the whole group (social component). Individuals are guided by the best individuals of the group and the overall movement is also influenced by the historically best points found in the search space. [18, 19]

There are other different swarm algorithms inspired by biology: Ant Colony, Bees Algorithm, Artificial Bee Colony Algorithm, Cuckoo, Blind Naked Mole-Rats (BNMR) algorithm, and the newly described Bison Seeker Algorithm [20].

2.6 Bison Algorithm

The Bison Algorithm is a multidimensional continuous optimization method introduced in [21]. The Bison Seeker Algorithm belongs to a family of swarm intelligence algorithms and explores the state space by mimicking behaviour patterns of bison herds. In this method, the bison population is divided into two groups: swarming group, running group.

Swarming individuals move in the direction of the weighted center calculated from the position of the best swarming individuals (elite group). The length of the move step towards the weighted center is affected by the overstep parameter. Runners move around the best swarming bison and test unexplored points in the search space. If a runner finds a better solution than a swarmer, it is promoted (copied) to the swarming group and the worst swarmers are completely removed from the population.

The parameters of the algorithm include:

- size of entire bison population,
- size of swarming group,
- size of elite swarming group (contains top/best swarmers),
- overstep parameter (it affects the swarmer movement towards the weighted center),
- boundaries of the search space in each dimension (min and max coordinates).

2.7 Bison Seeker Algorithm

The Bison Seeker Algorithm (BSA) is an extension of the Bison Algorithm and it uses the same set of parameters. The authors added the seeking phase to the original algorithm. When copying successful runners, in the next iterations, the runners become seekers and explore a promising solution. During this phase they behave like swarmers. Their weighted center is calculated from the positions of each successfully promoted bison. The length of the seeker phase is directly proportional to the number of promoted runners. For this phase, the authors recommend temporary reduction of the overstep parameter (from 3.5 to 2). After seeking, the runners return to their original formation and the algorithm continues with the original swarmer and runner group behaviour until other successful runners are promoted. More information about Bison Seeker Algorithm can be found in [21].

3 Methods

The aim of the experiment was to compare the basic version of GP symbolic regression without adaptation, to the Lamarckian hybrid version with lifetime learning based on the few iterations of the Bison Seeker Algorithm with small populations. For the purpose of the experiment, author created a dataset of inputs and outputs corresponding to the selected functions (1) and (2) with integer coefficients.

$$y = 2x^2 + 3x + 5 \tag{1}$$

$$y = \sin(x) \tag{2}$$

For experiments, author used a classic implementation of the generational genetic programming based on the evolution of syntactic trees. Fitness in the Lamarckian approach is the fitness of a phenotype with modified coefficients, calculated using the least squares method of differences between expected and actual outputs. Author used the MersenneTwister random number generator (the global genetic programming algorithm and local learning algorithm had their own random number generator). Author also decided to use pruning to control tree sizes.

Because author optimized integer coefficients, he used a discrete version of the Bison Seeker Algorithm. This version works the same as the original continuous version, only the real positions at the end of the calculation are rounded to integers. The initial population in the local learning algorithm was generated randomly in the neighbourhood of the original genotype coefficients.

For the first function (1), 15 points from -5 to 5 were generated and set of parameters is shown in Table 1 and Table 2:

Table 1: Parameters of genetic programming algorithm for the first function (1)

Parameter	Value
Population size	100
Elitism	1
Mutation rate	0.01
Initialisation method	grow method
Initialisation tree depth	3
Target fitness	0.0
Generation limit	500
Tree length for pruning	30
Selection method	deterministic tournament selection
Tournament size	2
Terminal set	x, 1.0, 2.0, 3.0, random integer from 1 to 10
Non-terminal set	* , +, -, % (protected division)

Table 2: Parameters of the Bison Seeker Algorithm for the first function (1)

Parameter	Value
Bison herd size	5 and 15
Swarmer group size	4 and 12
Elite group size	2 and 6
Overstep	2.0
Neighbourhood size	2
Min coordinates (all dimensions)	-5
Max coordinates (all dimensions)	5
Number of iterations	3

For the second function (2), 20 points from 1 to 15 were generated and set of parameters is shown in Table 3 and Table 4:

Table 3: Parameters of genetic programming algorithm for the second function (2)

Parameter	Value
Population size	100
Elitism	1
Mutation rate	0.01
Initialisation method	grow method
Initialisation tree depth	3
Target fitness	0.1
Generation limit	1000
Tree length for pruning	65
Selection method	deterministic tournament selection
Tournament size	2
Terminal set	x, 1.0, 2.0, 3.0, random integer from 1 to 10
Non-terminal set	* , +, -, % (protected division)

Table 4: Parameters of the Bison Seeker Algorithm for the second function (2)

Parameter	Value
Bison herd size	5
Swarmer group size	4
Elite group size	2
Overstep	2.0
Neighbourhood size	2
Min coordinates (all dimensions)	-5
Max coordinates (all dimensions)	5
Number of iterations	1 and 3

Datasets for experiments with both functions consisted of equidistantly generated points in given ranges for x value and corresponding function values for the y value.

4 Results

The performance of the individual configurations was measured as the number of generations needed to find the correct mathematical model. In every experiment author ran the symbolic regression 1,000 times. Author also tracked information about successful runs (fitness, successful generation, tree size).

Following tables (Table 5, Table 6, Table 7 and Table 8) show the results of the first set of experiments with first function and 3 iterations of the BSA (basic symbolic regression, BSA with 5 bisons and BSA with 15 bisons):

Table 5: Success rate of the experiments

Experiment	Success rate (%)
Basic version	96.5
Hybrid with BSA (5 bisons)	98.2
Hybrid with BSA (15 bisons)	100

Table 6: Results of the experiments with first function via the basic symbolic regression

	Fitness	Generation	Size
Median	0	40	19
Average	0	81.0	19.8
Min	0	3	9
Max	0	496	29
Std. dev.	0	95.0	4.8

Table 7: Results of the experiments with first function via hybrid symbolic regression with the BSA with population size set to 5 (bsa5)

	Fitness	Generation	Size
Median	0	25	17
Average	0	67.1	18.0
Min	0	1	9
Max	0	496	29
Std. dev.	0	91.5	5.4

Table 8: Results of the experiments with first function via hybrid symbolic regression with the BSA with population size set to 15 (bsa15)

	Fitness	Generation	Size
Median	0	12	19
Average	0	20.4	17.5
Min	0	1	9
Max	0	410	29
Std. dev.	0	31.1	5.2

Figure 1 shows comparison of the convergence of successful solutions of the first function (1) for the basic version of symbolic regression (basic), the BSA with 5 bisons and 3 iterations (bsa5) and for the BSA with 15 bisons and 3 iterations (bsa5 3x):

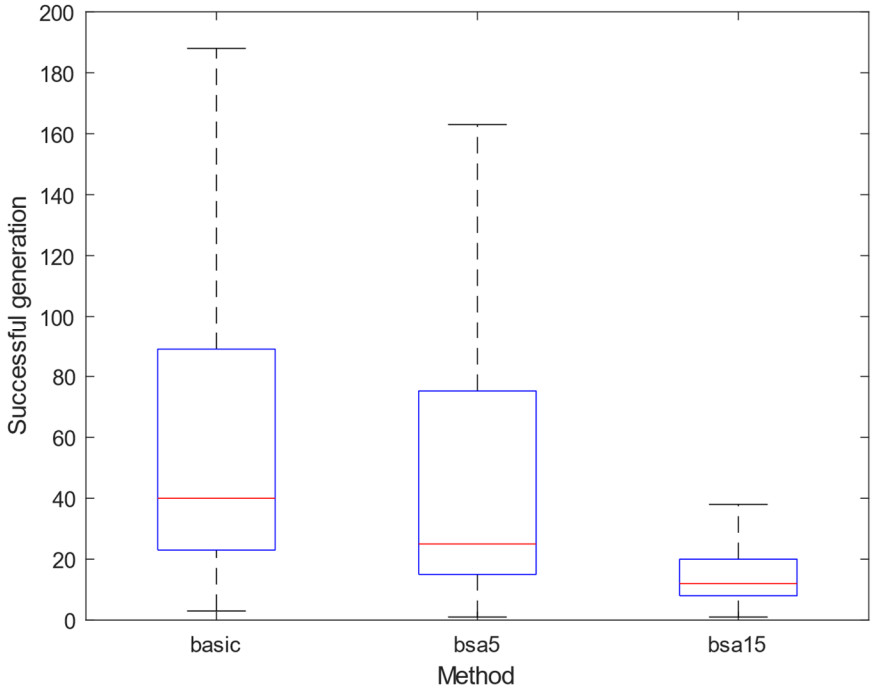

Figure 1: Comparison of the convergence of successful solutions of the first function (1) for individual methods

Table 9 shows the results of the second set of experiments with second function algorithm (basic symbolic regression, hybrid version with 1 and 3 iterations of the BSA with 5 bisons, hybrid version with 1 and 3 iterations of the BSA with 15 bisons):

Table 9: Success rate of the experiments

Experiment	Success rate (%)
Basic version	0
Hybrid with BSA (5 bisons, 1 iteration)	25.2
Hybrid with BSA (5 bisons, 3 iterations)	69.1
Hybrid with BSA (15 bisons, 1 iteration)	87.0
Hybrid with BSA (15 bisons, 3 iterations)	95.7

Basic version was not successful (best fitness was 0.167, median fitness was 0.286). Following tables (Table 10, Table 11, Table 12 and Table 13) show statistics only for hybrid versions:

Table 10: Results of the experiments with second function via the hybrid symbolic regression with the BSA with 5 bisons (1 iteration)

	Fitness	Generation	Size
Median	0.089	328	49
Average	0.081	388.6	46.8
Min	0.018	25	15
Max	0.100	996	65
Std. dev.	0.014	194.4	8.6

Table 11: Results of the experiments with second function via the hybrid symbolic regression with the BSA with 5 bisons (3 iterations)

	Fitness	Generation	Size
Median	0.087	268	49
Average	0.080	343.2	46.2
Min	0.008	8	13
Max	0.100	995	65
Std. dev.	0.017	209.3	10.5

Table 12: Results of the experiments with second function via the hybrid symbolic regression with the BSA with 15 bisons (1 iteration)

	Fitness	Generation	Size
Median	0.087	190	49
Average	0.080	287.3	47.8
Min	0.012	6	13
Max	0.100	992	65
Std. dev.	0.018	243.5	11.0

Table 13: Results of the experiments with second function via the hybrid symbolic regression with the BSA with 15 bisons (3 iterations)

	Fitness	Generation	Size
Median	0.086	110	49
Average	0.079	178.6	47.7
Min	0.012	8	11
Max	0.100	993	65
Std. dev.	0.018	180.1	11.4

Figure 2 shows comparison of the convergence of successful solutions of the second function (2) for the BSA with 5 bisons and 1 iteration (bsa5 1x), the BSA with 5 bisons and 3 iterations (bsa5 3x), the BSA with 15 bisons and 1 iteration and the BSA with 15 bisons (bsa15 1x) and 3 iterations (bsa15 3x).

5 Discussion

The results of the experiments support the fact that lifetime adaptation can (sometimes under certain conditions) accelerate the evolution of the mathematical model which describes the relationship between input and output data. For chosen experiment setting, using local learning via the Bison Seeker Algorithm, the success of finding the mathematical model of the first polynomial increased. This hybrid method reduced the number of generations needed to find the right model. Using more bisons (15), the convergence of the algorithm improved, but even a minimalist bison population (5) had a positive effect on the evolution rate.

Finding an equation for a sine function within a given limit without a hybrid version was unsuccessful. Local learning using the Bison Seeker Algorithm helped find solutions (even with only 1 iteration). For more difficult problems, the basic symbolic regression may not be enough, so there is a possibility that local learning can help find solutions. In the author's opinion, more complex mathematical functions could better show the behaviour of hybrid methods. The results of the current study support the fact that the local algorithm accelerates evolution even with a small number of iterations.

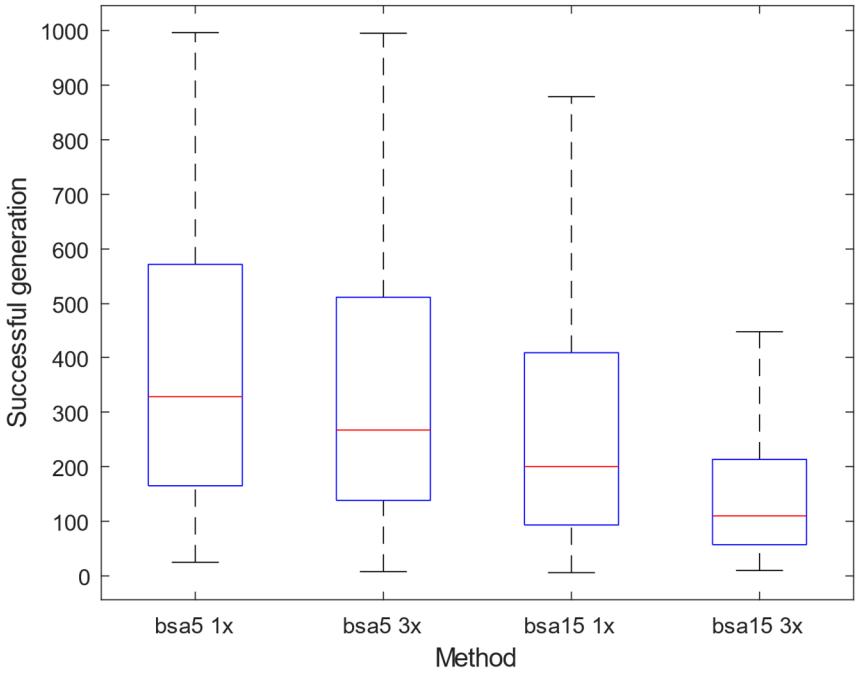

Figure 2: Comparison of the convergence of successful solutions of the second function (2) for individual methods

6 Conclusion

In this article, author has firstly done an overview of existing research in the field of hybrid evolutionary methods. Then author tried the possibilities of using the Bison Seeker algorithm to find a mathematical model via symbolic regression with genetic programming. For a more accurate comparison of individual lifetime adaptation approaches, multiple experiments are needed.

In the future, author is going to try to find models of different kinds of functions. Author would like to optimize real coefficients instead of integers. Author wants to compare this method with other local learning algorithms to optimize coefficients in hybrid symbolic regression. Also, author would like to explore the behaviour of the Bison Seeker Algorithm on other layers of symbolic regression optimization (for example, in ensemble methods for tuning the weights of individual classifiers or for dividing data into effective chunks), and use the findings from the experiments to solve other interesting problems.

Acknowledgement: The work has been supported by the Funds of University of Pardubice (by project "SGS 2019" No: SGS_2019_021), Czech Republic. This support is very gratefully acknowledged.

References

[1] **Iba** H., Sato T., **and** de Garis, H. **1995.** Recombination guidance for numerical genetic programming. In *Proceedings of 1995 IEEE International Conference on Evolutionary Computation*. IEEE, **p**p. 97. DOI: 10.1109/ICEC.1995.489292

[2] Nikolaev, N. Y. and Iba H. 2006. *Adaptive Learning of Polynomial Networks: Genetic Programming, Backpropagation and Bayesian methods*. Springer, New York, USA.

[3] Schoenauer M., Lamy, B., and Jouve, F. 1995. *Identification of Mechanical Behaviour by Genetic Programming Part II: Energy formulation*. Technical report, Ecole Polytechnique, France.

[4] Sharman, K. C., Esparcia-Alcazar, A. I., and Li, Y. 1995. Evolving Signal processing Algorithms by Genetic Programming. In *First International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications, GALESIA*. Volume 414, pp. 473–480, Sheffield, UK.

[5] Koza, J. R. 1992. *Genetic programming: On The Programming of Computers by Means of Natural Selection*. Bradford Book, Cambridge, UK.

[6] Poli, R., Langdon W. B., and McPhee, N. F. 2008. *A Field Guide to Genetic Programming*. Lulu Press, Morrisville, North Carolina, USA.

[7] Whitley, D., Gordon, S., and Mathias, K. 1994. Lamarckian Evolution, the Baldwin Effect and Function

Optimization. In *Parallel Problem Solving from Nature - PPSN III*. Springer, Berlin, pp. 6–15.
[8] Le, N., Brabazon, A., and O'Neill, M. 2018. How the "Baldwin Effect" Can Guide Evolution in Dynamic Environments. *Theory and Practice of Natural Computing* [online]. Lecture Notes in Computer Science. Cham: Springer International Publishing, pp. 164–175. DOI: 10.1007/978-3-030-04070-3_13
[9] Reďko, V. G., Mosalov, O. P., and Prokhorov, D. V. 2005. A Model of Evolution and Learning. *Neural Networks* 18, 5–6, pp. 738–745. DOI: 10.1016/j.neunet.2005.06.005
[10] Turney, P. D. 2002. *Myths and legends of the Baldwin effect*. arXiv: cs/0212036. Retrieved from https://arxiv.org/abs/cs/0212036
[11] Hinton, G. E. and Nowlan, S. J. 1987. How learning can guide evolution. *Complex Systems* 1, pp. 495–502.
[12] French, R. and Messinger, A. 1994. Genes, Phenes and the Baldwin Effect: Learning and Evolution in a Simulated Population. In *Artificial Life IV*. MIT Press, Cambridge, MA, USA.
[13] Topchy, A., and Punch, W. F. 2001. Faster Genetic Programming Based on Local Gradient Search of Numeric Leaf Values. In *Proceedings of the 3rd Annual Conference on Genetic and Evolutionary Computation (GECCO'01)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp. 155–162.
[14] Anderson R. W. 1995. Learning and evolution: A Quantitative Genetics Approach. *Journal of Theoretical Biology* 175, 1, pp. 89–101. DOI: 10.1006/jtbi.1995.0123
[15] Hashimoto, N., Kondo, N., Hatanaka, T., and Uosaki, K. 2008. Nonlinear System Modeling by Hybrid Genetic Programming. *IFAC Proceedings Volumes* 41, 2, pp. 4606–4611. DOI: 10.3182/20080706-5-KR-1001.00775
[16] Raidl, G. 1998. A Hybrid GP Approach for Numerically Robust Symbolic Regression. In *Proc. of the 1998 Genetic Programming Conference*. Madison, Wisconsin, pp. 323–328.
[17] Brandejský, T. 2019. Dependency of GPA-ES Algorithm Efficiency on ES Parameters Optimization Strength. In *AETA 2018: Recent Advances in Electrical Engineering and Related Sciences*. Springer, pp. 294–302. DOI: 10.1007/978-3-030-14907-9_29
[18] Kennedy, J., and Eberhart, R. **1995**. Particle Swarm Optimization. In: *Proceedings of ICNN'95 - International Conference on Neural Networks* .IEEE, **pp**. 1942–1948**.** DOI: 10.1109/ICNN.1995.488968
[19] Rosendo, M., and Pozo, A. 2010. Applying a Discrete Particle Swarm Optimization Algorithm to Combinatorial Problems. In *2010 Eleventh Brazilian Symposium on Neural Networks*. IEEE, pp. 235–240. DOI: 10.1109/SBRN.2010.48
[20] Xing, B., and Gao, W.-J. 2014. *Innovative Computational Intelligence: A Rough Guide to 134 Clever Algorithms*. Springer International Publishing, New York, NY, USA.
[21] Kazíková, A., Pluháček, M. and Šenkeřík, R. 2018. Regarding the Behavior of Bison Runners Within the Bison Algorithm. MENDEL 24, 1, pp. 63–70. DOI: 10.13164/mendel.2018.1.063