INFLUENCE OF (P)RNGS ONTO GPA-ES BEHAVIORS

T. Brandejsky[*]

Abstract: The main aim of this paper is to investigate if the evolutionary algorithms (EAs) can be influenced by Random Number Generators (RNGs) and pseudo Random Number Generators (pRNGs) and if different evolutionary operators applied within EAs requires different features of RNGs and pRNGs. Speaking both about RNGs and pRNGs, the abbreviation (p)RNGs will be used. This question is significant especially if genetic programming is applied to symbolic regression task with the aim to produce human expert comparable results because such task requires massive computations. Experiments were performed on GPA-ES algorithm combining genetic programming algorithm (GPA) for structure development and evolutionary strategy (ES) algorithm for parameter optimization. This algorithm is described bellow and it applies extended scale of different evolutionary operators (additional individuals generating, symmetric crossover, mutations, and one point crossover). These experiments solved problem of symbolic regression of dynamic system. The number of iterations needed for required quality of regression was used as the measure of (p)RNG influence. These experiments point that different (p)RNGs fit to different evolutionary operators, that some combinations (p)RNGs are better than others and that some theoretically excellent (p)RNGs produces poor results. Presented experiments point that the efficiency of evolutionary algorithms might be increased by application of more (p)RNGs in one algorithm optimised for each particular evolutionary operator.

Key words: *(pseudo) Random Number Generator ((p)RNG); Genetic Programming Algorithm (GPA); Evolutionary Strategy (ES); GPA-ES algorithm; sensitivity to (p)RNG properties*

Received: July 25, 2017
Revised and accepted: January 2, 2018

DOI: 10.14311/NNW.2017.27.033

1. Introduction

Evolutionary algorithms as genetic algorithms, self-organizing migrating algorithm (SOMA), differential evolution algorithms, evolutionary strategies and genetic programming algorithms are stochastic optimization algorithms. Their properties are strongly influenced by used (pseudo) Random Number Generator (p)RNG. On the

[*]Tomas Brandejsky; University of Pardubice, Department of Software Technologies, Faculty of Electrical Engineering and Informatics, CS Legies Square 565, Pardubice, Czech Republic, Tomas.Brandejsky@upce.cz

opposite side, there exist signals that on the place of random number generator it is possible to apply not only (pseudo) Random Number Generators [9], but also deterministic chaos systems and even deterministic functions as $\sin(x)$ without significant loss of evolution algorithm abilities. Such observations [1, 2, 7, 8, 15–18] opened question which property of function applied on the place of random number generator are significant, if these property significance does not vary during evolutionary process and in different evolutionary operators implemented by used evolutionary algorithm. It also means that there might be used different generator for initial population formation, parameter identification (in the case of GPA-ES algorithm [12]), particular evolutionary operators control etc. The (p)RNGs influence evolutionary algorithms as well as intensively studied evolutionary operators, see e.g. [3].

The problem of investigation of (p)RNG influence onto evolutionary algorithm behaviors [13] consist especially in the needed number of experiments. There are many stages of the algorithms function (e.g. initial population generating, evolutionary operator application) and many evolutionary operators like mutation or crossover which might be implemented many ways. The number of experiments is increased by the stochastic nature of these algorithms caused especially by their dependency on used (p)RNGs features.

Tests are implemented with GPA-ES [4] which is one version of standard GP algorithm and it is described in the next chapter. Its main difference lies in use of ES (evolutionary strategy) algorithm to optimize each population member constant magnitudes in each evolutionary cycle as it is described in the next chapter. The ES optimization cycle allows to decrease noise caused by constant identification (there is significantly decreased of ill-defined constant magnitudes) and thus increases preciseness of evolutionary operations (especially cross-over and mutation ones) influence measurement. This capability is significant to this study.

The main aim of presented research is to verify hypothesis that (p)RNGs significantly influence evolutionary algorithms efficiency and if requirements on these properties varies in different stages of evolutionary algorithm. During these tests, the sufficient number of experiments must be determined to recognize differences between different (p)RNGs reliably.

2. GPA-ES algorithm

Structure of hybrid GPA-ES system [3,4] is outlined at Fig. 1. Fig. 2 then illustrates relationship between individuals in GPA population and vector of ES populations (one population for each GPA individual) in the GPA-ES system. While Fig. 1 outlines control flow of hybrid algorithm, Fig. 2 is denoted to fundamental data structures of this system.

Left part of Fig. 1 describes master GPA algorithm, which is implemented in the way described by Koza [10, 11]. When execution of this algorithm enters into state of individual evaluation, slave Evolutionary Strategy described by right part of the Figure is called for each GPA individual. After the ES algorithm execution, fitness of the best individual is reasoned as fitness of the GPA individual.

The first line of boxes in Fig. 2 represents individuals of GPA algorithm. Each box in second line corresponds to related GPA individual. Each of these boxes

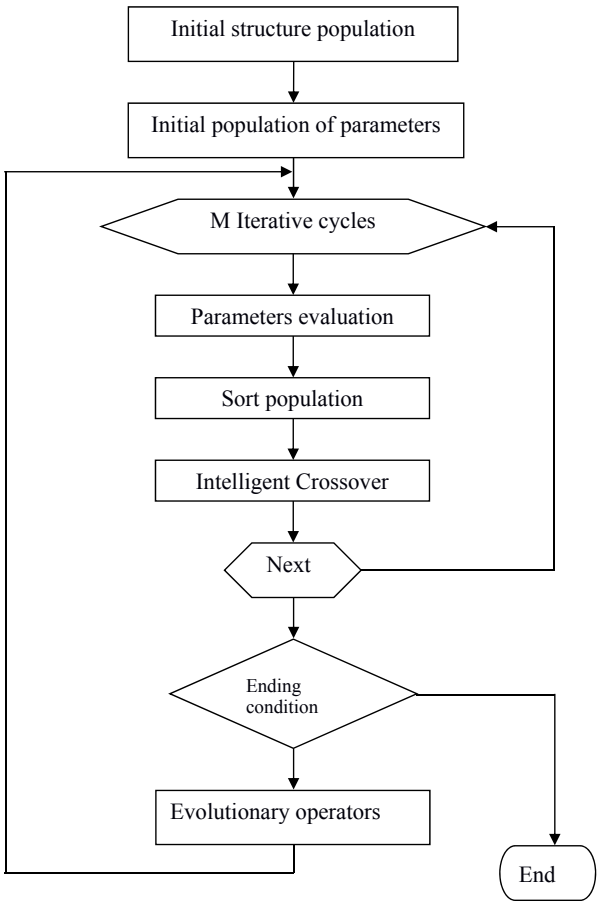

Fig. 1 *Hybrid GPA-ES system structure.*

represents separate population of ES individuals used for parameters optimization. Individuals of this ES population are represented as rows of magnitudes in the relevant boxes.

Computational complexity of GPA-ES is expressed as (1), as it was analyzed in [3]

$$O\left(GPAES\right) \simeq pqnm \log m + pqn \log n, \tag{1}$$

where

n is number of GPA individuals,

m is number of ES individuals,

l is complexity of structures created by GPA,

k is average number of constants in GPA genes, where $k = 2^{l-2}$,

p is number of GPA populations,

q is number of ES populations.

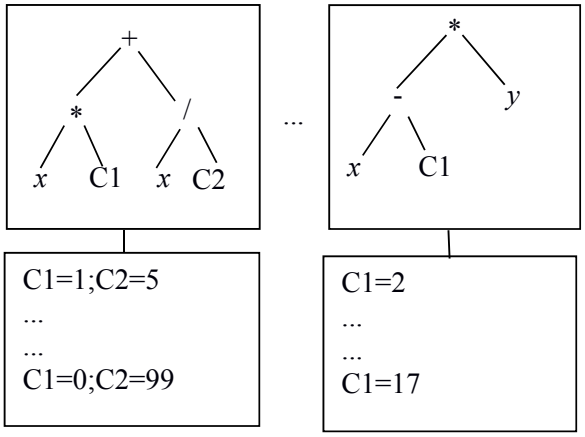

Fig. 2 *Relationship between GPA population individuals and vectors of ES populations.*

Experiment organization corresponds to this structure, there were 5 different (p)RNGs for GPA part and 3 for ES part of the algorithm are reasoned. These generators are listed in the following Tab. I.

– initial population of GPA generating
– regenerating of GPA population (except the best individual)
– mutation operators of GPA
– crossover operators of GPA (it is useful to use more different operators like symmetric crossover and one-point-crossover to prevent stagnation of evolution caused trapping in evolutionary cycle like it is demonstrated for simple case in [4])
– control of GPA operation
– initial ES population generating
– control of ES intelligent crossover operator
– addition of new individuals into ES population

Tab. I *Different (p)RNG use in the GPA-ES algorithm.*

There is hidden significant problem of comparability of particular (p)RNG influence to whole GPA-ES evolutionary system behaviour, see e.g. [14]. There is no possibility to imply e.g. if generator A is the best on position of mutation operation generator when other generators are of type B that this generator will be the best when type of the remaining generator types change. There are mutual influences of evolutionary operators given especially by common genome, that does not allow analysing of each operator individually. This is problem for choice of optimal (p)RNG combination but it is not the limit of presented project because it is still possible to analyse requirements of particular evolutionary operators to (p)RNGs.

The efficiency of GP algorithm is influenced by many powers. On the first places it is possible to mention population sizes, function sets, probabilities of

particular operations (like mutation or crossover), real independence of GPA and ES threads and particular (p)RNGs etc. In the presented study, the big effort was focused to elimination of these influences. Population sizes were constant as well as probabilities of particular operations. Thread independence is solved through openMP library. The biggest problem is thread independence of (p)RNGs. In presented experiments, <random>library of 11^{th} version of C++ standard is applied. This library declares thread independence, but many implementations even in the same GNU compiler version do not ensure this feature.

Because the significant role to reliability of presented data plays number of particular generator calls during single test, Tab. II enumerates their number when we are reasoning 100 individuals in GPA population, 100 individuals in each of 100 corresponding ES populations, constant number of 40 ES evolutionary cycles and 100 GPA evolutionary cycles in average:

There is significant influence of seed magnitudes observed in the experiments. Unfortunately, it was observed that large number of individuals in studied populations (e.g. populations of 1000 individuals) is not able to eliminate this factor. This influence shall be eliminated only by a large number of experiments with different seed magnitudes. The performed experiments concludes that it is need to use at least 2000 different seed magnitudes to decrease error of measurement to magnitude smaller than 0.1%.

3. Observed dynamics

The results of experiments are significantly influenced by regressed functions, especially their structure. To increase reliability of experiments, the large number of test functions is frequently applied by many researchers and benchmark sets were formed. Because it is need to use large number of seed magnitudes for each function and large populations in each experiment, there it is extreme computational power need and experiments are calculated on supercomputer. Fortunately, if structure of the algorithm is known, it is possible to estimate probability of occurrence of given structure in the population and transpose results from measured experiments to another test case, as it was presented in [6, 13, 18], where the first monograph presents many ways based especially on the schema theory, the second paper brings approach close to Markov processes.

As it was written above, the test set of herein presented experiments was restricted to three functions describing Lorenz attractor system (2 and 3):

$$\begin{aligned} x(t) &= \sigma(y(t) - x(t)), \\ y(t) &= (\rho - z(t)) - y(t), \\ z(t) &= x(t)\,y(t) - \beta z(t), \end{aligned} \tag{2}$$

$$\sigma = 16 \qquad \beta = 4 \qquad \rho = 45.91 \tag{3}$$

The regressed data are in the form of table of 600 points describing Lorenz attractor system movement sampled with period of 0.01 sec. The estimation process stops when sum of error squares across all 600 points is less or equal to 1×10^{-7}.

Lorenz attractor system is not a typical benchmark. Many researchers use different ones like [19]. These benchmarks are many times focused to different

– Initial GPA population generating – approximately 550 (p)RNG calls for average depth of initial operator tree between 2 and 3. This operation is applied only once.
– Regenerating of GPA population (except the best individual) - approximately influences 30% individuals in the each next population means 7500 (p)RNG calls in average. Regenerating operation is a generating of totally new individuals without any relation to existing ones in contrast to mutation operation which modifies previously existing ones.
– Mutation operators of GPA are called in 50% of evolutionary operator calls ad in represents 200000 random numbers.
– Crossover operators influence about 20% of GPA individuals when there are two versions called classical crossover used in even cycles and symmetric crossover in odd cycles. Symmetrical one-point crossover differ from other variants (brief reviews bring e.g. [4, 10, 11, 14]) in the same movement in both recombined structures. The purpose of this modification of crossover operator is to increase semantic similarity of produced structures (and efficiency of the algorithm).
– classical crossover – about 1740 (p)RNG calls in our model case.
– symmetric crossover – about 1740 (p)RNG calls in presented average case.
– Control of GPA operation – 200 (p)RNG calls.
– Initial ES population generating – because to each individual of GPA part of the algorithm it is need to create and evolve separate ES population, the average number of (p)RNG calls for above described parameters is equal to 5000000.
– Control of ES intelligent crossover operator (p)RNG call number depends on numbers of regenerated ES individuals, ES and GPA cycles and GPA populations and it is equal to 396000. This crossover operator uses linear interpolation between both individuals with overlaps below and over interval defined by original individual coordinate positions.
– Addition of new individuals into ES population then also represents 396000 (p)RNG calls.

Tab. II *Estimations of (p)RNG call numbers in separate parts of GPA-ES algorithm.*

problem than symbolic regression ones. There is also another problem that these benchmarks are not representing non-similar functions. As it was analysed in [6], it is necessary to analyse probability of particular structure in the population and it depends on the used algorithm. Thus results obtained from Lorenz attractor system symbolic regression might be transformed and then compared e.g. with results of [19].

The 1st equation describing x variable is linear and simple. It is easy to discover it and the needed number of iteration is the smallest. The Tab. III and Fig. 3. Used random number generators are following: rand denotes standard c random number generator – the simplest linear congruential one, minstd0 is multiplicative congruential pseudo-random number generator as well as minstd (they differs in

X	rand	minstd0	true random	randlux24	minstd	Lorenz _x	Lorenz _y	Lorenz _z
GPA initial population generating	21.84	21.47	21.94	22.04	22.19	22.12	21.88	22.90
GPA population regenerating	22.87	22.29	22.40	21.73	22.77	23.38	22.01	22.77
GPA mutation control	22.13	22.56	22.44	22.77	22.29	22.05	22.57	20.96
GPA crossover control	22.13	22.56	22.44	22.77	22.29	22.05	22.57	20.96
GPA operation control	22.24	22.35	21.92	22.62	22.08	19.78	21.92	13.75
ES initial population generation	22.05	20.97	21.76	20.54	20.67	21.78	21.28	21.52
ES crossover control	22.35	21.07	21.33	21.11	21.00	21.28	21.25	27.81
ES new individual addition control	22.69	21.77	21.24	22.05	21.98	8.40	7.44	17.27

Tab. III *Influence of Number Generators to GPA-ES efficiency for symbolic regression x-axis function of Lorenz attractor data.*

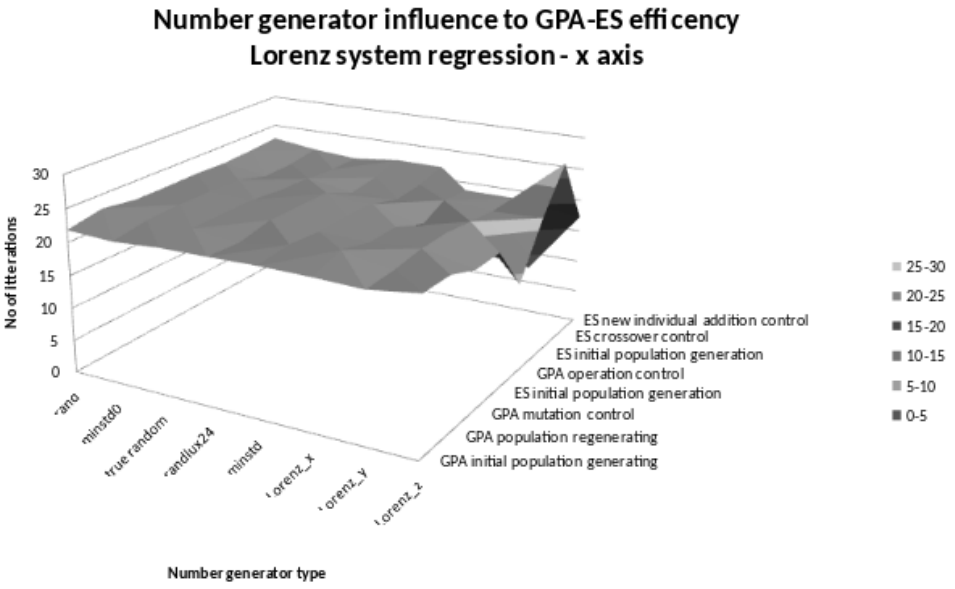

Fig. 3 *Number of iterations of symbolic regression process of x-variable of Lorenz attractor data in dependence on chosen number generator type.*

Y	rand	minstd0	true random	randlux24	minstd	Lorenz _x	Lorenz _y	Lorenz _z
GPA initial population generating	203.24	204.76	203.23	197.50	207.01	206.08	200.33	202.86
GPA population regenerating	198.14	209.89	204.21	203.08	201.93	204.14	198.89	203.35
GPA mutation control	204.41	205.25	202.19	204.18	204.11	197.84	198.88	295.83
GPA crossover control	204.41	205.25	202.19	204.18	204.11	197.84	198.88	295.83
GPA operation control	204.48	198.82	205.40	204.78	206.38	195.90	219.46	174.02
ES initial population generation	203.46	199.55	205.11	205.78	206.28	205.14	201.37	206.09
ES crossover control	200.23	196.50	205.20	202.80	202.19	277.62	268.48	377.75
ES new individual addition control	203.80	205.55	199.43	206.26	197.41	139.01	129.95	197.18

Tab. IV *Influence of Number Generators to GPA-ES efficiency for y-axis function symbolic regression of Lorenz attractor data.*

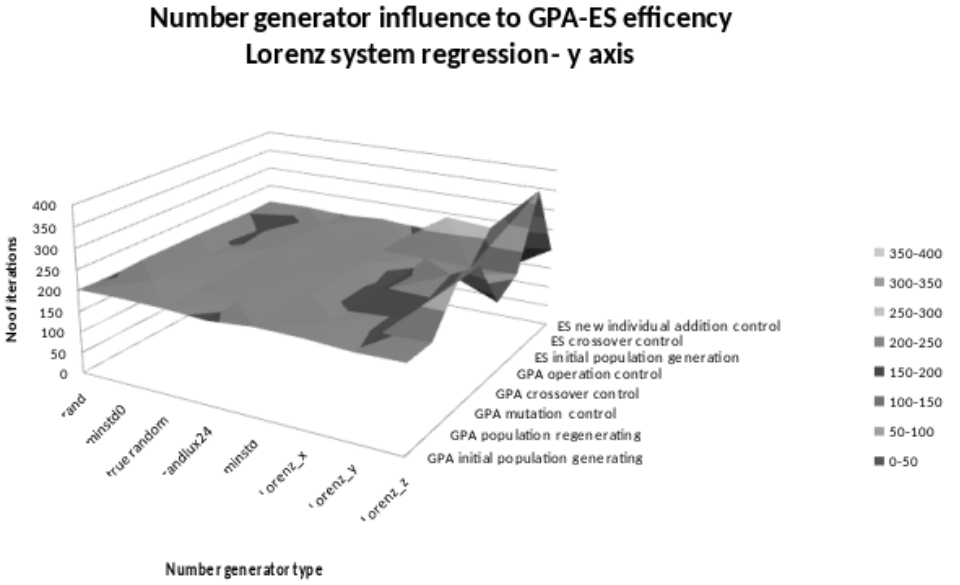

Fig. 4 *Number of iterations of symbolic regression process of y-variable of Lorenz attractor data in dependence on chosen number generator type.*

Z	rand	minstd0	true random	randlux24	minstd	Lorenz _x	Lorenz _y	Lorenz _z
GPA initial population generating	30.46	31.33	30.51	31.26	31.41	30.44	32.67	32.27
GPA population regenerating	30.74	31.59	30.94	30.52	31.27	30.97	31.78	30.71
GPA mutation control	30.98	31.15	31.31	30.37	30.93	31.09	32.34	29.90
GPA crossover control	30.98	31.15	31.31	30.37	30.93	31.09	32.34	29.90
GPA operation control	31.35	31.28	30.60	31.42	30.67	31.80	31.26	28.46
ES initial population generation	30.78	31.47	30.75	30.70	31.61	30.33	30.30	30.94
ES crossover control	31.60	31.31	30.39	30.21	31.72	31.04	31.50	34.83
ES new individual addition control	31.59	31.61	31.36	31.02	30.71	21.54	20.37	27.57

Tab. V *Influence of Number Generators to GPA-ES efficiency for z-axis of Lorenz attractor data.*

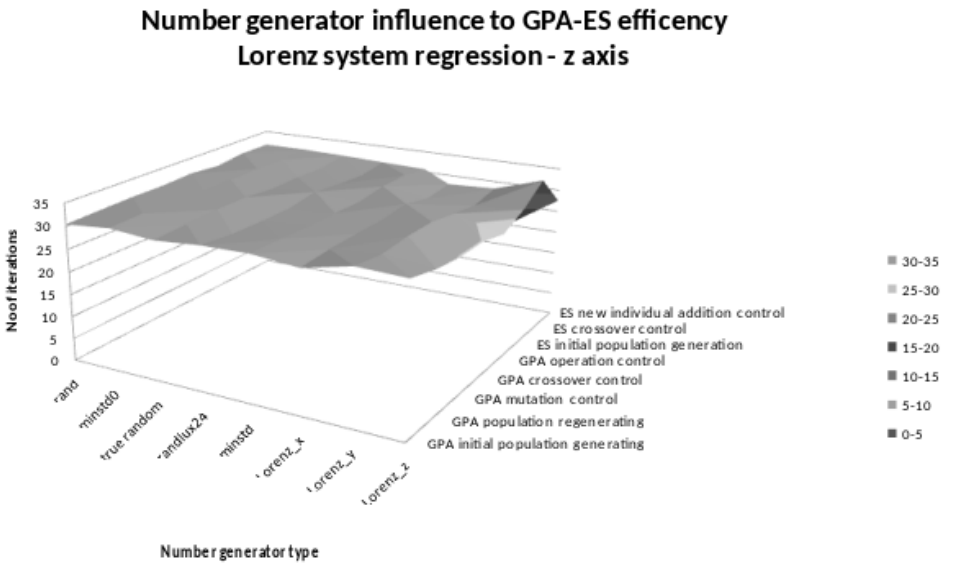

Fig. 5 *Number of iterations of symbolic regression process of x-variable of Lorenz attractor data in dependence on chosen number generator type.*

parameters, true random generator is based on stochastic processes to generate a sequence of uniformly distributed random numbers, randlux24 is a subtract-with-carry pseudo-random generator of 24-bit numbers and Lorenz _X, _Y and _Z generators uses differential equation describing Lorenz deterministic chaos system coordinate X, Y and Z respectivelly.

Above presented graphs points some interesting dependencies. The generators used to formation of initial population are able to influence whole symbolic regression process. There is strong evidence that requirements of GPA crossover and mutation operators to Number Generators are similar. ES part of the algorithm is extremely sensitive to application of non-random Lorenz attractor based Number Generators to influence symbolic regression process.

For the practical applicability of obtained results in multi-number generator evolutionary systems there will be crucial presence on some form of superposition of results obtained by investigation of single distinguish generator influence. Because it is possible to estimate, that composition of specific generators will be complicated and non-linear, this problem will be subject of future research too. Tab. VI brings example of superposition of generators and presented results gives chance that presented analysis of independent NGs is applicable to real multi-number generator evolutionary systems design.

The first two lines of this table presents situations when all NGs are rand() functions and start with seed 0 except one, which seed magnitudes vary from 0 up to 1999. These initial seed magnitude vectors are applied to GPA mutation and GPA operation control NGs. The next two lines replaces GPA mutation and GPA operation control NGs with minstd or Lorenz z-axis based NG respectively. The results produced by these generator vectors are better than in the first two cases. The last two lines of Tab. VI present combination of both previous NG vectors and when one seed magnitude vector is applied. The obtained numbers of iteration cycles are smallest, thus they confirm the presence of constrained form of superposition.

The above presented experiments points that there are only small differences between many (p)RNGs. On the opposite side, e.g. in the case of number generator used to generating of new additional individuals into ES population the best results are in some cases produced by Lorenz attractor systems based number generators.

4. Conclusion

Presented study points that there are no significantly distinguishable differences between standard (p)RNGs in the case of symbolic regression on linear or nearly linear systems symbolic regression like x a y coordinate of Lorenz attractor. Z axis based number generator produces more chaotic results.

The more significant information about importance of particular (p)RNG properties for evolutionary algorithms efficiency might be observed on non-standard generators like Lorenz attractor system based ones or non-linear systems symbolic regression. The problems of such system study are caused by computational power consumption because both non-standard number generators and symbolic regression of strongly non-linear systems symbolic regression require exhaustive computations.

Test case	x	y	z
rand() generator for all positions and 2000 different seed magnitudes for mutation one	22.78	205.60	32.10
rand() generator for all positions and 2000 different seed magnitudes for GPA operation controll	22.24	204.48	31.35
rand() generator for all positions except mutation one, where was minstd() tested for 2000 different seed magnitudes	21.22	203.50	31.23
rand() generator for all positions except GPA operation control, where was NG based on z axis of Lorenz attractor tested for 2000 different seed magnitudes	13.75	173.61	28.36
rand() generator for all positions except mutation and GPA operation control ones, where were NG based on z axis of Lorenz attractor for GPA operation control and minstd() (p)RNG for mutation tested for 2000 different seed magnitudes	15.40	159.53	28.22
rand() generator for all positions except mutation and GPA operation control ones, where were minstd() (p)RNG for mutation and NG based on z axis of Lorenz attractor for GPA operation control tested for 2000 different seed magnitudes	14.52	161.32	28.15

Tab. VI *comparison of numbers of needed iteration cycles for different sets of NGs.*

It is easier to analyse Evolutionary Strategy properties. For less complex ES system it is possible to form suggestion about number generators influencing control of ES intelligent crossover operator and addition of new individuals into ES population. In the case of intelligent crossover operator true (hardware) random device, subtract-with-carry pseudo-random generator of 24-bit numbers with accelerated advancement and minimal standard generator gives better results than frequently used rand() function. For regenerating of poor individuals in the population the minimal standard generator and true random device are better (under conditions of presented study) than standard rand() function of C++ stdlib function.

On the base of above presented results it is possible to conclude that the selection of (p)RNG (and other number generators) influence efficiency of evolutionary system. The hypothesis that different evolutionary operators need distinct number generator sets was confirmed too. From practical use viewpoint there is still problem that influence of particular number generator (e.g. of generator controlling mutation) is affected by other generators and thus whole generator set for each particular operation must be reasoned. It is not possible to simply find the best (p)RNG e.g. for mutation operator without respecting (p)RNGs influencing other evolutionary operators of the used evolutionary algorithm.

Acknowledgement

This work was supported by The Ministry of Education, Youth and Sports from the Large Infrastructures for Research, Experimental Development and Innovations project "IT4Innovations National Supercomputing Center – LM2015070", project "Random number generators influence onto Evolutionary Algorithms behaviors" in 6^{th} Open Access call.

References

[1] Bastos-Filho C.J.A., Oliveira Junior M.A.C., Nascimento D.N.O., Ramos A. D. Impact of the Random Number Generator Quality on Particle Swarm Optimization Algorithm Running on Graphic Processor Units. Conference: 10th International Conference on Hybrid Intelligent Systems (HIS 2010), Atlanta, GA, USA, August, 2010, 23-25, Curran Associates, Inc. ISBN 9781424473632, doi: 10.1109/HIS.2010.5601073.

[2] Brandejsky T. Limited randomness evolutionary strategy algorithm. In: Matousek R. (ed.): Mendel 2015. Springer, 2015, pp. 53–62, ISBN 978-3-319-19824-8.

[3] Brandejsky T. Multi-layered evolutionary system suitable to symbolic model regression. In: Nikos Mastorakis, Metin Demiralp, and N. A. Baykara (Eds.) Proceedings of the 2nd international conference on Applied informatics and computing theory (AICT'11). World Scientific and Engineering Academy and Society (WSEAS), Stevens Point, Wisconsin, USA, 2011, pp. 222–225. ISBN 978-1-61804-034-3

[4] Brandejsky T. Evolutionary system to model structure and parameters regression, Neural Network World. 2012, 22(2), pp. 181–194. ISSN 1210-0552, doi: 10.14311/NNW.2012.22.011.

[5] Brandejsky T. Analysis of Genetic Algorithm Behavior. In: Matousek R., ed. *Mendel 2012.* Mendel 2012 – 18th International Conference on Soft Computing. Brno, 27.06.2012 – 29.06.2012. Brno: VUT v Brně, Faculty of mechanical engineering, 2012, pp. 76–80. ISSN 1803-3814. ISBN 978-80-214-4540-6.

[6] Brandejsky T., Zelinka I. Specific behaviour of GPA-ES evolutionary system observed in deterministic chaos regression. In: Nostradamus : Modern Methods of Prediction, Modelling and Analysis of Nonlinear Systems. Heidelberg, Springer, 2013, pp. 73–81. ISSN 2194-5357. ISBN 978-3-642-33226-5, https://link.springer.com/chapter/10.1007/978-3-642-33227-2_10.

[7] Cantú-Paz E. On Random Numbers and the Performance of Genetic Algorithms. GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference, New York, USA, 2002, pp. 311–318, Morgan Kaufmann.

[8] Cárdenas-Montes M., Vega-Rodríguez M.A., Gómez-Iglesias A. Sensitiveness of Evolutionary Algorithms to the Random Number Generator. In: Dobnikar, A. Šter, U., Branko L. Adaptive and Natural Computing Algorithms: 10th International Conference, ICANNGA 2011, Ljubljana, Slovenia, April 14–16, Proceedings, Part I, 2011, pp. 371–380, Springer, Berlin, Heidelberg, ISBN 978-3-642-20282-7, doi: 10.1007/978-3-642-20282-7_38.

[9] James F. A review of pseudorandom number generators. Computer Physics Communications. 1990, 60, pp. 329–344, North-Holland.

[10] Koza J.R. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, MIT Press, 1992. ISBN 0-262-11170-5

[11] Koza J.R., Bennett F.H., Andre D., Keane M.A. *Genetic Programming III: Darwinian Invention and Problem Solving*, Morgan Kaufmann, 1999. ISBN 1-55860-543-6

[12] Krőmer P., Snášel V., Platoš J., Husek D. Genetic algorithms for the linear ordering problem, Neural Network World. 2009, 19, pp. 65–80, CTU in Prague.

[13] Langdon W.B., Poli R. Foundations of Genetic Programming. Springer, New York, eidelberg, Berlin, 1998. ISBN 978-3-662-04726-2

[14] Poli R., Langdon W.B., McPhee N.F. A field guide to genetic programming, Lulu Enterprises, UK Ltd (March 26, 2008), ISBN-13: 978-1409200734

[15] Senkerik R., Davendra D.D., Zelinka I., Pluhacek M., Kominkova-Oplatkova Z. Chaos Driven Differential Evolution with Lozi Map in the Task of Chemical Reactor Optimization. Lecture Notes in Computer Science. Volume 7895, Springer, 2013, pp. 56–66.

[16] Senkerik R., Davendra D.D., Zelinka I., Pluhacek M., Kominkova-Oplatkova Z.: On The Differential Evolution Driven By Selected Discrete Chaotic Systems: Extended Study. Mendel 2013: 19th International Conference on Soft Computing : June 26-28, 2013, Brno, Czech Republic, Brno University of Technology, 2013, pp. 137–144.

[17] Senkerik R., Pluhacek M., Davendra D.D., Zelinka I., Kominkova-Oplatkova Z.: Chaos Driven Evolutionary Algorithm: a New Approach for Evolutionary Optimization. Recent advances in systems, control and informatics: proceedings of the 2013 International conference on systems, control and informatics (SCI 2013) : September 28–30, 2013, Venice, Italy, WSEAS Press, 2013, pp. 117–122.

[18] Senkerik R., Pluhacek M., Kominkova-Oplatkova Z.: Simulation of time-continuous chaotic systems for the generating of random numbers. Latest Trends on Systems – Volume II. Proceedings of the 18th International Conference on Systems (part of CSCC '14). Santorini Island, Greece, July 17–21, 2014, pp. 557-561, ISSN: 1790-5117, ISBN: 978-1-61804-244-6.

[19] GPA benchmarks, http://www.gpbenchmarks.org/wiki/index.php?title$=$Problem_Classification, accessed 21st July 2017.