Dependency of GPA-ES Algorithm Efficiency on ES Parameters Optimization Strength

*Tomas BRANDEJSKY**

University of Pardubice, 532 10 Pardubice, Czech Republic

*Corresponding Author: Tomas BRANDEJSKY (Email: tomas.brandejsky@upce.cz)
(Received: 24-Dec-2018; accepted: 13-Mar-2019; published: 31-Mar-2019)
DOI: http://dx.doi.org/10.25073/jaec.201931.226

Abstract. *In herein presented work, the relation between number of ES iterations and convergence of the whole GPA-ES hybrid algorithm will be studied due to increasing needs to analyze and model large data sets. Evolutionary algorithms are applicable in the areas which are not covered by other artificial intelligence or soft computing techniques like neural networks and deep learning like search of algebraic model of data. The difference between time and algorithmic complexity will be also mentioned as well as the problems of multitasking implementation of GPA, where external influences complicate increasing of GPA efficiency via Pseudo Random Number Generator (PRNG) choice optimization.*

Hybrid evolutionary algorithms like GPA-ES uses GPA for solution structure development and Evolutionary Strategy (ES) for parameters identification are controlled by many parameters. The most significant are sizes of GPA population and sizes of ES populations related to each particular individual in GPA population. There is also limit of ES algorithm evolutionary cycles. This limit plays two contradictory roles. On one side bigger number of ES iterations means less chance to omit good solution for wrongly identified parameters, on the opposite side large number of ES iterations significantly increases computational time and thus limits application domain of GPA-ES algorithm.

Keywords

Genetic Programming Algorithm, Evolutionary Strategy, Hybrid Evolutionary System, Algorithm Efficiency, Optimization.

1. Introduction

Increasing amount of data to be processed forces needs for improving efficiency of existing algorithms. While artificial neural networks and deep learning technology can be reasoned rather as data representation or interpolation, recall and approximation tool, evolutionary algorithms are capable to transform data into models which can be understood and analyzed by humans, searched for optima, and solved many next tasks on the base of training data. In the area of model development by evolutionary algorithms called symbolic regression, Genetic Programming Algorithms (GPA), Analytic Programming [1] and related techniques are used.

Hybrid evolutionary algorithms like GPA-ES [2] uses GPA for solution structure development and Evolutionary Strategy (ES) for parameters identification are controlled by many parameters. The most significant are sizes of GPA population and sizes of ES populations related to each particular individual in GPA population. There is also limit of ES algorithm evolutionary cycles. This limit plays two contradictory

roles. On one side bigger number of ES iterations means less chance to omit good solution for wrongly identified parameters [3] and it was the main idea of GPA-ES hybrid algorithm development. On the opposite side large number of ES iterations significantly increases computational time and thus limits application domain of GPA-ES algorithm.

In this study, the relation between number of ES iterations and convergence of the whole GPA-ES hybrid algorithm will be studied. The difference between time and algorithmic complexity will be also mentioned as well as the problems of multitask and multithread implementation of GPA as algorithms where external influences complicates increasing of GPA efficiency via Pseudo Random Number Generator (PRNG) choice optimization.

2. Hybrid GPA-ES algorithm

GPA-ES hybrid algorithm combines GPA algorithm for solution structure development and ES algorithm for optimization of parameters of each individual in GPA population. Such design of evolutionary algorithm prevents situations when good structure solution (e.g. well composed equation) but with wrongly estimated constants (coefficient) is eliminated from population and replaced by individual of worse structure but better fitted constants, which has worse evolutionary potential.

Algorithm 1. Studied GPA algorithm.

1) FOR ALL individuals DO Initialize() END FOR;
2) FOR ALL individuals DO Evaluate() ⇒ fitness END FOR;
3) Sort(individuals);
4) IF Terminal_condition() THEN STOP END IF;
5) FOR ALL individuals DO
   SELECT Rand() OF
   CASE a DO Mutate()⇒ new_individuals;
   CASE b DO Symmetric_crossover() ⇒ new_individuals;
   CASE c DO One_point_crossover() ⇒ new_individuals;
   CASE d DO Re-gerating() ⇒ new_individuals;
   END SELECT;
   END FOR;
6) FOR ALL individuals DO
   New ES_algorithm_object;
   //for each GPA individual new independent parameter optimizer is created
   FOR ALL ES_individuals DO Initialize()END;
   evaluate() ⇒ ES_fitness
   FOR ALL ES_cycles DO
   FOR ALL ES_individuals DO
   Evaluate ⇒ ES_fitness
   END FOR;
   FOR ALL ES_individuals DO
   intelligent_crossover() ⇒ new_ES_individuals
   Evaluate ⇒ new_ES_fitness
   END FOR;
   FOR ALL ES_individuals DO
   IF new_ES_fitness>ES_fitness THEN
   ES_individual = new_ES_individual;
   fitness = new_fitness;
   END IF;
   Sort(ES_individuals, ES_fitness);
   END FOR;
   END FOR;
   new_individual=ES_individual[0];
   new_fitness = ES_fitness[0];
7) FOR ALL individuals DO IF new_fitness<fitness THEN
   individual = new_individual;
   fitness = new_fitness;
   END IF;
8) GOTO 3);

Size of GPA population and sizes of ES populations related to each particular individual in GPA population are the most significant parameters of GPA-ES algorithm. Influence of population sizes was studied in many publications,

see e.g. [2]. Small GPA populations forces evolutionary pressure and in the case of specific conditions it might speed up evolution. On the opposite side, in the small populations there is increased risk of stuck in local optima on the place of global one. Very large populations bring problems with small speed and low efficiency of evolution frequently. Problem is that precise meaning of terms small or big population depends on fitness function landscape. Extremely small GPA populations also bring big dispersion of needed evolutionary cycles and thus also of computational time. In the case of ES populations, analogical reasons are valid only if there are eliminated random influences of task switching and other sources etc.

It is possible to accept results of experiments concluding that large populations might be replaced by bigger number of generations and vice versa. But such reasons are about ability to find solution. When the number of fitness function evaluations (computational complexity) or computational time is evaluated, dependencies between them and population sizes are not the same and small populations are more efficient as it will be presented in the next sections.

There is also parameter representing number of ES algorithm evolutionary cycles. This number plays two contradictory roles. The bigger number of ES iterations means less chance to omit good solution for wrongly identified parameters [3]- [9] and it was the main idea of GPA-ES hybrid algorithm development but the large number of ES iterations significantly increases computational time and thus limits application domain of GPA-ES algorithm. Standard set-up of this algorithm in this study except results presented on Fig. 2 is 40 populations of ES algorithm for each GPA individual but in the last experiment series there are studied influences of ES population number and population sizes onto required number of GPA cycles (and thus whole GPA-ES algorithm number of iterations). Equivalent of pure GPA is none ES population cycle. The experiments published in the next section describe influence of this parameter. Previous study of PRNGs influence onto GPA dynamics pointed that results of experiments are similar and differences between different combinations of PRNGs are on the similar magnitude as observed noise.

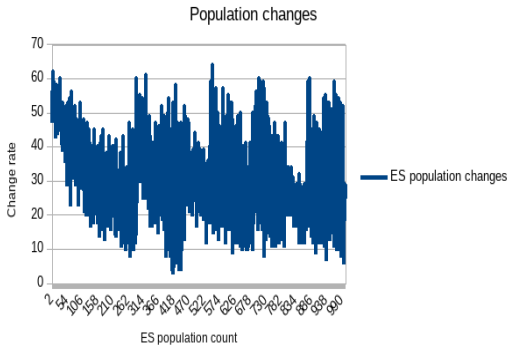

Fig. 1: Count of the ES population improvements for whole GPA population evaluated from the best to the worst individual.

These first experiments were executed with fixed parameters of 1000 different initial PRNGs seed magnitudes. Sizes of populations were 1000 individuals both in GPA and related ES populations. The number of ES algorithm cycles was fixed and equal to 40 (it was not shorten in fitness value of the best individual in the population reached residual error limit, in contrary to superior GP algorithm). Such large populations causes small numbers of iterations and thus small resolution of obtained results.

GPA-ES algorithm allows to apply not only different PRNGs for GPA and ES part of the algorithm but even eight different PRNGs for initialization, re-initialization, mutation and crossover operators and control of GPA and initialization, re-initialization and control including influencing of intelligent crossover operator of ES part.

Resulting average numbers of iterations are displayed on Fig. 2. It can be easily observable that differences between partial combinations are small and random. The meaning of used abbreviations is following: MINSTD0 means simple multiplicative congruential pseudo-random number generator as well as MINSTD one (with different parameter setting), RDEVICE is hardware random number generator that produces non-deterministic random numbers, RANLUX24 subtract-with-

carry pseudo-random generator of 24-bit numbers and RANLUX24 is the same type producing 48-bit numbers, RAND denotes standard C/C++ rand() function and MT19937 represents Mersenne Twister pseudo-random generator of 32-bit numbers with a state size of 19937 bits. Detail dynamics of GPA-ES system is com-

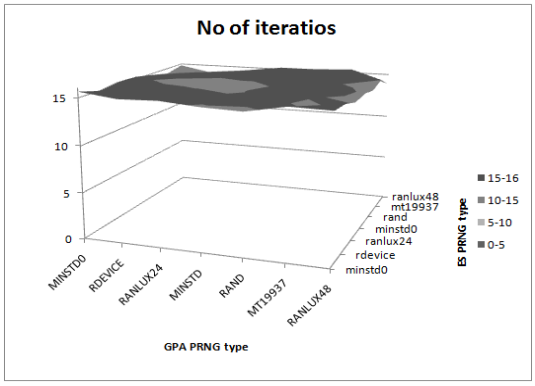

Fig. 2: Average number of iterations of GPA-ES algorithm depending on PRGNs influencing GPA and ES part of the algorithm.

plicated and influenced by many control parameters. Fig. 1 illustrates decrease of population activity in the time. We can identify periods of 100 individuals given by size of GPA population because individuals were evaluated sequentially. Fig. 1. also confirms that after first three evolutionary cycles when only mutation is applied evolutionary activity decreased. In the rest part of data influence of crossover operations which are not used to whole populations but only randomly. They cause periodic boosting of activity. Periods of 100 individuals of decreasing activity also illustrates that activity of individuals with better fitness function is bigger than activity of worse ones non looking to the fact that in each period to each individual one evolutionary operator is applied but this complicated microdynamics study is not the main subject of this work.

Small number of individuals in ES population represents analogy of small populations in GPA part of algorithm. Also there they can bring faster convergence in average (from the viewpoint of time), but because the number of generations is fixed, it cannot correspond to larger number of generations. Thus the quality of results (resulting fitness function magnitudes) must be worse for small ES populations due to ill-identified parameters. In the superior GPA it might cause worse recognition between good and wrong structures against original ideas of GPA-ES design. As the conclusion, if the number of ES population decreases, superior GPA will need more population to achieve the comparable quality results. The significant question is if the faster is to decrease of ES populations or to increase of GPA ones.

Paper [10] was focused to relations between GPA and ES population sizes of composed GPA-ES algorithm. Now the influence of ES population size and ES population number limit is studied applying modified methodology of experiments. Very low limit of generation number for constant optimizing embedded ES part of the algorithm was not studied yet. This modification focuses to elimination influences affecting PRNG used in evolutionary algorithm as

- influence of other tasks running on the same node of computing cluster
- PRNG number series stationarity
- thread switching (if the used number generator is hared between threads or if threads mutually communicate)
- task allocating on cluster nodes

The first point is caused by the fact that other tasks in the operating system influence in the next point described task switching and also function of (P)RNGs if they are not implemented as thread safe. Problem is that each operating system also runs at least different, it is possible to say "service", tasks like network communication support, cluster operation support, etc. These tasks cannot be stopped during these experiments and they can cause changes of parallely executed sub-task run order during experiments. There plays its role also task scheduler of used operating system. Linux systems frequently offers normal, batch, round-robin and FIFO CPU schedulers influencing order of tasks and threads. On clusters there is also the last mentioned source of non-determinism, allocation of tasks on computational nodes which also

might not be deterministic because it s influenced by computer network traffic etc.

Presented experiments eliminate some of these sources of non-determinism by the simplest way – by elimination of multi-thread execution. Each task was running ES for parameter optimization as single-thread one without any communication with others.

3. Experiments and obtained data

Lorenz attractor system served as test case for experiments with symbolic regression of differential equations describing this dynamic system on the base of pre-computed data set. Lorenz attractor system is described by (1) and (2).

$$\begin{cases} x'(t) = \sigma\,(y(t) - x(t)), \\ y'(t) = x(t)\,(\rho - z(t)) - y(t), \\ z'(t) = x(t)\;y(t) \;-\beta\, z(t) \end{cases} \tag{1}$$

$$\sigma = 16, \beta = 4, \rho = 45.91 \tag{2}$$

The limit of GPA algorithm iteration number was set to 10000 cycles, the termination condition was set to sum of error squares less than 10-8 for applied 599 samples of training data, while the number of ES algorithm iterations has varying between magnitudes 1,10 and 100, as well as the sizes of ES populations was varying magnitudes 10, 40 and 100. Size of GPA population was 64 individuals. Experiments were repeated 10000 times for different seeds of used PRNGs.

In the presented study we use two variable parameters, number of ES algorithm iterations and size of ES populations. They determine number of evaluations of fitness function both in ES and composed GPA-ES algorithms. Standard C and C++ function rand() served as PRNG in herein presented experiments. To analyze computational efficiency of GPA-ES algorithm, both number of needed GPA cycle iterations and computational time are measured (computational time can be replaced by another HW independent measure, number of fitness function iterations).

Results of experiments are presented on the following Fig. 3-8.

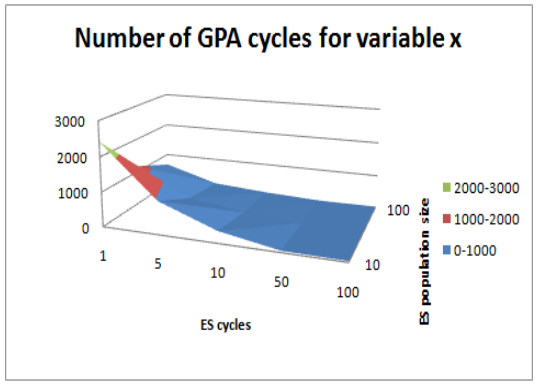

Fig. 3: Number of iterations of encapsulating GPA for x variable depending on ES.

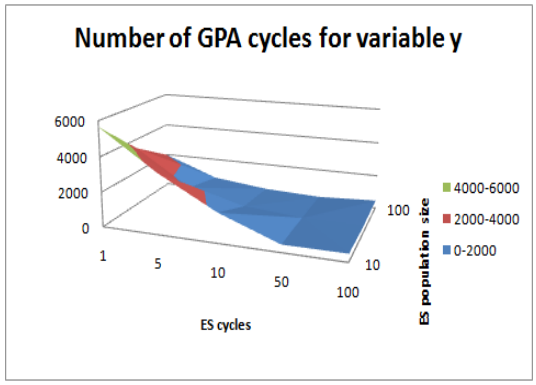

Fig. 4: Number of iterations of encapsulating GPA for y variable depending on ES cycle limit and ES population size.

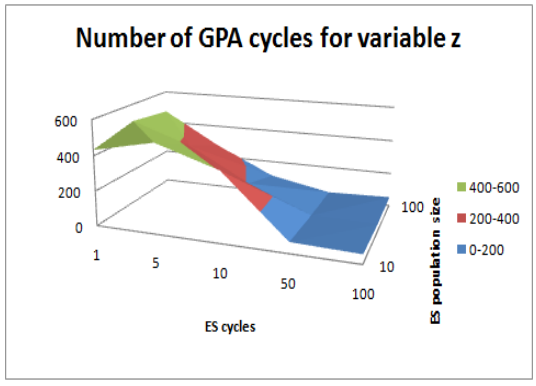

Fig. 5: Number of iterations of encapsulating GPA for z variable depending on ES cycle limit and ES population size.

Because above presented elimination of entropy sources lying out of used PRNG are not sufficient and e.g. running GPA-ES on the same

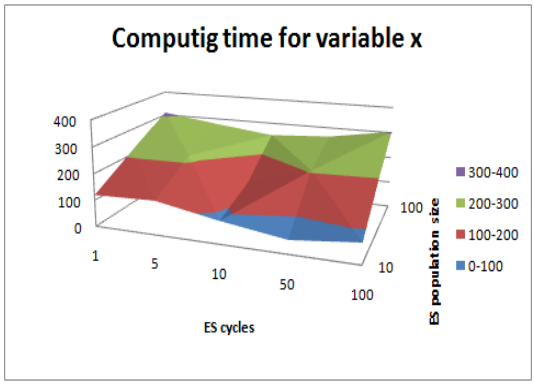

Fig. 6: Computing time of whole GPA-ES for x variable depending on ES cycle limit and ES population size.

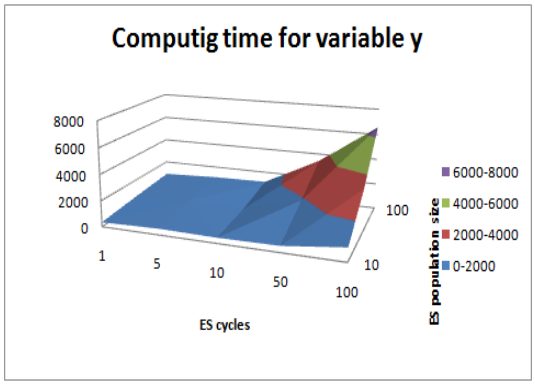

Fig. 7: Computing time of whole GPA-ES for y variable depending on ES cycle limit and ES population size.

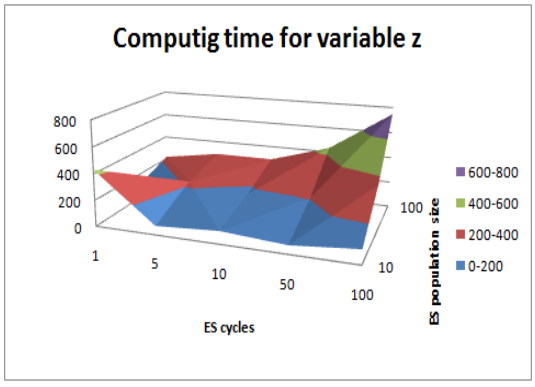

Fig. 8: Computing time of whole GPA-ES for z variable depending on ES cycle limit and ES population size.

data with the same used PRNG seed magnitudes give different results in each run, new provision was applied. The OpenMP was not limited to 1 thread but eliminated completely.

Evolutionary algorithm provides numerical computations and there in intensive mixing of cooperating individuals during the time of task solving. The last research of such dynamic systems points that there might be observed many types of chaotic attractors [11]. To prevent chaotic behavior of GPA-ES and fitness function evaluation, the numerical precision was extended from 64bit double data type to 80bit to long double one. GNU C++ compiler also offers 128bit_float128 data type but it is extremely slow, so it was not used. Table 1 presents increasing of repeated experiments variability with growing number of ES cycles. This situation can have only one explanation – presence of undiscovered source of entropy. Such source can be undiscovered interaction between application and operating system or numerical instability of regressed mode.

ES cycles No	x	y	z
5	0	0	0
10	57,93	53684,07	772,40
40	991,44	120123,97	1204,52

Tab. 1: Experiment variability depending on number of ES cycles for variables x, y and z

Observations summarized in the Table 1 points to Lamarckian inheritance [12] and especially to Baldwin effect [13]. Even small improvements of individual behavior can bring improvement when it is coded back into DNA. Thus it is possible to apply computationally easier search of parameters sub-optima on the place of computationally exhaustive searches.

4. Conclusions

Above presented data are depending on two parameters - number of ES algorithm generations and size of ES populations. While for simplest equation describing x variable in Fig. 2 was for smallest ES population of 10 individuals, more complex equations describing dynamics of y and

z variables required larger ES population of 100 individuals.

Computing time was different. There was smaller variance of results and the optima were between 5 and 50 populations and smallest populations of 10 individuals. Such results partially defers to above described expectations and they are caused by computational complexity of ES algorithm which depends quadratically on the number of individuals but the improvement of larger population to convergence is of the lower order.

Presented data points that GPA-ES algorithm behavior corresponds to expectations in computational complexity. Time complexity of algorithms corresponds less and it is given by used hardware properties.

Non looking that Fig. 1 presents decrease of GPA efficiency, small populations and higher numbers of iterations are interesting way of efficiency increasing except extremely low magnitudes. Moreover, with decreasing of population members number and with increasing of count of evolutionary cycles there increases dispersion of needed iterations and thus even if the average number is small, some runs can be extremely long.

Presented experiments also points that computational time is controversial measure of algorithm efficiency – from the algorithm complexity viewpoint numbers of individuals can be replaced by amount of iterations but GPA and ES algorithms has incomparable complexity and their implementation might have different efficiency. ES works faster, thus in time space comparison the results displayed on Figs. 5-8 are not significant and do not copy results from Figs. 2-4 representing numbers of iterations (evolutionary cycles).

Presented study uses relatively simple problem as case study. In the future, when more computational time will be obtained, there will be need to repeat herein presented experiments on more complex problems where the area of highest efficiency is expected far from the lowest population sizes and iteration numbers of ES.

Acknowledgements

The work was supported from ERDF/ESF " Cooperation in Applied Research between the University of Pardubice and companies, in the Field of Positioning, Detection and Simulation Technology for Transport Systems (PosiTrans)" (No. CZ.02.1.01/0.0/0.0/17_049/0008394).

Access to computing and storage facilities owned by parties and projects contributing to the National Grid Infrastructure MetaCentrum provided under the programme "Projects of Large Research, Development, and Innovations Infrastructures" (CESNET LM2015042), is greatly appreciated.

References

[1] Zelinka, I., Oplatkova, Z., & Nolle, L. (2005). Analytic programming–Symbolic regression by means of arbitrary evolutionary algorithms. Int. J. of Simulation, Systems, Science and Technology, 6(9), 44-56.

[2] Brandejsky, T. (2012). Evolutionary system to model structure and parameters regression. Neural Network World, 22(2), 181.

[3] Brandejsky, T. (2013). The Use of Local Models Optimized by Genetic Programming Algorithms in Biomedical-Signal Analysis. In Handbook of Optimization (pp. 697-716). Springer, Berlin, Heidelberg.

[4] Alander, J. T. (1992, May). On optimal population size of genetic algorithms. In 1992 Proceedings Computer Systems and Software Engineering (pp. 65-70). IEEE.

[5] Eiben, Á. E., Hinterding, R., & Michalewicz, Z. (1999). Parameter control in evolutionary algorithms. IEEE Transactions on evolutionary computation, 3(2), 124-141.

[6] Koumousis, V. K., & Katsaras, C. P. (2006). A saw-tooth genetic algorithm combining the effects of variable population size and reinitialization to enhance performance. IEEE Transactions on Evolutionary Computation, 10(1), 19-28.

[7] Lobo, F. J., Lima, C. F., & Michalewicz, Z. (Eds.). (2007). Parameter setting in evolutionary algorithms (Vol. 54). Springer Science & Business Media.

[8] Reeves, C. R. (1993, June). Using Genetic Algorithms with Small Populations. In ICGA (Vol. 590, p. 92).

[9] Piszcz, A., & Soule, T. (2006, July). Genetic programming: Optimal population sizes for varying complexity problems. In Proceedings of the 8th annual conference on Genetic and evolutionary computation (pp. 953-954). ACM.

[10] Brandejsky, T. (2013). Small populations in GPA-ES algorithm. In: Matousek, R., (ed.) MENDEL 2013. 19th Interence Conference on Soft Computing MENDEL 2013. Brno, 31-36.

[11] Isaeva, O. B., Kuznetsov, S. P., & Mosekilde, E. (2011). Hyperbolic chaotic attractor in amplitude dynamics of coupled self-oscillators with periodic parameter modulation. Physical Review E, 84(1), 016228.

[12] Ghiselin, M. T. (1994). The imaginary Lamarck: a look at bogus 'history' in schoolbooks. The Textbook Letter, 5(4).

[13] Baldwin, J. M. (1896). A new factor in evolution. The american naturalist, 30(354), 441-451.

About Authors

Tomas BRANDEJSKY obtained PhD from technical cybernetics at Czech Technical University in Prague and he now is Associated Professor at University of Pardubice. His research interest focuses to area of soft-computing, especially to interval valued fuzzy sets and genetic programming.