Investigating the Efficiency of Parallel Algorithms for Stochastic Optimization

Marat Voronukhin
Samara State Transport University
Samara, Russia
wape123@list.ru

Valery Zasov
Samara State Transport University
Samara, Russia
vzasov@mail.ru

***Abstract*—The paper presents the results of investigating the efficiency indicators of two parallel algorithms for stochastic optimization—the genetic algorithm and the particle-swarm algorithm. The paper discusses experimental relationships for the efficiency indicators of the parallel algorithms: runtime, acceleration, error, and solution stability, depending on the type and dimension of the target function, the number of agents used, and processor cores. The comparative estimates for the algorithms are given.**

Keywords—stochastic optimization, genetic algorithm, swarm algorithm, parallel computing, efficiency indicators, acceleration, stability, error, computational experiments, software package

I. Introduction

The effectiveness of optimal control depends significantly on the characteristics of optimization algorithms used and on how those algorithms are implemented. One difficulty arising in solving optimal-control problems is the expected uncertainty in the properties of objects and control processes. When information is incomplete, optimization problems are most efficiently solved through stochastic algorithms since the efficiency of deterministic algorithms largely depends on the conditions of the particular problem [1,2].

Random search is successfully used when the target function is multiextremal and discontinuous and the solution space is nonuniform, i.e. when it is impossible to identify the conditions of approaching the optimal solution [2,3]. Although stochastic algorithms (as well as deterministic ones) do not guarantee exact solutions, they normally allow, in a time that is reasonable for operational-control tasks, to find solutions that are close enough for practical use.

Among stochastic algorithms, we will discuss two groups of optimization algorithms borrowed from nature—evolutionary algorithms and swarm-intelligence algorithms. These algorithms are distinguished by the use of populations of individuals (agents) while searching for solutions and are focused on efficient multiagent programming processes [2,3]. The natural parallelism of these algorithms allows the more simpler implementation of their parallel variants intended to run on parallel computing systems. That is important for providing real-time conditions in control systems [4].

On the other hand, the application of stochastic optimization algorithms is not a trivial task because of their stochastic nature. Therefore, before practical use in applied problems, it is advisable to carry out computational experiments to determine the speed, stability, error, and other indicators of algorithm efficiency for the specific algorithmic implementation.

In addition, parallel optimization algorithms are more complicated than sequential ones, and the costs of organizing the information interaction between the threads increase significantly as the degree of parallelization increases. This factor adversely affects the increase rate of computation acceleration when the number of threads increases; therefore, evaluation of the parallelization efficiency for optimization algorithms also requires computational experiments.

The purpose of this paper is to compute and analyze efficiency indicators for two groups of parallel algorithms for stochastic discrete optimization—evolutionary algorithms and swarm-intelligence algorithms.

II. Parallel Stochastic Optimization Algorithms and Software for Their Investigation

The problem of stochastic optimization is one of determining the extremum for the target function $f(X)$:

$$f_{opt}(X) = \max_{X \in D} f(X) \text{ or } f_{opt}(X) = \min_{X \in D} f(X),$$

where X is the vector of variable parameters; and D is the tolerance range for X. For purposes of further discussion, we will introduce S as the number of population agents, each of which represents a certain point in the search space for problem solutions. The optimization process consists in the controlled movement of agents in this space.

Algorithms of evolutionary optimization—the classic genetic algorithm and its various modifications—involve the generation of new agent populations at each step (iteration), subject to the experience gained by previous agent populations.

The most common genetic algorithm (GA) consists of the following steps: generating the initial population of agents (species), S; calculating the target function $f(X)$ for each of the agents S; selecting and crossing the agents; observing random changes to the agents (mutation); and ending the algorithm if the termination condition is satisfied; otherwise, recalculating the target function $f(X)$ [2,5] is performed.

Swarm-intelligence algorithms (particle swarm, bee swarm, fish-swarm search, etc.) are based on organizing the decentralized movement of agents in one population without selecting, destroying, or generating agents, and these algorithms use a set of rules for the exchange of information between agents.

The particle-swarm optimization (PSO) algorithm, which is the most typical for this group, consists of the following steps: activating the initial states of the agents S; calculating the target function $f(X)$ for each of the agents S; migrating (moving) the agents subject to the rules for setting the magnitude of the speed vector directed to a new

978-1-7281-6700-8/19/$31.00 ©2019 IEEE

point; and updating the coordinates of the best solution found for each of the agents S and for all agents in general. Then the iteration is repeated until the termination condition is satisfied [2,6,7].

An analysis of the considered optimization algorithms allows to determine that the main method for creating parallel versions of the algorithms is to allocate for each agent a separate calculator—a processor core, which in turn can parallelize the algorithm steps (activating the agents, calculating the target function, etc.) as threads. To reduce the time needed for information interaction between agents, the use of shared memory is advisable.

To solve this problem, a test software package has been developed that allows one to perform computational experiments and obtain experimental relationships for the efficiency indicators of parallel algorithms for stochastic optimization, such as runtime, acceleration, and error depending on the type and dimension of the target function, the number of agents, and processor cores used, as well as comparative evaluations of these indicators for different algorithms.

The software package is designed to be installed on multicore computing systems that use multithread processing technology [8,9]. The scalability of parallel algorithms is improved by the CUDA (computer unified device architecture) technology [10,11], which allows low-cost high-performance parallel computing systems to be created based on graphics processors (GPU).

The use of computational platforms and programming languages that are conventional for control problems allows one to obtain not only qualitative but also quantitative parameters for the efficiency indicators of parallel optimization algorithms for specific computing systems.

This allows to assess more precisely the possibility of using optimization algorithms for the given application conditions.

The software package consists of modules for selecting a target function; selecting an optimization algorithm and setting its parameters; planning and carrying out computational experiments; calculating the efficiency indicators of optimization algorithms under specific conditions of use; generating summary reports in tabular and graphical forms based on the results of the experiments; and organizing archives for the results of the experiments.

The module used to select target optimization functions allows one to select a target function from the library of functions most commonly used for testing—the McCormick, Rastrigin, Beale, Matyas, Booth, Three-Hump Camel, or Easom function [5,12]—or to use a target function proposed by the developer.

The module for selecting optimization algorithms allows to select one of the sequential or parallel stochastic optimization algorithms from a library, and to set its parameters.

The planning module for computational experiments is designed to calculate the efficiency indicators for algorithms under specific conditions of use.

III. Computational Experiments—Results and Analysis

On the basis of the software package, computational experiments have been conducted to determine the efficiency indicators of parallel algorithms which conventionally represent evolutionary and swarm stochastic optimization algorithms.

We selected the genetic algorithm for the evolutionary group and the particle-swarm algorithm for the swarm-intelligence group. The McCormick function [5,12] is used as the target function for the experiment series.

Fig. 1 shows the experimental relationships between the runtime of sequential and parallel variants of the genetic and swarm algorithms and the number of agents.

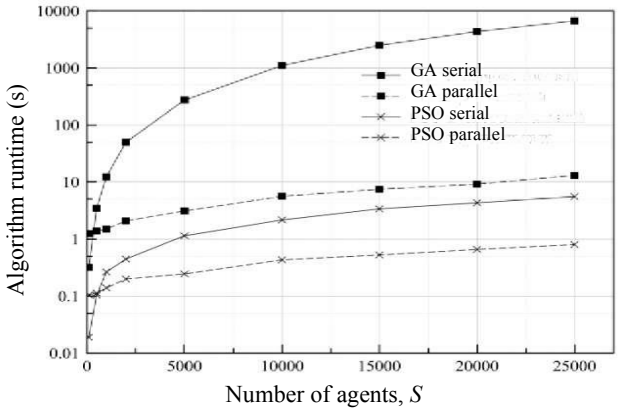

Fig. 1. Experimental relationships between the runtime of genetic and PSO algorithms and the number of agents

The condition to the completion of the algorithms is the execution of a given number of iterations (in the experiments, the number of iterations is 200). The runtime of the sequential and parallel variants of the GA is almost two orders longer than the runtime of the PSO.

This is caused by the fact that algorithms for the transformation of agents in genetic algorithms (crossing-over, mutations, change of agent generations) are significantly more complicated When the condition $S > 10,000$ is valid for the number of agents, the sequential genetic algorithm is practically unacceptable for the use in operational-control problems because of the long runtime.

On the other hand, a comparison of the runtimes for the sequential and parallel variants of the algorithms show the high efficiency of parallelizing the GA.

This conclusion is confirmed by the calculated acceleration values of the genetic and PSO algorithms presented in Fig. 2.

Algorithm acceleration [8] means the value of the ratio of the execution time for a given number of iterations (200 for the above experiment) by the sequential and parallel variants of the algorithms.

The acceleration achieved by the parallel GA is much higher than one obtained by the parallel PSO algorithm; this advantage increases with growing the number of agents.

Thus, the efficiency of using agents (processor cores), determined by the ratio of the acceleration value to the number of agents, is substantially greater in parallel genetic algorithms.

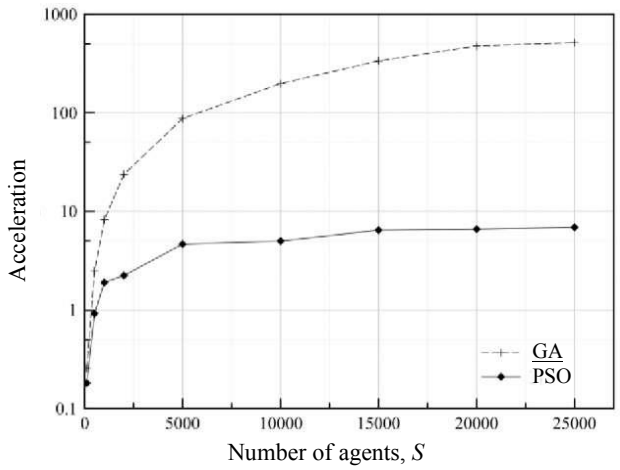

Fig. 2. Experimental relationships between the acceleration of parallel genetic/PSO algorithms and the number of agents

An important indicator of the effectiveness of optimization algorithms is the value of error in calculating the extremum of the target function $f_{opt}(X)$.

The magnitude of the error depends on the difference between the test target function's extremum calculated by the given algorithm and its known reference value.

The error value (called relative error) will be measured by the ratio of the diameter of the neighborhood at the extremum of the test target function to the magnitude of that extremum.

Given the stochastic nature of the algorithms under study, we investigated the relationship between the probability of calculating the extremum with specified relative error and the value of relative error.

Fig. 3 shows the results obtained from a series of experiments with 200 iterations for each of the algorithms.

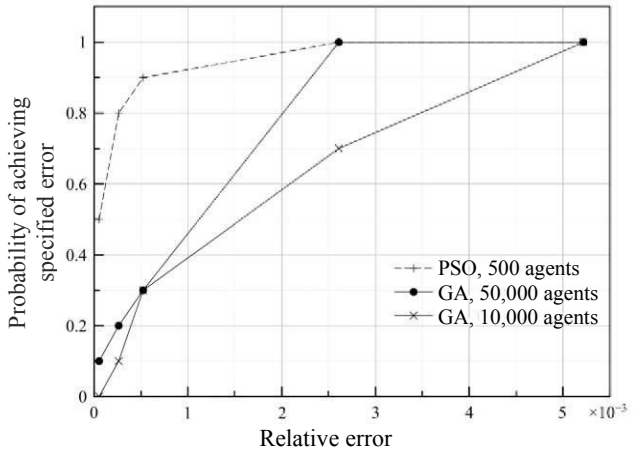

Fig. 3. Experimental relationships between the probability of calculating an extremum with specified error and the magnitude of relative error

The results indicate the evident advantage of the particle-swarm algorithm, even where the number of agents is small (500 for PSO and 50,000 for GA). Computational experiments to estimate the error of the sequential genetic and swarm algorithms have shown that the results are comparable with the results obtained for the parallel algorithms.

For practical applications, a relevant indicator of the efficiency of optimization algorithms is the time for approaching a stable solution (convergence of algorithms).

We propose measuring the stability of a solution by the number of consecutive solutions (calculations of the extremes of the target function) whose relative error does not exceed the specified error.

For specific problems, it is convenient to use the ratio of the stability indicator as described above to the stability indicator that is ideal for the given problem.

Fig. 4 shows examples of experimental relationships between the time for approaching a stable solution and the value of relative stability.

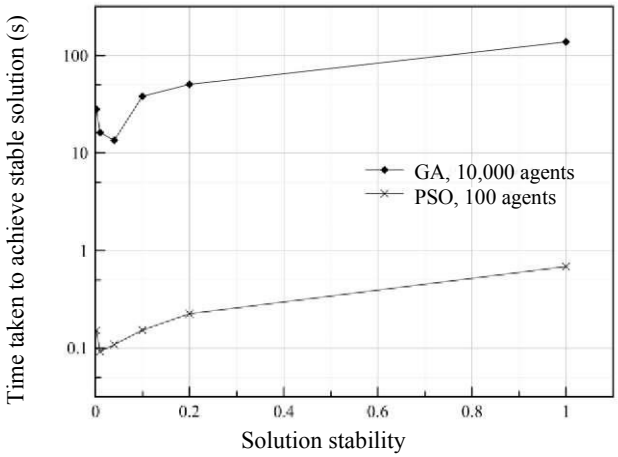

Fig. 4. Experimental relationships between the time taken to achieve a stable solution and the relative stability of the solution

The time for aproaching a stable solution for the particle-swarm algorithm is much shorter as compared with the time for genetic algorithm. However, the dependences are similar in their form, and an increase in relative stability leads to a significant increase in the time for calculating the extremum of the objective function.

The results were obtained for a relative error of 1×10^{-3}, while the ideal stability indicator for the problem was 5,000.

The experimentally obtained time characteristics of the parallel genetic algorithm and the particle-swarm algorithm for various target functions are listed in Table I.

TABLE I. ALGORITHM COMPLETION TIME FOR VARIOUS TARGET FUNCTIONS

Test target function	GA runtime (s)	PSO algorithm runtime (s)
Rastrigin	0.311	4.631
Beale	0.316	Wide scatter in results
Booth	0.333	0.132
Matyas	0.306	0.221
Three-Hump Camel	0.313	0.231
Easom	1.188	0.446
McCormick	1.185	0.189

The runtime of the relatively slow genetic algorithm under conditions of the expected uncertainty of the target function has a smaller scatter compared with the runtime of the particle-swarm algorithm, i.e. this algorithm is more stable.

Under the conditions of the small apriori uncertainty of the target function, it is possible to use the particle-swarm algorithm with the appropriate adjustment of the algorithm constants responsible for moving the agent at each iteration.

This algorithm is a more specialized one, and the solution for the particle-swarm algorithm may not fall into the region of the extremum when the constants are adjusted improperly. A large value of the agent step leads to an unstable operation of the particle swarm algorithm, this drawback is observed when testing the algorithm on the target Beal function.

An option for practical use of the PSO algorithm is shown in Fig. 5 as a block diagram for the adaptive parametric identification of the controlled object.

Contemporary monitoring and control are largely based on the parameters of models that describe controlled objects; therefore, the prompt determination of parameters of the models is an important control-related problem.

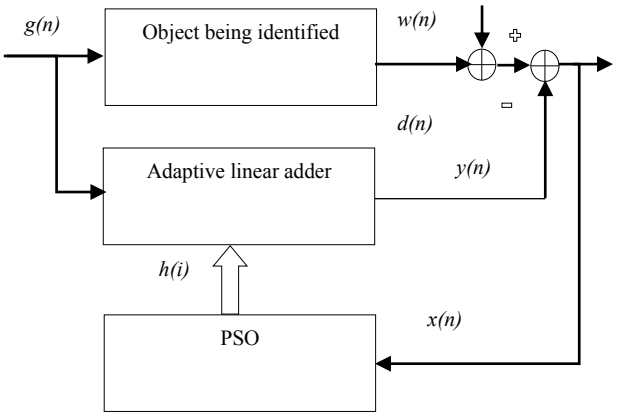

Fig.5. Block diagram of adaptive parametric identification based on the parallel swarm optimization algorithm

Once the adaptation process is complete, i.e. the extremum (the minimum value of the target function of the mean-square error $E\left\{\left|x(n)\right|^2\right\}$) is found, the parameters $h(i)$ of the adaptive linear adder will be close to the parameters of the identification object model. Then the calculated parameters are used in the control system.

The effectiveness of using a parallel swarm optimization algorithm in identification problems is confirmed by the results of computer simulation shown in Fig. 6.

Under the conditions of observation noise $w(n)$, given by uniform white noise with a dispersion of 0.08, an object was identified with previously known (reference) parameters, the number of which is $i = 32$.

A comparison using the criterion of the average absolute deviation of the calculated model parameters from the reference ones shows an identification error of ~ 4.4% acceptable for engineering applications.

The identification time under the conditions of observation noise $w(n)$ is 0.06 seconds. This time is two times shorter than the identification time, for example, by application of the widely used deterministic least-mean-square algorithm [13].

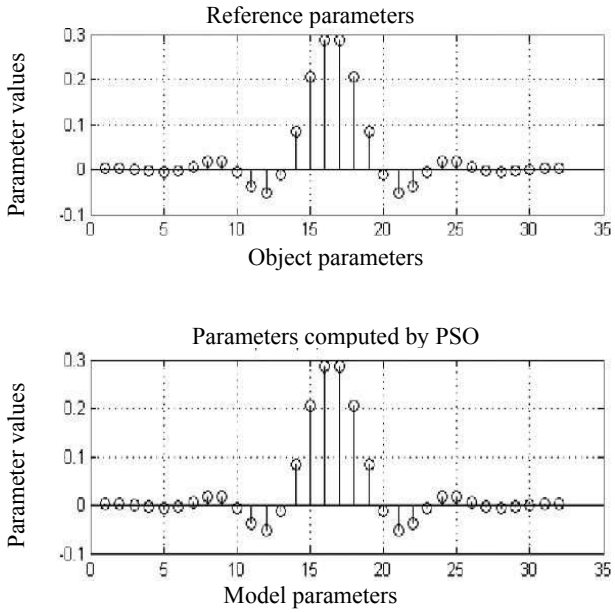

Fig.6. Reference and calculated model parameters

As the number of parameters (the dimension of the target function) increases, the time gain in comparison with the deterministic algorithm increases and reaches a factor of 2.8 at $i = 512$.

IV. Conclusion

The paper presents the results of investigating the efficiency indicators of two parallel algorithms for stochastic optimization—the genetic algorithm and the particle-swarm algorithm.

This paper discussed experimental relationships for the efficiency indicators of parallel algorithms: runtime, acceleration, error, and stability depending on the type and dimension of the target function, the number of agents, and processor cores used. Comparative evaluations for the algorithms are also represented.

An example of the practical application of the particle swarm optimization algorithm to solve problems of adaptive parametric identification under observation-noise conditions is considered, and it has shown advantages in terms of speed over the deterministic least-mean-square algorithm.

Computational experiments have been performed using a software package with multi-core architecture and the CUDA multithread processing technology.

References

[1] F.P. Vasil'ev, Metody optimizacii [Optimization Methods]. Moskva, Faktorial Press, 2002. (in Russian)

[2] P.V. Matrenin, M.G. Grif and V.G.Sekaev, Metody stohasticheskoj optimizacii: uchebnoe posobie [Methods for Stochastic Optimization: a Textbook]. Novosibirsk, Izd-vo NGTU, 2016. (in Russian)

[3] D. Rutkovskaya, M. Pilin'skij and L. Rutkovskij, Nejronnye seti, geneticheskie algoritmy i nechetkie sistemy [Neural Networks,

Genetic Algorithms, and Fuzzy Systems]. Moskva, Goryachaya liniya – Telekom, 2006. (in Russian)

[4] M.B. Gitman, Vvedenie v stohasticheskuyu optimizaciyu: ucheb. posobie [Introduction to Stochastic Optimization: a Textbook]. Perm', Izd-vo Perm. nac. issled.politekhn. un-ta, 2014. (in Russian)

[5] R.l. Haupt, and S.E. Haupt, "Practical Genetic Algorithms,"2ed, Hoboken, Naw Jersey, J. Wiley@Sons, Inc., 2004.

[6] I.C. Trelea, "The particle swarm optimization algorithm: convergence analysis and parameter selection," Information Processing Letters, no. 85, 2003, pp. 317–325.

[7] M.E. Voronuhin, "Issledovanie effektivnosti parallel'nogo algoritma roya chastic dlya zadach optimizacii," Analiticheskie i chislennye metody modelirovaniya estestvenno-nauchnyh i social'nyh problem: Trudy XIII mezhdunarod. nauchno-tekhn. konferencii ["Investigating the efficiency of the parallel particle-swarm algorithm for optimization problems," Analytical and Numerical Methods for Modeling Natural Science and Social Problems: Proceedings of XIIIth International Scientific and Technical Conference]. Penza, PGU, 2018, pp.145-150. (in Russian)

[8] V.P. Gergel', Teoriya i praktika parallel'nyh vychislenij: uchebnoe posobie [Theory and Practice of Parallel computations: a Textbook]. Moskva, Internet-Universitet Informacionnyh Tekhnologij: BINOM, Laboratoriya znanij, 2007. (in Russian)

[9] Yu.I. Demyanovich, I.G. Burova, T.O. Yevdokimova, O.N. Ivantsova and I.D. Miroshnichenko, Parallel'nye algoritmy. Razrabotka i realizaciya: uchebnoe posobie [Parallel Algorithms. Development and Implementation: a Textbook]. Moskva, Internet-Universitet Informacionnyh Tekhnologij: BINOM, Laboratoriya znanij, 2012. (in Russian)

[10] A.V. Boreskov and A.A. Kharlamov, Osnovy raboty s tekhnologiej CUDA [Basics of CUDA Technology]. Moscow, DMK Press, 2011. (in Russian)

[11] A.V. Boreskov, Parallel'nye vychisleniya na GPU. Arhitektura i programmnaya model' CUDA: uchebnoe posobie [GPU-Based Parallel Computations: The Architecture and Program Model of CUDA: a Textbook]. Moskva, MGU, 2012. (in Russian)

[12] S.K. Mishra, "Some New Test Functions for Global Optimization and Performance of Repulsive Particle Swarm Method," Shillong, North-Eastern Hill University, Shillong, p.26, August, 2006, MPRA Paper No. 2718, posted 13, April 2007.

[13] V.I. Dzhigan, Adaptivnaya fil'traciya signalov: teoriya i algoritmy [Adaptive Signal Filtering: Theory and Algorithms]. Moskva, Tekhnosfera, 2013. (in Russian)