The task of setting the parameters of metaheuristic optimization algorithms

N M Lugovaya[1], A S Mikhalev[1], V V Kukartsev[1], V S Tynchenko[1], V A Baranov[1], A O Kolbina[1] and E A Chzhan[1]

[1] Siberian Federal University, 79, Svobodny pr., Krasnoyarsk, 660041, Russia

E-mail: vlad_saa_2000@mail.ru

Abstract. When solving global optimization problems, many problems arise related to their multi-extremity, nonlinearity, high computation complexity, and other. To solve these problems, a large number of algorithms were created. This article focuses on analyzing the class of metaheuristic algorithms for searching for extrema: testing them on various functions and comparing their behavior at the most optimal parameters. Particle swarm algorithms are considered; bats, swarming bees, swarming fireflies, bacterial. In the framework of this article, a study of various metaheuristic extremum search algorithms, the selection of optimal parameters for each of them using various functions as an example was conducted. Also, these algorithms solved the problem of designing a cylindrical spring with a constant axial load.

1. Introduction

Today, one of the most pressing problems in the field of computational mathematics is the development of effective methods for solving global optimization problems. In the simplest case, the optimization problem is to maximize or minimize the objective function, that is, to ensure that we find the best option for all possible.

Global optimization methods have a greater advantage over standard local search methods, since they are often not able to go beyond the zone of attraction of local optima, and therefore are unable to detect the global optimum [1-3]. Using the found local solutions may not be appropriate due to the fact that the global optimum may provide a significant advantage in relation to the local optimum. [4-7]

When solving global optimization problems, many problems arise related to their multi-extremity, nonlinearity, high computation complexity, and other. To solve these problems, a large number of algorithms were created. Furthermore, among them there is no universal algorithm for solving any problem. Moreover, despite the abundance of algorithms, the task of researching and tuning existing global optimization algorithms remains relevant. [8-11]

This article focuses on the analysis of classical extremum search algorithms: testing them for various functions and comparing their behavior with the most optimal parameters. The following algorithms are considered [1, 2, 4, 6, 9, 10]:

- Particle swarming.
- Bats.
- Swarming bees.
- Swarm of fireflies.

- Bacterial.

Also, these algorithms solved the problem of designing a cylindrical spring with a constant axial load.

2. The task of finding a global extremum

In a general formulation, the global optimization problem is formulated as following:

$$f^* = f(X^*) = \min_{X \in R^m} f(x), \tag{1}$$

where $f(x)$ – objective function, X^* – globally optimal point or globally optimal solution, $X = \{x_1, x_2, \dots, x_m\}$ – m-dimensional vector of the sought variables, R^m – Euclidean space.

The area in which the solution of the global optimization problem is sought is given as follows:

$$D = \{X | x_i^{min} \le x_i \le x_i^{max}, i = \overline{1,m}\} \subset R^m. \tag{2}$$

The optimized function satisfies the Lipschitz condition with a constant L, i.e. inequality holds:

$$|f(x) - f(z)| \le L\|x - z\|_\infty, \ \|x\|_\infty = \max_{1 \le i \le n} |x_i|. \tag{3}$$

This condition limits the growth of the function. The function is limited and continuous almost everywhere on X.

Algorithms for solving global optimization problems are divided into two classes: deterministic and stochastic [8]. Stochastic algorithms are algorithms that use randomness when searching for optimum. Accident manifests itself in estimating the value of a function at various points in the search area, followed by processing the data. Deterministic algorithms are more rigorous, they get a global solution by exhaustive search on the entire search range. At the same time, they lose their accuracy, speed and efficiency with increasing dimension of the task. In turn, stochastic algorithms do not give an absolute guarantee of finding a global optimum [4, 6-8]. But despite this, the class of stochastic algorithms gives the highest efficiency in solving problems [3, 9]. Its efficiency, as a rule, varies widely depending on the initial approximation obtained at the stage of population initialization. In this regard, to evaluate the effectiveness of these algorithms, multiple runs of the algorithm based on different initial approximations are used. The main criteria for the effectiveness of population-based algorithms are the reliability of the algorithm — an estimate of the localization probability of the global extremum, as well as its rate of convergence — an estimate of the expectation of the required number of tests (calculations of the value of the function to be optimized).

3. Statement of the problem of finding optimal parameters

The efficiency of the algorithms when searching for a global optimum is due to the adjustment of the corresponding parameters. It is required to reveal a set of parameter values that is universal for the classes of functions under consideration.

Let $a(\vec{\theta})$ be a considered algorithm for solving problems of a given class, where $\vec{\theta}$ – vector of tunable algorithm parameters. Suppose that for each of the components of the parameter the intervals of its permissible values are set, so that the set of permissible combinations of the algorithm parameters is defined:

$$D_\theta = \{\theta_i | \theta_i^- \le \theta \le \theta_i^+, i \in [1: |B|]\}. \tag{4}$$

The set of possible settings for the algorithm parameters:

$$A = \{a(\vec{\theta}) | \vec{\theta} \in D_\theta\}. \tag{5}$$

For each class of functions, it is required to find a combination of algorithm parameters that provides the minimum value of the objective function of the form:

$$\min_{\vec{\theta} \in D_\theta} \mu\left(f, a(\vec{\theta})\right) = \mu\left(f, a(\vec{\theta}^*)\right). \tag{6}$$

The resulting task (6) is also an optimization task. Let us dwell on its solution on the random search method. The complex characteristic of the search for the global minimum was chosen to estimate the probability of finding the true solution $\hat{P}_{true}$. For each test task, the study was conducted using multiple runs of each algorithm with different combinations of the parameter under consideration with the others fixed. The estimate of the probability of finding a true solution is usually calculated as the ratio of the number of favorable outcomes to the total number of launches. At the same time, the launch is considered "favorable" if the found value of the objective function is as close as possible to the correct answer, i.e. if the condition is met:

$$\min_{\vec{\theta}\in D_\theta} \mu\left(f, a(\vec{\theta})\right) = \mu\left(f, a(\vec{\theta}^*)\right). \quad (7)$$

where x^*– found position of global minimum, x^{**}– true position of the global minimum.

For research, we selected fundamentally different test properties: De Jong, Goldman-Pryce, Shestigorbogo camel, Himmelblau, Schweifel, as well as multi-extremal potential function. All classes behave in their own special way; therefore, the best for them are their algorithm parameters.

4. Customizable parameters

According to the results of the selection of parameters for each algorithm, the following ranges of optimal parameters were obtained.

For Firefly swarm algorithm (FSA):

- $\mathrm{Num} = [200; 400]$ – number of fireflies.
- $\mathrm{Iter} = 1000$ – number of loop iterations.
- $\gamma = 1$ – gamma (local variable setting the boundaries for the firefly position).
- $\beta = 1$ – base beta (local variable setting the boundaries for the firefly position).
- $\alpha = 0.1$ – alpha (local variable setting the boundaries for the firefly position).

For Bacterial algorithm (BCA):

- $\mathrm{S} = [100; 700]$ – number of bacteria.
- $\mathrm{N_c} = [150;\ 200]$ – bacterium lifetime.
- $\mathrm{N_s} = [6;\ 8]$ – maximum number of steps in the chosen direction.
- $\mathrm{N_{ed}} = 4$– the number of the type of event destruction - «dispersion».
- $\mathrm{N_{re}} = 5$ – number of generations in the population.
- $\mathrm{P_{ed}} = 0.25$ – probability of killing bacteria.
- $\mathrm{C} = 0.05$ – basic movement length for each bacterium in one step.

For Bats algorithm (BTA):

- $\mathrm{Num} = [100; 600]$ – number of bats.
- $\mathrm{R} = 0.9$ – volume gain coefficient.
- $\mathrm{Iter} = [700; 1000]$ – number of iterations.
- $\mathrm{A} = 0.8$ – bat volume attenuation coefficient.
- $\lambda = [0; 2] - [1; 3]$ – bat wavelength.
- $\mathrm{r} = 0.8$ – pulse frequency.
- $\mathrm{a} = [0{,}1;\ 0{,}9]$ – bat volume.
- $\vartheta = [-2;\ 2] - [-3;\ 3]$ – bat flight speed.

For Bee swarm algorithm (BSA):

- $\mathrm{S} = [200; 300]$ – the number of scout bees.
- $\mathrm{Threshold} = 1$ – threshold value of the distance between the bees at the initial moment.
- $\Delta = [0{,}8;\ 0{,}85]$ – local scope parameter.
- $\mathrm{B} = [10;\ 12]$ – the number of bees sent to the «best areas».
- $\mathrm{P} = 3$ – the number of bees sent to «promising areas».

- $K = [80;\ 60]$ – maximum number of iterations.
- $b = 2$ – number of selected best values.
- $p = 3$ – the number of selected promising values.
- $R = 0.9$ – area reduction coefficient.

For Particle swarm algorithm (PSA):

- $K = [80; 120]$ – maximum number of iterations.
- $NP = [30; 100]$ – number of particles in the swarm.
- $NI_{min} = [10; 17]$ – least number of neighbours.
- $NI_{max} = [20; 35]$ – greatest number of neighbours.
- $\omega = 0.5$ – weight coefficient characterizing particle memory.
- $\alpha = 0.5$ и $\beta = 0.8$ – parameters used in calculating particle velocity.

5. Experimental study on the problem of a cylindrical spring construction.

The parameters found during the research were used to solve a real problem. The task was to design a cylindrical spring at a constant axial load (Figure 1). The goal is to minimize the total cost of the spring.

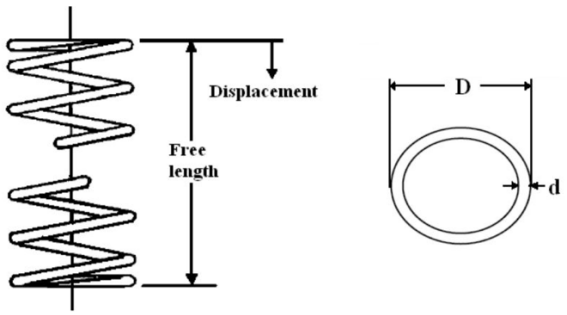

Figure 1. Cylindrical spring.

In the problem, it is necessary to calculate D (diameter of the ring), d (diameter of the wire) and N (the number of turns in the spring).

Let:

- x_1 – the number of turns in the spring (N), $5 \le x_1 \le 20$, x_1– integer.
- x_2 – diameter of the ring (D), $0.207 \le x_2 \le 0.5$.
- x_3 – diameter of the wire (d), $0.207 \le x_3 \le 0.5$.

Then the task is formulated as follows:

$$f(x) = \frac{\pi^2 x_2 x_3^2 (x_1+2)}{4}, \tag{7}$$

$$g_1(x) = \frac{8 c_f F_{max} x_2}{\pi x_3^3} - S, \tag{8}$$

$$g_2(x) = L_f - L_{max}, \tag{9}$$

$$g_3(x) = d_{min} - x_3, \tag{10}$$

$$g_4(x) = x_2 - D_{max}, \tag{11}$$

$$g_5(x) = 3 - C, \tag{12}$$

$$g_6(x) = \sigma_p - \sigma_{pm}, \tag{13}$$

$$g_7(x) = \sigma_p + \frac{F_{max} - F_p}{K} + 1.05(x_1 + 2) - L_f, \tag{14}$$

$$g_8(x) = \sigma_w + \frac{F_{max} - F_p}{K}, \quad (15)$$

$$g_i(x) \le 0, i = 1, \ldots, 8, \quad (16)$$

$$c_f = \frac{4C-1}{4C-4} + \frac{0.615x_3}{x_2}, \quad (17)$$

$$C = \frac{x_2}{x_3}, \quad (18)$$

$$K = \frac{Gx_3^4}{8x_1x_2^3} \quad (19)$$

$$\sigma_p = \frac{F_p}{K}, \quad (20)$$

$$L_f = \frac{F_{max}}{K} + 1.05(x_1 + 2)x_3. \quad (21)$$

Also:

- $F_{max} = 1\,000 -$ maximum workload.
- S = 189 000 – allowable maximum tension.
- $L_{max} = 14$ – maximum spring length.
- $d_{min} = 0.2$ – spring diameter.
- $D_{max} = 3.0$ – maximum outer spring diameter.
- $F_p = 300$ – pressing force.
- $\sigma_{pm} = 6$ – permissible maximum deviation under preload.
- $\sigma_w = 1.25$ – deviation from the preload position to the maximum load position.
- $G = 11.5 \cdot 10^7$ – material shear modulus.

The results of solving the problem, obtained using the investigated algorithms, are presented in Table 1. This table also shows the results of solving this problem, obtained by other authors using Branch-and-Bounds (BNB), AL, Genetic adaptive search (GA) and Hybrid genetics (HG).

Table 1. The results of problem solving.

Param	BNB	AL	GAS	HG	FSA	BCA	BTA	BSA	PSA
N	10	7	9	9	9	9	9	9	9
D	1.18	1.329	1.226	1.2253	1.231	1.223	1.201	1.223	1.202
d	0.283	0.283	0.283	0.283	0.283	0.283	0.283	0.283	0.283
g_1	–5.5001e+3	1.0169e+4	-713.51	-772.22	-73.5	-862.38	-5046.04	-1.36e-4	-675.577
g_2	–8.6523	–9.5436	-8.933	-8.9357	-8.90	-8.943	-8.632	-8.903	-8.932
g_3	–0.083	-0.083	-0.083	-0.083	-0.083	-0.083	-0.083	-0.083	-0.083
g_4	–1.8200	–1.6710	-1.491	-1.7747	-1.767	-1.776	-1.776	-1.767	-1.774
g_5	–1.1696	–1.700	-1.337	-1.3297	-1.356	-1.323	-1.185	-1.356	-1.333
g_6	–5.464	-5.464	-5.461	-5.4613	-5.452	-5.464	-5.459	-5.452	-5.46
g_7	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
g_8	0.026	0.026	–0.009	-0.007	-0.002	-0.004	–0.011	0.0	-0.01
$f(x)$	2.7995	2.365	2.665	2.6634	2.6740	2.6586	2.6113	2.676	2.6133

Sandgren used the branch and bound method in 1990, Deb and Goyal (1996) used combined genetic adaptive search, Rao and Shong used hybrid genetics in 2005. It should be noted that the decision of

Cannan and Kramer (1995), obtained using the Lagrange methods, is among the optimal ones listed. However, it violates g_1 constraint and cannot be considered a valid solution.

Among the algorithms studied, the bat method showed the best result compared to all the works listed - the method allowed to achieve the value of the objective function equal to 2.6116 and ensure that all the problem's constraints are satisfied.

6. Conclusion

In the framework of this article, a study was conducted of various metaheuric algorithms for the search for extrema, the selection of optimal parameters for each of them using various functions as an example. Later, the algorithms were compared by the example of a real problem, particularly the construction of a cylindrical spring.

The selection of the optimal parameters was carried out due to a complete search. Moreover, the task of selecting parameters requires the development of more optimal approaches to the selection of these parameters.

By adjusting the parameters of each algorithm, a high level of speed and accuracy of the search for extremes was achieved. As a result of the comparison, it was found that the best value was obtained using the bat algorithm. This algorithm showed high results when testing on various types of functions: from simple to complex (multiextreme).

References

[1] Anop M F, Katueva Ya V and Mikhalichuk V I 2015 Algorithms of swarm intelligence in the task of ensuring reliability by gradual refusals *Science and Education* **1** 144-157

[2] Yang X-S 2010 A New Metaheuristic Bat-Inspired Algorithm *Studies in Computational Intelligence* **284** 117-131

[3] Tynchenko V S, Tynchenko V V, Bukhtoyarov V V, Tynchenko S V and Petrovskyi E A 2016 The multi-objective optimization of complex objects neural network models *Indian Journal of Science and Technology* **9(29)** 99467

[4] Zhao O and Gao Z-M 2015 The bat algorithm and its parameters *Electronics, Communications and Networks IV* 1323-1326

[5] Klimenko A V, Zernov M M and Bobryakov A V 2015 The use of genetic algorithms for multiextremal search in fuzzy clustering problems *Bulletin of the Moscow power engineering institute* **1** 101-106

[6] Yang X-S 2010 Firefly algorithm, stochastic test functions and design optimization *International journal of bio-inspired computation* **2** 78-84

[7] Mirjalili S, Song Dong J, Lewis A and Sadiq A S 2020 Particle swarm optimization: Theory, literature review, and application in airfoil design *Studies in Computational Intelligence* **811** 167-184

[8] Tynchenko V S, Murygin A V, Petrenko V E, Emilova O A and Bocharov A N 2017 Optimizing the control process parameters for the induction soldering of aluminium alloy waveguide paths *IOP Conference Series: Materials Science and Engineering* **255(1)** 012017

[9] Hossain M A, Pota H R, Squartini S and Abdou A F Modified PSO algorithm for real-time energy management in grid-connected microgrids *Renewable Energy* **136** 746-757

[10] Zhou J, Yao X, Chan F T S, Lin Y, Jin H, Gao L and Wang X 2019 An individual dependent multi-colony artificial bee colony algorithm *Information Sciences* **485** 114-140

[11] Tynchenko V S, Petrovsky E A and Tynchenko V V 2016 The parallel genetic algorithm for construction of technological objects neural network models *2nd International Conference on Industrial Engineering, Applications and Manufacturing, ICIEAM* 7911573