Self-tuning Co-Operation of Biology-Inspired and Evolutionary Algorithms for Real-World Single Objective Constrained Optimization

Shakhnaz Akhmedova
Dept. of Higher Mathematics
Reshetnev Siberian State University of Science and Technology
Krasnoyarsk, Russian Fedeatiom
shahnaz@inbox.ru

Vladimir Stanovov
Dept. of Higher Mathematics
Reshetnev Siberian State University of Science and Technology
Krasnoyarsk, Russian Federation
vladimirstanovov@yandex.ru

***Abstract*—Solving single objective constrained real-parameter optimization problems via population-based algorithms has attracted much attention. In this paper, a new self-tuning meta-heuristic approach called Fuzzy Controlled Cooperative Heterogeneous Algorithm (FCHA), which was proposed for constrained optimization, is introduced. The developed approach combines competition and cooperation between biology-inspired and evolutionary algorithms, regulated by fuzzy controller. It should be noted, that the epsilon-constrained method is utilized to handle the constraints for the solved optimization problems. The performance of the proposed FCHA algorithm is evaluated on 57 real-world constrained problems submitted for CEC 2020 special session. Its workability and usefulness are demonstrated; also ways of algorithm improvement are discussed.**

Keywords—constrained optimization, biology-inspired algorithms, evolutionary algorithms, self-tuning, fuzzy controller

I. Introduction

Many problems in science and engineering can be formulated as constrained optimization problems [1], which can be linear or non-linear, multimodal or rotated. The single-objective constrained optimization problem can be described as follows:

$$f(x) \to opt, \tag{1}$$

$$\begin{cases} g_j(x) \le 0, j = \overline{1,r} \\ h_j(x) = 0, j = \overline{r+1,m} \end{cases}, \tag{2}$$

where $x = (x_1, \ldots, x_n)$, n is the number of space dimensions for a given problem, r is the number of inequality constraints and, finally, m is the total number of equality and inequality constraints, which define the feasible region for a problem in hand.

Evolutionary as well as biology-inspired algorithms have been successful for a wide range of constrained optimization problems, examples can be found in [2] or [3]. In the last years various modifications of algorithms such as Differential Evolution (DE) [4] or Particle Swarm Optimization (PSO) [5], for instance, have arisen as attractive optimization techniques due to their competitive results.

Significant effort in the area of evolutionary computation and swarm intelligence has been made to find a balance between minimization of the objective function and finding feasible solutions. Early works on constrained optimization utilized penalty factors for every constraint, resulting in a group of methods, such as static penalty [6], with fixed factors, dynamic penalty [7], with factors changing as the optimization process proceeds, death penalty [8], where the infeasible solutions are eliminated and adaptive penalty approaches, with factors depending on the search successfulness, as well as self-adaptive penalty approaches [9].

Another group of methods focuses on the superiority of feasible solutions, for example Deb's rule described in [10], stochastic ranking [11] and epsilon-constraint (EC) technique [12]. In this study a new modification of the EC method, similar to the one introduced in [13], is used.

Besides, mentioned constraint handling technique is applied to a new cooperative population-based algorithm with fuzzy controller as the self-tuning mechanism, called Fuzzy Controlled Cooperative Heterogeneous Algorithm (FCHA). Its basic idea is adopted from the meta-heuristic approach Co-Operation of Biology Related Algorithms or shortly COBRA [14].

Generally, the FCHA approach consists of parallel work of several evolutionary and biology-inspired algorithms, therefore, population-based algorithms, which compete and cooperate with each other. Their population sizes are automatically determined by the fuzzy controller and change during the optimization process.

Efficiency of the FCHA was examined on test problems taken from the CEC 2020 competition on real-world single objective constrained optimization [15]. Experimental results demonstrated the workability and usefulness of the proposed FCHA approach.

Thus, this paper has the following structure. In Section II, the developed algorithm is described. In Section III, the fuzzy controller is introduced. Experimental setup and numerical results are presented in Section IV. Finally, in Section V, the main conclusions and several future works are summarized.

II. Cooperative Algorithm for Constrained Optimization

As was already mentioned, the proposed FCHA approach is based on the idea introduced in [14], namely it is based on the cooperative work of five biology-inspired and evolutionary algorithms (in other words components). In this study the following components were used: SHADE algorithm [16] with

978-1-7281-6929-3/20/$31.00 ©2020 IEEE

two different mutation strategies (DE/rand/1 and DE/current-to-best/1) [17], Particle Swarm Optimization (PSO) [5], Cuckoo Search Algorithm (CSA) [18] and Bat Algorithm (BA) [19]. All listed algorithms were chosen due to their high efficiency. Moreover, mentioned evolutionary as well as biology-inspired algorithms have similar schemes, so it was easier to implement them together.

The originally proposed approach consists in generating one population for each component-algorithm, therefore five populations, which are then executed in parallel, cooperate with each other. The FCHA algorithm is a self-tuning meta-heuristic, so there is no need to choose the population size for each component-algorithm.

The number of individuals in the population of each algorithm can increase or decrease depending on the fitness values: if the overall fitness value was not improved during a given number of iterations, then the size of each population increased, and vice versa.

There is also one more rule for population size adjustment, whereby a population can "grow" by accepting individuals removed from other populations. This happens if the corresponding component-algorithm has the highest success rate on a given step. Therefore, in this study the "winner algorithm" is determined as an algorithm whose population has the best currently found fitness value. Then fuzzy controller makes decision about the population size changes for each component-algorithm.

It should be noted that the procedure of comparison of individuals is changed due to the necessity to take into account their feasibility. Thus, the individual x is considered to be better than the individual y (here it is denoted as $[x \succ$) in the following cases:

$$[x \succ \quad \begin{cases} f(x) < f(y), CV(x), CV(y) < \varepsilon \\ (x) < f(y), CV(x) = CV(y) \\ CV(x) < CV(y), \text{otherwise} \end{cases} . \qquad (3)$$

Here $f(x)$ and $f(y)$ are function values, $CV(x)$ and $CV(y)$ are two constraint violations, calculated according to the formula given in [15]. This means that the solutions with small violations (less than ε) are considered as feasible, and ordered according to their fitness function values. In the case of $\varepsilon = 0$, $CV(x)$ always precedes $f(x)$.

In this study ε was calculated on each iteration in the following way:

$$\theta = \theta_p \times NP, \qquad (4)$$

$$\varepsilon(t) = \begin{cases} CV_\theta \times \left(1 - \frac{NFE}{NFE_{\max}}\right)^{cp}, NFE < NFE_c \\ 0, \text{otherwise} \end{cases}, \qquad (5)$$

where t is the iteration number, NFE is the current number of function evaluations, NFE_{max} is the total available resource, NFE_c is the cut-off level set to $0.8 \times NFE_{max}$, θ_p is the control parameter from [0; 1] set to 0.8, θ is the index of an individual in an array sorted by constraint violation, thus, CV_θ is the constraint violation of the θ-th individual, NP is the population size of the component, cp is another control parameter, which is equal to 3.

Also populations interact with each other. The main goal of their communication is to prevent their preliminary convergence to their own local optimum. "Communication" was determined in the following way: populations exchange individuals in such a way that a part of the worst individuals of each population is replaced by the best individuals of other populations. Thus, the group performance of all algorithms can be improved.

III. Fuzzy Controller Design

The main idea of using a fuzzy controller is to implement a flexible tuning method to change the population sizes during the optimization process. Fuzzy controllers are well known for their ability to generate real-valued outputs using special fuzzification, inference and defuzzification schemes.

In this work success rates were used as inputs and population size changes as outputs for the fuzzy controller. To be more specific, the fuzzy controller had 6 input variables, including 5 success rates, one for each component, and an overall success rate, and 5 output variables, i.e. the number of solutions to be added to or removed from each component.

The success rate for all input variables except for the last one is evaluated as the best fitness value of its population. The last input variable was determined as the ratio of the number of iterations, during which the best found fitness value was improved, to the given number of iterations, which was a constant period. Thus, the process of population growth was automated by the fuzzy controller.

The Mamdani-type fuzzy inference was used to obtain the output values, and the rules had the following form:

$$R_q: \text{IF } u_1 \text{ is } A_{q1} \ \dots \ u_p \text{ is } A_{qn} \text{ THEN } v_1 \text{ is } B_{q1} \ \dots \ v_k \text{ is } B_{qk} \quad (6)$$

where R_q is the q-th fuzzy rule, $u = (u_1, \dots, u_p)$ is the set of controller's input values in p-dimensional space (in this study p is equal to 6), $v = (v_1, \dots, v_k)$ is the set of controller's outputs (k is set to 5), A_{qi} is the fuzzy set for the i-th input variable, B_{qj} is the fuzzy set for the j-th output variable.

The rule base contained 18 fuzzy rules, which had the following structure: each 3 rules described the case when one of the components gave better results than the others (as there were 5 components, 15 rules were established); the last 3 rules used the overall success of all components (variable 6) to add or remove solutions from all components, i.e. to regulate the computational resources. Example of the rule base is demonstrated in Table I.

TABLE I. Part of the Rule Base

№							
1	IF	u_1 is A_3	u_2-u_5 is A_4	u_6 is DC	THEN	v_1 is B_3	v_2-v_5 is B_1
2	IF	u_1 is A_2	u_2-u_5 is A_4	u_6 is DC	THEN	v_2 is B_3	v_2-v_5 is B_1
3	IF	u_1 is A_1	u_2-u_5 is A_4	u_6 is DC	THEN	v_3 is B_3	v_2-v_5 is B_1
…	…						

16	IF	u_1-u_5 is DC	u_6 is A_1	THEN	v_1 is B_1
17	IF	u_1-u_5 is DC	u_6 is A_2	THEN	v_1 is B_2
18	IF	u_1-u_5 is DC	u_6 is A_3	THEN	v_1 is B_3

The input variables were always in the range [0, 1], and fixed fuzzy terms of triangular shape were used for this case. In addition to the three classical fuzzy sets A_1, A_2 and A_3, the "Don't Care" (DC) condition and the A_4 term with the meaning "larger than 0" (opposite to A_1) were also used to decrease the number of rules and make them simpler.

FCHA Algoritm

begin
Fix minimal population size (min_size) of populations all together;
Fix maximal population size (max_size) of populations all together;
Randomly initialize of five populations P, D1, D2, C, B;
Initialize parameters of all component-algorithms;
Find best solution and its fitness for each component;
Find global best solution and its fitness value GBest;
while (*NFE* < NFE_{max})
Execute PSO for P;
Execute SHADE with rand/1 for D1;
Execute SHADE with current-to-best/1 for D2;
Execute CSA for population C;
Execute BA for population B;
Update best solution and its fitness for each component;
Update global best solution and its fitness value GBest;
if *(Gen = 7k, k=1,2,…)*
Estimate each component's success rate;
Estimate the last input variable for controller;
Generate controller's outputs;
Change population sizes and therefore populations;
Migration of the best individuals of all populations;
end if
if *(Gen = 10k, k=1,2,…)*
Migration of the best individuals of all populations;
end if
end while
Post-processing of the obtained results;
end

Fig. 1. Pseudo-code of the Fuzzy Controlled Cooperative Heterogeneous Algorithm (FCHA).

For the outputs, three fuzzy terms of triangular shape were used. The output fuzzy terms were symmetrical, and the positions and shapes were determined by two values, encoding the left and right position of the central term, as well as the middle position of the side terms in one value, and the left and right positions of the side term in another value. These two values were optimized using the PSO algorithm.

The defuzzification procedure was performed by calculating the centre of mass of the shape received by fuzzy inference. FCHA algorithm's pseudo-code is demonstrated in Fig. 1.

IV. Experimental Results

A. Experimental Setup

The FCHA algorithm's performance was evaluated on the CEC 2020 Competition on Real-World Single Objective Constrained Optimization [15]. The benchmark contains 57 real-world constrained optimization problems with various features such as large number of local optima, asymmetry, non-separability and so on. Also all problems have different dimensions, which vary from 2 to 158, and contain a wide variety of constraints.

As was mentioned, the FCHA approach is based on the cooperative work of five algorithms, which have their own parameters. Therefore, the initial values of the necessary parameters for all component-algorithms were taken from original papers dedicated to them and proposed by authors.

Parameters of the fuzzy controllers for the FCHA approach were found by PSO as described in [20], namely the following parameters were obtained: [–3; –2; 0; 10]. The last input variable for fuzzy controller was determined as the ratio of the number of iterations, during which the currently best found fitness value was improved, to the given number of iterations, which was set to 7, according to the previously conducted experiments. Competition and communication between components were conducted each time after 10 iterations.

To check the efficiency of the proposed algorithm, the maximum number of function evaluations varied depending on the problem in hand, namely the number of variables. To be more specific, the maximum number of function evaluations was set according to the CEC'2020 competition's rule. Also there were 25 program runs for each benchmark.

For the cooperative meta-heuristic FCHA the minimum population size for each component was set to 0, but if the total sum of population sizes was equal to 0, then all population sizes increased to 10. Additionally, the maximum total sum of population sizes was set to 300.

B. Numerical Results

Currently best found value of the objective function $f(x)$ and corresponding constraint violation $CV(x)$ were recorded for the achieved best solution x after $0.1\times NFE_{max}$, $0.2\times NFE_{max}$, $0.3\times NFE_{max}$, … , $0.9\times NFE_{max}$ and NFE_{max} function evaluations for each problem. To calculate the $CV(x)$ for solution x the formula from [15] was used.

Also, the feasibility rate (FR) and a vector c for each problem over 25 trials were calculated. Feasibility rate was defined as the ratio of the number of program runs, where at least one feasible solution was found under NFE_{max} function evaluations, to the maximum number of trials (25). The vector c is the vector of number of violated constraints at the median solution that have three elements indicate the number of violations (including inequality and equality constraints) by more than 1.0, in the range [0.01, 1.0] and less than 0.01 respectively.

The simulation results obtained for the different optimization problems are demonstrated in Tables II-X.

TABLE II. OUTCOMES FOR PROBLEMS RC01-RC07

		RC01	RC02	RC03	RC04	RC05	RC06	RC07
Best	*f*	-1.36	42E+2	-22E+3	-1.01	-39E+2	1.00	0.49
	v	0	0.72	0.39	0	0	78.87	72.24
Median	*f*	1.78	70E+2	-20E+3	-1.00	-20E+2	1.00	0.49
	v	20E+5	65E+4	486.19	0	93.09	82.87	102.76
Mean	*f*	60.02	11E+4	-18E+3	-0.80	-22E+2	1.00	0.59
	v	14E+3	57E+4	393.69	0.12	73.34	90.93	110.19
Worst	*f*	334.22	26E+5	-37E+2	-0.32	-736.48	1.00	1.03
	v	20E+5	10E+5	563.72	0.69	199.86	262.24	191.44
STD	*f*	93.75	50E+3	44E+2	0.26	967.65	0	0.21
	v	87E+3	32E+3	181.03	0.23	60.05	35.54	26.22
FR		12	0	0	72	36	0	0
c		5;2;1	6;0;3	3;1;10	0;0;5	1;0;5	2;10;20	10;9;19

TABLE III. OUTCOMES FOR PROBLEMS RC08-RC14

		RC08	RC09	RC10	RC11	RC12	RC13	RC14
Best	*f*	-1.00	0	0	-13	-0.07	0	0
	v	0	0	0	0	0	0	0
Median	*f*	-1	1.11	0.11	109.80	0	0	53E+3
	v	0	0	0	0.55	0	0	0
Mean	*f*	-0.99	1.17	0.30	100.22	0.24	10E+3	47E+3
	v	0	0	0	0.73	0	0	0.59
Worst	*f*	-0.83	2.63	0.90	179.87	3.15	27E+3	17E+3
	v	0	0	0	2.35	0	0	11.40
STD	*f*	0.03	0.96	0.30	41.69	0.83	13E+3	41E+3
	v	0	0	0	0.75	0	0	2.30
FR		100	100	100	16	100	100	100
c		0;0;2	0;0;2	0;0;3	0;5;3	0;0;9	0;0;3	0;0;10

TABLE IV. OUTCOMES FOR PROBLEMS RC15-RC21

		RC15	RC16	RC17	RC18	RC19	RC20	RC21
Best	*f*	0	-1E+13	0	0	0	-26.27	0
	v	0	0	0	0	0	0	0
Median	*f*	29E+2	-5.91	0	0	0	0	0
	v	0	0	0	0	0	0	0
Mean	*f*	20E+2	-5E+11	0.001	16.78	0.06	-2.97	0
	v	0	0.09	0	0	0	0	0
Worst	*f*	30E+2	3.15	0.01	92.16	1.24	0	0
	v	0	1.24	0	0	0	0	0
STD	*f*	14E+2	2E+12	0.002	28.78	0.25	7.99	0
	v	0	0.31	0	0	0	0	0
FR		100	96	100	100	100	100	100
c		0;0;11	0,0;15	0;0;4	0;0;4	0;0;5	0;0;3	0;0;8

TABLE V. OUTCOMES FOR PROBLEMS RC22-RC28

		RC22	RC23	RC24	RC25	RC26	RC27	RC28
Best	*f*	0	0	0	-59E+9	0	-40.68	71E+2
	v	0	0	0	0	10.29	0	0
Median	*f*	0	5.89	0	-32E+3	16.67	0	12E+3
	v	0	0	0	0	41.95	0	0
Mean	*f*	0.21	9.26	0.31	-26E+9	25.53	0.96	12E+3
	v	0	0.003	0	0.002	44.26	0	0
Worst	*f*	0.54	19.65	2.59	71E+2	63.38	53.81	15E+3
	v	0	0.01	0	0.04	91.62	0	0
STD	*f*	0.26	8.27	0.84	12+E9	19.42	15.72	24E+2
	v	0	0.005	0	0.007	24.21	0	0
FR		100	72	100	100	0	100	100
c		0;0;11	0;0;11	0;0;7	0;0;7	5;1;80	0;0;3	0;0;9

TABLE VI. OUTCOMES FOR PROBLEMS RC29-RC35

		RC29	RC30	RC31	RC32	RC33	RC34	RC35
Best	*f*	0	0	0	-32E+3	0	0.17	-738.79
	v	0	0	0	0	0	80.64	31E+2
Median	*f*	13E+5	0.0004	0	-31E+3	0	6.57	-425.16
	v	0	0	0	0	0	134.65	46E+2
Mean	*f*	12E+5	0.13	0	-31E+3	0	14.84	-334.04
	v	0	0	0	0	0	131.44	49E+2
Worst	*f*	30E+5	2.71	0	-31E+3	0	50.75	159.55
	v	0	0	0	0	0	163.53	93E+2
STD	*f*	12E+5	0.53	0	435.99	0	15.48	287.10
	v	0	0	0	0	0	18.67	13E+2
FR		100	100	100	100	100	0	0
c		0;0;1	0;0;8	0;0;2	0;0;6	0;0;30	30;77;1	72;75;1

TABLE VII. OUTCOMES FOR PROBLEMS RC36-RC42

		RC36	RC37	RC38	RC39	RC40	RC41	RC42
Best	*f*	-359.53	-48.79	-31.91	-48.27	19.55	0.07	-25E+2
	v	26E+2	116.16	127.75	89.91	214.21	643.58	52E+2
Median	*f*	-217.37	-28.80	-30.14	-46.37	636.41	1.09	-20E+2
	v	50E+2	187.58	176.93	196.12	645.91	21E+2	64E+2
Mean	*f*	-171.14	-29.49	-26.27	-40.26	714.65	23E+2	-20E+2
	v	50E+2	199.19	180.46	185.73	664.88	17E+2	68E+2
Worst	*f*	221.30	-8.48	-11.62	0.23	16E+2	57E+3	-15E+2
	v	82E+2	322.03	267.50	258.27	10E+2	22E+2	99E+2
STD	*f*	166.18	13.38	6.73	13.11	396.10	11E+3	329.45
	v	15E+2	50.31	36.43	38.94	178.23	678.87	11E+2
FR		0	0	0	0	0	0	0
c		74;71;3	35;79;2	39;74;3	51;34;31	63;13;0	15;58;1	73;3;0

TABLE VIII. OUTCOMES FOR PROBLEMS RC43-RC49

		RC43	RC44	RC45	RC46	RC47	RC48	RC49
Best	*f*	-18E+2	-62E+2	0.19	0.12	5.21E-16	0.11	0.04
	v	56E+2	0	0.00	1.61E-5	0.00	0.00	0.67

		RC43	RC44	RC45	RC46	RC47	RC48	RC49
Median	*f*	-16E+2	-60.E+2	0.65	0.31	0.28	0.34	0.2
	v	73E+2	0	38.28	29.03	1.92	80.53	69.43
Mean	*f*	-14E+2	-60E+2	0.78	0.38	0.30	0.35	0.23
	v	71E+2	0	39.04	26.90	25.28	65.12	67.46
Worst	*f*	-670.61	-58E+2	3.64	0.98	0.84	0.89	0.66
	v	80E+2	0	90.00	83.88	90.00	97.16	130.43
STD	*f*	299.06	110.06	0.65	0.22	0.17	0.19	0.12
	v	703.81	0	26.00	23.69	29.08	34.64	28.05
FR		0	100	16	20	16	8	0
c		76;0;0	0;0;91	7;1;17	6;6;13	1;0;24	13;2;15	15;4;11

TABLE IX. OUTCOMES FOR PROBLEMS RC50-RC53

		RC50	RC51	RC52	RC53
Best	*f*	0.05	24E+2	31E+2	25E+2
	v	0.08	0.23	0.01	0.04
Median	*f*	0.21	34E+2	44E+2	48E+2
	v	65.05	1.81	0.12	0.60
Mean	*f*	0.21	35E+2	47E+2	46E+2
	v	66.07	1.69	0.40	1.06
Worst	*f*	0.48	52E+1	96E+2	58E+2
	v	91.56	3.47	5.68	2.36
STD	*f*	0.12	708.53	14E+2	878.70
	v	23.36	0.64	1.09	0.88
FR		0	0	0	0
c		6;6;18	1;3;11	0;2;13	0;3;12

TABLE X. OUTCOMES FOR PROBLEMS RC54-RC57

		RC54	RC55	RC56	RC57
Best	*f*	18E+2	646.53	18E+2	968.99
	v	0.18	0.03	0.11	0.03
Median	*f*	29E+2	15E+2	37E+2	18E+2
	v	1.31	0.09	3.64	0.06
Mean	*f*	33E+2	23E+2	46E+2	24E+2
	v	1.10	0.45	3.99	0.69
Worst	*f*	58E+2	55E+2	12E+3	88E+2
	v	1.94	3.75	9.73	2.51
STD	*f*	12E+2	15E+2	24E+2	19E+2
	v	0.64	0.85	2.44	0.97
FR		0	0	0	0
c		1;1;13	0;2;4	1;4;1	0;2;4

The obtained results were compared with the ones presented in [15], to be more specific, in [15] results of three algorithms (IUDE [21], εMAgES [22] and iLSHADE$_\varepsilon$ [23]) are given. Comparison was conducted in the following way:

- firstly, the feasibility rates *FR* were compared, namely algorithm with higher value of *FR* was considered as the better one in comparison;
- if algorithms had the same value of *FR* for a given problem, then the Student's *t*-test with significance level $p = 0.05$ was applied to mean values of the objective function (this test was used due to the fact that only mean and standard deviation values are known for mentioned algorithms).

Comparison results are demonstrated in the Table XI. In this table "better" and "worse" mean that the proposed approach won and lost respectively compared to a given algorithm, "equal" means that there was no significant difference between results. Thus, each cell in the Table XI contains the number of times results, obtained by the proposed approach FCHA, were better, worse or statistically the same.

TABLE XI. COMPARISON BETWEEN FCHA AND OTHER METHODS

	IUDE	**εMAgES**	**iLSHADE$_\varepsilon$**
Better	34	28	35
Equal	4	5	2
Worse	19	24	20

Finally, the FCHA algorithm's complexity was estimated. The algorithmic complexity is calculated using the benchmark suite proposed in [15]. Besides, the procedures described in [15] were adopted to calculate the algorithmic complexity, namely the following values were calculated:

- the average time required to evaluate all functions for 100000 times (T_1);
- the average computation time required by algorithm for 100000 function evaluations for each problem (T_2);
- the algorithmic complexity $T_3 = (T_2 - T_1) / T_1$.

TABLE XII. ALGPRITHM'S COMPLEXITY

T_1	T_2	T_3
74.99	107.97	0.44

Thus, it was established that the proposed approach FCHA is capable to solve complex constrained optimization problems and outperforms alternative algorithms in most cases, and, therefore, it can be used instead of them.

V. CONCLUSIONS

In this study a new self-tuning approach called Fuzzy Controlled Cooperative Heterogeneous Algorithm or FCHA for solving constrained optimization problems was proposed. Its basic idea consists in the cooperative work of five evolutionary and bionic algorithms, whose population sizes are defined by the fuzzy controller.

The FCHA algorithm's performance was evaluated on the 57 constrained optimization problems submitted for the CEC 2020 Competition on Real-World Single Objective Constrained Optimization. Obtained results were compared with the ones provided by the competition organizers. Also algorithm's computational complexity was estimated.

Experimental results confirmed workability and usefulness of the proposed FCHA approach: it outperformed other algorithms on most of the problems according to the statistical test. However, still there were cases when feasible solutions weren't found. Thus, its ability to find feasible solutions can be improved. Moreover, the developed FCHA algorithm can be

modified for solving multi-objective constrained optimization problems.

ACKNOWLEDGMENT

This work was supported by the internal grant of Reshetnev Siberian State University of science and technology for the support of young researchers.

REFERENCES

[1] F. Rossi, P. van Beek, T. Walsh, "Chapter 1 – Introduction," Foundations of Artificial Intelligence, Handbook of Constraint Programming, Elsevier, vol. 2, pp. 3–12, 2006.

[2] Z. Michalewicz, N. Attia, "Evolutionary optimization of constrained problems," Proc. of the 3rd Annual Conference on Evolutionary Programming, pp. 98–108, 1994.

[3] C. Coello Coello, "Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art," Computer Methods in Applied Mechanics and Engineering, vol. 191 (11-12), pp. 1245-1287, 2002.

[4] R. Storn, K. Price, "Differential Evolution – a Simple and Efficient Heuristic for Global Optimization over Continuous Spaces", Journal of Global Optimization, vol. 11(4), pp. 341–359, 1997.

[5] J. Kennedy, R. Eberhart, "Particle Swarm Optimization," Poceedings of IEEE International Conference on Neyral networks, pp. 1942–1948 1995.

[6] A. Homaifar, C. X. Qi, S. H. Lai, "Constrained optimization via genetic algorithms," SIMULATION, vol. 62(4), pp. 242–253 , 1994.

[7] J. A. Joines, C. R. Houck, "On the use of nonstationary penalty functions to solve nonlinear constrained optimization problems with GA's," Proc. of the first IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence, vol. 2, pp. 579–584, 1994.

[8] T. Back, F. Homeister, H.-P. Schwefel, "A survey of evolution strategies," Proc. of the Fourth International Conference on Genetic Algorithms, pp. 2–9, 1991.

[9] R. Farmani, J. A. Wright, "Self-adaptivetness formulation for constrained optimization," IEEE Trans. Evolutionary Computation, vol. 7, pp. 445–455, 2003.

[10] K. Deb, "An efficient constraint handling method for genetic algorithms," Computer Methods in Applied Mechanics and Engineering, vol. 186(2), 311–338, 2000.

[11] T. P. Runarsson, X. Yao, "Stochastic ranking for constrained evolutionary optimization," IEEE Transactions on Evolutionary Computation, vol. 4(3), pp. 284–294, 2000.

[12] T. Takahama, S. Sakai, "Constrained optimization by the ε constrained differential evolution with gradient-based mutation and feasible elites," IEEE International Conference on Evolutionary Computation, pp. 1–8, 2006.

[13] V. Stanovov, Sh. Akhmedova, E. Semenkin, "Selective pressure in constrained differential evolution," Proc. of the 2019 Genetic and Evolutionary Computation Conference Companion, pp. 83–84, 2019.

[14] Sh. Akhmedova, E. Semenkin, "Co-Operation of Biology Related Algorithms", Proc. of the IEEE Congress on Evolutionary Computation (CEC 2013), June 20-23, 2013, pp. 2207–2214.

[15] A. Kumar, G. Wu, M. Z. Ali, R. Mallipeddi, P. N. Suganthan, S. Das, "A test-suite of non-convex constrained optimization problems from the real-world and some baseline results," Swarm and Evolutionary Computation, 100693, 2020.

[16] R. Tanabe, A. Fukunaga, "Success-History Based Parameter Adaptation for Differential Evolution," Proc. of the IEEE Congress on Evolutionary Computation, pp. 71–78, 2013.

[17] S. Das, S. S. Mullick, P. N. Suganthan, "Recent Advances in Differential Evolution – an Updated Survey," Swarm and Evolutionary Computation, vol. 27, pp. 1–30, 2016.

[18] X. S. Yang and S. Deb, "Cuckoo Search via Levy flights," Proc. of the World Congress on Nature & Biologically Inspired Computing (NaBic 2009), IEEE Publications, USA, pp. 210–214, 2009.

[19] X. S. Yang, "A new metaheuristic bat-inspired algorithm," Proc. of the Nature Inspired Cooperative Strategies for Optimization (NICSO 2010), Springer, SCI 284, pp. 65–74, 2010.

[20] Sh. Akhmedova, E. Semenkin, V. Stanovov, S. Vishnevskaya, "Fuzzy logic controller design for tuning the cooperation of biology-inspired algorithms," Tan, Y., Takagi, H., Shi, Y., Niu, B. (eds.) ICSI 2017, LNCS, vol. 10386, Springer, 2017, pp. 269–276.

[21] A. Trivedi, K. Sanyal, P. Verma, D. Srinivasan, "A unified differential evolution algorithm for constrained optimization problems," Proc. of the 2017 IEEE Congress on Evolutionary Computation (CEC), pp. 1231–1238, 2017.

[22] L. Huang, W. Zhao, B. R. Abidi, M. A. Abidi, "A Constrained Optimization Approach for Image Gradient Enhancement," IEEE Transactions on Circuits and Systems for Video Technology, vol. 28, no. 8, pp. 1707–1718, 2018.

[23] J. Brest, M. S. Maučec, B. Bošković, "iL-SHADE: Improved L-SHADE algorithm for single objective real-parameter optimization," Proc. of the IEEE Congress on Evolutionary Computation (CEC), pp. 1188–1195, 2016.