An Improved Backtracking Search Algorithm for Constrained Optimization Problems

Wenting Zhao[1], Lijin Wang[1,2], Yilong Yin[1,*], Bingqing Wang[1], Yi Wei[1], and Yushan Yin[1]

[1] School of Computer Science and Technology, Shandong University, Jinan, 250101, P.R. China
wenting.wentingzhaoid@gmail.com, yilong.ylyin@sdu.edu.cn, bingqing.wangbqing@qq.com, {yi.weiyi1991,yushan.yinyushande2012}@163.com

[2] College of Computer and Information Science, Fujian Agriculture and Forestry University, Fuzhou, 350002, P.R. China
lijin.lijinwang@fafu.edu.cn

Abstract. Backtracking search algorithm is a novel population-based stochastic technique. This paper proposes an improved backtracking search algorithm for constrained optimization problems. The proposed algorithm is combined with differential evolution algorithm and the breeder genetic algorithm mutation operator. The differential evolution algorithm is used to accelerate convergence at later iteration process, and the breeder genetic algorithm mutation operator is employed for the algorithm to improve the population diversity. Using the superiority of feasible point scheme and the parameter free penalty scheme to handle constrains, the improved algorithm is tested on 13 well-known benchmark problems. The results show the improved backtracking search algorithm is effective and competitive for constrained optimization problems.

Keywords: constrained optimization, backtracking search algorithm, differential evolution algorithm, breeder GA mutation operator, mutation.

1 Introduction

Decision science and the analysis of physical system attach great importance to optimization techniques. Optimization problems can be mathematically formulated as the minimization or maximization of objective functions subject to constraints on their variables. Recently, nature-inspired meta-heuristic algorithms designed for solving various global optimization problems have been changing dramatically, e.g. genetic algorithm (GA) [1], differential evolution algorithm (DE) [2], ant colony optimization algorithm (ACO) [3], particle swarm optimization algorithm (PSO) [4,5], artificial bee colony algorithm (ABC) [6], social emotion optimization algorithm (SEOA) [7,8,9,10,11], bat algorithm (BA) [12], firefly algorithm (FA) [13], harmony search

* Corresponding author.

R. Buchmann et al. (Eds.): KSEM 2014, LNAI 8793, pp. 222–233, 2014.

algorithm (HS) [14], biogeography-based optimization algorithm (BBO) [15], group search optimizer (GSO) [16], and backtracking search optimization algorithm (BSA) [17].

BSA, a new nature-inspired algorithm proposed by Civicoglu, is effective, fast and capable of solving different numerical optimization problems with a simple structure. It has been proved that BSA can solve the benchmark problems more successfully than the comparison algorithms e.g. PSO, CMAES, ABC and JDE [17]. To our knowledge, no one has so far attempted making research on the BSA algorithm for constrained optimization problems. In light of this, we propose an improved BSA algorithm for constrained optimization problems, called IBSA. IBSA divides the evolutionary process into two phases. In the first phase, the proposed algorithm employs the mutation and crossover operators used in the standard BSA to take advantage of information gained from previous population. In the second phase, the mutation and crossover operators employed in the standard differential evolution algorithm is used to accelerate convergence and guide algorithm to find the optimal solution. In addition, the breeder genetic algorithm mutation is utilized to improve the population diversity with a small probability in the later phase.

The remainder of this paper is organized as follows. Section 2 describes general formulation of constrained optimization problem and constraint handling method. Section 3 introduces improved backtracking search algorithm. Results are presented in Section 4 and the concluding remarks are made in Section 5.

2 Constraint Problem and Constraint Handling Method

2.1 Constraint Problem

In the field of decision science and the analysis of physical system, there are a bundle of constrained optimization problems. Generally speaking, a constrained optimization problem can be described as follows (without loss of generality minimization is considered here).

$$f_{\min} = \min\{ f(x) \mid x \in \Omega \} \tag{1}$$

Feasible region:

$$\Omega = \{ x \in \mathbb{R}^n \mid g_i(x) \le 0, h_j(x) = 0, l_m \le x_m \le u_m, \; for \; i = 1,...p \;\; j = 1,...,q \;\; \forall m \} \tag{2}$$

In the above equations, $\vec{x} = (x_1, x_2, ..., x_D) \in \Omega \subseteq S$ is a D-dimensional vector. Each variable x_m subjects to Lower bound l_m and upper bound u_m. $f(x)$ is the objective function, $g_i(x)$ is the i-th inequality constraint, $h_j(x)$ is the j-th equality constraint. We divide constraints into four categories broadly, linear inequality constraints, nonlinear inequality constraints, linear equality constraints and nonlinear equality constraints. Most constraint handling techniques tend to deal with inequality constraints only. Consequently, we transform equality constraints into inequality constraints of the form $|h_j(x)-\delta| \le 0$, where δ is the constraint violation tolerance (a small positive value close to zero).

2.2 Constraint Handling Method

There are lots of constrained handling methods used in constrained optimization problems, but the penalty function has been used most widely. The basic penalty function can be formulated as follows:

$$\hat{f}(x) = f(x) + R \times G(x) \tag{3}$$

$$G(x) = R\sum_{i=1}^{s} \max[0, g_j(x)]^q \tag{4}$$

where R is the penalty parameter, and $\hat{f}$ is called an exact penalty function.

The superiority of feasible points (SFP) scheme is based on the static penalty method but includes an additional term in formulation (1). The purpose of this additional function is to ensure that infeasible points always have worst fitness values than feasible points. Eq.(1) can be rewritten as follows, where T_k is the population composed of trial individuals v_i at the k-th iteration.

$$\hat{f}(v_i^k) = f(v_i^k) + R \times G(v_i^k) + \Theta_k(v_i^k),\ v_i^k \in T^k \tag{5}$$

$$\Theta_k(v_i^k) = \begin{cases} 0 & if\ T^k \cap \Omega = \Phi\ \ or\ \ v_i^k \in \Omega \\ \alpha & if\ T^k \cap \Omega \neq \Phi\ \ and\ \ v_i^k \notin \Omega \end{cases} \tag{6}$$

The value α is calculated by:

$$\alpha = \max[0, \max_{v \in T^k \cap \Omega} f(v) - \min \min_{z \in T^k \setminus (T^k \cap \Omega)} [f(z) + R \times G(z)]] \tag{7}$$

The method of parameter free penalty (PFP) scheme is a modification of the SFP Scheme. The most significant feature is the lack of a penalty coefficient R. The fitness function in the PFP scheme is as follows:

$$\hat{f}(v_i^k) = f(v_i^k) + G(v_i^k) + \Theta_k(v_i^k),\ v_i^k \in T^k \tag{8}$$

$$\Theta_k(v_i^k) = \begin{cases} 0 & if\ v_i^k \in \Omega \\ -f(v_i^k) & if\ T^k \cap \Omega = \Phi \\ -f(v_i^k) + \max_{y \in T^k \cap \Omega} f(v_i^k) & if\ T^k \cap \Omega \neq \Phi\ \ and\ \ v_i^k \notin \Omega \end{cases} \tag{9}$$

3 Improved Backtracking Search Algorithm

3.1 BSA

BSA is a population-based iterative evolutionary algorithm designed to be a global minimizer. BSA maintains a population of N individual and D-dimensional members

for solving bound constrained global optimization. Moreover, BSA possesses a memory in which it stores a population from a randomly chosen previous generation for use in generating the search-direction matrix [17]. To implement BSA, the following processes need to be performed.

BSA initials current population and history population according to Eq.(10) and (11) respectively where U is the uniform distribution.

$$P_{i,j} \sim U(l_j, u_j) \tag{10}$$

$$oldP_{i,j} \sim U(l_j, u_j) \tag{11}$$

At the start of each iteration, an *oldP* redefining mechanism is introduced in BSA through the rule defined by Eq.(12) and (13), where a, $b \sim U(0,1)$ is satisfied.

$$oldP = \begin{cases} P, \ a < b \\ oldP, \ otherwise \end{cases} \tag{12}$$

$$oldP := permuting(oldP) \tag{13}$$

BSA has a random mutation strategy that uses only one direction individual for each target individual. BSA generates a trial population, taking advantage of its experiences from previous generations. F controls the amplitude of the search-direction matrix. The initial form of the trial individual u_i is created by Eq.(14).

$$u_i = P_i + F \times (oldP_i - P_i) \tag{14}$$

Trial individuals with better fitness values for the optimization problem are used to evolve the target population individuals. BSA generates a binary integer-valued matrix called map guiding crossover directions. Eq.(15) shows BSA's crossover strategy.

$$V_{i,j} = \begin{cases} P_{i,j}, \ map_{i,j} = 1 \\ u_{i,j}, \ otherwise \end{cases} \tag{15}$$

At this step, a set of v_i which has better fitness values than the corresponding x_i are utilized to renew the current population as next generation population according to a greedy selection mechanism as shown in Eq.(16).

$$x_i^{next} = \begin{cases} v_i \ \ if \ f(v_i) \le f(p_i), \\ x_i \ \ otherwise. \end{cases} \tag{16}$$

3.2 Differential Evolution

Differential evolution (DE) is proposed by Storn and Price in 1995. So far, more than six mutation strategies have been proposed [18, 19] owing to its simple yet efficient properties. Compared with original DE mutation, "Rand-to-best" [18] mutation is able

to improve population convergence, guiding evolution towards better directions. "Rand-to-best" mutation strategy is introduced as follows:

$$\overrightarrow{u_i} = \vec{x}_{best} + F \times (\vec{x}_{r_1} - \vec{x}_{r_2}) \tag{17}$$

Where r_1, r_2 are integers randomly selected from 1 to N, and satisfy $r_1 \neq r_2$. The scaling factor F is a real number randomly selected between 0 and 1. $\vec{x}_{best}$ is the best individual in the current population, and $\overrightarrow{u_i}$ is the mutant vector.

Subsequently, the crossover operation is implemented to generate a trial vector v_i shown by Eq. (18). Where I_i is an integer selected randomly from 1 to D, r_j is selected randomly from 1 to 0 and j denotes the j-th dimension. The index k is the number of iteration, and C_r is the crossover control parameter.

$$v_{i,j}^k = \begin{cases} u_{i,j}^k, \ r_j \le c_r \ or \ j = I_i \\ x_{i,j}^k, otherwise \end{cases} \tag{18}$$

3.3 Breeder Genetic Algorithm Mutation Operator

$$v_{i,j} = \begin{cases} x_{i,j} \pm rang_i \times \sum_{s=0}^{15} \alpha_s 2^{-s} & rand < 1/D \\ x_{i,j} & otherwise \end{cases}, j = 1, ..., D \tag{19}$$

$$rang_i = (u(i) - l(i)) \times (1 - current\_gen / total\_gen)^6 \tag{20}$$

Improved version of breeder genetic algorithm mutation operator proposed in [20, 21] intends to produce a highly explorative behavior in the early stage and ensures the global convergence in the later stage. Where $U(0,1)$ is the uniform random real number generator between 0 and 1. The plus or minus sign is selected with a probability of 0.5, and $\alpha_s \in \{0,1\}$ is randomly generated with expression $P(\alpha_s = 1) = 1/16$. Current generation number is denoted as *current_gen*, and total generation number is denoted as *total_gen*. Individuals in the interval $[x_i - rang_i, x_i + rang_i]$ are generated after IBGA mutation.

3.4 IBSA

BSA has a powerful exploration capability but a relatively slow convergence speed, since the algorithm uses historical experiences to guide the evolution. Focusing on excellent convergent performance of "Rand-to-best" mutation, it is combined with BSA. Meanwhile, IBGA is utilized to expand population diversity. Pseudo–code of IBSA can be present as follows:

Step 1: Initialize population size N, mutation probability *pm*, stage control parameter *rate*, crossover probability *Cr*, total number of iteration *IterMax* and penalty coefficient R if SFP is used.

Step 2: Initialize population P, and historical population $oldP$ using Eq.(10) and Eq.(11), respectively.

Step 3: Evaluate the population P using Eq.(5) or Eq.(8).

Step 4: k=0;

Step 5: Update the historical population $oldP$ using Eq.(12) and Eq.(13).

Step 6: if k<$IterMax$*$rate$ then perform mutation and crossover operators according to Eq.(14) and Eq.(15), and goto Step 8.

Step 7: if pm<=0.05 then perform mutation operator using Eq.(19), else perform mutation and crossover operators according to Eq.(17) and Eq.(18), where the factor F is generated from the range of [-1,-0.4] and [0.4,1] uniformly.

Step 8: Evaluate the population P using Eq.(5) or Eq.(8), and select the best individual X_{best}.

Step 9: k=k+1, if k<$IterMax$ goto Step 5.

Step 10: output X_{best}.

4 Experiments

We use a set of 13 benchmark problems [22] in this paper to evaluate the performance of BSA, which were tested widely in evolution computation domain to show the performance of different algorithms for constrained optimization problems. The objective functions can be divided into 6 classes: quadratic, nonlinear, polynomial, cubic, linear, and exponential. Main characteristics of these functions are summarized in Tab.1.

Table 1. Main properties of benchmark functions(n: number of variables, |F|/|S|: the ratio of the feasible region to the given box constrained area, LI, NE, NI: number of linear inequality, nonlinear equality, and nonlinear inequality, a: number of active constraints at optimum)

	known optimal	n	Min/Max type	$f(x)$ type	|F|/|S|	LI	NE	NI	a
G01	-15	13	Minimum	quadratic	0.011%	9	0	0	6
G02	0.803619	20	Maximum	nonlinear	99.90%	1	0	1	1
G03	1	10	Maximum	polynomial	0.002%	0	1	0	1
G04	-30665.539	5	Minimum	quadratic	52.123%	0	0	6	2
G05	5126.4981	4	Minimum	cubic	0.000%	2	3	0	3
G06	-6961.8139	2	Minimum	cubic	0.006%	0	0	2	2
G07	24.306291	10	Minimum	quadratic	0.000%	3	0	5	6
G08	0.095825	2	Maximum	nonlinear	0.856%	0	0	2	0
G09	680.630057	7	Minimum	polynomial	0.512%	0	0	4	2
G10	7049.25	8	Minimum	linear	0.001%	3	0	3	3
G11	0.75	2	Minimum	quadratic	0.000%	0	1	0	1
G12	1	3	Maximum	quadratic	4.779%	0	0	9^3	0
G13	0.0539498	5	Minimum	exponential	0.000%	0	3	0	3

Additionally, for each problem, 30 independent runs were performed. Other parameters are given in Tab.2.

Table 2. Parameter values

	N	$IterMax$	C_r	p_m	$rate$	R	DIM_RATE
values	80	10000	0.9	0.05	0.6	10^{50}	1

Table 3. Results for IBSA using SFP

	optimal	best	mean	worst	std
G01	-15	**-15**	-15	-15	0
G02	0.803619	0.803614	0.788434	0.761742	0.009713
G03	1	**1.012555**	1.011447	0.992238	0.003785
G04	-30665.539	**-30665.539**	-30665.539	-30665.539	0
G05	5126.4981	**5126.484154**	5126.484154	5126.484154	0
G06	-6961.8139	**-6961.8139**	-6961.8139	-6961.8139	0
G07	24.306291	**24.306209**	24.306214	24.306279	0.000015
G08	0.095825	**0.095825**	0.095825	0.095825	0
G09	680.630057	**680.630057**	680.630057	680.630057	0
G10	7049.25	**7049.248021**	7049.248039	7049.248158	0.000037
G11	0.75	**0.7499**	0.7499	0.7499	0
G12	1	**1**	1	1	0
G13	0.0539498	**0.053942**	0.053942	0.053942	0

Table 4. Results for IBSA using PFP

	optimal	best	mean	worst	std
G01	-15	**-15**	-15	-15	0
G02	0.803619	0.803615	0.789926	0.75071	0.013549
G03	1	**1.01256**	1.010525	0.972599	0.007409
G04	-30665.539	**-30665.539**	-30665.539	-30665.539	0
G05	5126.4981	**5126.484154**	5126.484154	5126.484154	0
G06	-6961.8139	**-6961.8139**	-6961.8139	-6961.8139	0
G07	24.306291	**24.306209**	24.306213	24.306235	0.000007
G08	0.095825	**0.095825**	0.095825	0.095825	0
G09	680.630057	**680.630057**	680.630057	680.630057	0
G10	7049.25	**7049.248021**	7049.248051	7049.248432	0.000081
G11	0.75	**0.7499**	0.7499	0.7499	0
G12	1	**1**	1	1	0
G13	0.0539498	**0.053942**	0.06679	0.439383	0.070372

4.1 Results of the Proposed Method (IBSA)

Table 3 and Table 4 list the experimental results of the IBSA algorithm using SFP and PFP respectively. They include the known optimal solution for each test problem and the obtained best, mean, worst and standard deviation values. The best obtained by IBSA, which are same to or better than the known optimal, are marked in boldface.

As seen from Table 3, the IBSA using SFP algorithm can find the global optimal solutions for 12 problems except G02 in terms of the best results. However, result for problem G02 is very close to the optimal. The standard deviations for G01, G04, G05, G06, G08, G09, G11, G12 and G13 are zero, so we consider IBSA-PFP algorithm is under stable condition. G07 has found better solution than the known optimal solution. G03, G05, G11 and G13 are obtained better solutions as well, as a result of δ settings.

Table 4 shows the IBSA using PFP algorithm can also find the global optimization for 12 problems except G02 which is very close to the optimal. Within the allowed equality constraints violation ranges, better solutions have been found than the known optimal solutions such as G3, G5, G10, G11 and G07.As for problem G13, there are 29 out of 30 times finding better solution than the optimal.

Table 5. Results for BSA using SFP

	best of IBSA	best	mean	worst	std
G01	-15	-15	-15	-15	0
G02	**0.803614**	0.80358	0.79666	0.785462	0.00506
G03	**1.012555**	1.012298	1.002612	0.954234	0.016466
G04	-30665.539	-30665.539	-30665.539	-30665.539	0
G05	**5126.484154**	5126.4967	5198.319116	5415.4138	77.85568
G06	-6961.8139	-6961.8139	-6961.8139	-6961.8139	0
G07	**24.306209**	24.306242	24.321343	24.343854	0.018588
G08	0.095825	0.095825	0.095825	0.095825	0
G09	680.630057	680.630057	680.630057	680.630057	0
G10	**7049.248021**	7049.2555	7049.617975	7052.634	0.633024
G11	0.7499	0.7499	0.749901	0.749918	0.000003
G12	1	1	1	1	0
G13	**0.053942**	0.055766	0.633725	0.999993	0.305339

4.2 Comparison with BSA Using Penalty Function

We compare results of IBSA using penalty functions from Table 3 and Table 4 with BSA using penalty functions from Table 5 and Table 6 respectively. It can be found that IBSA performs far superior to BSA when solving problem G2, G3, G05, G07, and G10.When it comes to problem G13, IBSA successfully finds global optimization 29 times in 30 runs. However, there is only once that BSA finds the optimal. Improved strategies on BSA enhance the stability of the algorithm.

Table 6. Results for BSA using PFP

	best of IBSA	best	mean	worst	std
G01	-15	-15	-15	-15	0
G02	**0.803615**	0.803599	0.798024	0.792409	0.005266
G03	**1.01256**	1.012417	0.993189	0.899105	0.031189
G04	-30665.539	-30665.539	-30665.539	-30665.539	0
G05	**5126.484154**	5126.496714	5215.3329	5477.3847	114.543
G06	-6961.8139	-6961.8139	-6961.8139	-6961.8139	0
G07	**24.306209**	24.306244	24.323931	24.343783	0.018832
G08	0.095825	0.095825	0.095825	0.095825	0
G09	680.630057	680.630057	680.630057	680.630057	0
G10	**7049.248021**	7049.257983	7049.394	7049.7464	0.11844
G11	0.7499	0.7499	0.7499	0.749901	0
G12	1	1	1	1	0
G13	**0.053942**	0.069471	1.01215	13.782101	2.428104

Table 7. Comparison results with other algorithms using penalty function

	optimal	IBSA	MGSO	SAPF	OPA
G01	-15	-15	-15.000	-15.000	-15.000
G02	0.803619	0.803615	0.803457	0.803202	**0.803619**
G03	1	**1.012555**	1.00039	1	0.747
G04	-30665.539	**-30665.539**	**-30665.539**	-60,665.401	-30,665.54
G05	5126.4981	**5126.484154**	5,126.50	5,126.91	5,126.50
G06	-6961.8139	**-6961.8139**	**–6,961.8139**	–6,961.046	–6,961.814
G07	24.306291	**24.306209**	24.917939	24.838	24.306
G08	0.095825	**0.095825**	**0.095825**	**0.095825**	**0.095825**
G09	680.630057	**680.630057**	680.646	680.773	680.63
G10	7049.25	**7049.248021**	7,052.07	7,069.98	7,049.25
G11	0.75	**0.7499**	0.7499	0.749	0.75
G12	1	1	1	1	1
G13	0.0539498	0.053942	0.058704	**0.053941**	0.447118

4.3 Comparison with Other Algorithms Using Penalty Function

Constraint handling method highly affects the performance of intelligent algorithm for constrained optimization. To test the performance of the IBSA, we compare it with other algorithms.

The three algorithms are based on penalty function, such as modified group search optimizer algorithm with penalty function (MGSO) [23], genetic algorithm (GA) with a self-adaptive penalty function (SAPF) [24], over-penalty approach (OPA) [25]. Comparison results are summarized in Table 7. The best optimum values are listed.

As seen from Table 7, the performance of IBSA is much better than MGSO, SAPF, OPA on problems G03, G05, G07, G09 and G10 according to the best results obtained from algorithms with penalty function. Problem G01, G08, G11, G12 can be also solved successfully by IBSA. For problems g04, and g06, MGSO, OPA, and IBSA show the similar performance in terms of the best optimum, and better than SAPF. In all, using penalty function for handling constraint, IBSA shows better performance than other three algorithms for most problems.

4.4 Comparison with Other Algorithms Using Different Constraint Handling Methods

We employs three competitive evolutionary algorithms to further compare IBSA with other algorithms with different constraint handling methods, which were stochastic ranking (SR) method [22], improved stochastic ranking (ISR) [25] evolution strategy, and the simple multimember evolutionary strategy (SMES) [26]. Comparison results are summarized in Table 8.

Table 8. Comparison results with other algorithms using different constraint handling methods

	optimal	IBSA	SR	ISRES	SMES
G01	-15	**-15.000**	**-15.000**	**-15.000**	**-15.000**
G02	0.803619	0.803615	0.803516	**0.803519**	0.803601
G03	1	**1.012555**	1	1.001	1.000
G04	-30665.539	**-30665.539**	**-30,665.539**	**–30,665.539**	**-30,665.54**
G05	5126.4981	**5126.484154**	5,126.50	5,126.50	5,126.60
G06	-6961.8139	**-6961.8139**	–6,961.814	–6,961.814	–6,961.814
G07	24.306291	**24.306209**	24.306	24.306	24.327
G08	0.095825	**0.095825**	**0.095825**	**0.095825**	**0.095825**
G09	680.630057	**680.630057**	680.63	680.63	680.632
G10	7049.25	**7049.248021**	7,049.25	7,049.25	7,051.90
G11	0.75	**0.7499**	0.75	0.75	0.75
G12	1	**1**	1	1	1
G13	0.0539498	**0.053942**	0.053957	0.053957	0.053986

As seen from Table 8, Problem G01, G03, G04, G06, G08, G11 and G12 can be found optima by these four algorithms. IBSA performs better when dealing with G03, G05 and G13. For problems G07, G09 and G10, IBSA, SR, and ISRES show the similar performance in terms of the best optimum, and better than SMES.

5 Conclusion

This paper proposes an improved version BSA algorithm, called IBSA, which is very effective for solving constrained optimization problems. The proposed algorithm employs BSA operations, variant DE operations, and IBGA mutation at different stages. IBSA has better performance than BSA using PFP and SPF, and other three algorithms using penalty function to handle constraints according to experimental results. Comparing with other algorithms with different constraint handling methods, the results show that IBSA is comparable with its competitors. As a consequence, the effect of constraint handling methods on the performance of IBSA algorithm can be investigated in future works. Another direction is to improve algorithm to enhance accuracy of finding global optimum on some problems including equality constraint, such as G03 and G13.

Acknowledgement. This work was supported by the NSFC Joint Fund with Guangdong of China under Key Project U1201258, the Shandong Natural Science Funds for Distinguished Young Scholar under Grant No.JQ201316, and the Natural Science Foundation of Fujian Province of China under Grant No.2013J01216.

References

1. Holland, J.H.: Adaptation in natural and artificial systems. University of Michigan Press, Ann Arbor (1975)
2. Storn, R., Price, K.V.: Differential evolution: a simple and efficient adaptive scheme for global optimization over continuous spaces, Technical Report TR-95-012, Berkeley, CA (1995)
3. Dorigo, M., Maniezzo, V., Colorni, A.: The ant system: optimization by a colony of cooperating agents. IEEE Trans. on Systems. Man, and Cybernetics 26, 29–41 (1996)
4. Kennedy, J., Eberhart, R.C.: Particle swarm optimization. In: Proceedings of the IEEE International Conference on Neural Networks, pp. 1942–1948. IEEE Press (1995)
5. Eberhart, R.C., Kennedy, J.: A new optimizer using particle swarm theory. In: Proc. of the 6th International Symposium on Micro Machine and Human Science, pp. 39–43. IEEE Press (1995)
6. Karaboga, D.: An idea based on honey bee swarm for numerical optimization. Technical Report TR-06, Erciyes University, Engineering Faculty, Computer Engineering Department (2005)
7. Cui, Z.H., Cai, X.J.: Using social cognitive optimization algorithm to solve nonlinear equations. In: Proc. 9th IEEE Int. Conf. on Cog. Inf., pp. 199–203 (2010)
8. Chen, Y.J., Cui, Z.H., Zeng, J.H.: Structural optimization of Lennard-Jones clusters by hybrid social cognitive optimization algorithm. In: Proc. of 9th IEEE Int. Conf. on Cog. Inf., pp. 204–208 (2010)
9. Cui, Z., Shi, Z., Zeng, J.: Using social emotional optimization algorithm to direct orbits of chaotic systems. In: Panigrahi, B.K., Das, S., Suganthan, P.N., Dash, S.S. (eds.) SEMCCO 2010. LNCS, vol. 6466, pp. 389–395. Springer, Heidelberg (2010)
10. Wei, Z.H., Cui, Z.H., Zeng, J.C.: Social cognitive optimization algorithm with reactive power optimization of power system. In: Proc. of 2010 Int. Conf. Computational Aspects of Social Networks, pp. 11–14 (2010)

11. Xu, Y., Cui, Z., Zeng, J.: Social emotional optimization algorithm for nonlinear constrained optimization problems. In: Panigrahi, B.K., Das, S., Suganthan, P.N., Dash, S.S. (eds.) SEMCCO 2010. LNCS, vol. 6466, pp. 583–590. Springer, Heidelberg (2010)
12. Yang, X.S.: A new metaheuristic bat-inspired algorithm. Springer, Berlin (2010)
13. Yang, X.S.: Nature-Inspried Metaheuristic Algorithms, 2nd edn. Luniver Press, Frome (2010)
14. Geem, Z.W., Kim, J.H., Loganathan, G.V.: A new heuristic optimization algorithm: Harmony search. Simulation 76, 60–68 (2001)
15. Simon, D.: Biogeography-Based Optimization. IEEE Transactions on Evolutionary Computation 12, 702–713 (2008)
16. He, S., Wu, Q.H., Saunders, J.R.: A novel group search optimizer inspired by animal behavioral ecology. In: Proceedings of IEEE Congress on Evolutionary Computation, pp. 16–21 (2006)
17. Civicioglu, P.: Backtracking search optimization algorithm for numerical optimization problems. Applied Mathematics and Computation 219, 8121–8144 (2013)
18. Mezura-Montes, E., Miranda-Varela, M., Gómez-Ramón, R.: Differential evolution in constrained numerical optimization: An empirical study. Information Sciences 180, 4223–4262 (2010)
19. Neri, F., Tirronen, V.: Recent advances in differential evolution: a survey and experimental analysis. Artificial Intelligence Review 33, 61–106 (2010)
20. Wang, Y., Cai, Z., Guo, G., Zhou, Y.: Multiobjective optimization and hybrid evolutionary algorithm to solve constrained optimization problems. IEEE Transaction on Systems 37, 560–575 (2007)
21. Jia, G., Wang, Y., Cai, Z., Jin, Y.: An improved (μ+λ)-constrained differential evolution for constrained optimization. Information Sciences 222, 302–322 (2013)
22. Runarsson, T.P., Yao, X.: Stochastic ranking for constrained evolutionary optimization. IEEE Transactions on Evolutionary Computation 4, 284–294 (2000)
23. Wang, L., Zhong, Y.: A modified group search optimiser for constrained optimisation problems. Int. J. Modelling, Identification and Control 18, 276–283 (2013)
24. Tessema, B., Yen, G.: A self-adaptive penalty function based algorithm for constrained optimization. In: Proceedings 2006 IEEE Congress on Evolutionary Computation, pp. 246–253 (2006)
25. Runarsson, T.P., Yao, X.: Search biases in constrained evolutionary optimization. IEEE Transactions on Systems, Man, and Cybernetics 35, 233–243 (2005)
26. Mezura-Montes, E., Coello Coello, C.A.: A simple multimembered evolution strategy to solve constrained optimization problems. IEEE Transactions on Evolutionary Computation 9, 1–17 (2005)