Colliding bodies algorithm with adaptive parameter adjustment strategy

Chun Yu¹, Hui Liu², Bing Liu², Yong Li² and Jian Fan³

¹Sergeant School, Armored Academy of Army, Changchun, China
²College of Comp. Sci. and Eng, CCUT, Changchun, China
³Jilin Branch Center of National Internet Emergency Center, Changchun, China

Corresponding author and e-mail: Chun Yu, 201701064315lch@ncist.edu.cn

Abstract. Colliding Bodies Optimization (CBO) is a new meta-heuristic algorithm that uses collisions between objects to move to a better position so that the solution tends to be an optimal solution. Aiming at the shortcomings of CBO algorithm's poor optimization accuracy and prone to evolutionary stagnation in the later stage of iterations, this paper proposes a new Colliding Bodies Optimization algorithm with adaptive adjustment strategy. Firstly, a set of initial solutions of colliding bodies are generated by the good-point set strategy to improve the stability of the algorithm solution. Secondly, we adjust the linear strategy of the control parameters to a nonlinear strategy and combine it with the fitness value to form an adaptive parameter adjustment strategy. Finally, in order to make the algorithm escape from the local optimal, a mutation operation is performed on the position of the object. In order to test the effectiveness of the proposed algorithm, experiments were conducted on 23 benchmark functions with it and the comparison algorithm. The experimental results show that the performance of the proposed algorithm is better than other experimental algorithms.

1. Introduction

Intelligent optimization algorithms usually refers to leveraging a certain method to search for the optimal solution in the problem space. For optimization problem solving, this kind of algorithms can be mainly divided into two types: algorithms using mathematical methods and heuristic algorithms using natural phenomena such as biology and evolution in solving optimization problems [1]. In some optimization problems, mathematical algorithms are difficult to be applied to solve practical problems and time-, computation- and memory-consuming, so they have encountered certain obstacles in the application of practical problems. Later, people discovered that heuristic algorithms have obvious effects on dealing with complex problems. At the same time, heuristic algorithms do not need to perform derivative calculations, nor do they need to use approximate processing in the optimization model, therefore, scientists gradually replace mathematical algorithms with heuristic algorithms to solve optimization problems in practical applications [2]. Common heuristic algorithms include Genetic Algorithm (GA) [3], Differential Evolution Algorithm (DE) [4], Particle Swarm Optimization (PSO) [5] and so on.

Collision Body Optimization (CBO) algorithm was proposed by Kaveh and Mahdavi in 2014 based on the principle of one-dimensional collision law between bodies. Each candidate solution is regarded as a body with the specific mass and speed. When two bodies collide, the speed and position of the
body will change, which makes the body move towards the better position in the search space [6]. CBO algorithm has many advantages, such as simple theoretical structure, independent of any internal parameters and so on. Therefore, it can be applied in many fields, such as truss structure, urban planning layout, function optimization and so on. For example, Kaveh et al. in [7] used CBO algorithm to optimize the weight of truss structures with continuous variables; Partha et al. in [8] applied CBO algorithm to identify different types of nonlinear models; Kaveh et al. in [9] proposed an optimal domain decomposition method by combining CBO algorithm with k-median method. Nevertheless, the CBO algorithm is still limited by its search accuracy and easy approach to local optimal solutions in the later stage of iterations.

In order to improve the above shortcomings, this paper proposes a new Colliding Bodies Optimization algorithm with adaptive adjustment strategy, named ACBO. The ACBO algorithm changes the way of generating the initial population and takes points uniformly in the solution area to enhance the probability of obtaining the feasible solution. Meanwhile, an adaptive parameter adjustment strategy is adopted for the collision coefficient $\varepsilon$ to improve the convergence speed of the algorithm. In order to make the algorithm robust and diversified in the later stage of iterations, our proposed algorithm adds a mutation strategy to perturb the position of the collision body. Finally, the algorithm has a good search ability while maintaining the richness of the solution, thereby enhancing the effectiveness of the algorithm.

2. Principle of CBO algorithm
The CBO algorithm is inspired by the collision theory. Each candidate solution can be regarded as a collision body (CB). The CB is divided into the stationary object and the moving object. The moving object moves towards the stationary object and collides with it, which makes the two objects move to a better position [10-11]. The CBO algorithm can be briefly summarized as follows:

The position of each CB is initialized by formula (1):

$$x_i^0 = x_{\text{min}} + \text{rand}(x_{\text{max}} - x_{\text{min}}), i = 1, 2, ..., n$$

where $x_i^0$ is the position of the i-th CB after initialization, $x_{\text{max}}$ and $x_{\text{min}}$ is the upper and lower boundary of the search area. When generating CB, the position should not exceed the boundary. rand is a random number in [0, 1], and n is the number of CB.

The mass of each collision body can be defined as:

$$m_k = \frac{1}{\sum_{i=1}^{n} \frac{1}{\text{fit}(i)}}, k = 1, 2, 3, ... n$$

where fit(k) is the function result of the k-th CB. According to the function value, CB is arranged in ascending order and divided into two groups equally. The group with better candidate solution is static group, whose speed is equal to 0, and the group with worse candidate solution is mobile group, whose speed is given by formula (4). As shown in Figure 1, the CB of the moving group and the corresponding CB of the stationary group constitute a collision pair.

![Figure 1. The collision pair of the stationary object and the moving object.](image-url)
\[ v_i = 0, i = 1, 2, \ldots, \frac{n}{2} \]
\[ v_i = x_{i^* - \frac{n}{2}} - x_{i^*}, i = \frac{n}{2} + 1, \ldots, n \]
\[ (4) \]
where \( x_{i^*} \) and \( v_i \) are respectively the position vector and velocity vector of the \( i \)-th CB in the group, and \( x_{i^* - \frac{n}{2}} \) is the position vector of the \( \left(\frac{n}{2}\right) \)-th CB. After collision, we can use formula (5) and formula (6) to calculate the velocity of each collision body.
\[ v_i = \frac{(m_i - \epsilon m_i^*) v_{i^*}}{m_i + m_i^*}, i = \frac{n}{2} + 1, \ldots, n \]
\[ (5) \]
where \( v_i \) and \( v_{i^*} \) are the velocities before and after the collision of the \( i \)-th moving object respectively, \( m_i \) represents the mass of the \( i \)-th object of the moving group and \( m_i^* \) is the mass of the corresponding CB of the stationary group which forms a collision pair with the \( i \)-th CB of the moving group.

After the collision, the velocity of the corresponding CB of the stationary group is:
\[ v_i' = \frac{(m_i^* + \epsilon m_i v_i^*) v_{i^*}}{m_i^* + m_i}, i = 1, \ldots, \frac{n}{2} \]
\[ (6) \]
where \( m_i^* \) is the mass of the \( i \)-th CB of the stationary group, \( m_i^* \) \( \frac{n}{2} \) and \( v_i^* \) \( \frac{n}{2} \) are the mass and velocity of the corresponding CB of the moving group in the \( i \)-th collision pair, respectively. The collision coefficient (\( \epsilon \)) in formula (5) and (6) is as follows:
\[ \epsilon = 1 - \frac{\text{iter}}{\text{iter}_{\text{max}}} \]
\[ (7) \]
where \( \text{iter} \) is the current execution times and \( \text{iter}_{\text{max}} \) is the maximum execution times.

According to formula (8), the new position of the moving object is obtained.
\[ x_i^{\text{new}} = x_{i^* - \frac{n}{2}} + \text{rand} \circ v_{i^*}, i = \frac{n}{2} + 1, \ldots, n \]
\[ (8) \]
where \( x_i^{\text{new}} \) and \( v_{i^*} \) is the new position and the velocity after collision, respectively, and \( x_{i^* - \frac{n}{2}} \) is the old position of the corresponding CB of the moving group which forms a collision pair with the \( i \)-th CB of the stationary group, then the new position of the \( i \)-th CB of the stationary group is obtained by the following formula.
\[ x_i^{\text{new}} = x_i + \text{rand} \circ v_i', i = 1, \ldots, \frac{n}{2} \]
\[ (9) \]
Where \( x_i^{\text{new}} \), \( x_i \), \( v_i' \) are the new position, the old position and the velocity after collision of the \( i \)-th CB of the stationary group, and rand is a number generated randomly on (-1,1).
3. Colliding bodies optimization algorithm with adaptive adjustment strategy, ACBO
ACBO algorithm mainly improves CBO algorithm in the following three aspects. Firstly, the object is initialized by the good point set method, which can use as few particles as possible to more comprehensively represent the solution in the search area. Secondly, the adaptive parameter adjustment strategy is used to improve the CBO algorithm to generate solutions. Finally, to avoid the convergence of the solution to the local optimal solution in the later stage of iterations, the stochastic differential mutation operation is introduced to increase the diversity of samples. Next, the principle and process of ACBO algorithm will be described comprehensively.

3.1. Initialization with good point set method
In [12], Gao et al. proposed that the selection of initial solution plays an important role in intelligent optimization algorithm to obtain the optimal solution. There is no way to ensure the stability of the initial solution of the algorithm by randomly generating the initial position of the object. When the initial position of the object is far away from the actual solution, it is difficult to find the optimal solution, which indicates that the random initialization is difficult to traverse all kinds of situations in the search space. Good point set is a kind of mathematical method, which is used to get points evenly, and can select potential solutions as the set of initial solutions. So far, many optimization algorithms such as GA, PSO and GSA have successfully applied good point set method, and the performance of these optimization algorithms has been improved. Therefore, this paper uses the good point set method to improve the CBO algorithm. The main steps and related formulas of the good point set method are discussed in [13].

3.2. The adaptive parameter adjustment strategy
In meta-heuristic algorithm, the balance between exploration and development is very important [14]. In CBO algorithm, the collision coefficient \( \varepsilon \) is used to control the rate of exploration and development, which decreases linearly with the increase of the number of cycles. In the earlier stage of iterations, the energy loss is small and the value of \( \varepsilon \) is large, so the separation speed after collision is relatively fast, here we should pay attention to the global search ability of the algorithm. In the later stage of iterations, the energy loss in the process of motion is large and the value of \( \varepsilon \) is small, so the separation speed after collision is slow, here we should pay more attention to the development of the algorithm.

In the optimization process, we should gradually reduce the exploration and increase the development at the same time. Therefore, this paper optimizes the calculation formula of parameter \( \varepsilon \), which is changed into a nonlinear decreasing strategy, and the adaptive function value is combined to make it closer to the problem to be solved. In [15], the optimization effect of adjusting control parameters by sinusoidal curve is better than that of the original algorithm, and its formula is shown in formula (10). In addition, the adaptive value is also an important parameter in the process of searching the optimal solution, so this paper further fuses the adaptive value and the nonlinear strategy to form an adaptive adjustment strategy, as shown in formula (11).

\[
\varepsilon = \varepsilon_{ini} - (\varepsilon_{ini} - \varepsilon_{fin}) \cdot \sin\left(\frac{\pi}{2} \cdot \frac{t}{t_{max}}\right)
\]  

(10)

where \( \varepsilon_{ini} \) and \( \varepsilon_{fin} \) are the start and end values of the collision coefficient, the values of them are 2 and 0 respectively, \( t \) is the current execution times, \( t_{max} \) is the maximum of the execution times.

\[
\varepsilon = \begin{cases} 
\varepsilon_{ini} - (\varepsilon_{ini} - \varepsilon_{fin}) \cdot \sin\left(\frac{\pi}{2} \cdot \frac{t}{t_{max}}\right), & f_i \leq f_{avg} \\
1, & f_i > f_{avg}
\end{cases}
\]

(11)

where \( f_i \) is the adaptive value of the current collision body and \( f_{avg} \) is the mean value of the adaptive values. When the function value of the \( i \)-th collision body is better than the average function value, the
value of $\varepsilon$ is smaller, which make the collision body closer to a better position. On the contrary, when the control parameter is large, it can search for a wider space.

3.3. The stochastic differential mutation strategy
In the later stage of iterations, the collision individuals are close to the local optimal solution, resulting in the loss of diversity of samples and the stagnation of algorithm evolution. To avoid premature phenomenon, a disturbance strategy is proposed in [16], which mutates the collision individuals to produce new individuals with diversity, so as to improve the robustness of the algorithm. The improved CB position is shown in formula (12).

$$X_i^{\text{new}} = r(X_a - X_i) - r(X' - X_i)$$  \hspace{2cm} (12)

where $X_a$ is the best CB so far, and $X'$ is the CB randomly selected from all objects, the role of $r$ is equivalent to rand in the previous paper, and its range is $[0,1]$.

3.4. Description of ACBO algorithm
CBO algorithm mainly makes the solution tend to the global optimal solution through the collision between objects, but the improper selection of the initial solution is easy to cause the algorithm to fall into the local optimal in the later stage of iterations. The good point set theory is used to initialize the position of the collision body, which can make the solution more evenly distributed in the search space, but also has a certain stability. The collision coefficient $\varepsilon$ of CBO algorithm is linear, which does not consider the real-time change of the position during the iteration, so that the solution accuracy is low. The adaptive parameter adjustment strategy is used to accelerate the convergence speed of the algorithm, and the stochastic differential mutation operation is carried out to improve the convergence quality of the algorithm.

To sum up, the detail of CBO algorithm with adaptive parameter adjustment strategy are as follows:

Algorithm 1. The proposed ACBO algorithm.

| Step | Description |
|------|-------------|
| Step 1 | Initialize parameters of ACBO algorithm, such as the number of population, the maximum of execution times. |
| Step 2 | Initialize the object set $P$ of CB using the method described in Section 2.1, where $P = \{X_1, X_2, \ldots, X_n\}$. |
| Step 3 | Compute the function results of the collision body and arrange them in ascending order, and divide them into two equal groups. The result group with smaller function is the stationary group, and the result group with larger function is the moving group. At the same time, calculate the mean value of function results, then calculate the mass of the collision body according to formula (2), and record the current optimal object $X_a$. |
| Step 4 | Compute the velocities of the stationary group and the moving group before collision by formula (3) and formula (4). |
| Step 5 | Compare the current value and the mean value of the objective function of the collision body, and then calculate the collision coefficient $\varepsilon$ according to formula (11). |
| Step 6 | After the collision, the velocity and position of the collision body change. Calculate the velocity after the collision by the value of the collision coefficient and formulas (5) and (6). Update the positions of the moving group and the stationary group. |
| Step 7 | Perform stochastic differential mutation operations on the CB to generate a new CB. |
| Step 8 | Repeat this process until meeting termination condition and finding the optimal solution. |
4. Experimental analysis

4.1. The setup of experiment
To test the performance of ACBO algorithm in solving function optimization problems, we compare the ACBO algorithm with CBO, PSO, BBO [17] and GSA [18]. The parameter values of each algorithm are specified in Table 1, where CBO algorithm and ACBO algorithm do not depend on any internal parameters. In this paper, we use 23 benchmark functions in [10] for simulation experiments. Among them, f1-f7 is unimodal function, f8-f13 is multimodal function, f14-f23 is fixed dimensional multimodal function. To ensure the fairness of the experiment, all the algorithms are carried out under the same conditions, where the number of CB n=30 and the maximum of execution times tmax=500.

Table 1. The parameter settings of algorithms.

| Algorithm | Parameter     | Value     |
|-----------|---------------|-----------|
| BBO       | keepRate      | 0.2       |
|           | alpha         | 0.9       |
|           | pMutation     | 0.1       |
| PSO       | V<sub>max</sub>, V<sub>min</sub> | 6, -6     |
|           | w<sub>max</sub>, w<sub>min</sub> | 0.9, 0.2  |
|           | c1, c2        | 1.496, 1.496 |
| GSA       | Elitistcheek  | 1         |
|           | kpower        | 1         |

4.2. Experimental results and analysis
In simulation experiments, we use Matlab 2017 and the running environment is based on Intel (R) Core (TM) i5-4200U CPU @ 1.60GHz-2.30GHz. Each algorithm runs 30 times separately, and the average precision AVE and standard deviation STD are recorded, whose calculation formulas are as follows:

\[
AVE = \frac{1}{\text{runs}} \sum_{i=1}^{\text{runs}} \text{value}_i
\]

\[
STD = \sqrt{\frac{1}{\text{runs} - 1} \sum_{i=1}^{\text{runs}} (\text{value}_i - \text{Ave})^2}
\]

where \( \text{value}_i \) is the i-th running result of each algorithm.

Table 2 shows the AVE and STD of the five algorithms executed 30 times separately. From the experimental results in Table 2, it can be concluded that ACBO algorithm performs better on 20 functions than the other four algorithms, BBO algorithm performs best on function f3, CBO algorithm performs best on function f7. On the function f14, the AVE of ACBO, CBO and PSO all converge to the theoretical optimal value, but the STD of ACBO algorithm is inferior to that of CBO and PSO algorithm. The AVE of the proposed ACBO algorithm converges to the theoretical optimal value in the functions f9, f16, f17, f18, f19, f20, f22 and f23, and the STD of the proposed ACBO algorithm converges to the theoretical optimal value in the functions f9 and f17, which shows that the ACBO algorithm has obvious optimization degree in the functions f8- f23. In general, ACBO algorithm is better than other algorithms on the accuracy of solving most functions.
To observe the performance of the proposed algorithm more conveniently, the experimental results of six test functions selected are shown in Figure 2. Different curves represent the experimental results of different algorithms. The abscissa is the number of iterations, and the ordinate is the optimal solution. In order to make the experimental effect of the algorithms more obvious, the ordinates of functions f1, f2, f9 and f10 are in logarithmic form. In Figure 2, PSO algorithm has the best convergence effect to function f2, BBO algorithm has the best convergence effect to function f10, and ACBO algorithm has the best convergence effect to the remaining four functions. The convergence speed of ACBO algorithm on six functions is faster than that of CBO algorithm. The above analyses can indicate that ACBO algorithm has better search accuracy and global convergence accuracy than the other four algorithms in most functions.

### Table 2. Comparison of experimental results between ACBO algorithm and other comparison algorithms on functions f₁₋f₁₀ (n=30).

| Func. | ACBO | CBO | BBO | PSO | GSA |
|-------|------|-----|-----|-----|-----|
| f₁    | 1.09e-20 | 1.46e-20 | 1.13e-08 | 5.74e-08 | 1.17e-05 | 8.20e-05 | 2.38e-04 | 7.89e-05 | 2.90e-16 | 1.48e-16 |
| f₂    | 8.15e-13 | 6.19e-13 | 7.69e-09 | 1.86e-08 | 1.64e-03 | 6.81e-04 | 9.45e-01 | 2.82e-01 | 3.77e-01 | 8.20e-01 |
| f₃    | 2.11e+01 | 1.46e+01 | 9.2751 | 1.68e+01 | 3.80e-01 | 3.96e-01 | 5.46e+01 | 8.96e+01 | 8.92e+02 | 3.67e+03 |
| f₄    | 2.67e-03 | 1.52e-03 | 6.15e-02 | 2.61e-02 | 4.01e-01 | 4.78e-02 | 4.4960 | 7.4881 | 7.8124 | 3.9762 |
| f₅    | 7.2190 | 4.5342 | 7.05e+01 | 7.03e+01 | 7.93e+01 | 7.55e+01 | 8.54e+01 | 2.35e+02 | 7.08e+01 | 1.07e+02 |
| f₆    | 9.23e+21 | 1.41e+20 | 5.91e-08 | 1.55e-07 | 1.73e-04 | 9.61e-05 | 1.63e-06 | 2.51e-05 | 1.39e-03 | 7.82e-03 |
| f₇    | 6.14e-03 | 2.11e-03 | 1.83e-03 | 3.55e-03 | 7.78e-03 | 2.58e-02 | 1.44e-02 | 2.08e-01 | 1.90e-01 | - |
| f₈    | 4.18e+03 | 3.00e+01 | 4.16e+03 | 2.48e+02 | 3.28e+03 | 2.78e+02 | 3.03e+02 | 2.36e+02 | 2.39e+03 | 5.95e+02 |
| f₉    | 0.00 | 9.2531 | 3.4183 | 1.17e+01 | 5.5820 | 1.61e+01 | 6.37e+01 | 3.53e+01 | 9.8401 | - |
| f₁₀   | 5.71e-11 | 3.91e-11 | 4.88e-01 | 5.97e-01 | 4.09e-03 | 1.89e-03 | 4.3283 | 7.12e-01 | 3.56e-02 | 1.03e-01 |
| f₁₁   | 7.06e-04 | 3.24e-03 | 3.80e-02 | 2.08e-02 | 5.63e-02 | 2.98e-02 | 8.69e-02 | 3.89e-02 | 2.86e-01 | 5.6698 |
| f₁₂   | 3.13e-22 | 3.14e-22 | 4.00e-01 | 9.65e-01 | 2.20e-06 | 2.64e-06 | 2.95e-02 | 5.89e-01 | 2.1311 | 1.9489 |
| f₁₃   | 1.76e-21 | 1.96e-21 | 7.28e-01 | 1.3425 | 5.59e-06 | 4.38e-06 | 3.66e-01 | 3.01e-01 | 8.2796 | 7.5315 |
| f₁₄   | 9.80e-02 | 4.2301 | 9.80e-02 | 3.0749 | 7.9398 | 5.6543 | 9.80e-02 | 4.1233 | 4.7406 | 3.7462 |
| f₁₅   | 7.91e-04 | 3.61e-04 | 3.05e-03 | 2.03e-03 | 5.08e-03 | 7.74e-03 | 4.84e+01 | 7.91e-03 | 9.12e-03 | 4.93e-03 |
| f₁₆   | -1.0316 | 6.34e-16 | -1.0316 | 6.71e-16 | -1.0316 | 1.53e-12 | -1.0316 | 7.52e-16 | -1.0316 | 4.51e-15 |
| f₁₇   | 3.98e-01 | 0.00 | 3.98e-01 | 0.00 | 3.98e-01 | 1.39e-09 | 3.98e-01 | 0.00 | 3.98e-01 | 0.00 |
| f₁₈   | 3.0000 | 1.19e-15 | 3.0000 | 1.69e-15 | 5.7000 | 8.2385 | 3.0000 | 1.73e-15 | 3.1589 | 8.56e-01 |
| f₁₉   | -3.8628 | 2.71e-17 | -3.8628 | 2.71e-16 | -3.8628 | 2.11e-15 | -3.8628 | 2.63e-15 | -3.2404 | 4.23e-01 |
| f₂₀   | -3.3220 | 2.17e-02 | -3.2220 | 3.33e-02 | -3.2943 | 5.11e-02 | -3.2784 | 5.83e-02 | -1.5849 | 5.30e-01 |
| f₂₁   | -10.0269 | 3.72e-01 | -9.2377 | 2.9420 | -6.0675 | 3.6827 | -7.7200 | 3.1300 | -5.0187 | 3.6955 |
| f₂₂   | -10.4029 | 1.56e-02 | -9.3802 | 2.2770 | -7.0456 | 3.6895 | -8.7750 | 2.7800 | -7.9605 | 2.8875 |
| f₂₃   | -10.5364 | 5.38e-01 | -9.6510 | 2.3635 | -6.6068 | 3.7825 | -7.8725 | 3.6300 | -9.2375 | 3.2873 |
To summarize, compared with the other four algorithms, our proposed ACBO algorithm has obvious optimization performance. In addition, ACBO algorithm has simple structure, high optimization accuracy, strong robustness, and is not easy to tend to local optimal solution. ACBO algorithm can also coordinate the improvement efficiency in different periods of the algorithm, so it can be applied to most of the function optimization problems.

5. Conclusions
In this paper, we propose an improved CBO algorithm with adaptive parameter adjustment strategy, which not only improves the accuracy of CBO algorithm, but also coordinates the exploration and development of CBO algorithm. The experimental data show that most of the experimental results of ACBO algorithm are better than those of CBO, BBO, PSO and GSA algorithm. However, the convergence speed of ACBO algorithm on individual functions is worse than that of PSO algorithm, and the experimental results of ACBO algorithm on some unimodal functions are not as obvious as those on multimodal functions, so we will combine PSO and ACBO algorithm to further accelerate the convergence speed in the future work. In addition, we plan to leverage some other strategies to optimize ACBO algorithm in unimodal function.

References
[1] H.R.E.H. Bouchekara, A.E. Chaib, M.A. Abido. Optimal power flow using an Improved Colliding Bodies Optimization algorithm [J]. Applied Soft Computing, 2016, 42: 119-131.
[2] Taher Niknam, Mohammad rasoul Narimania, Masoud Jabbarib, et al. A modified shuffle frog leaping algorithm for multi-objective optimal power flow [J]. Energy, 2011, 36(11): 6420-6432.
[3] S.R. Paranjothi, K. Anburaja. Optimal power flow using refined genetic algorithm [J]. Electric Power Components and Systems 2002, 30(10): 1055-1063.
[4] A.A. Abou El Ela, M.A. Abido, S.R. Spea. Optimal power flow using differential evolution...
algorithm [J]. Electrical Engineering Berlin, 2009, 91(2): 69-78.

[5] M.A. Abido. Optimal power flow using particle swarm optimization [J]. International Journal of Electrical Power and Energy Systems, 2002, 24(7):563-571.

[6] A. Kaveh, V.R. Mahdavi. Colliding bodies optimization: A novel meta-heuristic method [J]. Computers and Structures, 2014, 139(15): 18-27.

[7] A. Kaveh, V.R. Mahdavi. Colliding Bodies Optimization method for optimum design of truss structures with continuous variables [J]. Advance in Engineering Software, 2014, 70: 1-12.

[8] S. Pal, Partha, R. Kar, D. Mandal. An efficient identification approach for stable and unstable nonlinear systems using Colliding Bodies Optimization algorithm [J]. ISA Transactions, 2015, 59: 85 – 104.

[9] A. Kaveh, V.R. Mahdavi. Optimal domain decomposition using Colliding Bodies Optimization and k-median method [J]. Finite Elements in Analysis and Design, 2015, 98:41 – 49.

[10] Wu X P, Coliding bodies optimization and its application [D]. Nanning City, Guangxi Province: GuangXi University for Nationalities, 2018.

[11] A. Kaveh, M. Ilchi Ghazaa. Enhanced colliding bodies optimization for design problems with continuous and discrete variables [J]. Advance in Engineering Software, 2014, 77: 66-75

[12] Gao W F, Liu S Y, Huang L L. Inspired artificial bee colony algorithm for Global optimization problems[J]. Acta Electronica Sinica, 2012, 40(12) :2396—2403.

[13] Chen Y X, Liang X M, Huang Y F. Improved quantum particle swarm optimization based on good-point set [J]. Journal of Central South University (Science and Technology), 2013, 44(4): 1409-1414.

[14] Sun B W, Wei S Y. DV-Hop Localization algorithm based on grey wolf optimization algorithm with adaptive adjustment strategy [J]. Computer Science, 2019, 46(5): 77-82.

[15] Zhou M, Li T Y. Nonlinear adjustment strategy of inertia weight in particle swarm optimization algorithm [J]. Computer Engineering, 2011, 37(5):204-207.

[16] Xu S J, Long W. Improved grey wolf optimizer algorithm based on stochastic convergence factor and differential mutation [J]. Science Technology and Engineering, 2018,18(23):1671-1815.

[17] P.K. Roy, S.P. Ghoshal, S.S. Thakur. Biogeography based optimization for multi-constraint optimal power flow with emission and non-smooth cost function [J]. Expert Systems with Applications, 2010, 37(12): 8221-8228

[18] A. Bhattacharya, P.K. Roy. Solution of multi-objective optimal power flow using gravitational search algorithm [J]. IIET Generation Transmission and Distribution, 2012, 6(8): 751-763.