Chapter 2
Colliding Bodies Optimization Algorithms

2.1 Introduction

This chapter consists of two parts. In part 1, the recently developed one dimensional Colliding Bodies Optimization (1D-CBO) algorithm is presented [1]. This is a multi-agent metaheuristic algorithm that is conceptualized using the one-dimensional collisions between bodies, with each agent solution being considered as an object or body with mass. The main advantage of this version is that it does not require tuning internal parameters for the decision variables (no parameter tuning).

In part 2, the two dimensional version of the CBO, denoted as 2D-CBO, is described. In this version, a memory is added to the CBO formulation to improve the performance of the algorithm [2]. This addition increases the exploitation ability and convergence rate of the CBO. Comparative studies illustrate the superiority of the 2D-CBO algorithm compared to those previously reported in the literature.

An inequality constrained optimization problem can formally be stated as:

$$\begin{array}{ll} \text{Find} & X = [x_1, x_2, x_3, \ldots, x_n] \\ \text{to minimize} & Mer(X) = f(X) \times f_{penalty}(X) \\ \text{subjected to} & g_k(X) \leq 0, \quad k = 1, 2, \ldots, m \\ & x_{jmin} \leq x_j \leq x_{jmax} \end{array} \tag{2.1}$$

where X is the vector of design variables with n unknowns, g_k is the kth constraint from m inequality constraints and $Mer(X)$ is the merit function; $f(X)$ is the objective function. Also, x_{jmin} and x_{jmax} are the lower and upper bounds of variable vector, respectively. $f_{penalty}(X)$ is the penalty function which transforms the constrained optimization problem into an unconstrained one as follows:

$$f_{penalty}(X) = 1 + \gamma_p \sum_{k=1}^{m} \max(0, g_k(x)) \tag{2.2}$$

where γ_p is penalty multiplier.

A. Kaveh and V.R. Mahdavi, *Colliding Bodies Optimization*,
DOI 10.1007/978-3-319-19659-6_2

2.2 One-Dimensional Colliding Bodies Optimization

The one-dimensional Colliding Bodies Optimization (CBO) algorithm is a new physical-inspired multi-agent metaheuristic algorithm. This algorithm is conceptualized using the one-dimensional collisions between bodies, with each agent solution being considered as an object or body with mass. After a collision of two moving bodies having specified masses and velocities, these bodies are separated with new velocities. This collision causes the agents to move toward better positions in the search space. The CBO algorithm does not require tuning internal parameters for the decision variables (no parameter tuning). Furthermore, instead of gradient-based algorithms, the CBO algorithm uses a stochastic random search and simple formulation to find minimum or maximum of functions. Compared to earlier meta-heuristic optimization algorithms, numerical results show that CBO is competitive and can be easily adopted for various types of engineering optimization problems [1].

Since CBO algorithm mimics one-dimensional collision, this part starts with describing the physical laws of one-dimensional collision between two objects to formulate the standard CBO algorithm. Then the mathematical formulation of this algorithm is presented, and its flowchart is shown. In the rest of this part, we first show performance of this algorithm on optimization of well-known test problems, then compare the obtained results of this algorithm with other state-of-the-art metaheuristic methods. Thereafter, we describe the features and advantages of this algorithm.

2.2.1 The Physical Laws of Collision

Collision is a physical process which has occurred in our daily life such as two objects colliding with each other, two balls colliding in the billiard game, or collision of cars in accident events.

Collisions between bodies are governed by the conversation laws of momentum and energy. Consider two bodies with masses of m_1 and m_2, which are moving in 1-dimensional space. These two bodies collide with each other as shown in Fig. 2.1. Provided that there are no net external forces acting upon the objects, the momentum of all objects before the collision equals the momentum of all objects after the collision.

The conservation of the total momentum demands that the total momentum before the collision is the same as the total momentum after the collision, and can be expressed by the following equation:

$$m_1 v_1 + m_2 v_2 = m_1 v_1' + m_2 v_2' \tag{2.3}$$

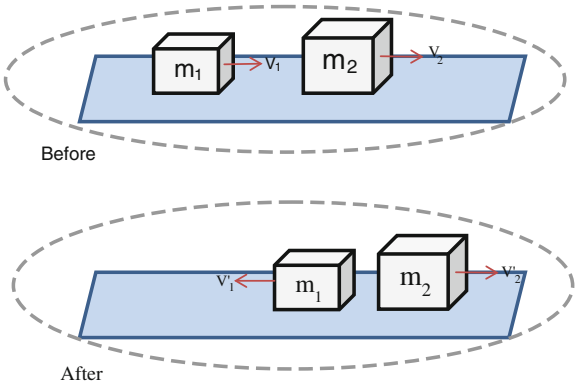

Fig. 2.1 The collision between two bodies: before and after the collision

Likewise, the conservation of the total kinetic energy is expressed as:

$$\frac{1}{2}m_1v_1^2+\frac{1}{2}m_2v_2^2=\frac{1}{2}m_1v_1'^2+\frac{1}{2}m_2v_2'^2+Q \tag{2.4}$$

where v_1 is the initial velocity of the first object before impact, v_2 is the initial velocity of the second object before impact, v_1' is the final velocity of the first object after impact, v_2' is the final velocity of the second object after impact, m_1 is the mass of the first object, m_2 is the mass of the second object and Q is the loss of kinetic energy due to the impact [3].

The formulas for the velocities after a one-dimensional collision are:

$$v_1'=\frac{(m_1-\varepsilon m_2)v_1+(m_2+\varepsilon m_2)v_2}{m_1+m_2} \tag{2.5}$$

$$v_2'=\frac{(m_2-\varepsilon m_1)v_2+(m_1+\varepsilon m_1)v_1}{m_1+m_2} \tag{2.6}$$

where ε is the Coefficient Of Restitution (COR) of the two colliding bodies, defined as the ratio of relative velocity of separation to relative velocity of approach:

$$\varepsilon=\frac{\left|v_2'-v_1'\right|}{\left|v_2-v_1\right|}=\frac{v'}{v} \tag{2.7}$$

According to the coefficient of restitution, there are two special cases of any collision as follows:

1. A perfectly elastic collision is defined as the one in which there is no loss of kinetic energy in the collision ($Q=0$ and $\varepsilon=1$). In reality, any macroscopic collision between objects will convert some kinetic energy to internal energy and other forms of energy. In this case, after collision, the velocity of separation is high.

2. An inelastic collision is the one in which part of the kinetic energy is changed to some other form of energy in the collision. Momentum is conserved in inelastic collisions (as it is for elastic collisions), but one cannot track the kinetic energy through the collision since some of it will be converted to other forms of energy. In this case, coefficient of restitution does not equal to one ($Q \neq 0$ and $\varepsilon \leq 1$). In this case, after collision the velocity of separation is low.

For the most real objects, the value of ε is between 0 and 1.

2.2.2 Mathematical Formulation of the CBO Algorithm

2.2.2.1 Theory

The main objective of the present chapter is to formulate a new simple and efficient meta-heuristic algorithm which is called Colliding Bodies Optimization (CBO). In CBO, each solution candidate X_i containing a number of variables (i.e. $X_i = \{x_{i,j}\}$) is considered as a colliding body (CB). The CBs are composed of two main equal groups; i.e. stationary and moving objects, where the moving objects move to follow stationary objects and a collision occurs between pairs of objects. This is done for two purposes: (i) to improve the positions of moving objects; (ii) to push stationary objects towards better positions. After the collision, new positions of colliding bodies are updated based on new velocity by using the collision laws as discussed in Sect. 2.2.1.

The CBO procedure can briefly be outlined as follows:

1. The initial positions of CBs are determined with random initialization of a population of individuals in the search space:

$$x_i^0 = x_{\min} + rand(x_{\max} - x_{\min}), \quad i = 1, 2, \ldots, 2n, \tag{2.8}$$

where, x_i^0 determines the initial value vector of the ith CB. $x_{\min}$ and $x_{\max}$ are the minimum and the maximum allowable values vector for the variables; *rand* is a random number in the interval [0,1]; and $2n$ is the number of CBs.

2. The magnitude of the body mass for each CB is defined as:

$$m_k = \frac{\frac{1}{fit(k)}}{\sum_{i=1}^{n} \frac{1}{fit(i)}}, \quad i = 1, 2, \ldots, 2n \tag{2.9}$$

where $fit(i)$ represents the objective function value of the ith agent; $2n$ is the number of population size. Obviously a CB with good values exerts a larger mass than the bad ones. Also, for maximizing the objective function, the term $\frac{1}{fit(i)}$ is replaced by $fit(i)$.

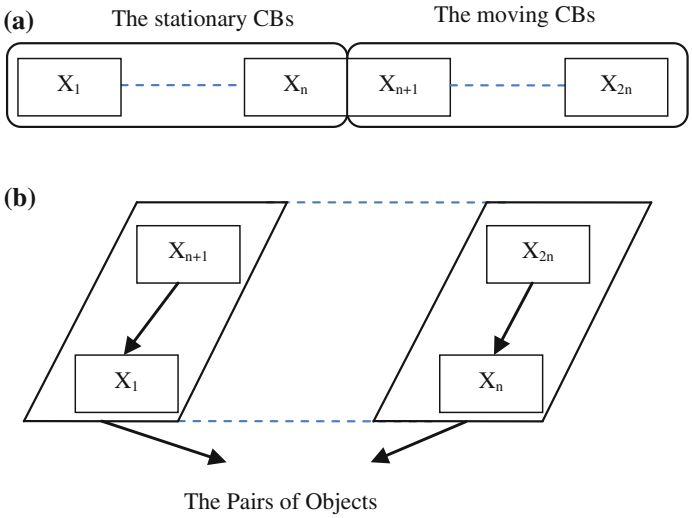

Fig. 2.2 **a** CBs sorted in increasing order, **b** colliding object pairs

3. The arrangement of the CBs objective function values is performed in ascending order (Fig. 2.2a). The sorted CBs are equally divided into two groups:

   - The lower half of CBs (stationary CBs); These CBs are good agents which are stationary and the velocity of these bodies before collision is zero. Thus:

$$v_i = 0, \quad i = 1, \ldots, n \tag{2.10}$$

   - The upper half of CBs (moving CBs): These CBs move toward the lower half. Then, according to Fig. 2.2b, the better and worse CBs, i.e. agents with upper fitness value, of each group will collide together. The change of the body position represents the velocity of these bodies before collision as:

$$v_i = x_{i-n} - x_i, \quad i = n+1, \ldots, 2n \tag{2.11}$$

     where, v_i and x_i are the velocity and position vector of the ith CB in this group, respectively; x_{i-n} is the ith CB pair position of x_i in the previous group.

4. After the collision, the velocities of the colliding bodies in each group are evaluated utilizing Eqs. (2.5) and (2.6), and the velocity before collision. The velocity of each moving CBs after the collision is obtained by:

$$v_i' = \frac{(m_i - \varepsilon m_{i-n})v_i}{m_i + m_{i-n}}, \quad i = n+1, \ldots, 2n \tag{2.12}$$

where, v_i and v_i' are the velocity of the ith moving CB before and after the collision, respectively; m_i is mass of the i th CB; m_{i-n} is mass of the ith CB pair. Also, the velocity of each stationary CB after the collision is:

$$v_i' = \frac{(m_{i+n} + \varepsilon m_{i+n})v_{i+n}}{m_i + m_{i+n}}, \quad i = 1, \ldots, n \tag{2.13}$$

where, v_{i+n} and v_i' are the velocity of the ith moving CB pair before and the ith stationary CB after the collision, respectively; m_i is mass of the ith CB; m_{i+n} is mass of the ith moving CB pair; ε is the value of the COR parameter whose law of variation will be discussed in the next section.

5. New positions of CBs are evaluated using the generated velocities after the collision in position of stationary CBs.
   The new positions of each moving CB is:

$$x_i^{new} = x_{i-n} + rand \circ v_i', \quad i = n+1, \ldots, 2n \tag{2.14}$$

   where, x_i^{new} and v_i' are the new position and the velocity after the collision of the ith moving CB, respectively; x_{i-n} is the old position of ith stationary CB pair. Also, the new positions of stationary CBs are obtained by:

$$x_i^{new} = x_i + rand \circ v_i', \quad i = 1, \ldots, n \tag{2.15}$$

   where, x_i^{new}, x_i and v_i' are the new position, old position and the velocity after the collision of the ith stationary CB, respectively. *rand* is a random vector uniformly distributed in the range [−1,1] and the sign "∘" denotes an element-by-element multiplication.
6. The optimization is repeated from Step 2 until a termination criterion, such as maximum iteration number, is satisfied. It should be noted that, a body's status (stationary or moving body) and its numbering are changed in two subsequent iterations.
   Apart from the efficiency of the CBO algorithm, which is illustrated in the next section through numerical examples, parameter independency is an important feature that makes CBO superior over other meta-heuristic algorithms. Also, the formulation of CBO algorithm does not use the memory which saves the best-so-far solution (i.e. the best position of agents from the previous iterations).

The flowchart of the CBO algorithm is also shown in Fig. 2.3, and its main steps are as follows:

Level 1: Initialization

Step 1: *Initialization*. Initialize an array of CBs with random positions and their associated values of the objective function (Eq. 2.8).

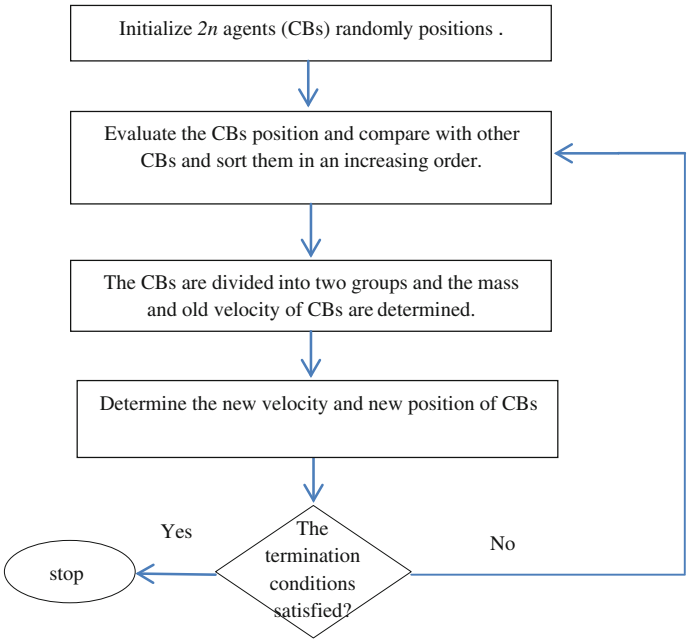

Fig. 2.3 The flowchart of the CBO algorithm

Level 2: Search

Step 1: *CBs ranking*. Compare the values of the objective function for each CB, and sort them in an ascending order.

Step 2: *Groups creation*. CBs are divided into two equal groups: (i) stationary group, (ii) moving group. Then, the pairs of CB are defined for collision (Fig. 2.2).

Step 3: *Evaluations before the collision*. The values of mass and velocity of each CB for each group are evaluated before the collision [Eqs. (2.9)–(2.11)].

Step 4: *Evaluations after the collision*. The values of velocity for each CB in each group are evaluated after the collision [Eqs. (2.12) and (2.13)].

Step 5: *CBs updating*. The new positions of CBs are calculated [Eqs. (2.14) and (2.15)].

Level 3: Terminating criterion control

Step 1: Repeat search level steps until a terminating criterion is satisfied.

The pseudo-code for the CBO algorithm is as follow:

Randomly initialize the positions of variable $X(0)$;
for i = 1 to (the maximum of iteration)

Evaluate the objective function of new position (*X*) and arrange these in an ascending order;
Evaluate the coefficient of restitution (ε) = i/the maximum of iteration
for j = 1 to (the numbers of bodies)/2

Set the moving body pair number (n1) = (the numbers of bodies)/2 + j
Set the stationary body pair number (n2) = j
Evaluate the velocity of moving body pair before collision (v_{n1}) = $X_{n2}(j-1) - X_{n1}(j-1)$
Evaluate the velocity of stationary body pair before collision (v_{n2}) = 0
Evaluate the velocity of moving body pair after collision $(v'_{n1}) = \frac{(m_{n1}-\varepsilon m_{n2})v_{n1}}{m_{n1}+m_{n2}}$
Evaluate the velocity of stationary body pair after collision $(v'_{n2}) = \frac{(m_{n1}+\varepsilon m_{n2})v_{i+\frac{n}{2}}}{m_{n1}+m_{n2}}$
Evaluate the position of moving body pair after collision $(x_{n1}(j)) = x_{n2}(j-1) + rand \cdot v'_{n1}$
Evaluate the position of stationary body pair after collision $(x_{n2}(j)) = x_{n2}(j-1) + rand \cdot v'_{n2}$

end

end

2.2.2.2 The Coefficient of Restitution

The meta-heuristic algorithms have two phases: exploration of the search space and exploitation of the best solutions found. In the meta-heuristic algorithm it is very important to have a suitable balance between the exploration and exploitation [4]. In the optimization process, the exploration should be decreased gradually while simultaneously exploitation should be increased.

In this algorithm, an index is introduced in terms of the coefficient of restitution (COR) to control exploration and exploitation rate. In fact, this index is defined as the ratio of the separation velocity of two agents after collision to approach velocity of two agents before collision. Efficiency of this index will be shown using one numerical example.

In this section, in order to have a general idea about the performance of COR in controlling local and global searches, a benchmark function (Aluffi-Pentiny) chosen from Ref. [5] is optimized using the CBO algorithm. Three variants of COR values are considered. Figure 2.4 is prepared to show the positions of the current CBs in 1st, 50th and 100th iteration for these cases. These three typical cases result in the following:

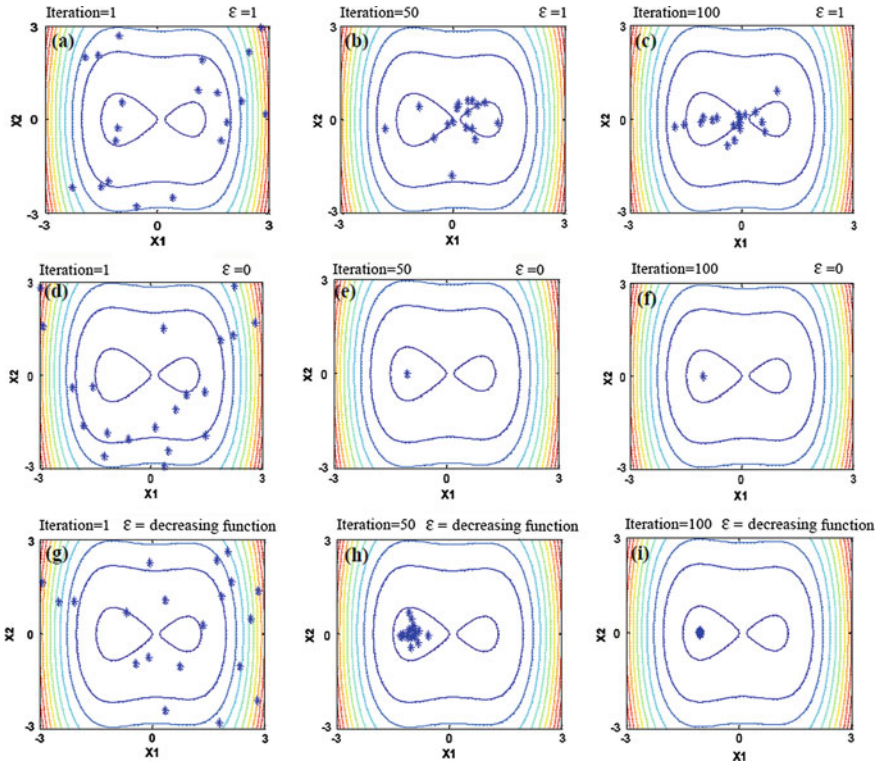

Fig. 2.4 Evolution of the positions of CBs during optimization history for different definitions of the coefficient of restitution (Aluffi-Pentiny benchmark function)

1. The perfectly elastic collision: In this case, COR is set equal to unity. It can be seen that in the final iterations, the CBs investigate the entire search space to discover a favorite space (global search).
2. The hypothetical collision: In this case, COR is set equal to zero. It can be seen that in the 50th iterations, the movements of the CBs are limited to very small space in order to provide exploitation (local search). Consequently, the CBs are gathered in a small region of the search space.
3. The inelastic collision: In this case, COR decreases linearly to zero and ε is defined as:

$$\varepsilon = 1 - \frac{iter}{iter_{\max}} \tag{2.16}$$

where *iter* is the actual iteration number and $iter_{max}$ is the maximum number of iterations. It can be seen that the CBs get closer by increasing iteration. In this way a good balance between the global and local search is achieved. Therefore, in the optimization process COR is considered such as the above equation.

2.2.3 The Features of the CBO Algorithm

In this section, firstly CBO is achieved for optimization of well-known mathematical and engineering problems to show the performance of the proposed algorithm. Then, the features, properties and advantages of this algorithm are mentioned.

2.2.3.1 Numerical Examples

Three well-studied engineering design problems and two mathematical optimization problems taken from the optimization literature are used to show the performance of the proposed algorithm. These examples have been previously studied using a variety of other techniques, which are useful to show the validity and effectiveness of the proposed algorithm. In order to assess the effect of the initial population on the final result, these examples are independently optimized with different initial populations.

For engineering design examples, 30 independent runs were performed for CBO, considering 20 individuals and 200 iterations; the corresponding number of function evaluations is 4000. The number of function evaluations set for the GA-based algorithm developed by Deb [6], the PSO-based method developed by He and Wang [7], the evolution strategies developed by Montes and Coello [8] is 900,000, 200,000 and 25,000, respectively. Similar to CBO, the number of function evaluations for the charged system search algorithm developed by Kaveh and Talatahari [4] is 4000.

Example 1: Design of welded beam As the first example, design optimization of the welded beam shown in Fig. 2.5 is carried out. The welded beam design problem was often utilized to evaluate performance of different optimization methods [4, 6–11]. The objective is to find the best set of design variables to minimize the total fabrication cost of the structure subject to shear stress (τ), bending stress (σ), buckling load (Pc), and end deflection (δ) constraints. Assuming $x_1 = h$, $x_2 = l$, $x_3 = t$, and $x_4 = b$ as the design variables, the mathematical formulation of the problem can be expressed as:

Find

$$\{x_1, x_2, x_3, x_4\} \tag{2.17}$$

To minimize

$$\cos t(x) = 1.10471 x_1^2 x_2 + 0.04811 x_3 x_4 (14 + x_2) \tag{2.18}$$

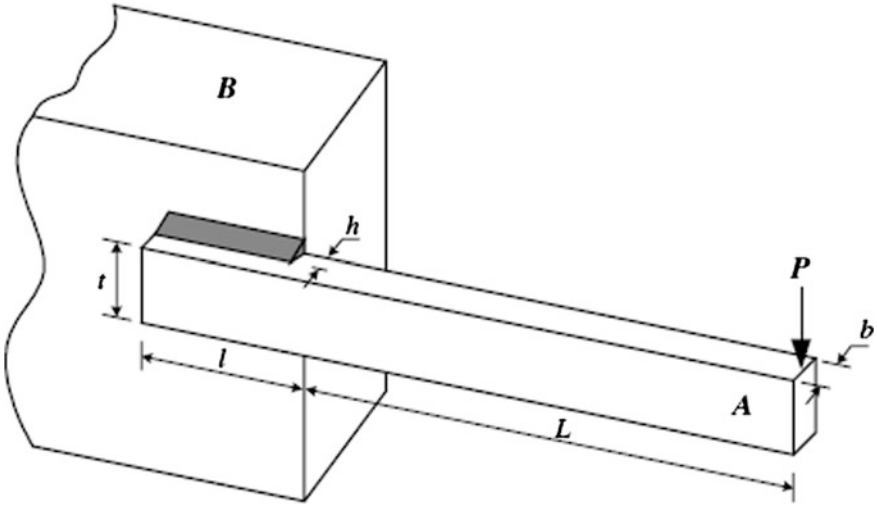

Fig. 2.5 Schematic of the welded beam structure with indication of design variables

Subjected to

$$\begin{aligned} g_1(x) &= \tau(x) - \tau_{max} \le 0 \\ g_2(x) &= \sigma(x) - \sigma_{max} \le 0 \\ g_3(x) &= x_1 - x_4 \le 0 \\ g_4(x) &= 0.10471 x_1^2 + 0.04811 x_3 x_4 (14 + x_2) - 5 \le 0 \\ g_5(x) &= 0.125 - x_1 \le 0 \\ g_6(x) &= \delta(x) - \delta_{max} \le 0 \\ g_7(x) &= p - p_c(x) \le 0 \end{aligned} \tag{2.19}$$

The bounds on the design variables are:

$$0.1 \le x_1 \le 2, \quad 0.1 \le x_2 \le 10, \quad 0.1 \le x_3 \le 10, \quad 0.1 \le x_4 \le 2 \tag{2.20}$$

$$\begin{aligned} \tau(x) &= \sqrt{(\tau')^2 + 2\tau'\tau''\frac{x_2}{2R} + (\tau'')^2} \\ \tau' &= \frac{P}{\sqrt{2}x_1x_2} \quad \tau'' = \frac{MR}{J} \quad M = P(L + \frac{x_2}{2}) \quad R = \sqrt{\frac{x_2^2}{4} + (\frac{x_1 + x_3}{2})^2} \\ J &= 2\left\{\sqrt{2}x_1x_2\left[\frac{x_2^2}{12} + (\frac{x_1 + x_3}{2})^2\right]\right\} \quad \sigma(x) = \frac{6PL}{x_4x_3^2} \quad \delta(x) = \frac{4PL^3}{Ex_3^3x_4} \\ P_c(x) &= \frac{4.013\sqrt{E(x_3^2x_4^6/36)}}{L^2}(1 - \frac{x_3}{2L}\sqrt{\frac{E}{4G}}) \end{aligned} \tag{2.21}$$

where the constants in Eqs. (2.19) and (2.20) are chosen as follows:

P = 6000 lb, L = 14 in., $E = 30 \times 10^6$ psi, $G = 12 \times 10^6$ psi, τ_{max} = 13,600 psi, σ_{max} = 30,000 psi, and δ_{max} = 0.25 in.

Radgsdell and Phillips [9] compared optimal results of different optimization methods which were mainly based on mathematical optimization algorithms. Deb [6], Coello [10], and Coello and Montes [11] solved this problem using GA-based methods. Also, He and Wang [7] used effective co-evolutionary particle swarm optimization, Montes and Coello [8] solved this problem utilizing evolution strategies, and Kaveh and Talatahari [4] employed charged system search.

Table 2.1 compares the optimized design and the corresponding cost obtained by CBO with those obtained by other meta-heuristic algorithms documented in literature. It can be seen that the best solution obtained by CBO is better than those quoted for the other algorithms. The statistical data on 30 independent runs reported in Table 2.2 also demonstrate the better search ability of CBO with respect to the other algorithms: in fact the best, worst and average cost, and the standard deviation (S.D.) of the obtained solutions are better than literature. The lowest standard deviation achieved by CBO proves that the present algorithm is more robust than other meta-heuristic methods.

Example 2: Design of a pressure vessel Design optimization of the cylindrical pressure vessel capped at both ends by hemispherical heads (Fig. 2.6) is considered as the second example. The objective of optimization is to minimize the total manufacturing cost of the vessel based on the combination of welding, material and forming costs. The vessel is designed for a working pressure of 3000 psi and a minimum volume of 750 ft^3 regarding the provisions of *ASME* boiler and pressure vessel code. Here, the shell and head thicknesses should be multiples of 0.0625 in. The thickness of the shell and head is restricted to 2 in. The shell and head thicknesses must not be less than 1.1 and 0.6 in., respectively. The design variables of the problem are x_1 as the shell thickness (T_s), x_2 as the spherical head thickness (T_h), x_3 as the radius of cylindrical shell (R), and x_4 as the shell length (L). The problem formulation is as follows:

Find

$$\{x_1, x_2, x_3, x_4\} \tag{2.22}$$

To minimize

$$\cos t(x) = 0.6224x_3x_1x_4 + 1.7781x_3^2x_2 + 3.1611x_1^2x_4 + 19.8621x_3x_1^2 \tag{2.23}$$

Subject to

$$\begin{aligned} g_1(x) &= 0.0193x_3 - x_1 \leq 0 \\ g_2(x) &= 0.00954x_3 - x_2 \leq 0 \\ g_3(x) &= 750 \times 1728 - \pi x_3^2 x_4 - \frac{4}{3}\pi x_3^3 \leq 0 \\ g_4(x) &= x_4 - 240 \leq 0 \end{aligned} \tag{2.24}$$

Table 2.1 Comparison of CBO optimized designs with literature for the welded beam problem

Design variables	Best solution found							
	Ragsdell and Phillips [9]	Deb [6]	Coello [10]	Coello and Montes [11]	He and Wang [7]	Montes and Coello [8]	Kaveh and Talatahari [4]	Present work
x_1(h)	0.245500	0.248900	0.208800	0.205986	0.202369	0.202369	0.20582	0.205722
x_2(l)	6.1960000	6.173000	3.420500	3.471328	3.544214	3.544214	3.468109	3.47041
x_3(t)	8.273000	8.178900	8.997500	9.020224	9.04821	9.04821	9.038024	9.037276
x_4(b)	0.245500	0.253300	0.210000	0.20648	0.205723	0.205723	0.205723	0.205735
f(x)	1.728024	2.433116	−1.748310	1.728226	1.728024	1.728024	1.724866	1.724663

Table 2.2 Statistical results from different optimization methods for the welded beam design problem

Std dev	Worst result	Average optimized cost	Best result	Methods
N/A	N/A	N/A	2.385937	Ragsdell and Phillips [9]
N/A	N/A	N/A	2.433116	Deb [6]
0.011220	1.785835	1.771973	1.748309	Coello [10]
0.074713	1.993408	1.792654	1.728226	Coello and Montes [11]
0.012926	1.782143	1.748831	1.728024	He and Wang [7]
0.070500	1.994651	1.813290	1.737300	Montes and Coello [8]
0.008064	1.759479	1.739654	1.724866	Kaveh and Talatahari [4]
0.0002437	1.725059	1.725707	1.724662	Present work

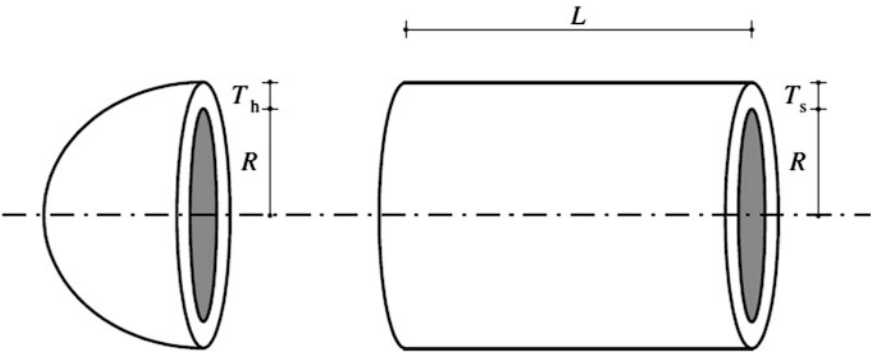

Fig. 2.6 Schematic of the spherical head and cylindrical wall of the pressure vessel with indication of design variables

The bounds on the design variables are:

$$1.125 \leq x_1 \leq 2, \quad 0.625 \leq x_2 \leq 2, \quad 10 \leq x_3 \leq 240, \quad 10 \leq x_4 \leq 240 \qquad (2.25)$$

It can be seen from Table 2.3 that the present algorithm found the best design overall which is about 3 % lighter than the best known design quoted in literature (5889.911 vs. 6059.088 of Ref. [4]). The statistical data reported in Table 2.4 indicate that the standard deviation of CBO optimized solutions is the third lowest among those quoted for the different algorithms compared in this test case. Statistical results given in Table 2.4 indicate that CBO is in general more robust than the other meta-heuristic algorithms. However, the worst optimized design and standard deviation found by CBO are higher than for CSS.

Example 3: Design of a tension/compression spring This problem was first described by Belegundu [12] and Arora [13]. It consists of minimizing the weight of a tension/compression spring subject to constraints on shear stress, surge frequency, and minimum deflection as shown in Fig. 2.7. The design variables are the wire diameter d (= x_1); the mean coil diameter D (= x_2), and the number of active coils N (= x_3). The problem can be stated as follows:

Table 2.3 Comparison of CBO optimized designs with literature for the pressure vessel problem

$X_4(L)$	$X_3(R)$	$X_2(T_h)$	$X_1(T_s)$	Methods
117.7010	47.70000	0.625000	1.125000	Sandgren [20]
43.6900	58.29100	0.625000	1.125000	Kannan and Kramer [21]
112.6790	48.32900	0.500000	0.937500	Deb and Gene [22]
200.0000	40.32390	0.437500	0.812500	Coello [10]
176.6540	42.09739	0.437500	0.812500	Coello and Montes [11]
176.7465	42.09126	0.437500	0.812500	He and Wang [7]
176.6405	42.09808	0.437500	0.812500	Montes and Coello [8]
176.572656	42.103624	0.437500	0.812500	Kaveh and Talatahari [4]
198.76232	40.409065	0.385560	0.779946	Present work

Table 2.4 Statistical results from different optimization methods for the pressure vessel problem

Std dev	Worst result	Average optimized cost	Best result	Methods
N/A	N/A	N/A	8129.103	Sandgren [20]
N/A	N/A	N/A	7198.042	Kannan and Kramer [21]
N/A	N/A	N/A	6410.381	Deb and Gene [22]
7.4133	6308.149	6293.843	6288.744	Coello [10]
130.9297	6469.322	6177.253	6059.946	Coello and Montes [11]
86.4545	6363.804	6147.133	6061.077	He and Wang [7]
426.0000	7332.879	6850.004	6059.745	Montes and Coello [8]
10.256	6085.476	6067.906	6059.088	Kaveh and Talatahari [4]
63.5417	6213.006	5934.201	5889.911	Present work

Fig. 2.7 Schematic of the tension/compression spring with indication of design variables

Find

$$\{x_1, x_2, x_3\} \tag{2.26}$$

To minimize

$$\cos t(x) = (x_3 + 2)x_2x_1^2 \tag{2.27}$$

Subject to

$$\begin{aligned} g_1(x) &= 1 - \frac{x_2^3 x_3}{71785 x_1^4} \leq 0 \\ g_2(x) &= \frac{4x_2^2 - x_1 x_2}{12566(x_2 x_1^3 - x_1^4)} + \frac{1}{5108 x_1^2} - 1 \leq 0 \\ g_3(x) &= 1 - \frac{140.45 x_1}{x_2^2 x_3} \leq 0 \\ g_4(x) &= \frac{x_1 + x_2}{1.5} - 1 \leq 0 \end{aligned} \tag{2.28}$$

The bounds on the design variables are:

$$0.05 \leq x_1 \leq 2, \quad 0.25 \leq x_2 \leq 1.3, \quad 2 \leq x_3 \leq 15, \tag{2.29}$$

This problem has been solved by Belegundu [12] using eight different mathematical optimization techniques. Arora [13] also solved this problem using a numerical optimization technique called a constraint correction at the constant cost. Coello [10] as well as Coello and Montes [11] solved this problem using GA-based method. Additionally, He and Wang [7] utilized a co-evolutionary particle swarm optimization (CPSO). Recently, Montes and Coello [8], Kaveh and Talatahari [4] used evolution strategies and the CSS to solve this problem, respectively.

Tables 2.5 and 2.6 compare the best results obtained in this work and those of the other researches. Once again, CBO found the best design overall. In fact, the lighter design found by Kaveh and Talatahari in [4] actually violates the first two optimization constraints. The statistical data reported in Table 2.6 show that the standard deviation on optimized cost seen for CBO is fully consistent with literature.

2.2.4 The Features of CBO

This section presents a novel efficient meta-heuristic optimization algorithm called Colliding Bodies Optimization (CBO). From the results obtained of our method, we draw the following conclusions:

(i) Using the governing laws from physics, we proposed a simple and efficient algorithm, where these laws determine the movement process of the objects. In this algorithm, each agent solution is considered as the colliding body. After a collision of two moving bodies which have specified masses and velocities, these bodies are separated with new velocities.

(ii) Most of the meta-heuristic have some constant parameters. It is very important to note that in these algorithms the constant parameters should carefully be chosen. In fact the algorithms are very sensitive with respect to these

Table 2.5 Comparison of CBO optimized designs with literature for the tension/compression spring problem

f(x)	Constraints				Optimal design variables			Methods
	$g_4(x)$	$g_3(x)$	$g_2(x)$	$g_1(x)$	x_3(N)	x_2(D)	x_1(d)	
0.0128334	−0.756067	−3.938302	−0.003782	−0.000014	14.250000	0.315900	0.050000	Belegundu [12]
0.0127303	−0.698283	−4.123832	−0.000018	−0.053396	9.185400	0.399180	0.053396	Arora [13]
0.0127048		−4.026318	−0.000110	−0.002080	11.632201	0.351661	0.051480	Coello [10]
0.0126810	−0.722698	−4.061338	−0.000021	−0.000013	10.890522	0.363965	0.051989	Coello and Montes [11]
0.0126747	−0.727090	−4.051300	−1.2600e−05	−0.000845	11.244543	0.357644	0.051728	He and Wang [7]
0.012698	−0.728664	−4.039301	−0.0000567	−0.001732	11.397926	0.355360	0.051643	Montes and Coello [8]
0.0126384	−0.726483	−4.063371	0.0011043	8.78603e−6	11.165704	0.358532	0.051744	Kaveh and Talatahari [4]
0.0126697	−0.724287	−4.061846	−1.4189e−5	−3.1073e−4	11.007846	0.3616740	0.051894	Present work

Table 2.6 Statistical results from different optimization methods for tension/compression string problem

Std dev	Worst result	Average optimized cost	Best result	Methods
N/A	N/A	N/A	0.0128334	Belegundu [12]
N/A	N/A	N/A	0.0127303	Arora [13]
3.9390e−5	0.012822	0.012769	0.0127048	Coello [10]
5.9000e−5	0.012973	0.0127420	0.0126810	Coello and Montes [11]
5.1985e−5	0.012924	0.012730	0.0126747	He and Wang [7]
9.6600e−4	0.16485	0.013461	0.012698	Montes and Coello [8]
8.3564e−5	0.013626	0.012852	0.0126384	Kaveh and Talatahari [4]
5.00376e−5	0.0128808	0.01272964	0.0126697	Present work

parameters and therefore the algorithm should be run with different values of these parameters until the best values are identified. However the present algorithm apart from being efficient, it is independent of such parameters. This is an important superiority of the CBO algorithm.

(iii) In this algorithm, an index is introduced in terms of the coefficient of restitution (COR) to control of the exploration and exploitation rates.

(iv) The formulation of the CBO algorithm does not use the memory which saves the best-so-far solution (i.e. the best position of agents from the previous iterations).

(v) The proposed approach performed well comparing the numerical results of five classical test problems. The results are compared to those generated with other techniques reported in the literatures.

2.3 Two-Dimensional Colliding Bodies Optimization

This part focuses on the new version of the one-dimensional Colliding Bodies Optimization, and its utility for optimization of truss structures. The idea of the standard CBO is derived from one-dimensional collisions between bodies, which does not use the internal parameter and memory in its formulation. However, the exploitation phase of the CBO is weak due to not using a memory for saving the best-so-far solution in its formulation. Here, in the two dimensional version of CBO, denoted by 2D-CBO, a memory is added to the standard CBO formulation to improve the performance of the latter algorithm. This addition increases the exploitation ability and convergence rate of the CBO. Comparative studies illustrate the superiority of the 2D-CBO algorithm compared to those previously reported in the literature [2].

Although the development of meta-heuristic algorithms within the framework of evolutionary algorithms (EAs) has been successful in the recent years, the development of more efficient algorithms for minimizing the construction cost and

evaluation efforts is still a challenging subject in the field of optimization of engineering problems. The existing algorithms have often two phases: exploration of the search space and exploitation of the best solutions found. Apparently, one of the main problems in developing an efficient meta-heuristic algorithm is to keep a reasonable balance between the exploration and exploitation abilities [14–16].

The present section develops an algorithm based on two dimensional collision laws. In this algorithm, the saved memories, i.e. the agents with best values, is been added to CBO. In fact, the bodies move toward the best saved agents and therefore it increases the exploitation.

The present section is organized as follows: In next section, two dimensional collision laws are briefly introduced. The new method is then presented, followed by a section consisting of the study of optimization of two mathematical constrained functions. Conclusions are derived in the final section.

2.3.1 Formulation of the Two-Dimensional Collision

Since our purpose in this section is to achieve the known physical equations in formulation of an optimization algorithm, this subsection briefly describes the mathematical formulation of elastic collision law between two bodies in two-dimensional space. Consider the model of a collision, where one body with mass m_1 and velocity V_1 moves toward to second body with mass m_2 which is at rest, Fig. 2.8. It should be noted that if the velocity direction of first body be coincident with the direction of center of bodies ($\alpha = \pi$), then the collision is one-dimensional, otherwise it is two-dimensional.

The velocity and direction of bodies are derived using the conversation laws of momentum and energy. When a two-dimensional collision occurs in a laboratory system, the total momentum of the system in each direction is conserved, Fig. 2.9. The directions of velocity of bodies after collision are expressed as:

$$\tan \vartheta_1 = \frac{m_2 \sin \theta}{m_1 + m_2 \cos \theta}, \quad \vartheta_2 = \frac{\pi - \theta}{2} \tag{2.30}$$

where ϑ_1 and ϑ_2 are the directions of the velocities of the first and second body after impact, respectively, m_1 is the mass of the first object, m_2 is the mass of the second object, and θ is the angle through which the direction of motion of the first object is turned [17] (Fig. 2.9). The magnitudes of the velocity of bodies can be expressed by the following equations:

$$V_1' = \frac{\sqrt{m_1^2 + m_2^2 + 2m_1 m_2 \cos(\theta)}}{m_1 + m_2} V_1, \quad V_2' = \frac{2m_1}{m_1 + m_2} \sin(\frac{\theta}{2}) V_1 \tag{2.31}$$

where V_1' and V_2' are the final velocities of the first and second object after collision, respectively [17].

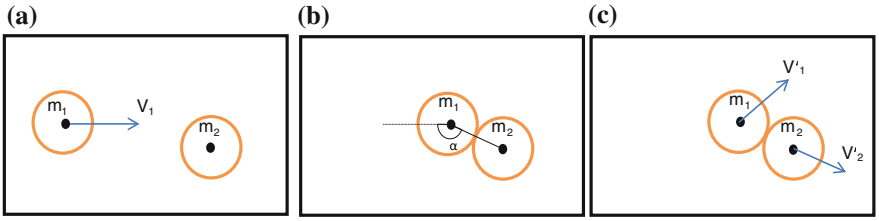

Fig. 2.8 Two dimensional collision between two bodies. **a** Before the collision, **b** the collision occurring, **c** after the collision

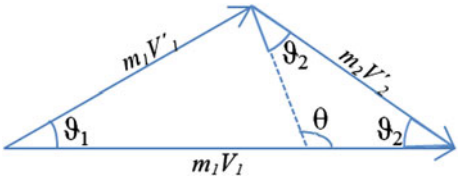

Fig. 2.9 Schematic of the conservation law of momentum

2.3.2 The 2-Dimensional Version of the CBO Algorithm

In this study, the laws of collision between two bodies in 2-dimensional space, as mentioned in Sect. 2.3.1, are implemented leading to a new optimization search strategy. The main algorithm is based on standard CBO, where some changes being added. Here, two features are added to the formulation of the standard CBO: (i) the saving memories are added to increase the exploitation ability of standard CBO algorithm, (ii) the COR of standard CBO is ignored due to considering the elastic collision formulation.

The proposed algorithm can now be described as follows:

Step 1: Initial populations, consisting of $2n$ individuals and their associated velocities, are created by means of the randomly generated individuals.

Step 2: The arrangement of the CBs and "mating process" is performed as shown in Fig. 2.2.

Step 3: The moving CBs move toward the stationary CBs. Therefore, the velocity of CBs before collision is derived:

$$v_i = \begin{cases} 0, & i = 1, \ldots, n \\ x_{i-n} - x_i, & i = n+1, \ldots, 2n \end{cases} \tag{2.32}$$

Step 4: The magnitude of the body mass for each CB is defined as:

$$m_k = \frac{\frac{1}{fit(k)}}{\sum_{i=1}^{n} \frac{1}{fit(i)}}, \quad k = 1, 2, \ldots, 2n \tag{2.33}$$

Step 5: Compute the velocity direction of the stationary CBs after the collision such that it moves toward the best object position:

$$\vartheta_i = \arctan\left(\frac{sbv - fit(i)}{sbp - x_i + \varepsilon}\right), \quad i = 1, \ldots, n \tag{2.34}$$

where ϑ_i is the velocity direction vector of the ith stationary CB after impact; $fit(i)$ represents the objective function value of the ith stationary CB; x_i is the position vector of the ith stationary CB; sbp and sbv are the saved best position of CBs and its corresponded value, respectively; ε is a small positive number to avoid singularities, and arctan is the inverse of tangent function between 0 and π.

Step 6: Then the angles θ between pairs of bodies is derived using Eqs. (2.30) and (2.34):

$$\theta_i = \pi - 2\vartheta_i, \quad i = 1, \ldots, n \tag{2.35}$$

Step 7: Compute the velocity of CBs after the collision using Eqs. (2.31) and (2.35):

$$v_i' = \begin{cases} \frac{2m_i}{m_i + m_{i+n}} \sin\left(\frac{\theta_i}{2}\right) v_i, & i = 1, \ldots, n \\ \frac{\sqrt{m_{i-n}^2 + m_i^2 + 2m_i m_{i-n} \cos(\theta_{i-n})}}{m_i + m_{i-n}} v_{i-n}, & i = n+1, \ldots, 2n \end{cases} \tag{2.36}$$

Step 8: The new positions of CBs are obtained using the generated velocities after the collision in the position of stationary CBs:

$$x_i^{new} = \begin{cases} x_i + rand \circ v_i', & i = 1, \ldots, n \\ x_{i-n} + rand \circ v_i', & i = n+1, \ldots, 2n \end{cases} \tag{2.37}$$

Here, *rand* is a random vector uniformly distributed in the range [−1, 1], and the sign "∘" denotes an element-by-element multiplication.

Step 9: The optimization is repeated starting with Step 2 until a termination criterion, until the maximum number of iteration, is satisfied.

2.3.3 Numerical Examples

This section discusses the computational examples used to investigate the performance of the proposed algorithm. To present a detailed comparison two mathematical function examples are utilized. These examples have been previously solved using a variety of other techniques, which is useful to show the validity and effectiveness of the proposed algorithm. In order to assess the effect of the initial population on the final result, 20 independent runs are carried out using the present algorithm and standard CBO with different initial populations. To show the effect of number of agents on the final results, for both algorithms two types of population sizes are considered. The algorithms are implemented in MATLAB.

Example 1: Constrained function I Optimization of the constrained function expressed in Eq. (2.38) is considered as the first example. This problem has seven variables and four nonlinear inequality constraints. This problem is defined as:

Find

$$\{x_1, x_2, x_3, x_4, x_5, x_6, x_7\} \tag{2.38}$$

To minimize

$$\begin{aligned} f(x) =& (x_1 - 10)^2 + 5(x_2 - 12)^2 + x_3^4 + 3(x_4 - 11)^2 + 10x_5^6 + 7x_6^2 + x_7^4 \\ & - 4x_6x_7 - 10x_6 - 8x_7 \end{aligned} \tag{2.39}$$

Subjected to

$$\begin{aligned} g_1(x) &= 127 - 2x_1^2 - 3x_2^4 - x_3 - 4x_4^2 - 5x_5 \geq 0, \\ g_2(x) &= 282 - 7x_1 - 3x_2 - 10x_3^2 - x_4 + x_5 \geq 0, \\ g_3(x) &= 196 - 23x_1 - x_2^2 - 6x_6^2 + 8x_7 \geq 0, \\ g_4(x) &= -4x_1^2 - x_2^2 + 3x_1x_2 - 2x_3^2 - 5x_6 + 11x_7 \geq 0. \end{aligned} \tag{2.40}$$

The bounds on the design variables are:

$$-10 \leq x_i \leq 10 \quad (i = 1\text{-}7) \tag{2.41}$$

This problem has been solved by Deb [18] using an efficient constraint handling method for the GA, and Lee and Geem [19] employed the harmony search (HS) algorithm.

In this problem, population size (n) is set to 40 and 60 individuals. The maximum number of optimization iterations is also considered as 800. The obtained values and statistical results of this problem for each algorithm are given in Tables 2.7 and 2.8, respectively. It can be seen that the best solutions obtained by CBO and 2D-CBO are better than HS algorithm-based method with less fitness function evaluations. Although best and average results of 20 independent runs are

Table 2.7 Optimal variables obtained by different researchers for the constrained function I

Optimal design variables (x)	Deb [18]	Lee and Geem [19]	Present study (CBO)		Present study (2D-CBO)	
			$\boldsymbol{n}$ = 40	$\boldsymbol{n}$ = 60	$\boldsymbol{n}$ = 40	$\boldsymbol{n}$ = 60
x_1	Unavailable	2.323456	2.336792	2.318452	2.3215438	2.313524
x_2		1.951242	1.948239	1.945693	1.9511246	1.948511
x_3		−0.44847	−0.43323	−0.49955	−0.496029	−0.4801
x_4		4.36192	4.370184	4.385752	4.3675236	4.377405
x_5		−0.63008	−0.62052	−0.63714	−0.612266	−0.62322
x_6		1.03866	1.058898	1.045751	1.0444193	1.048238
x_7		1.605384	1.604781	1.590209	1.5900506	1.580425

Table 2.8 Statistical results from different optimization methods for the constrained function I

Methods		Best objective function	Average objective function	Std dev	Fitness function evaluations
Deb [18]		680.6344	680.6417	N/A	350,070
Lee and Geem [19]		680.6413	N/A	N/A	160,000
Present study [1] (CBO)	$\boldsymbol{n}$ = 40	680.6404	680.919	0.2003	24,000
	$\boldsymbol{n}$ = 60	680.6465	680.735	0.104	36,000
Present study [2] (2D-CBO)	$\boldsymbol{n}$ = 40	680.6352	680.845	0.211643	24,000
	$\boldsymbol{n}$ = 60	680.6373	680.7643	0.123451	36,000

more than those of Deb [18], however, the difference of fitness evaluations is significant. Therefore, the obtained statistical results using 2D-CBO are better than those of CBO algorithm. The best results obtained using 2D-CBO shows that this algorithm is less sensitive to the population size compared to the CBO. Figure 2.10 shows the convergence curves of the two algorithms, and it indicates high convergence rate for the 2D-CBO compared to the CBO algorithm.

Example 2: Constrained function II This is a 10-variable problem which challenges the algorithm ability to deal with the problem of optimization. This problem also has eight nonlinear inequality constraints. This problem is defined as:

Find

$$\{x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}\} \tag{2.42}$$

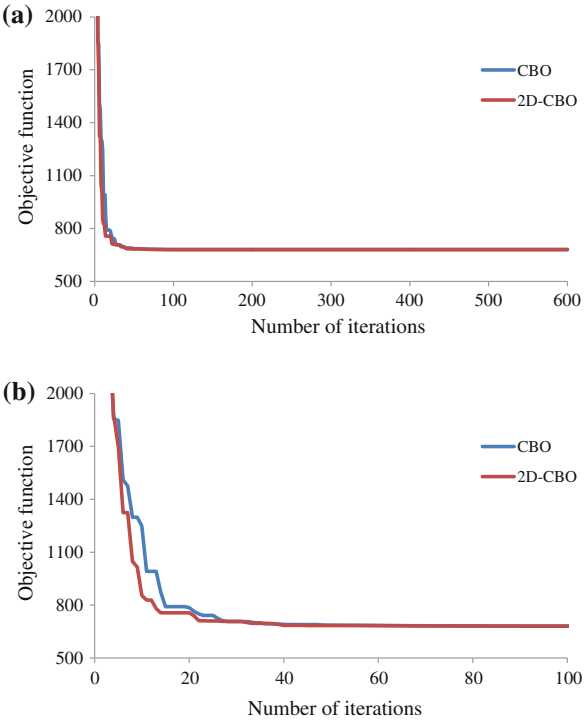

Fig. 2.10 Comparison of the convergence curves of the two algorithms for the constrained function I. **a** All iterations, **b** 1–100 iterations

To minimize

$$f(x) = x_1^2 + x_2^2 + x_1x_2 - 14x_1 - 16x_2 + (x_3 - 10)^2 + 4(x_4 - 5)^2 + (x_5 - 3)^2 + 2(x_6 - 1)^2 + 5x_7^2 + 7(x_8 - 11)^2 + 2(x_9 - 10)^2 + (x_{10} - 7)^2 + 45 \tag{2.43}$$

Subjected to

$$\begin{aligned}
g_1(x) &= 105 - 4x_1 - 5x_2 + 3x_7 - 9x_8 \geq 0,\\
g_2(x) &= -10x_1 + 8x_2 + 17x_7 - 2x_8 \geq 0,\\
g_3(x) &= 8x_1 - 2x_2 - 5x_9 + 2x_{10} + 12 \geq 0,\\
g_4(x) &= -3(x_1 - 2)^2 - 4(x_2 - 3)^2 - 2x_3^2 + 7x_4 + 120 \geq 0,\\
g_5(x) &= -5x_1^2 - 8x_2 - (x_3 - 6)^2 + 2x_4 + 40 \geq 0,\\
g_6(x) &= -x_1^2 - 2(x_2 - 2)^2 + 2x_1x_2 - 14x_5 + 6x_6 \geq 0,\\
g_7(x) &= -0.5(x_1 - 8)^2 - 2(x_2 - 4)^2 - 3x_5^2 + x_6 + 30 \geq 0,\\
g_8(x) &= 3x_1 - 6x_2 - 12(x_9 - 8)^2 + 7x_{10} \geq 0.
\end{aligned} \tag{2.44}$$

Table 2.9 Optimal design variables obtained by different researchers for the constrained function II

Optimal design variables (x)	Deb [18]	Lee and Geem [19]	Present study (CBO)		Present study (2D-CBO)	
			$\boldsymbol{n}$ = 80	$\boldsymbol{n}$ = 100	$\boldsymbol{n}$ = 80	$\boldsymbol{n}$ = 100
x_1	Unavailable	2.155225	2.140682	2.142755	2.169335	2.163967
x_2		2.407687	2.447447	2.441786	2.378239	2.387446
x_3		8.778069	8.770297	8.772559	8.75873	8.761691
x_4		5.102078	5.083513	5.089189	5.083288	5.070258
x_5		0.967625	1.00314	0.976804	0.958985	0.987816
x_6		1.357685	1.424841	1.36545	1.349924	1.413288
x_7		1.287760	1.25747	1.261765	1.31204	1.303916
x_8		9.800438	9.774706	9.778372	9.81862	9.813183
x_9		8.187803	8.249837	8.196755	8.285297	8.221302
x_{10}		8.256297	8.509323	8.362651	8.414142	8.284834

Table 2.10 Statistical results from different optimization methods for the constrained function II

Fitness function evaluations	Std dev	Average objective function	Best objective function		Methods
350,070	N/A	24.40940	24.37248		Deb [18]
230,000	N/A	N/A	24.36679		Lee and Geem [19]
80,000	0.609358	24.90411	24.39129	$\boldsymbol{n}$ = 80	Present study (1D-CBO)
100,000	0.580431	24.86188	24.38470	$\boldsymbol{n}$ = 100	
80,000	0.369562	24.69515	24.33491	$\boldsymbol{n}$ = 80	Present study (2D-CBO)
100,000	0.362542	24.70537	24.33704	$\boldsymbol{n}$ = 100	

The bounds on the design variables are:

$$-10 \leq x_i \leq 10 \quad (i = 1\text{-}10) \tag{2.45}$$

This problem has been solved by Deb [18] utilizing an efficient constraint handling method for the GA, and Lee and Geem [19] employed the harmony search algorithm.

In this case, the population size (n) is set as 80 and 100 individuals. The maximum number of optimization iterations is also considered as 1000. Tables 2.9 and 2.10 compare the optimized variables and the statistical results obtained by CBO and 2D-CBO with those obtained by other meta-heuristic algorithms documented in literature. According to these tables, the best results reported in the literature is 24.36679. While, the 2D-CBO found the best results as 24.33491 after 80,000 fitness evaluations, with the standard deviation and average being 0.369562

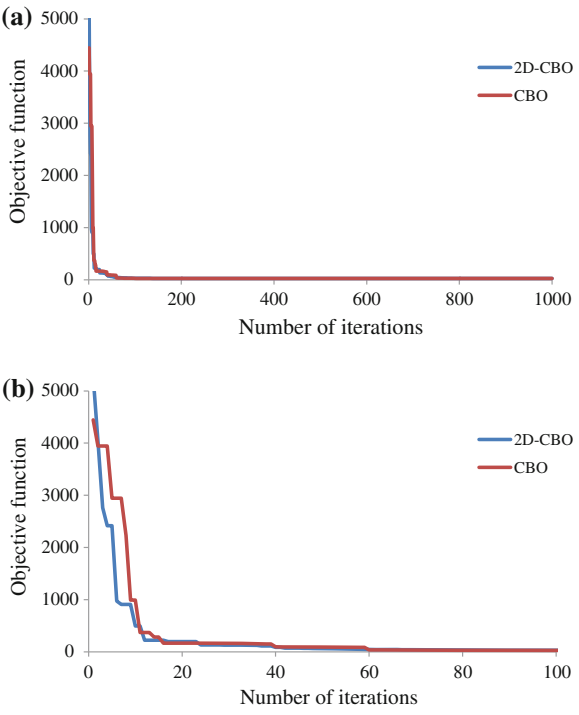

Fig. 2.11 Comparison of the convergence curves of the two algorithms for the constrained function II. **a** All iterations, **b** 1–100 iterations

and 24.69515, respectively. Although, the average result of the 2D-CBO is more than that of Deb [18]. The best results obtained using 2D-CBO with less population size is better than those of the CBO. Figure 2.11 shows the convergence curves for the CBO and 2D-CBO algorithms.

2.3.4 Discussions

In this part, a new optimization algorithm, named as 2D-CBO, is developed. The proposed algorithm is a method based on the reformation of the standard CBO algorithm. The CBO is recently developed meta-heuristic algorithm which mimics the laws of collision between objects which do not use a memory for saving the best-so-far solution in its formulation. In the 2D-CBO, bodies or agents move toward to the saved best particles and collide to them to promote the exploitation ability of the CBO.

A comparative study of 2D-CBO algorithm on two constrained functions is presented. The results of applying examples clearly indicate that 2D-CBO is competitive and even outperforms most of the standard algorithm. The results obtained by the 2D-CBO are lighter than those of other algorithms with same

population size on all examples. The comparison of the function evaluation shows that 2D-CBO requires less evaluation and population size than the standard CBO algorithm on some of the selected benchmark test problems. In both examples, the convergence rate of 2D-CBO also is higher than the CBO due to adding a memory in the formulation of the algorithm. It should be noted that the no internal parameter tuning is needed in the proposed algorithm.

References

1. Kaveh A, Mahdavai VR (2014) Colliding bodies optimization: a novel meta-heuristic method. Comput Struct 139:18–27
2. Kaveh A, Mahdavi VR (2015) Two-dimensional colliding bodies algorithm for optimal design of truss structures. Adv Eng Softw 83:70–79
3. Tolman RC (1979) The principles of statistical mechanics. Clarendon Press, Oxford (Reissued)
4. Kaveh A, Talatahari S (2010) A novel heuristic optimization method: charged system search. Acta Mech 213:267–289
5. Tsoulos IG (2008) Modifications of real code genetic algorithm for global optimization. Appl Math Comput 203:598–607
6. Deb K (1991) Optimal design of a welded beam via genetic algorithms. AIAA J 29:2013–2015
7. He Q, Wang L (2007) An effective co-evolutionary particle swarm optimization for constrained engineering design problem. Eng Appl Artif Intell 20:89–99
8. Montes EM, Coello CAC (2008) An empirical study about the usefulness of evolution strategies to solve constrained optimization problems. Int J Gen Syst 37:443–473
9. Ragsdell KM, Phillips DT (1976) Optimal design of a class of welded structures using geometric programming. ASME J Eng Ind Ser B 98:1021–1025
10. Coello CAC (2000) Use of a self-adaptive penalty approach for engineering optimization problems. Comput Ind 41:113–127
11. Coello CAC, Montes EM (1992) Constraint-handling in genetic algorithms through the use of dominance-based tournament. IEEE Trans Reliab 41:576–582
12. Belegundu AD (1982) A study of mathematical programming methods for structural optimization. PhD thesis, Department of Civil and Environmental Engineering, University of Iowa, Iowa, USA
13. Arora JS (1989) Introduction to optimum design. McGraw-Hill, New York
14. Holland JH (1975) Adaptation in natural and artificial systems. University of Michigan Press, Ann Arbor
15. Kaveh A (2014) Advance in metaheuristic algorithms for optimal design of structures. Springer, Switzerland
16. Saka MP (2014) Shape and topology optimization design of skeletal structures using metaheuristic algorithms: a review. Comput Technol Rev 9:31–68
17. Landau LD, Lifshitz EM (1976) Mechanics(course of theoretical physics), vol 224. Butterworth, London
18. Deb K (2000) An efficient constraint handling method for genetic algorithms. Comput Methods Appl Mech Eng 186:311–338
19. Lee KS, Geem ZW (2005) A new meta-heuristic algorithm for continuous engineering optimization: harmony search theory and practice. Comput Methods Appl Mech Eng 194:3902–3933
20. Sandgren E (1988) Nonlinear integer and discrete programming in mechanical design. In: Proceedings of the ASME design technology conference, Kissimine, FL, pp 95–105

21. Kannan BK, Kramer SN (1994) An augmented Lagrange multiplier based method for mixed integer discrete continuous optimization and its applications to mechanical design. Trans ASME J Mech Des 116:318–320
22. Deb K, Gene AS (1997) A robust optimal design technique for mechanical component design. In: Dasgupta D, Michalewicz Z (eds) Evolutionary algorithms in engineering applications. Springer, Berlin, pp 497–514

http://www.springer.com/978-3-319-19658-9

Colliding Bodies Optimization
Extensions and Applications
Kaveh, A.; Mahdavi, V.R.
2015, XI, 284 p. 162 illus., 106 illus. in color., Hardcover
ISBN: 978-3-319-19658-9