Computers and Structures 169 (2016) 1–12

Contents lists available at ScienceDirect

Computers and Structures

journal homepage: www.elsevier.com/locate/compstruc

A novel metaheuristic method for solving constrained engineering optimization problems: Crow search algorithm

Alireza Askarzadeh *

Department of Energy Management and Optimization, Institute of Science and High Technology and Environmental Sciences, Graduate University of Advanced Technology, Kerman, Iran

ARTICLE INFO

Article history:
Received 27 September 2015
Accepted 3 March 2016
Available online 23 March 2016

Keywords:
Metaheuristic optimization
Crow search algorithm
Constrained engineering optimization

ABSTRACT

This paper proposes a novel metaheuristic optimizer, named crow search algorithm (CSA), based on the intelligent behavior of crows. CSA is a population-based technique which works based on this idea that crows store their excess food in hiding places and retrieve it when the food is needed. CSA is applied to optimize six constrained engineering design problems which have different natures of objective functions, constraints and decision variables. The results obtained by CSA are compared with the results of various algorithms. Simulation results reveal that using CSA may lead to finding promising results compared to the other algorithms.

1. Introduction

Engineering design is defined as a decision making process to build products that satisfy specified needs. Most often, engineering design problems include complicated objective functions with a large number of decision variables. The feasible solutions are the set of all designs characterized by all possible values of the design parameters (decision variables). An optimization technique tries to find the optimal solution from all available feasible solutions.

Conventional search methods have long been applied to solve engineering design problems. Although these methods find promising results in many real problems, they may fail in more complex design problems. In real design problems, the number of decision variables can be very large and their effect on the objective function can be very complicated. The objective function may have many local optima, whereas the designer is interested in the global optimum. Such problems cannot be handled by conventional methods that only find local optima. In these cases, efficient optimization methods are needed.

Metaheuristic algorithms have shown promising performance for solving most real-world optimization problems that are extremely nonlinear and multimodal. All metaheuristic algorithms use a certain tradeoff of randomization and local search [1]. These algorithms can find good solutions for difficult optimization problems, but there is no guarantee that optimal solutions can be reached. It is hoped that these algorithms work most of the time, but not all the time. Metaheuristic algorithms could be suitable for global optimization [2]. Based on Glover's convention, all the modern nature-inspired methods are called metaheuristics [3].

Current trend is to utilize nature-inspired metaheuristic algorithms to tackle difficult problems and it has been shown that metaheuristics are surprisingly very efficient [1,2]. For this reason, the literature of metaheuristics has expanded tremendously in the last two decades [4,5]. Some of the well-known metaheuristic algorithms are as follows: genetic algorithm (GA) based on natural selection [6], particle swarm optimization (PSO) based on social behavior of bird flocking and fish schooling [7], harmony search (HS) based on music improvisation process [8], cuckoo search algorithm based on the brood parasitism of some cuckoo species [9], bat algorithm (BA) based on echolocation behavior of microbats [10], group search optimizer (GSO) based on animal searching behavior [11], firefly algorithm (FA) based on the flashing light patterns of tropic fireflies [12], etc. To date, researchers have only used a very limited characteristics inspired by nature and there is room for development of more algorithms. One of the main motivations of this paper is to develop a user-friendly (simple concept and easy implementation) metaheuristic technique by which we may obtain promising results when solving optimization problems.

Crows are widely distributed genus of birds which are now considered to be among the world's most intelligent animals [13,14]. As a group, crows show remarkable examples of intelligence and often score very highly on intelligence tests. They can memorize faces, use tools, communicate in sophisticated ways and hide and retrieve food across seasons [13,15].

* Tel./fax: +98 342 6233176.
E-mail addresses: a.askarzadeh@kgut.ac.ir, askarzadeh_a@yahoo.com

http://dx.doi.org/10.1016/j.compstruc.2016.03.001
0045-7949/

In a crow flock, there is a behavior which has many similarities with an optimization process. According to this behavior, crows hide their excess food in certain positions (hiding places) of the environment and retrieve the stored food when it is needed. Crows are greedy birds since they follow each other to obtain better food sources. Finding food source hidden by a crow is not an easy work since if a crow finds another one is following it, the crow tries to fool that crow by going to another position of the environment. From optimization point of view, the crows are searchers, the environment is search space, each position of the environment is corresponding to a feasible solution, the quality of food source is objective (fitness) function and the best food source of the environment is the global solution of the problem. Based on these similarities, CSA attempts to simulate the intelligent behavior of the crows to find the solution of optimization problems.

2. Crow search algorithm

Crows (crow family or corvids) are considered the most intelligent birds. They contain the largest brain relative to their body size. Based on a brain-to-body ratio, their brain is slightly lower than a human brain. Evidences of the cleverness of crows are plentiful. They have demonstrated self-awareness in mirror tests and have tool-making ability. Crows can remember faces and warn each other when an unfriendly one approaches. Moreover, they can use tools, communicate in sophisticated ways and recall their food's hiding place up to several months later [13–16].

Crows have been known to watch other birds, observe where the other birds hide their food, and steal it once the owner leaves. If a crow has committed thievery, it will take extra precautions such as moving hiding places to avoid being a future victim. In fact, they use their own experience of having been a thief to predict the behavior of a pilferer, and can determine the safest course to protect their caches from being pilfered [17].

In this paper, based on the above-mentioned intelligent behaviors, a population-based metaheuristic algorithm, CSA, is developed. The principles of CSA are listed as follows:

- Crows live in the form of flock.
- Crows memorize the position of their hiding places.
- Crows follow each other to do thievery.
- Crows protect their caches from being pilfered by a probability.

It is assumed that there is a d-dimensional environment including a number of crows. The number of crows (flock size) is N and the position of crow i at time (iteration) iter in the search space is specified by a vector $x^{i,\mathrm{iter}}$ $(i = 1, 2, \ldots, N;\ \mathrm{iter} = 1, 2, \ldots, \mathrm{iter}_{\max})$ where $x^{i,\mathrm{iter}} = \left[x_1^{i,\mathrm{iter}},\ x_2^{i,\mathrm{iter}},\ \ldots,\ x_d^{i,\mathrm{iter}}\right]$ and $\mathrm{iter}_{\max}$ is the maximum number of iterations. Each crow has a memory in which the position of its hiding place is memorized. At iteration iter, the position of hiding place of crow i is shown by $m^{i,\mathrm{iter}}$. This is the best position that crow i has obtained so far. Indeed, in memory of each crow the position of its best experience has been memorized. Crows move in the environment and search for better food sources (hiding places).

Assume that at iteration iter, crow j wants to visit its hiding place, $m^{j,\mathrm{iter}}$. At this iteration, crow i decides to follow crow j to approach to the hiding place of crow j. In this case, two states may happen:

State 1: Crow j does not know that crow i is following it. As a result, crow i will approach to the hiding place of crow j. In this case, the new position of crow i is obtained as follows:

$$x^{i,\mathrm{iter}+1} = x^{i,\mathrm{iter}} + r_i \times fl^{i,\mathrm{iter}} \times \left(m^{j,\mathrm{iter}} - x^{i,\mathrm{iter}}\right) \tag{1}$$

where r_i is a random number with uniform distribution between 0 and 1 and $fl^{i,\mathrm{iter}}$ denotes the flight length of crow i at iteration iter.

Fig. 1 shows the schematic of this state and the effect of fl on the search capability. Small values of fl leads to local search (at the vicinity of $x^{i,\mathrm{iter}}$) and large values results in global search (far from $x^{i,\mathrm{iter}}$). As Fig. 1(a) shows, if the value of fl is selected less than 1, the next position of crow i is on the dash line between $x^{i,\mathrm{iter}}$ and $m^{j,\mathrm{iter}}$. As Fig. 1(b) indicates, if the value of fl is selected more than 1, the next position of crow i is on the dash line which may exceed $m^{j,\mathrm{iter}}$.

State 2: Crow j knows that crow i is following it. As a result, in order to protect its cache from being pilfered, crow j will fool crow i by going to another position of the search space.

Totally, states 1 and 2 can be expressed as follows:

$$x^{i,\mathrm{iter}+1} = \begin{cases} x^{i,\mathrm{iter}} + r_i \times fl^{i,\mathrm{iter}} \times \left(m^{j,\mathrm{iter}} - x^{i,\mathrm{iter}}\right) & r_j \geqslant AP^{j,\mathrm{iter}} \\ \text{a random position} & \text{otherwise} \end{cases} \tag{2}$$

where r_j is a random number with uniform distribution between 0 and 1 and $AP^{j,\mathrm{iter}}$ denotes the awareness probability of crow j at iteration iter.

Metaheuristic algorithms should provide a good balance between diversification and intensification [2]. In CSA, intensification and diversification are mainly controlled by the parameter of awareness probability (AP). By decrease of the awareness probability value, CSA tends to conduct the search on a local region where a current good solution is found in this region. As a result, using small values of AP, increases intensification. On the other hand, by increase of the awareness probability value, the probability of searching the vicinity of current good solutions decreases and CSA tends to explore the search space on a global scale (randomization). As a result, use of large values of AP increases diversification.

3. CSA implementation for optimization

Pseudo code of CSA is shown in Fig. 2. The step-wise procedure for the implementation of CSA is given in this section.

Step 1: Initialize problem and adjustable parameters

The optimization problem, decision variables and constraints are defined. Then, the adjustable parameters of CSA (flock size (N), maximum number of iterations ($\mathrm{iter}_{\max}$), flight length (fl) and awareness probability (AP)) are valued.

Step 2: Initialize position and memory of crows

N crows are randomly positioned in a d-dimensional search space as the members of the flock. Each crow denotes a feasible solution of the problem and d is the number of decision variables.

$$\mathrm{Crows} = \begin{bmatrix} x_1^1 & x_2^1 & \ldots & x_d^1 \\ x_1^2 & x_2^2 & \ldots & x_d^2 \\ \vdots & \vdots & \vdots & \vdots \\ x_1^N & x_2^N & \ldots & x_d^N \end{bmatrix} \tag{3}$$

The memory of each crow is initialized. Since at the initial iteration, the crows have no experiences, it is assumed that they have hidden their foods at their initial positions.

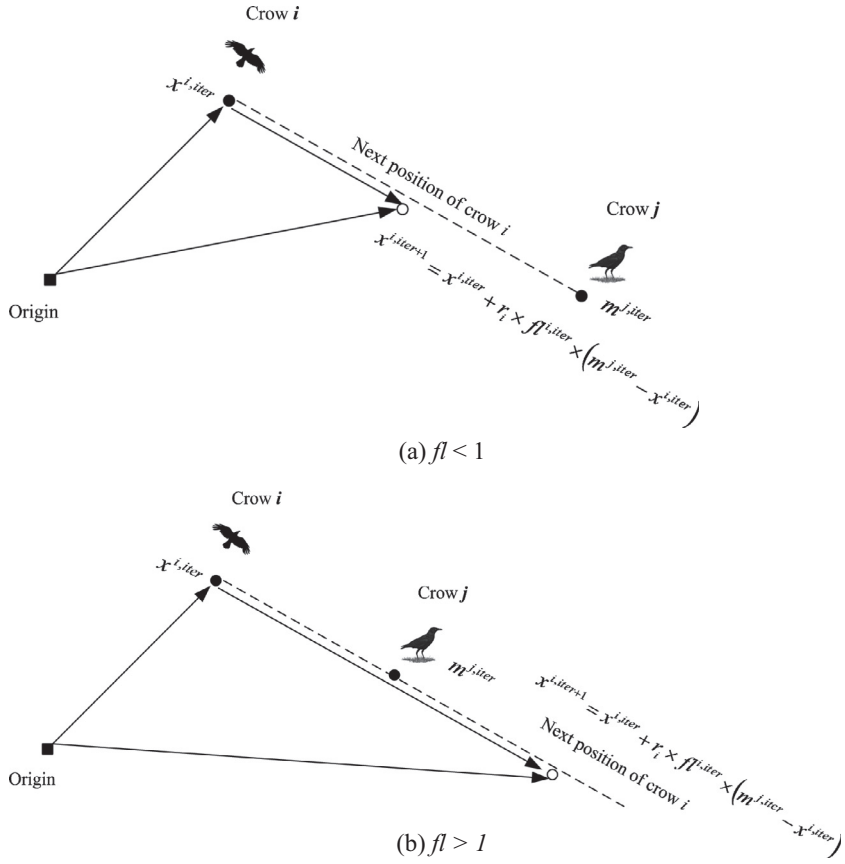

Fig. 1. Flowchart of state 1 in CSA (a) $fl < 1$ and (b) $fl > 1$. Crow i can go to every position on the dash line.

Crow search algorithm

Randomly initialize the position of a flock of N crows in the search space
Evaluate the position of the crows
Initialize the memory of each crow
while $iter < iter_{max}$
  for $i = 1 : N$ (*all N crows of the flock*)
    Randomly choose one of the crows to follow (*for example j*)
    Define an awareness probability
      if $r_j \geq AP^{j,iter}$
        $x^{i,iter+1} = x^{i,iter} + r_i \times fl^{i,iter} \times \left(m^{j,iter} - x^{i,iter}\right)$
      else
        $x^{i,iter+1} = a\ random\ position\ of\ search\ space$
      end if
  end for
  Check the feasibility of new positions
  Evaluate the new position of the crows
  Update the memory of crows
end while

Fig. 2. Pseudo code of the proposed CSA.

$$\text{Memory} = \begin{bmatrix} m_1^1 & m_2^1 & \dots & m_d^1 \\ m_1^2 & m_2^2 & \dots & m_d^2 \\ \vdots & \vdots & \vdots & \vdots \\ m_1^N & m_2^N & \dots & m_d^N \end{bmatrix} \quad (4)$$

Step 3: Evaluate fitness (objective) function

For each crow, the quality of its position is computed by inserting the decision variable values into the objective function.

Step 4: Generate new position

Crows generate new position in the search space as follows: suppose crow i wants to generate a new position. For this aim, this crow randomly selects one of the flock crows (for example crow j) and follows it to discover the position of the foods hidden by this crow (m^j). The new position of crow i is obtained by Eq. (2). This process is repeated for all the crows.

Step 5: Check the feasibility of new positions

The feasibility of the new position of each crow is checked. If the new position of a crow is feasible, the crow updates its position. Otherwise, the crow stays in the current position and does not move to the generated new position.

Step 6: Evaluate fitness function of new positions

The fitness function value for the new position of each crow is computed.

Step 7: Update memory

The crows update their memory as follows:

$$m^{i,\mathrm{iter}+1} = \begin{cases} x^{i,\mathrm{iter}+1} & f(x^{i,\mathrm{iter}+1}) \text{ is better than } f(m^{i,\mathrm{iter}}) \\ m^{i,\mathrm{iter}} & \text{o.w.} \end{cases} \quad (5)$$

where $f(\cdot)$ denotes the objective function value.

It is seen that if the fitness function value of the new position of a crow is better than the fitness function value of the memorized position, the crow updates its memory by the new position.

Step 8: Check termination criterion

Steps 4–7 are repeated until $\mathrm{iter}_{\max}$ is reached. When the termination criterion is met, the best position of the memory in terms of the objective function value is reported as the solution of the optimization problem.

4. Comparison of CSA with GA, PSO and HS

Like the other well-known algorithms such as GA, PSO and HS, CSA makes use of a population of seekers to explore the search space. By use of a population the probability of finding a good solution and escaping from local optima increases. In addition to population size and maximum number of iterations (generations), optimization algorithms have some other parameters which should be adjusted. Parameter setting is one of the drawbacks of optimization algorithms since it is a time-consuming work. Algorithms which have fewer parameters to adjust are easier to implement. In CSA, flight length and awareness probability should be tuned (2 parameters). In PSO algorithm the adjustable parameters are inertia weight, maximum value of velocity, individual learning factor and social learning factor (4 parameters). HS requires the value of harmony memory considering rate, pitch adjusting rate and bandwidth of generation (3 parameters). In GA, selection method, crossover method, crossover probability, mutation method, mutation probability and replacement method should be determined (6 parameters).

Like GA and PSO, CSA is not a greedy algorithm since if a crow generates a new position which is not better than its current position, it will move to the new position. Non-greedy algorithms can increase the diversity of generated solutions. In HS, a new solution is accepted if its fitness value is better than the fitness of the worst harmony of memory.

Like HS and PSO, CSA includes memory in which good solutions are memorized. In PSO, each particle is attracted towards the best position ever found by itself and the best position ever found by the group. As a result, at each iteration, the best solutions found so far are directly used. At each iteration of CSA, each crow selects randomly one of the flock crows (it may be itself) and moves towards its hiding place (the best solution found by that crow). This means that at each iteration of CSA, the best positions found so far are directly used to find better positions.

5. Numerical examples

It has been proved that under certain assumptions, no single search algorithm is the best on average for all problems [18,19]. In other words, an algorithm might solve some problems better and some problems worse than the other algorithms. In order to evaluate the optimization power of the proposed CSA without a biased conclusion, six engineering design problems are considered and solved, including three-bar truss, pressure vessel, tension/compression spring, welded beam, gear train and Belleville spring. All the considered problems have different natures of objective functions, constraints and decision variables. CSA has been executed in the MATLAB environment on a PC with Pentium 4 CPU 2.1 G 2 GB RAM. Fig. 3 shows the flowchart of CSA implementation. Table 1 shows the parameter setting of CSA for solving these problems. It is worthwhile to mention that no attempt has made to optimize the parameter setting of CSA.

In this paper, for each problem the constraints are directly handled. It means that each solution that cannot satisfy the constraints altogether, will be considered as infeasible and abandoned. However, the rejection of the infeasible solutions may have serious disadvantages for problems which include many design variables and/or for problems whose design space is heavily dominated by constraints since generating a feasible design in such problems may take an excessive number of successive trials. One of the ways to handle constraints of a constrained optimization problem is using penalty function. By use of penalty function, a constrained optimization problem will be converted to an unconstrained one. Since the results obtained by CSA are compared with the results reported in the literature and most methods have used direct control of constraints, the penalty method has not been used here.

5.1. Engineering optimization problems

5.1.1. Three-bar truss design problem

The objective of this problem is to minimize the volume of a statistically loaded three-bar truss subject to stress (σ) constraints on each of the truss members by adjusting cross sectional areas (x_1 and x_2). Fig. 4 represents the schematic of three-bar truss design problem. This optimization problem has a nonlinear fitness function with three nonlinear inequality constraints and two continuous decision variables as follows:

$$\begin{aligned} &\text{Min. } f(x) = (2\sqrt{2}x_1 + x_2) \times l \\ &\text{S.t.} \\ &g_1(x) = \frac{\sqrt{2}x_1 + x_2}{\sqrt{2}x_1^2 + 2x_1x_2} P - \sigma \leqslant 0 \\ &g_2(x) = \frac{x_2}{\sqrt{2}x_1^2 + 2x_1x_2} P - \sigma \leqslant 0 \\ &g_3(x) = \frac{1}{\sqrt{2}x_2 + x_1} P - \sigma \leqslant 0 \\ &0 \leqslant x_i \leqslant 1, \quad i = 1,2 \\ &l = 100 \text{ cm}, P = 2 \text{ kN/cm}^2, \ \sigma = 2 \text{ kN/cm}^2 \end{aligned} \quad (6)$$

Table 2 represents the best solution obtained by CSA for three-bar truss design problem over 50 independent runs. Table 3 shows the comparison of the statistical results obtained by CSA and those obtained by society and civilization (SC) algorithm [20], hybridizing PSO with differential evolution (PSO-DE) [21], dynamic stochastic selection with multimember differential evolution (DSS-MDE) [22] and mine blast algorithm (MBA) [23]. As the results show, CSA produces promising results in comparison with the other methods on the three-bar truss design problem. In terms of the best index, CSA produces better results than SC and MBA. In this case, the results produced by CSA are same as the results

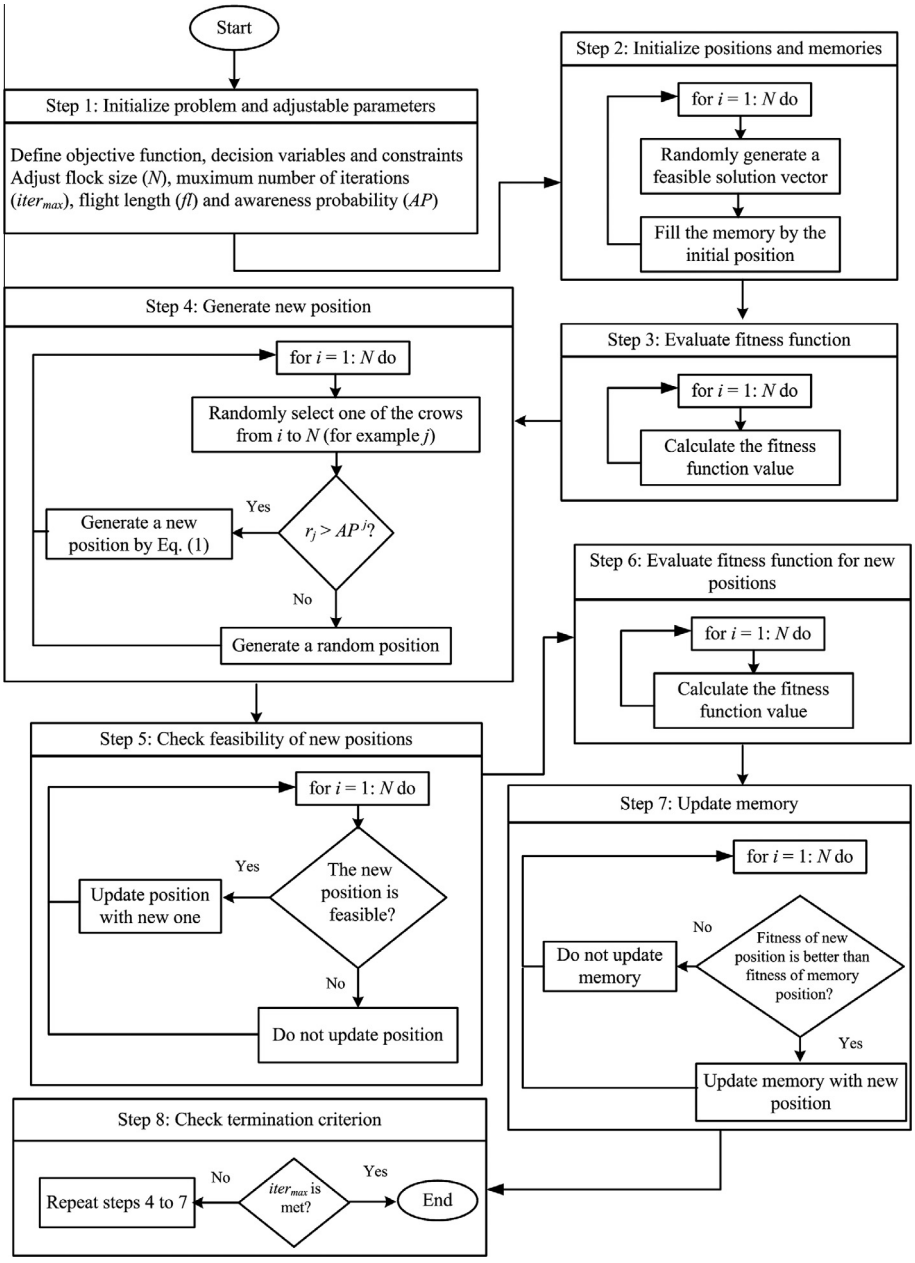

Fig. 3. Flowchart of CSA for doing optimization.

Table 1
Parameter setting of CSA for solving the design problems.

Design problem	*N*	$iter_{max}$	*fl*	*AP*
Three-bar truss	50	500	2	0.1
Pressure vessel	50	5000	2	0.1
Tension/compression spring	50	1000	2	0.1
Welded beam	50	2000	2	0.1
Gear train	20	500	2	0.1
Belleville spring	50	1000	2	0.1

obtained by PSO-DE and DSS-MDE. In terms of the worst and mean indices, the results obtained by CSA are promising. The minimal value of Std. index denotes the high robustness of CSA.

Fig. 5 illustrates the convergence rate of the CSA for finding the best solution of the three-bar truss design problem. It can be seen the convergence rate of CSA is good since this algorithm finds rapidly a good region in less than 50 iterations.

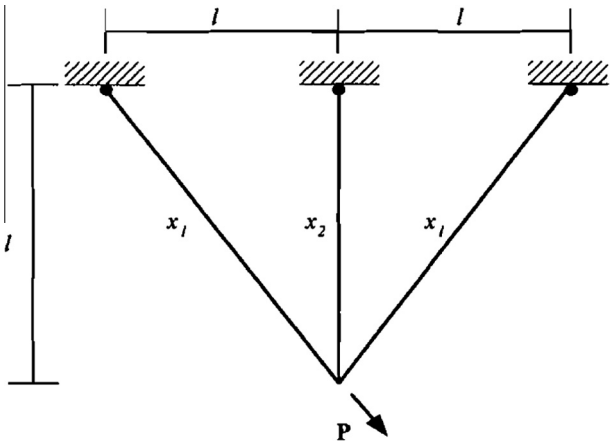

Fig. 4. Schematic of three-bar truss design problem.

Table 2
The best solution obtained by CSA for three-bar truss design problem.

Parameter	x_1	x_2	f
Value	0.7886751284	0.4082483080	263.8958433765
Parameter	g_1	g_2	g_3
Value	−1.687539e−14	−1.4641015952	−0.5358984048

5.1.2. Pressure vessel design problem

In this design problem, the goal is to minimize the total cost of a pressure vessel including material, forming and welding costs. As Fig. 6 shows, this optimization problem consists of four decision variables: thickness of the shell (x_1 or T_s), thickness of the head (x_2 or T_h), inner radius (x_3 or R) and length of the cylindrical section of the vessel (x_4 or L). Among the four decision variables, x_1 and x_2 are discrete (integer multiplies of 0.0625 in) and x_3 and x_4 are continuous. The pressure vessel design problem has a nonlinear fitness function with linear and one nonlinear inequality constraints as follows:

$$\begin{aligned}
&\text{Min. } f(x) = 0.6224x_1x_3x_4 + 1.7781x_2x_3^2 + 3.1661x_1^2x_4 + 19.84x_1^2x_3\\
&\text{S.t.}\\
&g_1(x) = -x_1 + 0.0193x_3 \leqslant 0\\
&g_2(x) = -x_2 + 0.00954x_3 \leqslant 0\\
&g_3(x) = -\pi x_3^2x_4 - \tfrac{4}{3}\pi x_3^3 + 1,296,000 \leqslant 0\\
&g_4(x) = x_4 - 240 \leqslant 0\\
&0 \leqslant x_i \leqslant 100, \quad i = 1,2\\
&10 \leqslant x_i \leqslant 200, \quad i = 3,4
\end{aligned} \tag{7}$$

Table 4 indicates the performance of the CSA on this problem. This table shows the optimal values of the decision variables and the constraint values corresponding to the best solution obtained by CSA over 50 independent runs. It can be seen that x_1 and x_2 are integer multiplies of 0.0625 and x_3 and x_4 are in the possible range. Table 5 compares the statistical results obtained by CSA and those obtained by the other methods of the literature which have reported feasible solutions (especially for x_1 and x_2), namely, GA based on co-evolution model (GA3) [24], GA based on dominance-based tour tournament selection (GA4) [25], co-evolutionary PSO (CPSO) [26], hybrid PSO (HPSO) [27], PSO [28], quantum-behaved PSO (QPSO) [28], QPSO combined with mutation operator (G-QPSO) [28], $(\mu + \lambda)$ evolution strategy ($(\mu + \lambda)$-ES) [29], unified PSO (UPSO) [30], co-evolutionary differential evolution (CDE) [31], hybridization of PSO and DE (PSO-DE) [21], artificial bee colony (ABC) [32] and teaching–learning-based optimization (TLBO) [33]. As can be seen, in terms of the Best index, CSA outperforms GA3, GA4, CPSO, G-QPSO, QPSO, PSO, CDE, UPSO and ABC. In this case, the results produced by CSA are same as the results obtained by PSO-DE and CSA is outperformed by HPSO, $(\mu + \lambda)$-ES and TLBO. However, the values found by HPSO and TLBO are nearly same as that found by CSA.

Fig. 7 illustrates the convergence rate of the CSA for finding the best solution of the pressure vessel problem. This figure shows the value of best-so-far at each iteration. It can be seen that the convergence rate of CSA is good.

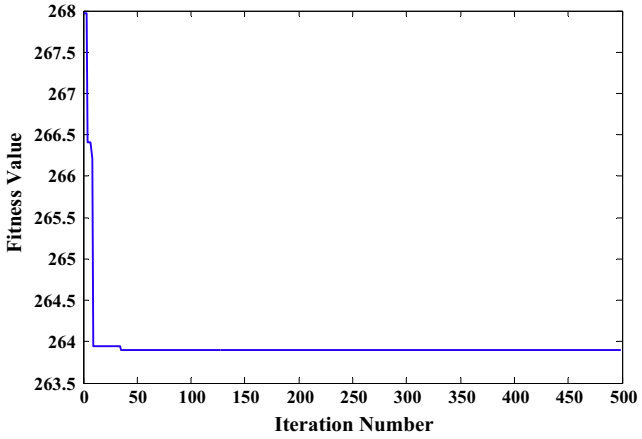

Fig. 5. Convergence rate of CSA for finding the best solution of three-bar truss design problem.

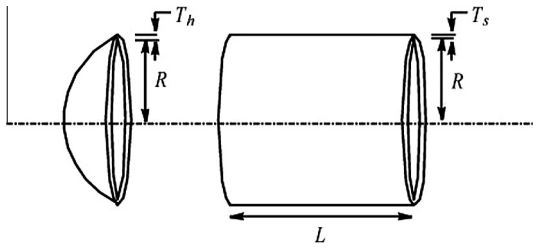

Fig. 6. Schematic of the pressure vessel design problem.

5.1.3. Tension/compression spring design problem

As Fig. 8 indicates, the objective of this problem is to minimize the weight of a tension/compression spring with respect to one linear and three nonlinear inequality constraints. This problem consists of three continuous decision variables, namely, wire diameter (d or x_1), mean coil diameter (D or x_2) and number of active coils (P or x_3).

$$\begin{aligned}
&\text{Min. } f(x) = (x_3 + 2)x_2x_1^2\\
&\text{S.t.}\\
&g_1(x) = 1 - \frac{x_2^3x_3}{71,785x_1^4} \leqslant 0\\
&g_2(x) = \frac{4x_2^2 - x_1x_2}{12,566\left(x_2x_1^3 - x_1^4\right)} + \frac{1}{5108x_1^2} - 1 \leqslant 0\\
&g_3(x) = 1 - \frac{140.45x_1}{x_2^2x_3} \leqslant 0\\
&g_4(x) = \frac{x_1 + x_2}{1.5} - 1 \leqslant 0\\
&0.05 \leqslant x_1 \leqslant 2,\ 0.25 \leqslant x_2 \leqslant 1.3,\ 2 \leqslant x_3 \leqslant 15,
\end{aligned} \tag{8}$$

Table 6 shows the best solution and the values of the constraints obtained by CSA for this design problem over 50 independent runs. Table 7 shows the comparison of the statistical results obtained by CSA and those obtained by the other algorithms. In terms of the best index, CSA outperforms GA3 [24], GA4 [25], CPSO [26], QPSO [28], PSO [28], SC [20], UPSO [30] and $(\mu + \lambda)$-ES [29]. In this case, the results produced by CSA are same as those obtained

Table 3
Comparison of statistical results obtained by CSA and other algorithms for three-bar truss design problem (50 runs).

Algorithm	Worst	Mean	Best	Std.
SC	263.969756	263.903356	263.895846	1.3e−2
PSO-DE	263.895843	263.895843	263.895843	4.5e−10
DSS-MDE	263.895849	263.895843	263.895843	9.72e−7
MBA	263.915983	263.897996	263.895852	3.93e−3
CSA	263.8958433770	263.8958433765	263.8958433765	1.0122543402e−10

Table 4
The best solution obtained by CSA for pressure vessel problem.

Parameter	x_1	x_2	x_3	x_4
Value	0.812500 (13 × 0.0625)	0.437500 (7 × 0.0625)	42.09844539	176.63659855
Parameter	g_1	g_2	g_3	g_4	f
Value	−4.02409828e−9	−0.03588083	−7.12266192e−4	−63.36340145	6059.71436343

Table 5
Comparison of statistical results obtained by CSA and other algorithms for pressure vessel problem (50 runs). N.A. means not available.

Algorithm	Worst	Mean	Best	Std.
GA3	6308.4970	6293.8432	6288.7445	7.4133
GA4	6469.3220	6177.2533	6059.9463	130.9297
CPSO	6363.8041	6147.1332	6061.0777	86.45
HPSO	6288.6770	6099.9323	6059.7143	86.20
G-QPSO	7544.4925	6440.3786	6059.7208	448.4711
QPSO	8017.2816	6440.3786	6059.7209	479.2671
PSO	14076.3240	8756.6803	6693.7212	1492.5670
CDE	6371.0455	6085.2303	6059.7340	43.0130
UPSO	9387.77	8016.37	6154.70	745.869
PSO-DE	N.A.	6059.714	6059.714	N.A.
ABC	N.A.	6245.308144	6059.714736	205
$(\mu + \lambda)$-ES	N.A.	6379.938037	6059.701610	210
TLBO	N.A.	6059.71434	6059.714335	N.A.
CSA	7332.84162110	6342.49910551	6059.71436343	384.94541634

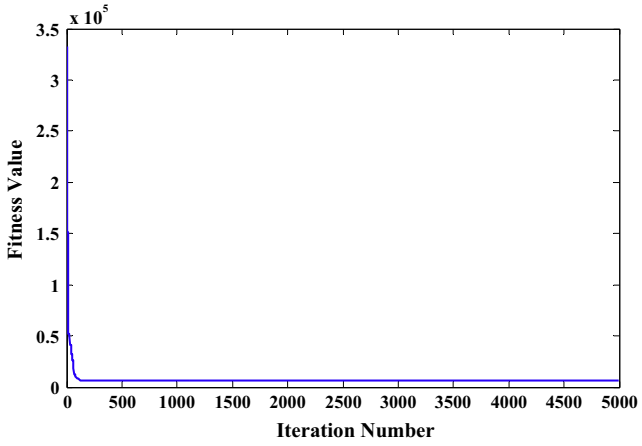

Fig. 7. Convergence rate of CSA for finding the best solution of pressure vessel problem.

Fig. 8. Schematic of tension/compression spring design problem.

by HPSO [27], G-QPSO [28], DSS-MDE [22], PSO-DE [21], ABC [32], TLBO [33] and MBA [23]. In terms of the Mean index, CSA produces better results than all the other algorithms except PSO-DE and TLBO. Based on the Std. index, after PSO-DE, CSA has the most robustness. Fig. 9 illustrates the convergence rate of the CSA for finding the best solution of the tension/compression spring design problem.

5.1.4. Welded beam design problem

The objective of this design problem is to minimize the cost of a welded beam. There are four continuous design variables with two linear and five nonlinear inequality constraints based on shear stress (τ), bending stress in the beam (σ), bucking load on the bar (P_b), end deflection of the beam (δ) and side constraints. Fig. 10 illustrates the schematic of welded beam design problem. Four decision variables of this problem are x_1 or h, x_2 or l, x_3 or t and x_4 or b.

$$\text{Min. } f(x) = 1.10471x_1^2x_2 + 0.04811x_3x_4(14 + x_2)$$

S.t.

$$g_1(x) = \tau(x) - \tau_{\max} \leqslant 0$$
$$g_2(x) = \sigma(x) - \sigma_{\max} \leqslant 0, \quad g_3(x) = x_1 - x_4 \leqslant 0$$
$$g_4(x) = 0.10471x_1^2 + 0.04811x_3x_4(14 + x_2) - 5 \leqslant 0$$
$$g_5(x) = 0.125 - x_1 \leqslant 0$$
$$g_6(x) = \delta(x) - \delta_{\max} \leqslant 0, \quad g_7(x) = P - P_c(x) \leqslant 0$$

where

Table 6
The best solution obtained by CSA for tension/compression spring problem.

Parameter	x_1	x_2	x_3	g_1
Value	0.0516890284	0.3567169544	11.2890117993	−4.44089210e−16
Parameter	g_2	g_3	g_4	f
Value	−4.10782519e−15	−4.05378408	−0.72772934	0.0126652328

Table 7
Comparison of statistical results obtained by CSA and other algorithms for tension/compression spring problem (50 runs). N.A. means not available.

Algorithm	Worst	Mean	Best	Std.
GA3	0.0128220	0.0127690	0.0127048	3.94e−5
GA4	0.0129730	0.0127420	0.0126810	5.90e−5
CPSO	0.0129240	0.0127300	0.0126747	5.20e−4
HPSO	0.0127190	0.0127072	0.0126652	1.58e−5
G-QPSO	0.017759	0.013524	0.012665	0.001268
QPSO	0.018127	0.013854	0.012669	0.001341
PSO	0.071802	0.019555	0.012857	0.011662
DSS-MDE	0.012738262	0.012669366	0.012665233	1.25e−5
PSO-DE	0.012665304	0.012665244	0.012665233	1.2e−8
SC	0.016717272	0.012922669	0.012669249	5.9e−4
UPSO	N.A.	0.02294	0.01312	7.2e−3
$(\mu + \lambda)$-ES	N.A.	0.013165	0.012689	3.9e−4
ABC	N.A.	0.012709	0.012665	0.012813
TLBO	N.A.	0.01266576	0.012665	N.A.
MBA	0.012900	0.012713	0.012665	6.3e−5
CSA	0.0126701816	0.0126659984	0.0126652328	1.357079e−6

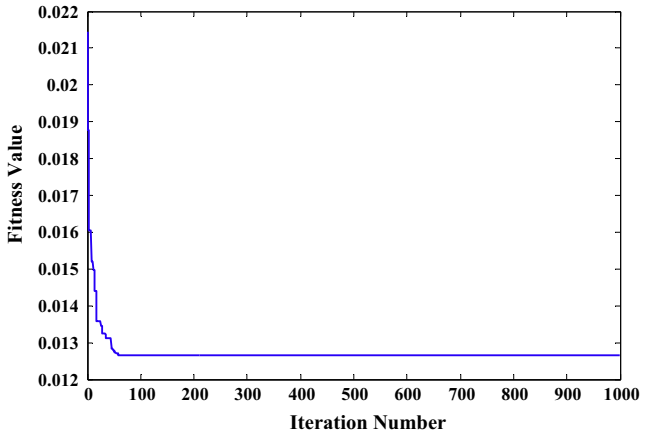

Fig. 9. Convergence rate of CSA for finding the best solution of tension/compression spring problem.

$$
\begin{aligned}
&\tau(x) = \sqrt{(\tau')^2 + 2\tau'\tau''\frac{x_2}{2R} + (\tau'')^2}\\
&\tau' = \frac{P}{\sqrt{2}x_1x_2}, \quad \tau'' = \frac{MR}{J}, \quad M = P\left(L + \frac{x_2}{2}\right)\\
&R = \sqrt{\frac{x_2^2}{4} + \left(\frac{x_1+x_3}{2}\right)^2}, \quad \delta(x) = \frac{4PL^3}{Ex_3^3x_4}\\
&J = 2\left[\sqrt{2}x_1x_2\left\{\frac{x_2^2}{12} + \left(\frac{x_1+x_3}{2}\right)^2\right\}\right], \quad \sigma(x) = \frac{6PL}{x_4x_3^2}\\
&P_c(x) = \frac{4.013E\sqrt{\frac{x_3^2x_4^6}{36}}}{L^2}\left(1 - \frac{x_3}{2L}\sqrt{\frac{E}{4G}}\right)\\
&P = 6000 \text{ lb}, L = 14 \text{ in}, \quad E = 30e6 \text{ psi}\\
&G = 12e6 \text{ psi}, \tau_{\max} = 13,600 \text{ psi}, \sigma_{\max} = 30,000 \text{ psi}\\
&\delta_{\max} = 0.25 \text{ in}, \quad 0.1 \leqslant x_1 \leqslant 2, 0.1 \leqslant x_2 \leqslant 10\\
&0.1 \leqslant x_3 \leqslant 10, 0.1 \leqslant x_4 \leqslant 2
\end{aligned}
\tag{9}
$$

Table 8 shows the best solution and the corresponding constraint values obtained by CSA for this design problem over 50 independent runs. Table 9 shows the comparison of the statistical results obtained by CSA and those obtained by the other algorithms. From the results, it is observed that in terms of the Best index, CSA produces better results than GA3 [24], GA4 [25], CPSO [26], SC [20], UPSO [30], CDE [31] and MBA [23]. The results obtained by CSA are same as those found by HPSO [27], PSO-DE [21], $(\mu + \lambda)$-ES [29], ABC [32] and TLBO [33]. In terms of the mean index, PSO-DE and CSA outperforms the other algorithms. The Std. value found by CSA reveals the robustness of this algorithm. Fig. 11 shows the convergence rate of the CSA for finding the best solution of the tension/compression spring design problem.

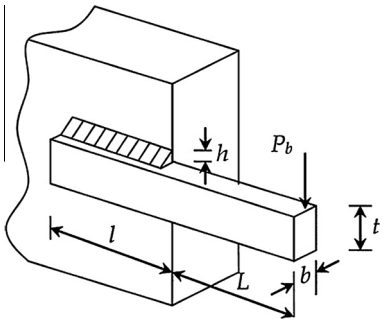

Fig. 10. Schematic of welded beam design problem.

5.1.5. Gear train design problem

The objective of this design problem is to minimize the cost of the gear ratio of the gear train shown in Fig. 12. This problem has only boundary constraints in the parameters. The decision variables are in discrete form since each gear has to have an integral number of teeth. Handling with discrete variables may increase the complexity of the problem. The decision variables are n_A (x_1), n_B (x_2), n_D (x_3) and n_F (x_4) and the gear ration is defined as $n_B\, n_D/n_F\, n_A$.

$$
\begin{aligned}
&\text{Min. } f(x) = \left((1/6.931) - (x_3x_2/x_1x_4)\right)^2\\
&\text{S.t.}\\
&12 \leqslant x_i \leqslant 60
\end{aligned}
\tag{10}
$$

Table 10 shows the best solution found by CSA for gear train design problem. Table 11 compares the statistical results obtained by CSA and those found by UPSO [30], ABC [32] and MBA [23]. In terms of the Best index, all the algorithms have generated same results. In terms of the mean index, CSA outperforms UPSO and MBA and is outperformed by ABC. Fig. 13 shows the convergence rate of the CSA for finding the best solution of the tension/compression spring design problem.

5.1.6. Belleville spring design problem

The objective of Belleville spring design problem is to minimize the weight of a Belleville spring subject to a number of constraints. As Fig. 14 shows, this problem has four decision variables including external diameter of the spring (De), internal diameter of the spring (Di), thickness of the spring (t) and the height of the spring

Table 8
The best solution obtained by CSA for welded beam problem.

Parameter	x_1	x_2	x_3	x_4	g_1	g_2
Value	0.2057296398	3.4704886656	9.0366239104	0.2057296398	0	0
Parameter	g_3	g_4	g_5	g_6	g_7	f
Value	0	−3.43298379	−0.08072964	−0.23554032	−3.63797881	1.7248523086

Table 9
Comparison of statistical results obtained by CSA and other algorithms for welded beam problem (50 runs).

Algorithm	Worst	Mean	Best	Std.
GA3	1.785835	1.771973	1.748309	1.12e−2
GA4	1.993408	1.792654	1.728226	7.47e−2
CPSO	1.782143	1.748831	1.728024	1.29e−2
HPSO	1.814295	1.749040	1.724852	4.01e−2
PSO-DE	1.724852	1.724852	1.724852	6.7e−16
SC	6.3996785	3.0025883	2.3854347	9.6e−1
UPSO	N.A.	2.83721	1.92199	0.683
CDE	N.A.	1.76815	1.73346	N.A.
$(\mu + \lambda)$-ES	N.A.	1.777692	1.724852	8.8e−2
ABC	N.A.	1.741913	1.724852	3.1e−2
TLBO	N.A.	1.72844676	1.724852	N.A.
MBA	1.724853	1.724853	1.724853	6.94e−19
CSA	1.7248523086	1.7248523086	1.7248523086	1.19450917e−15

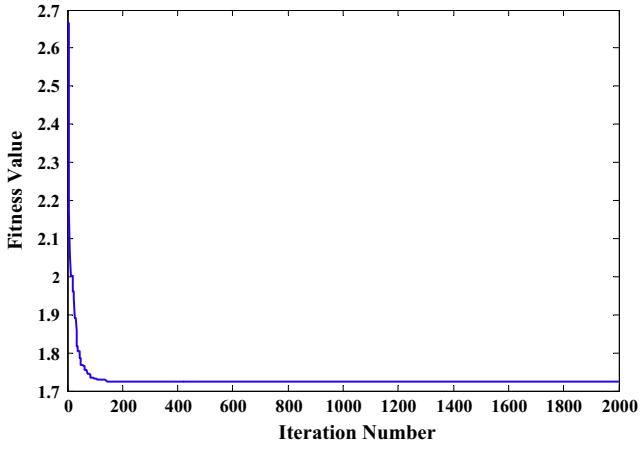

Fig. 11. Convergence rate of CSA for finding the best solution of welded beam problem.

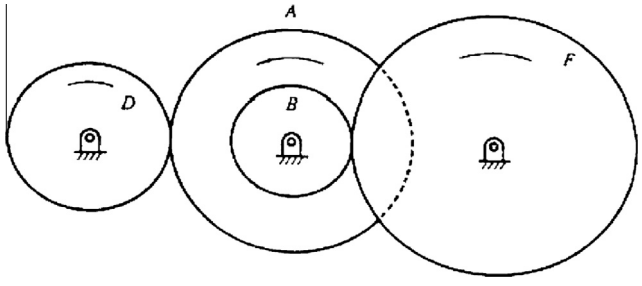

Fig. 12. Schematic of gear train design problem.

(h). The constraints are for compressive stress, deflection, height to deflection, height to maximum height, outer diameter, inner diameter and slope.

Table 10
The best solution obtained by CSA for gear train problem.

Parameter	x_1	x_2	x_3	x_4	f
Value	49	19	16	43	2.7008571489e−12

$$
\begin{aligned}
&\text{Min. } f(x) = 0.07075\pi(D_e^2 - D_i^2)t \\
&\text{S.t.} \\
&g_1(x) = S - \tfrac{4E\delta_{\max}}{(1-\mu^2)\alpha D_e^2}\left[\beta\left(h - \tfrac{\delta_{\max}}{2}\right) + \gamma t\right] \geqslant 0 \\
&g_2(x) = \left(\tfrac{4E\delta_{\max}}{(1-\mu^2)\alpha D_e^2}\left[\left(h - \tfrac{\delta}{2}\right)(h-\delta)t + t^3\right]\right)_{\delta=\delta_{\max}} - P_{\max} \geqslant 0 \\
&g_3(x) = \delta_1 - \delta_{\max} \geqslant 0 \\
&g_4(x) = H - h - t \geqslant 0 \\
&g_5(x) = D_{\max} - D_e \geqslant 0 \\
&g_6(x) = D_e - D_i \geqslant 0 \\
&g_7(x) = 0.3 - \tfrac{h}{D_e - D_i} \geqslant 0
\end{aligned}
\tag{11}
$$

where

$$
\begin{aligned}
&\alpha = \tfrac{6}{\pi \ln K}\left(\tfrac{K-1}{K}\right)^2,\ \beta = \tfrac{6}{\pi \ln K}\left(\tfrac{K-1}{\ln K} - 1\right) \\
&\gamma = \tfrac{6}{\pi \ln K}\left(\tfrac{K-1}{2}\right), P_{\max} = 5400 \text{ lb},\ E = 30e6 \text{ psi} \\
&\delta_{\max} = 0.2 \text{ in},\ \mu = 0.3,\ S = 200 \text{ KPsi},\ H = 2 \text{ in} \\
&D_{\max} = 12.01 \text{ in},\ K = D_e/D_i,\ \delta_1 = f(a)a, a = h/t
\end{aligned}
$$

Table 12 indicates the value of $f(a)$ with a. Table 13 shows the best solution found by CSA for the Belleville spring design problem. This table shows the optimal values of the decision variables and the values of the constraints corresponding to the best solution. Table 14 represents the comparison of the CSA performance with the other algorithms. It is seen that in terms of the best index, the results obtained by CSA are same as the results found by ABC [32], TLBO [33] and MBA [23]. In terms of the mean and Std. indices, CSA produces more promising results than the other algorithms. Fig. 15 shows the convergence rate of the CSA for finding the best solution of this design problem.

Table 15 represents the computational cost of CSA for finding the solution of each design problem over 50 runs in second (s). This table shows the minimum, average and maximum values of the

Table 11
Comparison of statistical results obtained by CSA and other algorithms for gear train problem (50 runs).

Algorithm	Worst	Mean	Best	Std.
UPSO	N.A.	3.80562e−8	2.700857e−12	1.09e−7
ABC	N.A.	3.641339e−10	2.700857e−12	5.52e−10
MBA	2.06290e−8	2.471635e−9	2.700857e−12	3.94e−9
CSA	3.1847379289e−8	2.0593270182e−9	2.70085714889e−12	5.059779e−9

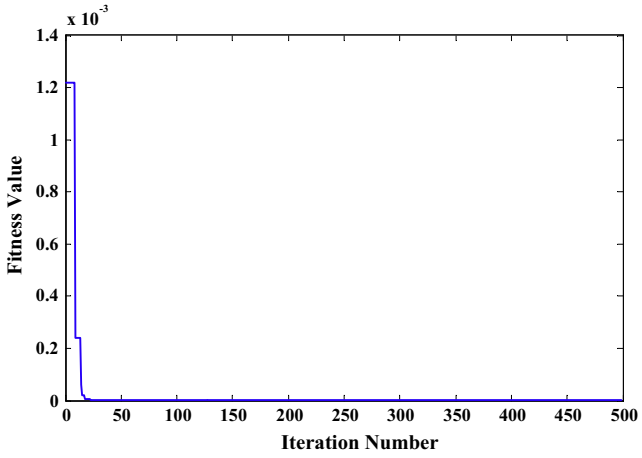

Fig. 13. Convergence rate of CSA for finding the best solution of gear train problem.

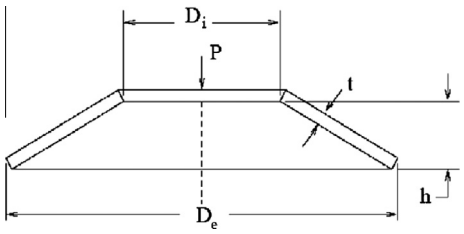

Fig. 14. Schematic of Belleville spring design problem.

runs for each design problem. On average, it is observed that CSA converges to the optimal solution less than 1 s.

5.2. Benchmark functions

Although the results obtained for the engineering problems prove that CSA shows a competitive performance with the existing methods in the literature, but still, there exists a question regarding the performance of CSA in larger-scale problems. In order to evaluate the performance of CSA on larger-scale optimization problems, five well-known benchmark functions shown in Table 16 are solved in 10 dimensions. In CSA, as before, *AP* and *fl* values have been set to 0.1 and 2, respectively. In PSO, the velocity is controlled and the learning factors (individual and social) are set to 2.

Table 14
Comparison of statistical results obtained by CSA and other algorithms for Belleville spring problem (50 runs).

Algorithm	Worst	Mean	Best	Std.
ABC	2.104297	1.995475	1.979675	0.07
TLBO	1.979757	1.97968745	1.979675	0.45
MBA	2.005431	1.984698	1.9796747	7.78e−3
CSA	1.97984321	1.97968106	1.9796747571	2.43810425e−5

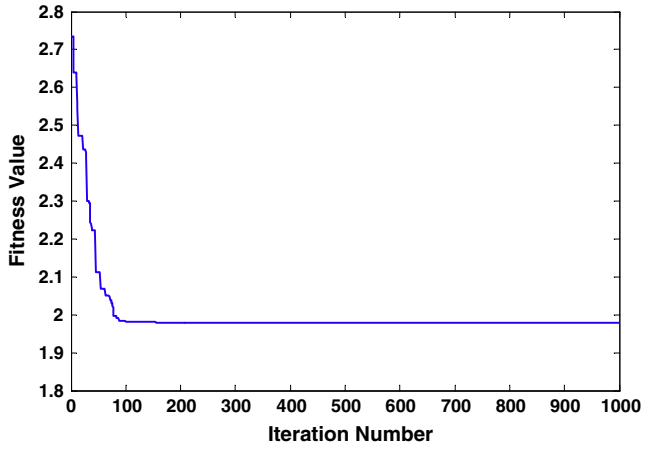

Fig. 15. Convergence rate of CSA for finding the best solution of Belleville spring problem.

Table 12
Variation of $f(a)$ with a.

a	$\leqslant 1.4$	1.5	1.6	1.7	1.8	1.9	2	2.1	2.2	2.3	2.4	2.5	2.6	2.7	$\geqslant 2.8$
$f(a)$	1	0.85	0.77	0.71	0.66	0.63	0.6	0.58	0.56	0.55	0.53	0.52	0.51	0.51	0.5

Table 13
The best solution obtained by CSA for Belleville spring problem.

Parameter	D_e	D_i	t	h	f
Value	12.0099999994	10.0304732892	0.2041433542	0.2000000000	1.9796747571
Parameter	g_1	g_2	g_3	g_4	g_5
Value	1.43883517e−6	2.8230715543e−9	0.7797037028	1.5958566458	6.02041083e−10
Parameter	g_6				g_7
Value	1.9795267102				0.1989657482

Table 15
Time consumed by CSA for finding the solution of the design problems (50 runs).

Design problem	Minimum (s)	Average (s)	Maximum (s)
Three-bar truss	0.06	0.09	0.11
Pressure vessel	0.58	0.62	0.73
Tension/compression spring	0.20	0.24	0.28
Welded beam	0.45	0.48	0.58
Gear train	0.02	0.04	0.08
Belleville spring	0.64	0.78	1.54

Moreover, the inertia weight decreases linearly from 0.9 to 0.4 during iterations. In GA, convex crossover (with the coefficients of 0.25 and 0.75), uniform mutation and tournament selection are used. Crossover and mutation probabilities are set to 0.9 and 0.005, respectively. In CSA, PSO and GA, population size is set to 20 and maximum number of iterations is selected 2000. As a result, in the investigations, number of fitness evaluations (NFEs) is 40,000.

Table 17 shows the results obtained by CSA in comparison with the results found by PSO and GA over 30 independent runs. It is seen, on all the functions, CSA outperforms the other algorithms in terms of the best index. In this table, the average time of the runs is shown. It is seen that CSA consumes less computational time than PSO and GA over same number of fitness evaluations.

As another investigation, the effect of different parameter setting of CSA is investigated on its performance. Tables 18 and 19 indicate the effect of different *AP* and *fl* values on the obtained results for f_1, f_2 and f_3. From Table 18, it is seen that $AP = 0$ leads to weak performance of CSA since the diversification ability of

Table 16
Test functions used in this study, their search range and the optimal value.

Test function	Search space	Optimal value
$f_1(x) = \sum_{i=1}^{n} x_i^2$	$[-100, 100]^{10}$	0
$f_2(x) = \sum_{i=1}^{n-1} (100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2)$	$[-30, 30]^{10}$	0
$f_3(x) = \frac{1}{4000}\sum_{i=1}^{n}(x_i)^2 - \prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	$[-600, 600]^{10}$	0
$f_4(x) = \sum_{i=1}^{n}\lvert x_i\rvert + \prod_{i=1}^{n}\lvert x_i\rvert$	$[-10, 10]^{10}$	0
$f_5(x) = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos 2\pi x_i\right) + 20 + e$	$[-32, 32]^{10}$	0

Table 17
Comparison of CSA, PSO and GA on test functions in 10 dimensions (30 runs). The parameters of *AP* and *fl* are set to 0.1 and 2, respectively.

Function	Index	CSA	PSO	GA
Sphere function (f_1)	Best	9.54×10^{-13}	6.45×10^{-7}	0.09
	Mean	4.09×10^{-11}	3.10×10^{-5}	2.01
	Std.	6.17×10^{-11}	4.54×10^{-5}	2.21
	Avg. time (s)	0.67	0.98	1.86
Rosenbrock function (f_2)	Best	1.52	2.85	42.98
	Mean	10.86	18.33	496.78
	Std.	22.76	39.43	769.00
	Avg. time (s)	0.87	1.11	1.91
Griewank function (f_3)	Best	0.0099	0.01	0.41
	Mean	0.21	0.12	0.86
	Std.	0.12	0.08	0.20
	Avg. time (s)	1.14	1.37	2.15
Schwefel function (f_4)	Best	9.37×10^{-6}	4.05×10^{-4}	0.10
	Mean	6.27×10^{-3}	3.58×10^{-3}	0.28
	Std.	1.99×10^{-2}	2.45×10^{-3}	0.11
	Avg. time (s)	0.76	1.03	1.84
Ackley function (f_5)	Best	1.02×10^{-6}	7.79×10^{-4}	0.32
	Mean	1.90	2.94	1.34
	Std.	0.79	1.90	0.70
	Avg. time (s)	0.87	1.17	1.95

Table 18
The effect of using different *AP* values on the performance of CSA.

Function	Index	$AP = 0$, $fl = 2$	$AP = 0.05$, $fl = 2$	$AP = 0.2$, $fl = 2$	$AP = 0.3$, $fl = 2$
Sphere function (f_1)	Best	125.28	2.90×10^{-16}	7.11×10^{-8}	6.11×10^{-6}
	Mean	900.61	2.67×10^{-14}	3.25×10^{-7}	5.75×10^{-5}
	Std.	541.93	5.21×10^{-14}	2.98×10^{-7}	3.29×10^{-5}
Rosenbrock function (f_2)	Best	305.37	0.24	3.16	0.99
	Mean	1.29×10^{5}	43.12	10.46	14.04
	Std.	1.97×10^{5}	71.76	18.16	19.47
Griewank function (f_3)	Best	3.03	0.07	5.41×10^{-6}	0.009
	Mean	12.31	0.23	0.14	0.11
	Std.	8.86	0.15	0.07	0.08

Table 19
The effect of *AP* and *fl* on the performance of CSA.

Function	Index	*AP* = 0.05 *fl* = 1.5	*AP* = 0.05 *fl* = 2.5	*AP* = 0.2 *fl* = 1.5	*AP* = 0.2 *fl* = 2.5
Sphere function (f_1)	Best	1.35×10^{-16}	8.14×10^{-14}	6.04×10^{-10}	2.33×10^{-6}
	Mean	8.48×10^{-14}	2.64×10^{-12}	2.47×10^{-8}	1.88×10^{-5}
	Std.	1.79×10^{-13}	3.59×10^{-12}	2.35×10^{-8}	3.18×10^{-5}
Rosenbrock function (f_2)	Best	4.10	0.63	0.96	1.08
	Mean	47.72	4.87	64.40	20.06
	Std.	64.42	1.94	149.46	35.33
Griewank function (f_3)	Best	0.13	0.05	0.02	0.03
	Mean	0.45	0.18	0.20	0.12
	Std.	0.28	0.08	0.14	0.07

the algorithm has been eliminated. Considering the best index, from the studied parameter setting approaches the best one for f_1, f_2 and f_3 is (*AP* = 0.05 and *fl* = 1.5), (*AP* = 0.05 and *fl* = 2.5) and (*AP* = 0.2 and *fl* = 2), respectively. Considering the mean index, from the studied parameter setting approaches the best one for f_1, f_2 and f_3 is (*AP* = 0.05 and *fl* = 2), (*AP* = 0.05 and *fl* = 2.5) and (*AP* = 0.3 and *fl* = 2), respectively. Among the test functions, f_1 and f_2 are unimodal while f_3 is multimodal. It seems that for unimodal functions, small values of *AP* lead to better results while for multimodal functions, it is better to use larger values for *AP* to escape local optima. If a fixed value for *AP* is used (*AP* = 0.05) and the value of *fl* is increased from 1.5 to 2.5, on average the performance of CSA on f_1, f_2 and f_3 improves. If the value of *AP* is set to 0.2 and the value of *fl* is increased from 1.5 to 2.5, on average the performance of CSA on f_1 does not improve while on f_3 the performance of CSA improves. As a result, like other optimization techniques, fine-tuning of CSA is a problem dependent issue which should be done by trial.

6. Conclusion

Based on the intelligent behavior of crows, a novel metaheuristic algorithm, called CSA, is proposed in this paper. CSA is population-based optimization algorithm which is rather simple with two adjustable parameters (flight length and awareness probability) only, which in turn makes it very attractive for applications in different engineering areas. In CSA, the parameter of awareness probability is directly used to control the diversity of the algorithm. In comparison with GA, PSO and HS, CSA has fewer parameters to adjust and hence is easier to implement. The usefulness of CSA is evaluated by solving different engineering design problems which have different natures of objective functions, constraints and decision variables. Simulation results show that the performance of the proposed new algorithm is promising since it has produced competitive results in comparison with the other studied algorithms. On a set of benchmark functions, it is observed that although PSO is known as a fast technique among population-based algorithms, it is outperformed by CSA. From the results it is seen that the convergence rate of CSA is good and this algorithm finds the solution of the investigated problems in around 1 s.

Acknowledgement

The author would like to thank anonymous reviewers for their constructive comments and suggestions.

References

[1] Blum C, Roli A. Metaheuristics in combinatorial optimization: overview and conceptual comparison. ACM Comput Surv 2003;35:268–308.
[2] Yang XS. Metaheuristic optimization. Scholarpedia 2011;6. 11472.
[3] Glover F. Future paths for integer programming and links to artificial intelligence. Comput Oper Res 1986;13:533–49.
[4] Yang XS. Nature-inspired metaheuristic algorithms. Luniver Press; 2008.
[5] Yang XS. Engineering optimization: an introduction with metaheuristic applications. Wiley; 2010.
[6] Holland J. Adaptation in natural and artificial systems. Ann Anbor: University of Michigan Press; 1975.
[7] Kennedy J, Eberhart RC. Particle swarm optimization. In: Proc of IEEE international conference on neural networks, Piscataway, NJ; 1995. p. 1942–48.
[8] Geem ZW, Kim JH, Loganathan GV. A new heuristic optimization algorithm: harmony search. Simulation 2001;76:60–8.
[9] Yang X-S, Deb S. Cuckoo search via L_evy flights. In: Proceedings of world congress on nature & biologically inspired computing (NaBIC), Coimbatore, India; 2009. p. 210e4.
[10] Yang XS. A new metaheuristic bat-inspired algorithm. In: Gonzalez JR et al., editors. Nature-inspired cooperative strategies for optimization (NICSO 2010). Springer, SCI 284; 2010. p. 65–74.
[11] He S, Wu QH, Saunders JR. Group search optimizer: an optimization algorithm inspired by animal searching behavior. IEEE Trans Evol Comput 2009;13:973–90.
[12] Yang XS. Firefly algorithm, stochastic test functions and design optimisation. Int J Bio-Inspired Comput 2010;2(2):78–84.
[13] <https://en.wikipedia.org/wiki/Corvus_%28genus%29>.
[14] Rincon, Paul, Science/nature|crows and jays top bird IQ scale, BBC News.
[15] Prior H, Schwarz A, Güntürkün O. Mirror-induced behavior in the magpie (pica pica): evidence of self-recognition. PLoS Biol 2008;6(8):e202.
[16] <https://en.wikipedia.org/wiki/Hooded_crow>.
[17] Clayton N, Emery N. Corvide cognition. Curr Biol 2005;15:R80–1.
[18] Wolpert DH, Macready WG. No free lunch theorems for search. Citeseer; 1995.
[19] Wolpert DH, Macready WG. No free lunch theorems for optimization. IEEE Trans Evol Comput 1997;1:67–82.
[20] Ray T, Liew KM. Society and civilization: an optimization algorithm based on the simulation of social behavior. IEEE Trans Evol Comput 2003;7:386–96.
[21] Liu H, Cai Z, Wang Y. Hybridizing particle swarm optimization with differential evolution for constrained numerical and engineering optimization. Appl Soft Comput 2010;10:629–40.
[22] Zhang M, Luo W, Wang X. Differential evolution with dynamic stochastic selection for constrained optimization. Inf Sci 2008;178:3043–74.
[23] Sadollah A, Bahreininejad A, Eskandar H, Hamdi M. Mine blast algorithm: a new population based algorithm for solving constrained engineering optimization problems. Appl Soft Comput 2013;13:2592–612.
[24] Coello CAC. Use of a self-adaptive penalty approach for engineering optimization problems. Comput Ind 2000;41:113–27.
[25] Coello CAC, Mezura Montes E. Constraint-handling in genetic algorithms through the use of dominance-based tournament selection. Adv Eng Inf 2002;16:193–203.
[26] He Q, Wang L. An effective co-evolutionary particle swarm optimization for constrained engineering design problems. Eng Appl Artif Intell 2007;20:89–99.
[27] He Q, Wang L. A hybrid particle swarm optimization with a feasibility-based rule for constrained optimization. Appl Math Comput 2007;186:1407–22.
[28] Coelho LDS. Gaussian quantum-behaved particle swarm optimization approaches for constrained engineering design problems. Expert Syst Appl 2010;37:1676–83.
[29] Mezura-Montes E, Coello CAS. Useful infeasible solutions in engineering optimization with evolutionary algorithms. Advances in artificial intelligence of LNCS, 3789. Berlin: Springer-Verlag; 2005.
[30] Parsopoulos K, Vrahatis M. Unified particle swarm optimization for solving constrained engineering optimization problems. Advances in natural computation LNCS, 3612. Berlin: Springer-Verlag; 2005. p. 582–91.
[31] Huang FZ, Wang L, He Q. An effective co-evolutionary differential evolution for constrained optimization. Appl Math Comput 2007;186(1):340–56.
[32] Akay B, Karaboga D. Artificial bee colony algorithm for large-scale problems and engineering design optimization. J Intell Manuf 2012;23:1001–14.
[33] Rao RV, Savsani VJ, Vakharia DP. Teaching-learning-based optimization: a novel method for constrained mechanical design optimization problems. Comput Aided Des 2011;43:303–15.