© 2018 IJSRST | Volume 4 | Issue 2 | Print ISSN: 2395-6011 | Online ISSN: 2395-602X
Themed Section: Science and Technology

A Review on Using Crow Search Algorithm in Solving the Problems of Constrained Optimization

B. Surya Samantha , M. Trupthi, U. Sairam
Assistant Professor Department of Information Technology, CBIT, Hyderabad, Telangana , India

ABSTRACT

This paper proposes a novel metaheuristic optimizer, named crow search algorithm (CSA), based on the wise conduct of crows. CSA is a population-based technique which works based on this thought crows store their abundance sustenance secluded from everything places and recover it when the nourishment is required. CSA is connected to improve six constrained engineering design problems which have diverse natures of target capacities, requirements and choice factors. Simulation results uncover that utilizing CSA may prompt finding promising results contrasted with alternate algorithms.

Keywords: Constrained Engineering Optimization, Metaheuristic Optimization, Crow Search Algorithm.

I. INTRODUCTION

Engineering design is characterized as a decision making procedure to construct items that fulfill indicated needs. Frequently, building design problems incorporate entangled target capacities with countless variables. The doable arrangements are the arrangement of all designs portrayed by every single conceivable estimation of the design parameters (decision variables). An optimization technique tries to locate the ideal arrangement from all accessible plausible arrangements. Traditional search techniques have for quite some time been connected to take care of engineering design problems. In spite of the fact that these strategies find promising results in numerous genuine problems, they may flop in more mind boggling design problems. In genuine design problems, the quantity of decision variables can be extensive and their impact on the target capacity can be extremely convoluted. The target capacity may have numerous neighborhood optima, while the designer is occupied with the worldwide ideal. Such problems can't be taken care of by traditional techniques that exclusive discover nearby optima. In these cases, productive optimization techniques are required. Metaheuristic algorithms have demonstrated promising execution for tackling most certifiable optimization problems that are extremely nonlinear and multimodal. All metaheuristic algorithms utilize a specific tradeoff of randomization and neighborhood search [1]. These algorithms can discover great answers for troublesome optimization problems, yet there is no certification that ideal arrangements can be come to. It is trusted that these algorithms work more often than not, yet not constantly. Metaheuristic algorithms could be reasonable for worldwide optimization [2]. Based on Glover's tradition, all the cutting edge nature-propelled techniques are called metaheuristics [3]. Current pattern is to use nature-motivated metaheuristic algorithms to handle troublesome problems and it has been demonstrated that metaheuristics are shockingly extremely productive [1,2]. Hence, the writing of metaheuristics has extended immensely over the most recent two decades [4,5]. A portion of the outstanding metaheuristic algorithms are as per the

following: genetic algorithm (GA) based on normal determination [6], molecule swarm optimization (PSO) based on social conduct of feathered creature rushing and fish tutoring [7], harmony search (HS) based on music act of spontaneity process [8], cuckoo search algorithm based on the brood parasitism of some cuckoo species [9], bat algorithm (BA) based on echolocation conduct of microbats [10], bunch search optimizer (GSO) based on creature searching conduct [11], firefly algorithm (FA) based on the blazing light examples of tropic fireflies [12], and so forth. To date, researchers have just utilized an extremely restricted qualities propelled by nature and there is space for improvement of more algorithms. One of the primary inspirations of this paper is to build up an easy to use (straightforward idea and simple execution) metaheuristic technique by which we may get promising results when tackling optimization problems.

Crows are generally appropriated variety of feathered creatures which are presently considered to be among the world's most wise creatures [13,14]. As a gathering, crows demonstrate wonderful cases of insight and frequently score exceedingly on knowledge tests. They can retain faces, utilize apparatuses, convey in complex ways and stow away and recover nourishment crosswise over seasons [13,15].

In a crow run, there is a conduct which has numerous likenesses with an optimization procedure. As indicated by this conduct, crows conceal their abundance sustenance in specific positions (concealing spots) of the earth and recover the put away nourishment when it is required. Crows are avaricious winged creatures since they take after each other to acquire better sustenance sources. Discovering sustenance source covered up by a crow isn't a simple work since if a crow finds another is tailing it, the crow tries to trick that crow by heading off to another position of the earth. From optimization perspective, the crows are searchers, the environment is search space, each position of the earth is comparing to an achievable arrangement, the nature of sustenance source is objective (wellness) work and the best nourishment wellspring of the earth is the worldwide arrangement of the issue. Based on these comparativeities, CSA endeavors to recreate the insightful conduct of the crows to discover the arrangement of optimization problems.

II. CROW SEARCH ALGORITHM

They contain the biggest cerebrum in respect to their body estimate. Based on a mind to-body proportion, their cerebrum is marginally lower than a human cerebrum. Confirmations of the cunning of crows are ample. They have exhibited mindfulness in reflect tests and have instrument making capacity. Crows can recall confront s and caution each other when an unpleasant one methodologies. In addition, they can utilize devices, impart in modern ways and review their sustenance's concealing spot up to a while later [13– 16]. Crows have been known to watch different winged animals, watch where alternate flying creatures shroud their nourishment, and take it once the proprietor clears out. On the off chance that a crow has submitted burglary, it will play it safe, for example, moving concealing spots to abstain from being a future casualty. Truth be told, they utilize their own particular experience of having been a hoodlum to foresee the conduct of a pilferer, and can decide the most secure course to professional tect their stores from being appropriated [17]. In this paper, based on the previously mentioned keen behaviors, a population-based metaheuristic algorithm, CSA, is developed. The standards of CSA are recorded as takes after:

- Crows live as run.
- Crows retain the position of their concealing spots.
- Crows take after each other to do burglary.
- Crows shield their reserves from being stolen by a likelihood.

It is accepted that there is a d-dimensional condition including various crows. The quantity of crows (rush size) is N and the position of crow I at time (cycle). At cycle iter, the position of concealing spot of crow i is appeared by $m^{i;iter}$. This is the best position that crow I has acquired up until now. For sure, in memory of each crow the position of its best experience has been remembered. Crows move in nature and search for better sustenance sources (concealing spots).

State 1: Crow j does not realize that crow I is tailing it. Accordingly, crow I will way to deal with the concealing spot of crow j. In this

$x^{i;iter}þ1 ¼ x^{i;iter} þ r_i \times fl^{i;iter} \times ðm^{j;iter} — x^{i;iter}þ$

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

Figure 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 $x^{i;iter}$) and large values results in global search (far from $x^{i;iter}$). As Figure 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 $x^{i;iter}$ and $m^{j;iter}$. As Figure 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 $m^{j;iter}$.

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.

Metaheuristic algorithms ought to give a decent harmony amongst broadening and heightening [2]. In CSA, increase and broadening are principally controlled by the parameter of mindfulness likelihood (AP). By reduction of the mindfulness probability esteem, CSA tends to direct the search on a neighborhood area where an ebb and flow great arrangement is found in this district. Accordingly, utilizing little estimations of AP, builds escalation. Then again, by increment of the mindfulness likelihood esteem, the likelihood of searching the region of flow great arrangements reductions and CSA has a tendency to investigate the search space on a worldwide scale (irregularization). Thus, utilization of vast estimations of AP expands broadening.

III. CSA IMPLEMENTATION FOR OPTIMIZATION

Pseudo code of CSA is appeared in Figure 2. The progression savvy methodology for the usage of CSA is given in this area.

Stage 1: Initialize issue and flexible parameters
The optimization issue, decision variables and limitations are characterized

Stage 2: Initialize position and memory of crows
N crows are arbitrarily situated in a d-dimensional search space as the individuals from the rush. Each crow indicates a doable arrangement of the issue and d is the quantity of decision variables.

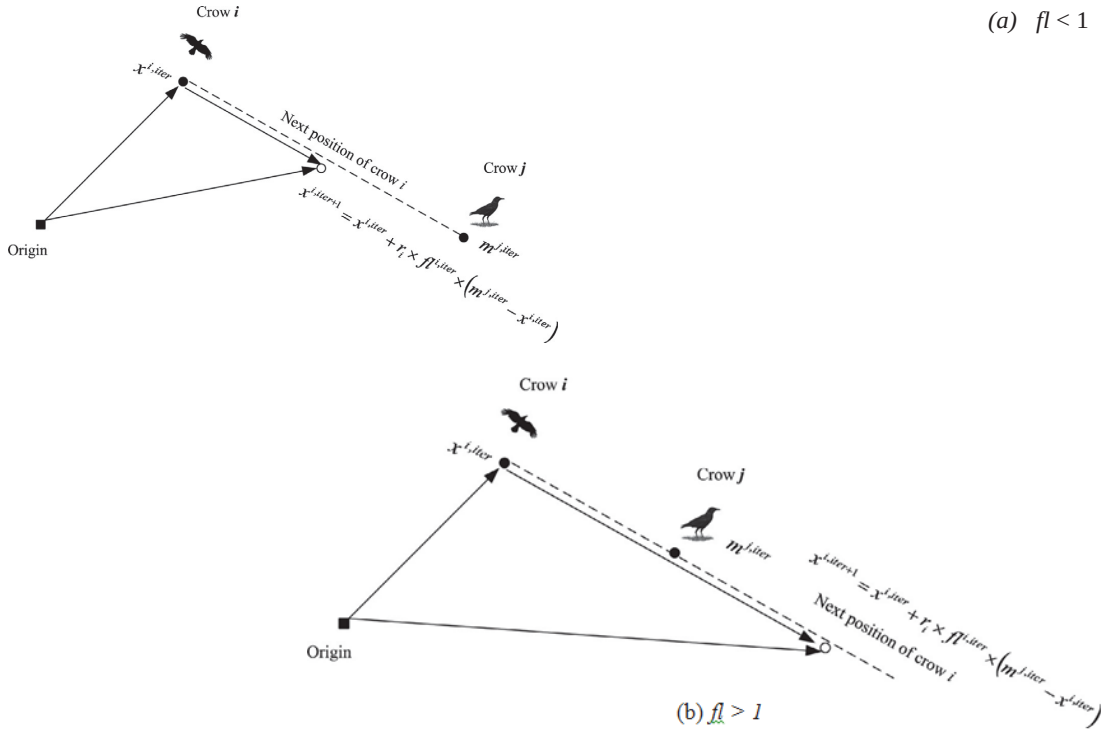

Figure 1. Flowchart of state 1 in CSA (a) $fl < 1$ and (b) $fl > 1$. Crow i can go to every position on the dash line.

Crow search algorithm :

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 $r_i \geq AP^{j,iter}$

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

else
$x^{i,iter+1}$ = *a random position of search space*
end if
end for
check the feasibility of new position of the crows
update the memory of crows

Stage 3: Evaluate wellness (objective) work

For each crow, the nature of its position is processed by embeddings the decision variable esteems into the goal work.

Stage 4: Generate new position

Crows create new position in the search space as takes after: assume crow I needs to produce another position. For this point, this

crow (mj). The new position of crow I is gotten by Eq. (2). This procedure is rehashed for every one of the crows.

Stage 5: Check the attainability of new positions

The attainability of the new position of each crow is checked. In the event that the new position of a crow is doable, the crow refreshes its position. Something else, the crow remains in the present position and does not move to the created new position.

Stage 6: Evaluate wellness capacity of new positions

The wellness work an incentive for the new position of each crow is processed.

Stage 7: Update memory

It is seen that if the wellness work estimation of the new position of a crow is superior to anything the wellness work estimation of the retained position, the crow refreshes its memory by the new position.

Stage 8: Check end paradigm

Stages 4– 7 are rehashed until the point when itermax is come to. At the point when the termi-country rule is met, the best position of the memory as far as the target work esteem is accounted for as the arrangement of the optimization issue.

IV. COMPARISON OF CSA WITH GA, PSO AND HS

Like other algorithms, for example, PSO , GA and HS, CSA influences utilization of a population of searchers to investigate the search to space. By utilization of a population the likelihood of finding a decent solution and getting away from neighborhood optima increments. Notwithstanding pop-ulation size and most extreme number of cycles (ages), optimization algorithms have some different parameters which ought to be balanced. Parameter setting is one of the downsides of optimization algorithms since it is a tedious work. Algorithms which have less parameters to alter are less demanding to implement. In CSA, flight length and mindfulness likelihood ought to be tuned (2 parameters). In PSO algorithm the customizable parameters are idleness weight, greatest estimation of speed, singular learning component and social learning factor (4 parameters). HS requires the estimation of harmony memory thinking about rate, pitch changing rate and transfer speed of age (3 parameters). In GA, determination strategy, hybrid technique, hybrid likelihood, change technique, transformation likelihood and substitution technique ought to be resolved (6 parameters). In HS, another arrangement is acknowledged if its wellness esteem is superior to the wellness of the most noticeably bad harmony of memory.

Like HS and PSO, CSA incorporates memory in which great arrangements are remembered. In PSO, every molecule is pulled in towards the best position at any point found without anyone else's input and the best position at any point found by the gathering. Therefore, at every cycle, the best arrangements discovered so far are straightforwardly utilized. At every emphasis of CSA, each crow chooses arbitrarily one of the run crows (it might act naturally) and moves towards its concealing spot (the best arrangement found by that crow). This implies at every emphasis of CSA, the best positions discovered so far are straightforwardly used to discover better positions.

V. NUMERICAL EXAMPLES

It has been demonstrated that under specific suspicions, no single search algorithm is the best by and large for all problems [18,19]. As it were, an algorithm may tackle a few problems preferred and a few problems more terrible over alternate algorithms. So as to assess the optimization energy of the proposed CSA without a one-sided conclusion, six engineering design problems are considered and unraveled, including three-bar truss, weight vessel, pressure/pressure spring, welded shaft, equip prepare and Belleville spring. All the considered problems have diverse natures of target capacities, requirements and decision variables. CSA has been executed in the MATLAB condition on a PC with Pentium 4 CPU 2.1 G 2 GB RAM. Figure 3 demonstrates the flowchart of CSA execution.

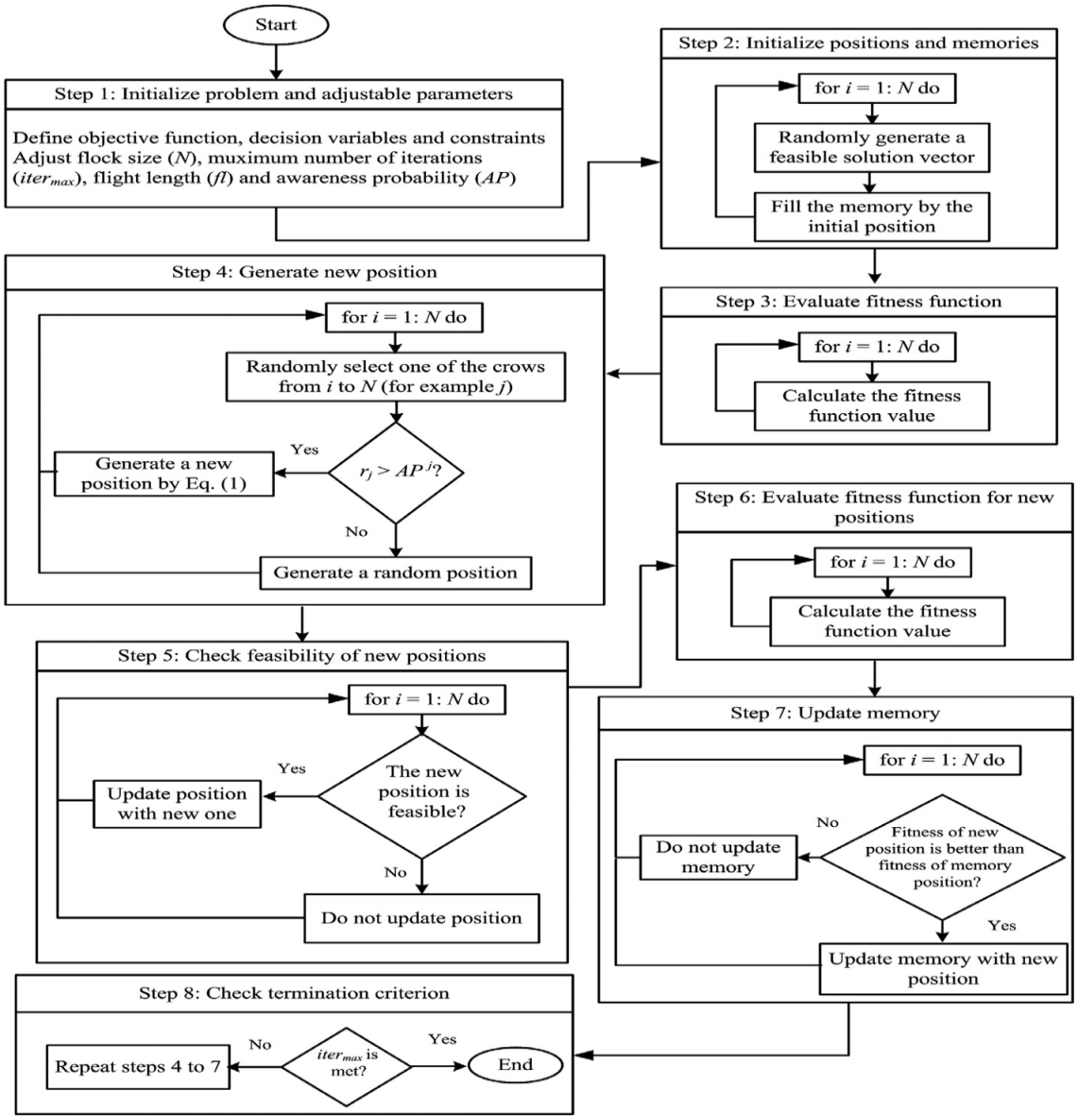

Figure 2. Flowchart of CSA for doing optimization.

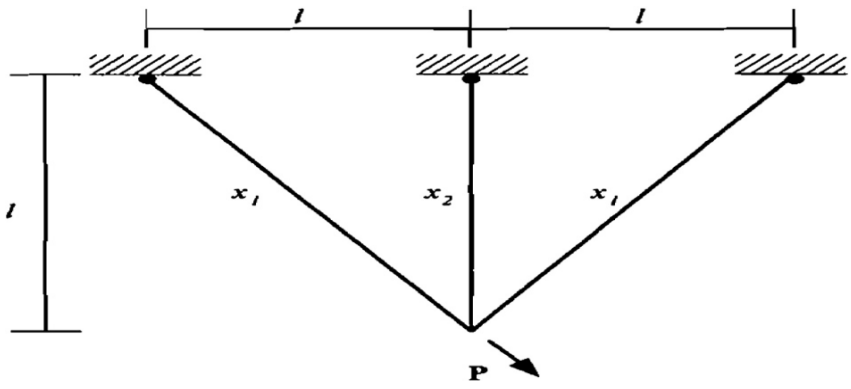

Figure 3. Schematic of three-bar truss design problem.

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

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

Pressure vessel design problem

In this design issue, the objective is to limit the aggregate cost of a weight vessel including material, shaping and welding costs. As Fig. 6 appears, this optimization issue comprises of four decision variables: thickness of the shell (x_1 or T_s), thickness of the head (x_2 or T_h), internal sweep (x_3 or R) and length of the tube shaped segment of the vessel (x_4 or L). Among the four decision variables, x_1 and x_2 are discrete (whole number increases of 0.0625 in) and x_3 and x_4 are continuous.

Table 4 shows the execution of the CSA on this issue. This table demonstrates the ideal estimations of the decision variables and the limitation esteems relating to the best arrangement acquired by CSA more than 50 autonomous runs. It can be seen that x_1 and x_2 are whole number duplicates of 0.0625 and x_3 and x_4 are in the conceivable range. Table 5 looks at the measurable results acquired by CSA and those got by alternate techniques for the writing which have revealed attainable arrangements (particularly for x_1 and x_2).

Figure 7 delineates the joining rate of the CSA for finding the best arrangement of the weight vessel issue. This figure demonstrates the estimation of best-so-far at every emphasis. It can be seen that the merging rate of CSA is great.

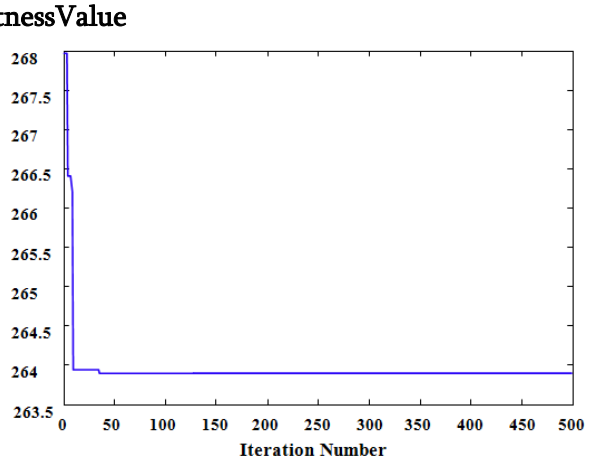

Figure 4. Convergence rate of CSA for finding the best solution of three-bar truss design problem.

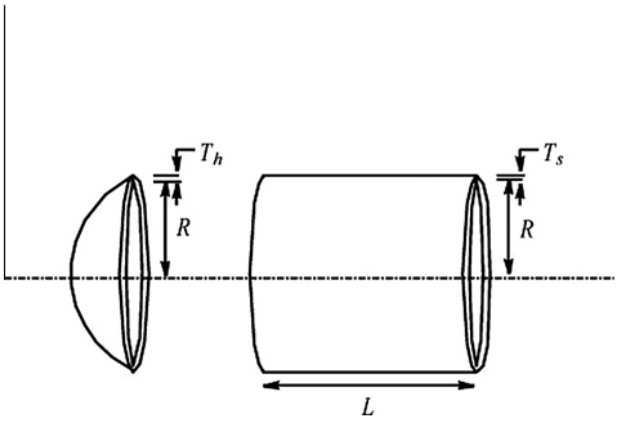

Figure 5. Schematic of the pressure vessel design problem.

Table 3. Comparison with statistical results obtained by CSA with 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 1.0122543402e—10	263.8958433765	263.8958433765	

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	176.636598 55
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. Comparitive analysys 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
$(1+k)$-ES	N.A.	6379.938037	6059.701610	210
TLBO	N.A.	6059.71434	6059.714335	N.A.
CSA	7332.84162110	6342.49910551	6059.71436343	384.9454163 4

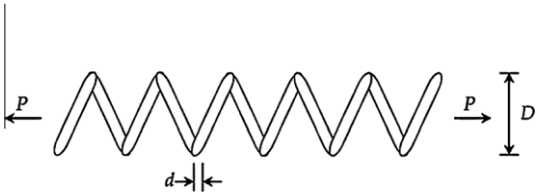

Figure 6. Schematic of tension/compression spring design problem

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
(μ + k)-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	0.0126599984	0.0126652328	1.357079e—6

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
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.
(μ + k)-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

Table 10. 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
CSA	3.184737928 9e—8	2.059327018 2e—9	2.700857148 89e—12	5.059779e—9

Table 11. 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

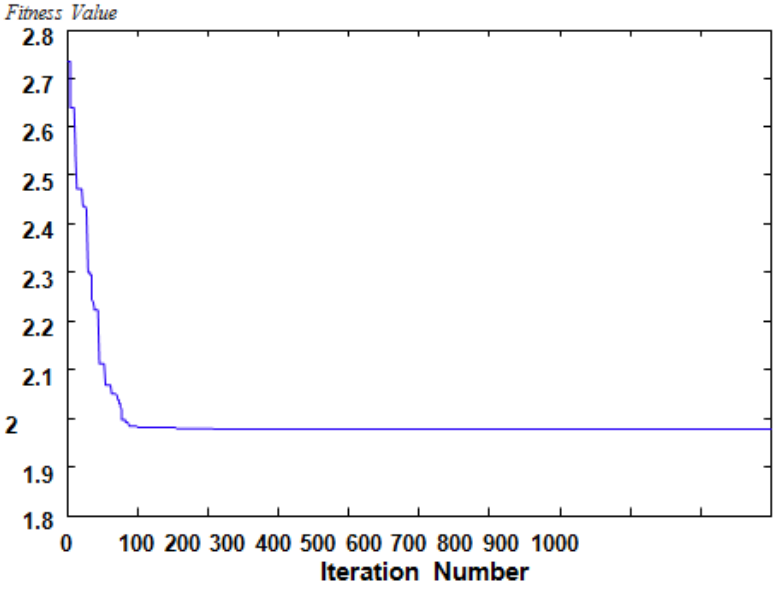

Figure 7. Convergence rate of CSA for finding the best solution of Belleville spring problem.

Despite the fact that the results got for the engineering problems demonstrate that CSA demonstrates a focused execution with the current strategies in the writing, yet, there exists an inquiry respect ing the execution of CSA in bigger scale problems. With a specific end goal to assess the execution of CSA on bigger scale optimization problems, five surely understood benchmark capacities appeared in Table 15 are tackled in 10 measurements. In CSA, as previously, AP and fl esteems have been set to 0.1 and 2, individually. In PSO, the speed is controlled and the learning factors (individual and social) are set to 2.

Table 12. 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

Also, the latency weight diminishes directly from 0.9 to 0.4 during cycles. In GA, curved hybrid (with the coefficients of 0.25 and 0.75), uniform transformation and competition choice are utilized. Hybrid and transformation probabilities are set to 0.9 and 0.005, individually. In CSA, PSO and GA, population measure is set to 20 and most extreme number of emphasess is chosen 2000. Thus, in the examinations, number of wellness assessments (NFEs) is 40,000. Table 15 demonstrates the results got by CSA in examination with the results found by PSO and GA more than 30 free runs. It is seen, on every one of the capacities, CSA beats alternate algorithms as far as the best list. In this table, the normal time of the runs is appeared. It is seen that CSA expends less computational time than PSO and GA over same number of wellness assessments.

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

Table 14. 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) 5.75 × 10^{-5}	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}	
	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 15. 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

From Table 15, it is seen that AP = 0 prompts feeble execution of CSA since the expansion capacity of 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 uni- modal 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. Considering the best file, from the examined 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), individually. Considering the mean record, from the contemplated 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), individually. Among the test capacities, f_1 and f_2 are uni-modular while f3 is multimodal. It appears that for unimodal capacities, little estimations of AP prompt better results while for multimodal functions, it is smarter to utilize bigger esteems for AP to escape nearby optima. On the off chance that a settled an incentive for AP is utilized (AP = 0.05) and the estimation of fl is expanded from 1.5 to 2.5, by and large the execution of CSA on f1, f2 and f3 makes strides. On the off chance that the estimation of AP is set to 0.2 and the estimation of fl is expanded from 1.5 to 2.5, by and large the execution of CSA on f_1 does not enhance while on f_3

the execution of CSA moves forward. Accordingly, as other optimization techniques, tweaking of CSA is an issue subordinate issue which ought to be finished by trial.

VI. CONCLUSION

Based on the wise conduct of crows, a novel metaheuristic algorithm, called CSA, is proposed in this paper. CSA is population-based optimization algorithm which is fairly straightforward with two flexible parameters (flight length and mindfulness prob-capacity) just, which thusly makes it extremely alluring for applications in various engineering zones. In CSA, the parameter of mindfulness likelihood is straightforwardly used to control the assorted variety of the algorithm. In examination with GA, PSO and HS, CSA has less parameters to change and thus is simpler to execute. The value of CSA is assessed by taking care of various engineering design problems which have distinctive natures of target capacities, limitations and decision variables. Simulation results demonstrate that the execution of the proposed new algorithm is promising since it has delivered aggressive results in examination with the other contemplated algorithms. On an arrangement of benchmark capacities, it is watched that in spite of the fact that PSO is known as a quick technique among population-based algorithms, it is outflanked by CSA. From the results it is seen that the meeting rate of CSA is great and this algorithm finds the arrangement of the examined problems in around 1 s.

VII. REFERENCES

[1]. Blum C, Roli A. Metaheuristics in combinatorial optimization: overview and conceptual comparison. ACM Comput Surv 2003;35:268-308.

[2]. Yang XS. Nature-inspired metaheuristic algorithms. Luniver Press; 2008.

[3]. Yang XS. Engineering optimization: an introduction with metaheuristic applications. Wiley; 2010.

[4]. Holland J. Adaptation in natural and artificial systems. Ann Anbor: University of Michigan Press; 1975.

[5]. Geem ZW, Kim JH, Loganathan GV. A new heuristic optimization algorithm: harmony search. Simulation 2001;76:60-8.

[6]. 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.

[7]. 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.

[8]. 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.

[9]. Yang XS. Firefly algorithm, stochastic test functions and design optimisation. Int J Bio-Inspired Comput 2010;2(2):78-84.

[10]. https://en.wikipedia.org/wiki/Corvus_%28genus%29.

[11]. Prior H, Schwarz A, Güntürkün O. Mirror-induced behavior in the magpie (pica pica): evidence of self-recognition. PLoS Biol 2008;6(8):e202.

[12]. https://en.wikipedia.org/wiki/Hooded_crow.

[13]. Clayton N, Emery N. Corvide cognition. Curr Biol 2005;15:R80-1.

[14]. Wolpert DH, Macready WG. No free lunch theorems for search. Citeseer; 1995.

[15]. Wolpert DH, Macready WG. No free lunch theorems for optimization. IEEE Trans Evol Comput 1997;1:67-82.

[16]. 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.
[17]. 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.
[18]. Zhang M, Luo W, Wang X. Differential evolution with dynamic stochastic selection for constrained optimization. Inf Sci 2008;178:3043-74.
[19]. 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.
[20]. Coello CAC. Use of a self-adaptive penalty approach for engineering optimization problems. Comput Ind 2000;41:113-27.
[21]. He Q, Wang L. An effective co-evolutionary particle swarm optimization for constrained engineering design problems. Eng Appl Artif Intell 2007;20:89-99.
[22]. He Q, Wang L. A hybrid particle swarm optimization with a feasibility-based rule for constrained optimization. Appl Math Comput 2007;186:1407-22.
[23]. Coelho LDS. Gaussian quantum-behaved particle swarm optimization approaches for constrained engineering design problems. Expert Syst Appl 2010;37:1676-83.
[24]. 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.
[25]. 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.