Iranian Journal of Numerical Analysis and Optimization
Vol. 12, No. 2, pp 397–425
DOI: DOI:10.22067/ijnao.2022.71418.1049
https://ijnao.um.ac.ir/

How to cite this article
Research Article

Crocodile Hunting Strategy (CHS): A comparative study using benchmark functions

A.R. Balavand

Abstract

The crocodiles have a good strategy for hunting the fishes in nature. These creatures are divided into two groups of chasers and ambushers when hunting. The chasers direct prey toward shallow water with a powerful splash of its tail without catching them, and the ambushers wait in the shallow and try to snatch the fishes. Such behavior inspires the development of a new population-based optimization algorithm called the crocodile hunting strategy (CHS). In order to verify the performance of the CHS, several classical benchmark functions and four constrained engineering design optimization problems are used. In the classical benchmark function, the comparisons are performed using ant colony optimization, differential evolution, genetic algorithm, and particle swarm optimization. Constrained engineering design problems are compared with firefly algorithm, harmony search, shuffled frog-leaping algorithm, and teaching-learning-based optimization. The results of the comparison show that different operators designed in the CHS algorithm lead to fast algorithm convergence and show better results compared to other algorithms.

AMS subject classifications (2020): 45D05; 42C10; 65G99.

Keywords: Crocodile hunting strategy; Optimization algorithms; Numerical optimization; Classical benchmark functions; Constrained engineering design problem.

1 Introduction

Optimization is an art that finds the best solution among the set of solutions, and the optimization techniques are tools that converge to the optimal

Received 11 July 2021; revised 18 January 2022; accepted 5 February 2022

Alireza Balavand
Department of Industrial Engineering, Science and Research Branch, Islamic Azad University. e-mail: a.balavand@srbiau.ac.ir

solutions by searching the solution space. Finding the best solution to mathematical problems has been a challenge for researchers, especially in complicated problems in the field of engineering. Therefore, convergence to the best solution in the least time with high accuracy has always been interesting to researchers in the field of optimization. Among the optimization tools, metaheuristic algorithms are widely being used for a large number of real-world engineering problems due to their acceptable accuracy and low time consumption. Their popularity derives from the following aspects. Firstly, all of these optimization techniques have some fundamental theories and mathematical models proven to be reasonable, which come from the real world and are inspired by all kinds of physical phenomena or biological behaviours [26, 7]. Secondly, metaheuristics are usually able to solve NP-hard problems with a high number of decision variables. They are very flexible and versatile since one can change the structures and parameters of algorithms to obtain better solutions. Thirdly, metaheuristic algorithms can effectively escape local optima, which is very valuable for multimodal engineering problems that have many local optima in their structures. In addition, their variants can be developed by absorbing the advantages of other algorithms to improve the accuracy of solutions in a reasonable time. Most metaheuristic algorithms can solve different types of problems. Some of them are inherently powerful because of efficient operators and other types of them have been well-hybridized by the classical algorithms [35]. These problems are divided into various categories, which are included constrained and unconstrained in terms of problem structure, discrete and continuous in terms of solutions, and single or multiobjective in terms of objective function [27]. Generally, metaheuristics are divided into two categories such as single-solution-based and population-based algorithms. In single-solution-based algorithms, the searching process starts with one candidate solution, whereas in population-based algorithms, the optimization process performs using a set of solutions (i.e., population). Population-based metaheuristics have advantages over single-solution. These are as follows:

1. The searching process starts with several random solutions as the initial population.

2. population-based metaheuristics can share the information with each other around the search space that, most of the time, leads to escape from local optimal.

3. The exploration capability of population-based metaheuristics has had better than single-solution-based techniques.

In a general view, population-based metaheuristics are classified into seven categories, including biology-based algorithms, physics-based algorithms, social-based algorithms, chemistry-based algorithms, math-based algorithms, sport-based algorithms, and music-based algorithms [1]. Biology-based algorithms are divided into three categories of evolution-inspired algorithms, swarm intelligence algorithms, and artificial immune systems [21]. The evolution-inspired algorithm is a generic population-based metaheuristic that is inspired by biological evolution that includes crossover, mutation, and selection. The second category is swarm-intelligence-based algorithms. These algorithms are based on the interaction of swarm with each other. Swarm-based algorithms are easier to implement than evolutionary-based algorithms due to including a less number of operators (i.e., selection, crossover, mutation). Apart from this, there are some advantages of swarm-based algorithms, which are as follows:

Swarm-based algorithms use the flow of information among solutions during iterations, whereas evolutionary-based algorithms do not use the information of the previous generations.

Swarm-based algorithms have few input parameters as compared to evolutionary techniques.

Swarm-based algorithms utilize less memory space for reaching the best optimal solutions [3, 36].

The artificial immune systems algorithm was introduced in [11]. These algorithms mimic the principle of the biological immune system, including the negative selection algorithm, clonal selection algorithm, the immune network model, and danger theory.

Solving the test functions has been used for comparing metaheuristics. Test functions include classical test functions, constrained engineering design problems, and CEC functions. Classical test functions have been solved in several dimensions using metaheuristics that are the most practical and simple type of test function. Constrained engineering design problems are real-world engineering problems that are the proper criterion for comparing metaheuristics. CEC test functions are classified into complicated test functions that have always been a challenge to metaheuristics.

In the remainder of this study, in Section 2, the literature review will be explained. The concepts and operators of crocodile hunting strategy (CHS) are explained in Section 3. Then in Section 4, the CHS is investigated in various aspects. In that section, the classical benchmark functions and constrained engineering design problems are solved using the CHS algorithm and compared with other well-known metaheuristics. Finally, in Section 5, the conclusion and discussion will be presented.

2 Literature review

Swarm intelligence of nature-inspired algorithms includes intensive techniques that mimic the social behavior of groups of animals. The power of swarm intelligence algorithms was appeared by Kennedy and Eberhart by developing the particle swarm optimization (PSO) algorithm [24]. The main idea of the PSO algorithm is inspired by the social behavior of the bird population. Ant colony optimization (ACO) [13] is inspired by the collective behavior of the ants, namely finding the shortest path from nest to food sources. Termite algorithm [29] is based on the behavior of termites. The shuffled fog-leaping algorithm [16] stems from the search for food by the frog group. Monkey search simulates a monkey that climbs trees and looks for food [34]. Cuckoo search [42] simulates the forced child parasitic behavior in combination with the levy flying behavior of some birds and fruit flies. The main idea of the firefly algorithm (FA) [41] is inspired by the optical connection between fireflies. Migrating birds optimization [14] simulates the process of the social evolution of birds. The fruit fly optimization algorithm [37] is inspired by the foraging behavior of fruit flies. Artificial bee colony [2] simulates the strategy of bees to find food from the nest to source food. Dolphin echolocation (DE) [23] is based on the behavior of dolphins in hunting, in which the dolphin finds its prey using the distance between its click and prey. The main idea of the bat algorithm [43] is based on the distance between the reflection of the bat's sound to source food. In social spider optimization [10], the spider receives and analyzes vibrations published on the web to determine the potential direction of a food source. Grey wolf optimizer [18, 33] is a nature-inspired metaheuristic algorithm that simulates the behavior of gray wolfs and the hierarchy of leadership in their hunting. Bird mating optimizer [4] is inspired by the mating strategies of bird species during the mating season. The tree-seed algorithm [25] is based on the relation between trees and their seeds. Moth-flame optimization [31] simulates the spiral motion of butterflies around the flame. Whale optimization algorithm [32] simulates the bubble net hunting method that is performed from humpback whales. Crow search algorithm [5] is modeled based on finding the food of other birds and stealing it by crows. The grasshopper optimization algorithm [40] is a nature-inspired metaheuristic algorithm that mimics the movement of Grasshoppers groups to find food sources. Queuing search algorithm [45] is inspired by human activities in queues. Pathfinder algorithm [44] is modeled by the collective movement of the animal group and mimics the leadership hierarchy of swarms to find the best food area or prey. The main inspiration of Harris hawks optimization [19] is the cooperative behavior and chasing style of Harris' hawks in nature called surprise pounce. The squirrel search algorithm [20] imitates the dynamic foraging behavior of southern flying squirrels and their efficient way of locomotion known as gliding. Marine predators algorithm [17] is inspired by widespread foraging strategies, namely Lévy and Brownian of predators in the ocean.

In this study, the new algorithm called CHS (crocodile optimization algorithm) is investigated using two kinds of benchmark functions. This algorithm was first proposed by Balavand [6] to find the optimum binary solution of a feature clustering method. The CHS is based on the swarm intelligence of crocodiles in hunting that simulates the behavior of crocodiles in the hunting of fishes— dividing crocodiles into different groups and creating the right strategy while hunting is the main idea of this algorithm. These creatures are divided into two groups when they are hunting fish. These groups include the chasers and the ambushers. These groups try to direct the fish to the attacking area by using the information flow and proper cooperation. After arriving prey in the attacking area, all members of the groups attack the fish and catch them. The details of how to encircle and attack by two groups will be explained in the next sections.

3 Crocodile optimization algorithm (CHS)

In this section, the optimization algorithm called CHS optimization algorithm is investigated, which was first used as one of the steps of feature clustering in [6]. The CHS is placed in the population-based algorithm that can solve optimization problems using swarm intelligence of crocodiles in hunting. Like other algorithms in this field, the CHS algorithm with exchanging information and creating an information flow through the collaboration of the members, converges to the optimum solution. This algorithm is based on the behavior of crocodiles that simulates the hunting strategy of crocodiles. Dinets [12] divided crocodiles into two kinds of chasers and ambushers when hunting. The chasers, which are bigger than other crocodiles, follow and chase fishes toward the shore with a powerful splash of their tail without catching them. Ambushers, which are smaller and more agile, wait in the shallow and try to snatch the fish. According to [12], crocodiles have a complicated strategy to snatch prey. According to the bottom section of Figure 1, the ambushers swim in a circular pattern around prey and take turns cutting through the center of the gradually shrinking circle, snatching the prey. According to the top section of Figure 1, chasers direct prey toward ambusher and attacking area [12]. In the following, the behaviors of the two groups will be simulated in mathematical equations.

3.1 Initializing

Like other metaheuristic algorithms, before entering the main phases, the initializing phase is done. In the initializing phase, several initial solutions are randomly generated. Indeed, these randomized solutions constitute the initial population of crocodiles. These solutions are generated with uniform random distribution between the lower and upper bounds. These solutions are generated according to the following equation:

$$x = LB + r \times (UB - LB)\,. \tag{1}$$

After determining initial parameters such as the number of population, the number of max iterations, the lower bound of variables, and the upper bounds of variables, random solutions (x) are generated based on (2), where LB and UB are lower and upper bounds of the problem, respectively. Also, r is a uniform random variable that is generated between zero and one. Then these solutions are evaluated based on the objective function. In fact, in operators of CHS, the solutions are evaluated based on the objective function. If a better solution is found, then it is replaced instead of the best solution. The better solution has the minimum objective function value, which is known as the best solution (x_{prey}).

Figure 1: Crocodiles hunting strategy

3.2 Chasing the prey

In this section, the behavior of the chasers is simulated. As mentioned before, the population is divided into two sections. Hence, each section makes up 50 percent of the population. The group of chasers includes the first 50 percent of the solutions. Accordingly, the second part of 50 percent of the population includes ambushers. These two groups are randomly divided into two groups of chasers and ambushers. The idea behind simulating the behavior of chasers is based on the distance between prey and crocodiles that simulates the behavior of chasers. As previously mentioned, the chasers are another group of hunters that follow the prey without catching it and direct the prey to the shore and shallow area. The following equations are proposed to simulate this behavior:

$$d^{i,t} = \left| x^{t}_{prey} - x^{i,t}_{chaser} \right|, \qquad \text{for all } i, \tag{2}$$

$$\begin{cases} x^{i,t+1}_{chaser} = \left(x^{i,t}_{chaser} - \beta . d \right) & \text{for all } i \ (\frac{\ln(it)}{\ln(maxit)} \times r) \geq 0.5, \quad (a) \\ x^{i,t+1}_{chaser} = \left| \beta - (\beta . d) \ \right| & \text{for all } i \ (\frac{\ln(it)}{\ln(maxit)} \times r) < 0.5. \quad (b) \end{cases} \tag{3}$$

Here x^{t}_{prey} denotes the prey position at iteration t and $x^{i,t}_{chaser}$ indicates the position of the chaser ith at iteration t. The coefficient β has a uniform distribution that changes between $[-3, \ 3]$. This r is a random number between zero and one. Also, it is the global iteration number and i is local iteration number in inner loops. These intervals have been obtained experimentally after successive iterations of the algorithm. According to (3), d is the distance between the prey and ith chaser. According to (3), two states occur: First, if $\frac{it}{maxit} \times r \geq 0.5$, then (3)($a$) is implemented, which means that each chaser in the first group moved toward the prey with a positive coefficient, and if $\frac{it}{maxit} \times r < 0.5$, then a random search is done. In brief, (3) shows that by an intelligent approach, in the initial iterations, (3)(b) is more implemented, and a random search is done. In the last iterations, (3)(a) is more implemented, and the chasers close to prey by random approach.

3.3 Attacking to prey

The final position of the prey is where the ambushers are waiting to catch the prey. Indeed, the ambushers camouflage in the final position and the chasers try to direct the prey to this place or the attacking area. In order to simulate the attacking phase, it is assumed that attackers are forced to update their position according to the following equations:

$$d^{i,t} = \left| x^{t}_{prey} - x^{i,t}_{ambusher} \right|, \qquad \text{for all } i, \tag{4}$$

$$A = \frac{apc + apa + x_{prey}}{3}, \tag{5}$$

$$\begin{cases} x^{i,t+1}_{ambusher} = \left(d . \cos(2r\pi) \ + x^{t}_{prey} \right), & \text{for all } i \ \ (p) \geq 0.5, \quad (a) \\ x^{i,t+1}_{ambusher} = \left(x^{i,t}_{ambusher} - \beta (A - x^{i,t}_{ambusher}) \right), & \text{for all } i \ \ (p) < 0.5, \quad (b) \end{cases} \tag{6}$$

in which $p = \frac{\ln(it)}{\ln(maxit)} \times r$ Here $x^{i,t}_{ambusher}$ shows the position of ith ambusher at iteration t and d denotes the distance between the ambushers and the prey.

Moreover, apc is the average position of chasers, apa is the average position of ambushers, and x_{prey} is best position or prey position. Crocodiles update their position according to the positions of the prey or average of the position of all groups. In this way, if $(\frac{\ln(it)}{\ln(maxit)} \times r) < 0.5$, then $(8)(a)$ is implemented that means that each ambusher swims around the prey with a rotational motion, and if $(\frac{\ln(it)}{\ln(maxit)} \times r) \geq 0.5$, then ambushers update their position according to the average of the position of all groups of chasers, ambushers, and the position of prey. How to calculate the average position of all groups is shown in (7).

3.4 Flowchart and pseudocode of the CHS

Given that the CHS algorithm is implemented in MATLAB software 2019b, in order to provide more explanation of the structure of the programming of the CHS algorithm, the pseudocode is presented as Algorithm 2, and the flowchart is shown in Figure 3. According to the pseudocode, after determining the initial parameters, such as the number of population, the maximum number of the iteration, the number of population of each group of the crocodiles, the lower and upper bound of the decision variable, and so on, the initializing phase is done. In the initializing phase, to create the number of the population, the random solutions are generated based on the uniform distribution that all of the solutions are between the lower and upper bounds of the objective function. These random solutions are evaluated based on the objective function. Before entering the main phases, the population is divided into two groups. Fifty percent of the population includes chasers, and the other fifty percent includes ambushers' population. It is assumed that the solution with the minimum objective value is the best solution, and other members of the population are considered crocodiles. Therefore, the population number is equal to $npop = nc + ng$, which is always an even number. Because the number of the population is halved, the number of all members of the population is always an even number.

In the following pseudocode, the first phase is performed based on (3) for each solution of the chasers population. Then each solution is evaluated based on the objective function. In this phase, if a better solution is found, then it will be replaced by the best solution. In the second phase, for each solution of the ambusher's population, (8) is performed and each solution is evaluated. In this phase, a better solution will be replaced instead of the best solution. According to the flowchart, after passing all phases of the algorithm, the stopping Criteria's are checked, if the stopping Criteria's are satisfied, then the best solutions, along with the objective value, will be reported. Otherwise, the number of iterations is added to a unit, and from the first phase, all steps are repeated.

3.5 Exploration and exploitation

In designing a metaheuristic algorithm, the balancing of the two criteria of exploration and exploitation increases the efficiency of the algorithm [3, 36, 28]. The experiences show that in the initial iterations, the ability of exploration should be increased, and in the final iterations, the power of exploitation should be increased. This means that in the initial iterations, the different parts of the solution space should be searched, and in the last iterations, the optimal solution should be searched more accurately. This rule has been observed in the CHS algorithm. In this way, in the exploitation phases, small mutations occur, and in the exploration phases, longer mutations occur. The CHS is divided into two phases, and in each phase, there is the main equation. Equation (3) is the main equation of the first phase and (8) is the second one. In the first iterations, part (b) of the main equations is more implemented because, in most solutions, the value of $\left(\frac{\ln(it)}{\ln(maxit)} \times r\right)$ is less than 0.5 in most cases. Therefore (b) is implemented in most times that (b)'s increase the power of exploration. Whereas in the last iterations, part (a) of (3) and (8) is implemented because in most cases the value of $\left(\frac{\ln(it)}{\ln(maxit)} \times r\right)$ is greater than 0.5 that increase the power of exploitation.

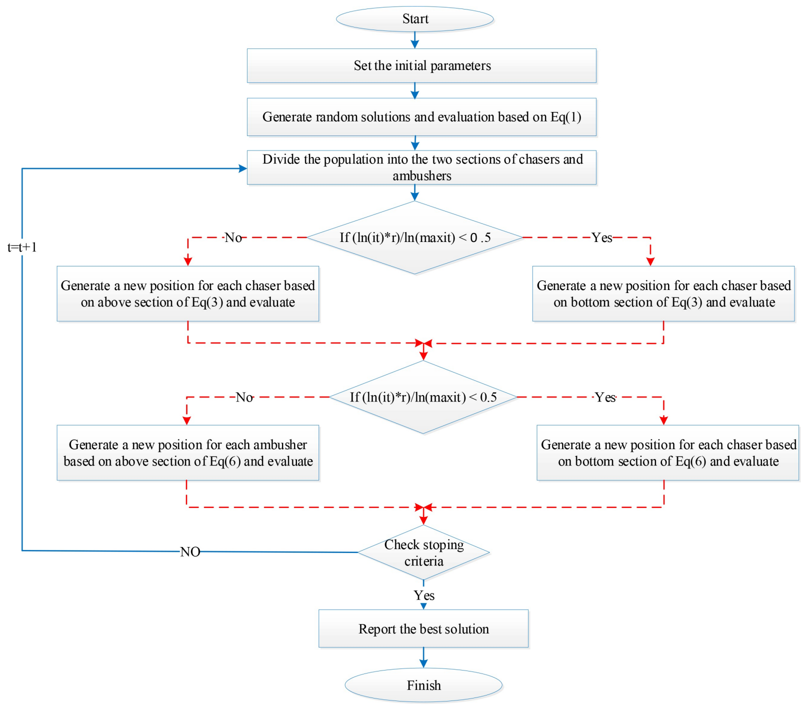

Start
Set the initial parameters
Generate random solutions and evaluation based on Eq(1)
Divide the population into the two sections of chasers and ambushers
If (ln(it)*r)/ln(maxit) < 0 .5
No
Yes
Generate a new position for each chaser based on above section of Eq(3) and evaluate
Generate a new position for each chaser based on bottom section of Eq(3) and evaluate
If (ln(it)*r)/ln(maxit) < 0.5
No
Yes
Generate a new position for each ambusher based on above section of Eq(6) and evaluate
Generate a new position for each chaser based on bottom section of Eq(6) and evaluate
t=t+1
Check stoping criteria
NO
Yes
Report the best solution
Finish

Figure 2: The flowchart of the CHS

Algorithm 1 The pseudocode of the CHS

```
set initial parameters
best solution=+$\infty$
Initializing
For each population
Generate random position between lower and upper bounds
Evaluation of random position
Update the best solution and current solution
End for
update the best solution
dividing the population into the chasers and ambushers
Main loop
While nfe $\leq$ maxnfe
-First phase
For each population of chasers
If $\left(\frac{\ln(it)}{\ln(maxit)} \times r\right) > 0.5$
Implementing $(3)(a)$
Else if $\left(\frac{\ln(it)}{\ln(maxit)} \times r\right) \leq 0.5$
Implementing $(3)(b)$
end if
Update the best solution and current solution
End for
-Second phase
For each population of ambushers
If $\left(\frac{\ln(it)}{\ln(maxit)} \times r\right) > 0.5$
Implementing $(8)(a)$
Else if $\left(\frac{\ln(it)}{\ln(maxit)} \times r\right) \leq 0.5$
Implementing $(8)(b)$
end if
Update the best solution and current solution
End for
It=it+1
End while (main loop)
Report the best solution
```

4 Evaluation of the CHS algorithm

In this section, two types of test functions are used to evaluate the efficiency of the CHS algorithm. The first category includes classical benchmark functions, and second category includes constrained engineering design problems. In addition, several well-known metaheuristics are used to compare with the CHS algorithm. In the following, classical test functions will be explained and metaheuristics will be compared using test functions. Then, the comparative results of metaheuristics will be shown in the constrained engineering design problems.

4.1 Classic test functions

In Table 1, 20 well-known test functions are shown, which are used to evaluate the CHS algorithm. In this table, the code of each test function, the name of each function, the lower and upper bounds of each function, the dimensions investigated, the formula, the minimum value of each function, and the type of each function in terms of unimodal and multimodal are shown, respectively. According to Table 1, there are six unimodal and 14 multimodal functions. In particular, unimodal functions are useful tools for benchmarking exploitation. In contrast, multimodal functions have many local optima and are suitable tools to evaluate the ability to the exploration of metaheuristics.

In Table 4, perspective views of the classic test functions are displayed. In this study, the number of function evaluations (NFE) is used for the stopping criteria. The value of NFE is equal to $NFE = maxit \times npop$ that $maxit$ denotes a maximum of iteration, and $npop$ denotes the number of population of the algorithm. For the functions, f_1–f_{12}, the value of NFE is 50,000, and for f_{13}–f_{20}, the value of NFE is 180,000. Because the functions f_1–f_{12} are investigated in two dimensions, and the functions f_{13}–f_{20} are investigated in thirty dimensions. Given that as the dimensions increase, it becomes harder to reach the optimal solution; accordingly, it should give more time to the algorithms for reaching better solutions. Therefore, the value of NFE in the functions of f_{13}–f_{20} is greater than f_1–f_{12}. In order to validate the results, the CHS algorithm is compared with four well-known high-performance algorithms that are ACO, DE, genetic algorithm (GA), and PSO. Table 2 shows the results of the study on classical functions in two dimensions. Among these functions, there are three unimodal and nine multimodal functions. In addition, in Tables 2 and 3, three indicators of the best cost, the average of the best costs, and the standard deviation of the best costs have been used to investigate the performance of the CHS algorithm in the functions f_1–f_{12}. The average of the best cost is calculated based on $A = \frac{\sum_{i=1}^{maxit} bestcost_i}{maxit}$ that $bestcost_i$ is the value of the best cost in ith iteration. Also, the standard

Table 1: Benchmark functions (NF= number of functions, D= dimension, U=unimodal, M = multimodal)

NF	Function	Range	D	Formulation	Min	type
(f_1)	Beale	$[-4.5, 4.5]$	2	$f(x) = (1.5 - x_1 + x_1x_2)^2 + (2.25 - x_1 + x_1x_2^2)^2 + (2.625 - x_1 + x_1x_2^3)^2$	0	U
(f_2)	Bird	$[-2\pi, 2\pi]$	2	$f(x) = sin(x_1) . \left[\left\lvert \exp[1-\cos(x_2)]^2 \right\rvert + \cos(x_2)\right] .exp\left[1 - sin(x_1)^2\right] + (x_1 - x_2)^2$	-106.76	M
(f_3)	Booth	$[-10, 10]$	2	$f(x) = (x_1 + 2x_2 - 7)^2 + (2x_1 + x_2 - 5)^2$	0	M
(f_4)	Carrom- Table	$[-10, 10]$	2	$f(x) = \left[\frac{[-\cos^2(x_1) .\cos^2(x_2)].\exp\left[\left\lvert\frac{1-\sqrt{x_1^2+x_2^2}}{\pi}\right\rvert\right]^2}{30}\right]$	-24.157	M
(f_5)	Cross-in-tray	$[-10, 10]$	2	$f(x) = -0.0001. \left[\left\lvert \sin(x_1) .\sin(x_2) .\exp\left[\left\lvert 100 - \frac{\left(\sqrt{x_1^2+x_2^2}\right)}{\pi}\right\rvert\right] + 1\right\rvert\right]^{0.1}$	-2.0626	M
(f_6)	Cross-Leg -Table	$[-10, 10]$	2	$f(x) = -\frac{1}{\left[\left\lvert \sin(x_1) .\sin(x_2) .\exp\left[\left\lvert 100 - \frac{\left(\sqrt{x_1^2+x_2^2}\right)}{\pi}\right\rvert\right] + 1\right\rvert\right]^{0.1}}$	-1	M
(f_7)	Crowned cross	$[-10, 10]$	2	$f(x) = 0.0001 \left[\left\lvert \sin(x_1) .\sin(x_2) .\exp\left[\left\lvert 100 - \frac{\left(\sqrt{x_1^2+x_2^2}\right)}{\pi}\right\rvert\right] + 1\right\rvert\right]^{0.1}$	0.0001	M
(f_8)	Easom	$[-100, 100]$	2	$f(x) = -\cos(x_1) .\cos(x_2) .\exp(\left[-(x_1-\pi)^2 - (x_2-\pi)^2\right]$	-1	U
(f_9)	Himmelblau	$[-5, 5]$	2	$f(x) = (x_1^2 + x_2 - 11)^2 + (x_1 + x_2^2 - 7)^2$	0	M
(f_{10})	Matyas	$[-10, 10]$	2	$f(x) = 0.26\left(x_1^2 + x_2^2\right) - 0.48x_1x_2$	0	U
(f_{11})	Pen-holder	$[-11, 11]$	2	$f(x) = -\exp\left[-\left\lvert \cos(x_1) .\cos(x_2) .\exp\left[\left\lvert 1 - \sqrt{\frac{x_1^2+x_2^2}{\pi}}\right\rvert\right]\right\rvert\right]^{-1}$	-0.9635	M
(f_{12})	Six-hump-camel	$[-5, 5]$	2	$f(x) = \left(4 - 2.1x_1^2 + \frac{x_1^4}{3}\right)x_1^2 + x_1x_2 + \left(4x_2^2 - 4\right)x_2^2$	-1.0316	M
(f_{13})	Ackley	$[-35, 35]$	30	$f(x) = -20\exp\left[-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right] - \exp\left[\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right] + 20 + \exp(1)$	0	M
(f_{14})	Griewank	$[-600, 600]$	30	$f(x) = \sum_{i=1}^{n}\frac{x_i^2}{4000} - \prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	0	M
(f_{15})	Michalewicz	$[0, \pi]$	30	$f(x) = -\sum_{i=1}^{n} sin(x_i) sin^{2m}\left(\frac{ix_i^2}{\pi}\right)$	-29.630	M
(f_{16})	Rastrigin	$[-5.12, 5.12]$	30	$f(x) = 10n + \sum_{i=1}^{n}\left[x_i^2 - 10\cos(2\pi x_i)\right]$	0	M
(f_{17})	Rosenbrock	$[-100, 100]$	30	$f(x) = \sum_{i=1}^{n-1} 100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2$	0	M
(f_{18})	Sphere	$[-100, 100]$	30	$f(x) = \sum_{i=1}^{n} x_i^2$	0	U
(f_{19})	Step	$[-5.12, 5.12]$	30	$f(x) = \sum_{i=1}^{n}(x_i + 0.5)^2$	0	U
(f_{20})	Quartic	$[-1.28, 1.28]$	30	$f(x) = \sum_{i=1}^{n} ix_i^4 + rand(0, 1)$	0	U

deviation of the best cost is calculated based on $std = std(bestcosts)$, in which *bestcosts* is the vector of the best costs. In Table 3, the best cost, the average of the best costs, and the standard deviation of the best costs are calculated. In this table, the functions of Ackley, Griewank, Michalewicz, Rastrigin, Rosenbrock, Sphere, Step, and Quartic are investigated in the thirty dimensions. The Ackley function has plenty of local minimal, and a large hole is located in the middle of this function. The Griewank function has a lot of local minimal. The specific structure of the Michalewicz function makes that this function becomes a hard function. The Rastrigin function is a high multimodal function, but its local minima are distributed regularly. The well-known function of Rosenbrock is a hard function that converging to its optimal global solution has always been a challenge for metaheuristics. The sphere function is unimodal and convex. The Step function is the same as the Sphere function, with the difference that its center is moved. The Quartic function is a unimodal function that is always a hard function among convex functions.

The search history of the CHS algorithm in Ackley, Michalewicz, Rastrigin, and Quartic is shown in Table 5. This table shows that the CHS algorithm uses exploration to search almost all local optimizations, and after approaching the global solution, it has searched for better solutions around it. The results obtained in Table 2 show that, in all test functions, the CHS algorithm has a superior performance and reaches the optimal global solution. According to this table, except for f_8, the CHS has had a good performance, and in functions f_6, f_7, f_{10}, the CHS has had the best performance compared to

Table 2: Comparative results of the CHS with ACO, DE, GA, and PSO in two-dimensional benchmark functions.

Function	parameters	ACO	DE	GA	CHS	PSO
f_1	Best	3.7285E-15	**0.0000E+00**	5.3573E-09	**0.0000E+00**	**0.0000E+00**
	Mean	3.0446E-04	1.7499E-04	1.4618E-04	2.8217E-03	6.1959E-04
	Variance	3.6041E-03	1.4767E-03	2.3619E-03	3.9028E-02	9.8299E-03
f_2	Best	**-1.067E+02**	**-1.067E+02**	**-1.067E+02**	**-1.067E+02**	**-1.067E+02**
	Mean	-1.0657E+02	-1.0670E+02	-1.0671E+02	-1.0670E+02	-1.0675E+02
	Variance	2.3277E+00	6.2301E-01	6.1592E-01	8.0510E-01	1.4927E-01
f_3	Best	**0.0000E+00**	**0.0000E+00**	6.6867E-05	**0.0000E+00**	**0.0000E+00**
	Mean	1.1495E-03	1.3656E-02	2.1538E-04	6.3135E-03	1.9529E-03
	Variance	2.0576E-02	1.8990E-01	1.7110E-03	1.0428E-01	5.6387E-02
f_4	Best	**-2.4157E+01**	**-2.4157E+01**	**-2.4157E+01**	**-2.4157E+01**	**-2.4157E+01**
	Mean	-2.4152E+01	-2.4154E+01	-2.4154E+01	-2.4146E+01	-2.4154E+01
	Variance	9.7867E-02	6.5639E-02	5.5639E-02	2.5004E-01	7.9855E-02
f_5	Best	**-2.0626E+00**	**-2.0626E+00**	**-2.0626E+00**	**-2.0626E+00**	**-2.0626E+00**
	Mean	-2.0625E+00	-2.0624E+00	-2.0625E+00	-2.0626E+00	-2.0626E+00
	Variance	2.2683E-03	2.9736E-03	3.3138E-03	5.8323E-04	1.0088E-03
f_6	Best	-8.4778E-02	-8.4778E-02	-8.4778E-02	**-1.0000E+00**	-8.4778E-02
	Mean	-7.4230E-02	-7.3789E-02	-7.3789E-02	-1.0000E+00	-5.7640E-02
	Variance	2.7300E-02	2.7295E-02	2.7295E-02	0.0000E+00	3.6339E-02
f_7	Best	2.3790E-01	1.1795E-03	1.7427E-02	**1.0000E-04**	**1.0000E-04**
	Mean	4.4454E-01	3.2020E-02	2.8825E-02	1.0092E-03	6.1287E-02
	Variance	9.1072E-02	1.0853E-01	6.8593E-02	2.8750E-02	1.6701E-01
f_8	Best	**-1.0000E+00**	**-1.0000E+00**	**-1.0000E+00**	**-1.0000E+00**	**-1.0000E+00**
	Mean	-9.9294E-01	-9.4630E-01	-9.4630E-01	-9.8772E-01	-9.8969E-01
	Variance	7.2850E-02	2.1878E-01	2.1878E-01	8.9564E-02	9.2793E-02
f_9	Best	**0.0000E+00**	7.8886E-31	7.8886E-31	7.8886E-31	7.8886E-31
	Mean	5.5989E-03	1.2939E-02	1.3149E-02	1.4854E-02	5.6025E-03
	Variance	7.2871E-02	1.6169E-01	1.7269E-01	1.4631E-01	1.2813E-01
f_{10}	Best	3.6536E-14	1.0988E-70	1.0988E-70	**0.0000E+00**	9.3840E-83
	Mean	2.9606E-04	8.5572E-04	8.5572E-04	6.5336E-06	6.4553E-04
	Variance	3.8089E-03	1.1494E-02	1.1494E-02	2.0661E-04	1.4383E-02
f_{11}	Best	**-9.6353E-01**	**-9.6353E-01**	**-9.6353E-01**	**-9.6353E-01**	**-9.6353E-01**
	Mean	-9.6345E-01	-9.6352E-01	-9.6352E-01	-9.6347E-01	-9.6353E-01
	Variance	1.0913E-03	1.1988E-04	1.2088E-03	5.8280E-04	2.5529E-05
f_{12}	Best	**-1.0316E+00**	**-1.0316E+00**	**-1.0316E+00**	**-1.0316E+00**	**-1.0316E+00**
	Mean	-1.0303E+00	-1.0311E+00	-1.0307E+00	-1.0301E+00	-1.0308E+00
	Variance	3.5421E-02	5.7551E-03	1.7266E-02	3.3696E-02	2.3828E-02

Table 3: Comparative results of the CHS with ACO, DE, GA, and PSO in three-dimensional benchmark functions.

Function	parameters	ACO	DE	GA	CHS	PSO
f_{13}	Best	4.4409E-15	1.5099E-14	4.0861E-02	**8.8818E-16**	1.5099E-14
	Mean	3.4812E-01	1.3290E+00	8.8208E-01	4.3144E-03	3.1990E-01
	Variance	2.0814E+00	4.0717E+00	1.9128E+00	1.3531E-01	1.4827E+00
f_{14}	Best	9.8573E-03	**0.0000E+00**	5.9285E-02	**0.0000E+00**	1.2321E-02
	Mean	2.2289E+00	7.7930E+00	1.5710E+00	3.6101E-03	6.6917E-01
	Variance	2.5471E+01	4.4016E+01	1.1665E+01	8.9211E-02	1.0457E+01
f_{15}	Best	-1.5282E+01	-2.8588E+01	**-2.9588E+01**	-2.4338E+01	-2.5882E+01
	Mean	-1.4375E+01	-2.5639E+01	-2.8840E+01	-2.3855E+01	-2.5667E+01
	Variance	1.0662E+00	3.5141E+00	2.2322E+00	1.8568E+00	1.5003E+00
f_{16}	Best	8.9546E+00	3.5499E+01	2.5984E-02	**0.0000E+00**	3.6813E+01
	Mean	3.5185E+01	7.8142E+01	1.2294E+01	1.2679E+00	4.1114E+01
	Variance	6.3628E+01	5.0830E+01	3.1570E+01	1.7385E+01	2.4295E+01
f_{17}	Best	2.3585E+01	2.7599E+01	1.8210E+02	**1.5166E+00**	2.3434E+01
	Mean	7.7561E+07	2.2186E+08	9.5396E+06	3.7934E+06	8.7383E+06
	Variance	1.1298E+09	1.6832E+09	2.0580E+08	1.5632E+08	2.8500E+08
f_{18}	Best	5.9710E-107	8.2297E-28	1.5372E-02	**0.0000E+00**	3.2359E-40
	Mean	2.3492E+02	8.6412E+02	1.5473E+02	1.6111E+01	6.0851E+01
	Variance	2.7076E+03	5.0574E+03	1.2821E+03	5.5154E+02	1.1066E+03
f_{19}	Best	2.7733E-32	1.7133E-30	6.2775E-05	2.9041E-07	**0.0000E+00**
	Mean	5.8458E-01	2.0791E+00	2.4766E-01	9.7130E-02	1.8939E-01
	Variance	6.9647E+00	1.2470E+01	2.4888E+00	1.4607E+00	2.9788E+00
f_{20}	Best	1.6655E-03	9.6106E-03	1.9030E-02	**3.5024E-05**	4.3593E-03
	Mean	2.6485E-01	9.6546E-01	7.6241E-02	2.1654E-03	6.1895E-02
	Variance	3.8147E+00	7.1113E+00	7.0836E-01	6.8662E-02	1.3975E+00

other metaheuristics in terms of best costs. The results in Table 3 show the efficiency of the algorithms in the test functions f_{13} to f_{20}. Based on this table, the performance of the CHS algorithm has been better than other algorithms in terms of the best solution in the functions $\mathrm{f}_{13}, \mathrm{f}_{14}, \mathrm{f}_{16}$, f_{17}, $\mathrm{f}_{18}, \mathrm{f}_{20}$. Furthermore, the CHS algorithm has reached the optimal solution in $\mathrm{f}_{14}, \mathrm{f}_{16}$, f_{18}, f_{20}, and the difference between the optimal solution and the obtained result in the functions $\mathrm{f}_{13}, \mathrm{f}_{15}, \mathrm{f}_{17}, \mathrm{f}_{19}$ is low. The extreme convergence of the CHS algorithm in multimodal functions $\mathrm{f}_{13}, \mathrm{f}_{14}, \mathrm{f}_{16}, \mathrm{f}_{17}$ and the unimodal functions $\mathrm{f}_{18}, \mathrm{f}_{19}$, f_{20} are quite evident in Table 6. In total, among the 20 classic functions, the CHS algorithm has reached the optimal solution in 17 functions, where among 17 functions, 12 functions are multimodal and five of them are unimodal. Also, in the six functions, the CHS algorithm has had better results compared to the other algorithms. In all of the classical test functions except for f_9, in terms of the best solution, none of the algorithms is able to provide a better solution than the CHS. These results indicate a good balance between the exploration and exploitation in the CHS algorithm.

4.2 Constrained engineering design problems

In recent years, constrained engineering design problems have been used in various papers to examine the performance of metaheuristic algorithms. For example, six constrained engineering design problems to evaluate the crow

Table 4: Perspective view for benchmark functions

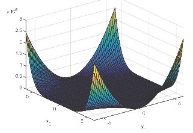
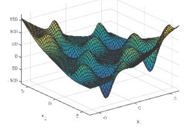
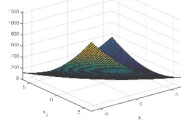
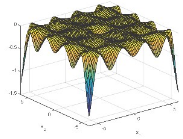
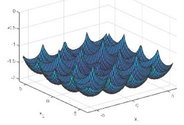
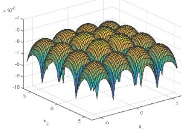
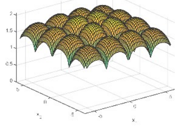
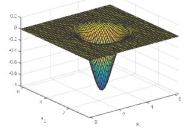
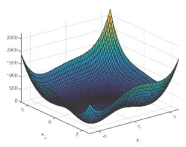
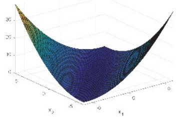
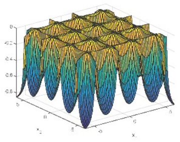
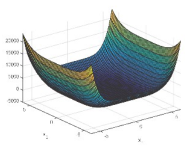
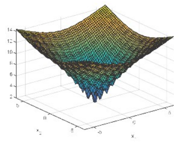
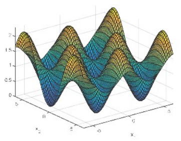
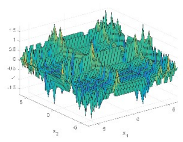
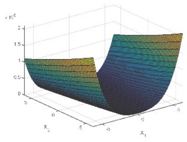
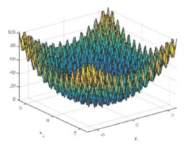
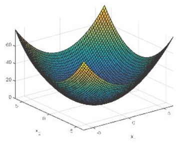
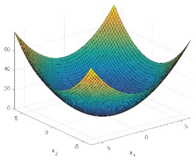
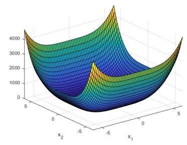
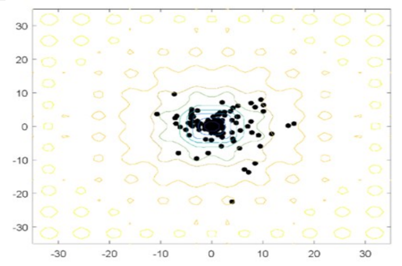
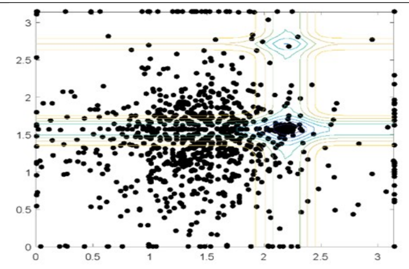
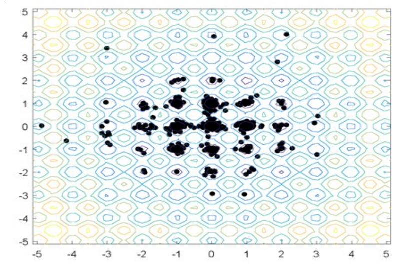
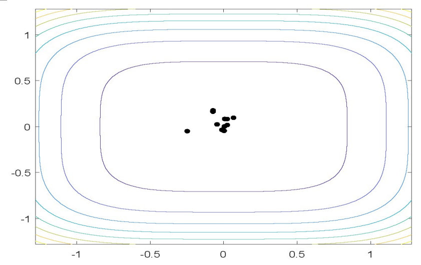
Beale(f_1)	Bird(f_2)	Booth(f_3)	Carrom-table(f_4)
			
Cross-in-tray(f_5)	Cross-leg-table(f_6)	Crowned cross (f_7)	Easom(f_8)
			
Himmelblau(f_9)	Matyas(f_{10})	Pen-holder(f_{11})	Six-hump-camel(f_{12})
			
Ackley(f_{13})	Griewank(f_{14})	Michalewicz(f_{15})	Rosenbrock(f_{16})
			
Rastrigin(f_{17})	Sphere(f_{18})	Step(f_{19})	Qartic(f_{20})
			

Table 5: Search history of the CHS

$f13$	$f15$
	
$f16$	$f20$
	

search algorithm were used in [5], five constrained engineering design problems were used to evaluate the league championship algorithm in [22], eight constrained engineering design problems to examine the mine blast algorithm were used in [39], and seven constrained engineering design problems were solved with water cycle algorithm in [15]. In this section, four constrained engineering design problems, including a three-bar design problem, pressure vessel design problem, Tension/compression spring design, and welded beam design problem, are used to study the performance of the CHS algorithm. Creating an effective balance in searching between acceptable and unacceptable solutions is one of the challenges in solving engineering design problems. Various strategies have emerged to handle unacceptable areas in evolutionary computing. Transferring constraints to the objective function with a penalty factor and transforming the optimization problem into an unstrained problem [9] is the basic idea of the penalty function. Additive penalty functions [9, 30] will be used to solve this problem in this paper, which is one of the famous methods. In order to investigate the performance of the CHS algorithm, several indicators are used, including the values of the decision variables, the constraint values, and the best solution. In order to compare the performance, the following four optimization algorithms were used:

1. FA (firefly algorithm)
2. Harmony Search (HS)
3. SFLA (shuffled frog-leaping algorithm)

Table 6: Compare performance algorithms

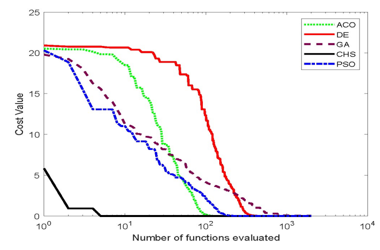
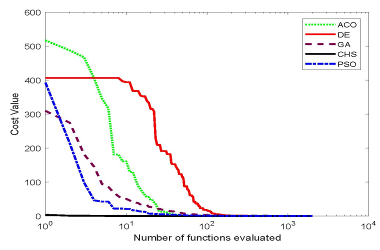
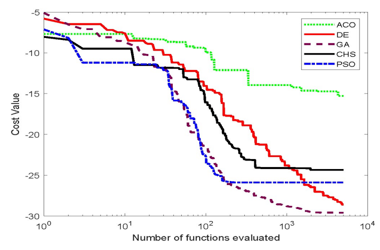
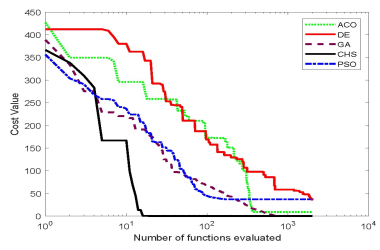
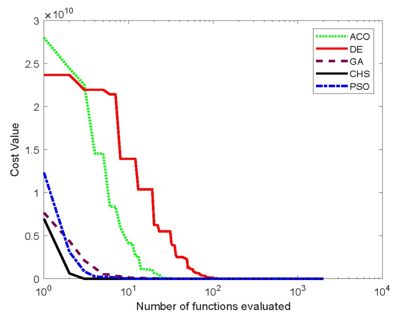
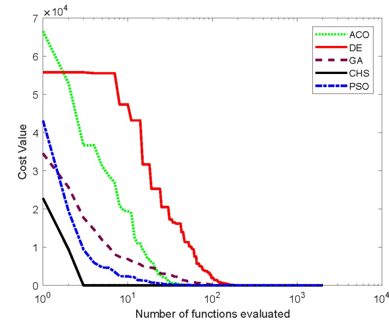
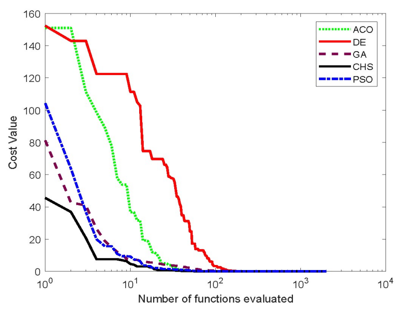
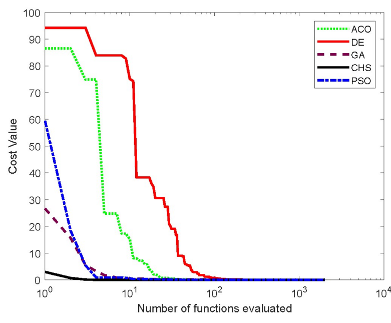
$\mathbf{f_{13}}$	$\mathbf{f_{14}}$
	
$\mathbf{f_{15}}$	$\mathbf{f_{16}}$
	
$\mathbf{f_{17}}$	$\mathbf{f_{18}}$
	
$\mathbf{f_{19}}$	$\mathbf{f_{20}}$
	

Table 7: Comparative results of the CHS with FA, HS, SFLA, and TLBO in three-bar truss design problem.

Parameters	FA	CHS	HS	SFLA	TLBO
x_1	7.886774424E-01	7.886783330E-01	7.880650859E-01	7.865946407E-01	7.886790873E-01
x_2	4.082417630E-01	4.082392461E-01	4.099765093E-01	4.141648543E-01	4.082371275E-01
g_1	-1.114419668E-11	-1.532325600E-09	4.653166741E-12	6.646239115E-12	-1.276911088E-08
g_2	-1.464109036	-1.464111898	-1.462138552	-1.457394322	-1.464114312
g_3	-5.358909643E-01	-5.358881039E-01	-5.378614480E-01	-5.426056785E-01	-5.358857010E-01
Best Value	**2.638958434E+02**	**2.638958436E+02**	**2.638961174E+02**	**2.638990472E+02**	**2.638958451E+02**

4. Teaching-learning-based optimization (TLBO)

The three-bar truss design problem Minimizing the volume of a statistically loaded three-bar truss with respect to stress limits (σ) [38] is the main objective in the three-bar truss design problem. The mathematical model of the problem is based on the following equation:

$$\begin{aligned}
\text{Min} f(x) &= (2\sqrt{2}x_1 + x_2) \times l \\
s.t. & \\
g_1(x) &= \frac{\sqrt{2}x_1 + x_2}{\sqrt{2}x_1^2 + 2x_1x_2} p - \sigma \leq 0, \\
g_2(x) &= \frac{x_2}{\sqrt{2}x_1^2 + 2x_1x_2} p - \sigma \leq 0, \\
g_3(x) &= \frac{1}{\sqrt{2}x_2 + x_1} p - \sigma \leq 0, \\
0 \leq & x_i \leq 1, \qquad i = 1, 2, \\
l = 100\ cm, \quad & p = 2\frac{kN}{cm^2}, \quad \sigma = 2\frac{kN}{cm^2}.
\end{aligned} \tag{7}$$

This problem has two continuous decision variables and three unequal constraints. In Figure 3, the three-bar truss design is represented schematically. Figure 4 shows the convergence rate of the CHS algorithm in finding an optimal solution to the three-bar truss design problem.

The results of Table 4 show that there are no significant difference between decision variables, constraints values, and the best solutions. The best result is related to the FA algorithm, and then the CHS algorithm outperforms HS, SFLA, and TLBO. The amount of NFE for solving this problem is assumed 24,000, and the results of the decision variables and the best solution are equal to

$$X^* = (0.7886783, 0.4082392),$$

$$f(X^*) = 263.8958436.$$

The Pressure vessel design problem As shown in Figure 5, in the design of a pressure vessel, a cylindrical pressure tank is used that has two joints in

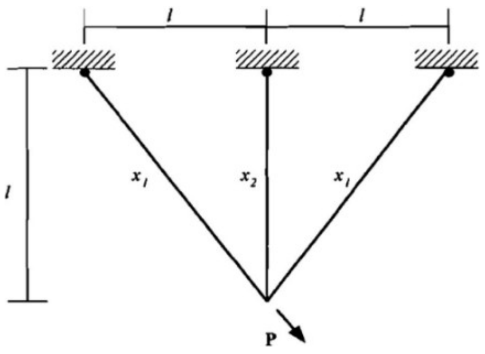


Figure 3: The three-bar truss design problem

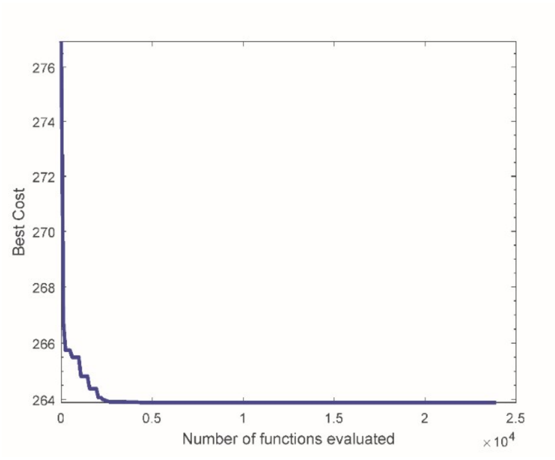


Figure 4: Convergence rate of the CHS for finding best solution of three-bar truss design problem

two cylindrical heads. The purpose of this problem is to minimize the weight of the pressure vessel, including the weight of the shell and the welding lines. The mathematical model of this problem is shown as follows:

$$
\begin{aligned}
\mathrm{Min} f(x) \;&= 0.6224x_1x_3x_4 + 1.7781x_2x_3^2 + 3.1661x_1^2x_4 + 19.84x_1^2x_3 \\
&s.t. \\
&g_1(x) = -x_1 + 0.0193x_3 \le 0, \\
&g_2(x) = -x_2 + 0.00954x_3 \le 0, \\
&g_3(x) = -\pi x_3^2 x_4 - \frac{4}{3}\pi x_3^3 + 1,296,000 \le 0 \\
&g_4(x) = x_4 - 240 \le 0, \\
&0 \le x_i \le 100, \quad i = 1,2, \\
&10 \le x_i \le 200, \quad i = 3,4.
\end{aligned} \tag{8}
$$

Based on this equation, (x_1 or T_s) and (x_2 or T_h) show the thickness of the shell and the thickness of the head, respectively. (x_3 or R) represents the inner radius, and (x_4 or L) shows the cylindrical section of the vessel. In order to solve this mixed-integer problem (MLP), the variables T_s and T_h are rounded, which are a coefficient of 0.0625, and R and L are continuous variables. Figure 6 shows the convergence rate of the CHS algorithm to find the best solution to this problem.

In order to solve this problem, the algorithms are implemented with NFE = 180,000. Based on Table 5, the CHS and TLBO algorithms have had better performance than the FA, HS, and SFLA algorithms. The values of decision variables and the value of the objective function at the best solution are

$$X* = (0.8125, 0.4375, 42.0984456, 176.6365958)\ ,$$

$$f(X*) = 6059.7143350.$$

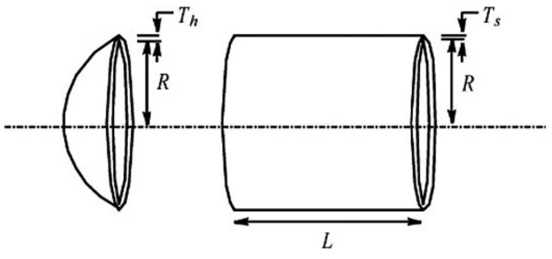


Figure 5: Pressure vessel design problem

Tension/compression spring design optimization problem The objective of this problem is to minimize tension/spring weight with a linear constraint and three nonlinear constraints [8]. Based on the following equation, the diameter of the wire (x_1), the mean diameter of the coil (x_3), and the number of active coils (x_2), are the decision variables of this problem:

Table 8: Comparative results of the CHS with FA, HS, SFLA, and TLBO in Pressure vessel design problem.

Parameters	FA	CHS	HS	SFLA	TLBO
x_1	$0.8750(14 \times 0.0625)$	$0.8125(13 \times 0.0625)$	$1(16 \times 0.0625)$	$1.1875(19 \times 0.0625)$	$0.8125(13 \times 0.0625)$
x_2	$0.4375(7 \times 0.0625)$	$0.4375(7 \times 0.0625)$	$0.5000(8 \times 0.0625)$	$0.5625(9 \times 0.0625)$	$0.4375(7 \times 0.0625)$
x_3	4.533678756E+01	4.209844560E+01	5.121941472E+01	5.896226415E+01	4.209844560E+01
x_4	1.402538467E+02	1.766365958E+02	8.895572796E+01	4.004432558E+01	1.766365958E+02
g_1	3.330669074E-16	2.220446049E-16	-1.146529598E-02	-4.952830189E-02	-1.744160372E-13
g_2	-4.987046632E-03	-3.588082902E-02	-1.136678361E-02	3.330669074E-16	-3.588082902E-02
g_3	0	0	-1.304185484E-03	0	-3.213062882E-08
g_4	-9.974615329E+01	-6.336340416E+01	-1.510442720E+02	-1.999556744E+02	-6.336340416E+01
Best Value	**6.090526202E+03**	**6.059714335E+03**	**6.466011401E+03**	**7.050673234E+03**	**6.059714335E+03**

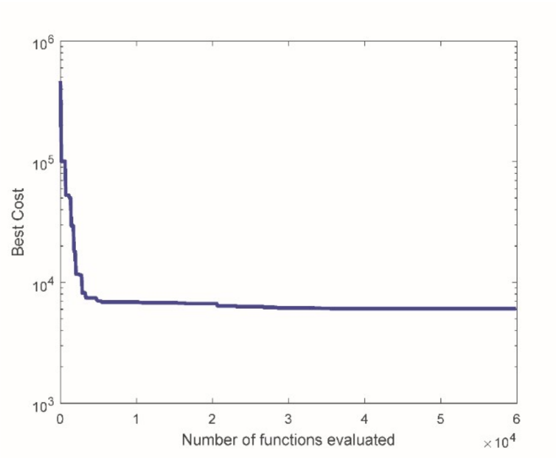


Figure 6: Convergence rate of the CHS for finding the best solution of pressure vessel design problem

$$\begin{aligned}
\text{Min} f\left(x\right) &= \left(x_3+2\right) x_2 x_1^2 \\
s.t. & \\
g_1\left(x\right) &= 1-\frac{x_2^3 x_3}{71,785 x_1^4} \leq 0, \\
g_2\left(x\right) &= \frac{4x_2^2 - x_1 x_2}{12,566\left(x_2 x_1^3 - x_1^4\right)} + \frac{1}{5108 x_1^2} - 1 \leq 0, \\
g_3\left(x\right) &= 1-\frac{140.45 x_1}{x_2^2 x_3} \leq 0, \\
g_4\left(x\right) &= \frac{x_1+x_2}{1.5} - 1 \leq 0, \\
0.05 \leq x_1 \leq 2, &\qquad 0.25 \leq x_2 \leq 1.3, \qquad 2 \leq x_3 \leq 15.
\end{aligned} \tag{9}$$

The schematic representation of the problem is shown in Figure 7. Figure 8 shows the convergence rate of the CHS algorithm in finding the optimal solution to the three-tension/compression spring design optimization problem.

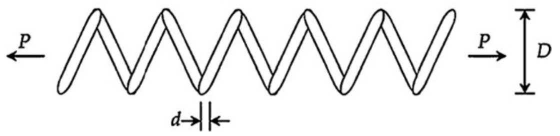


Figure 7: Tension/compression spring structure

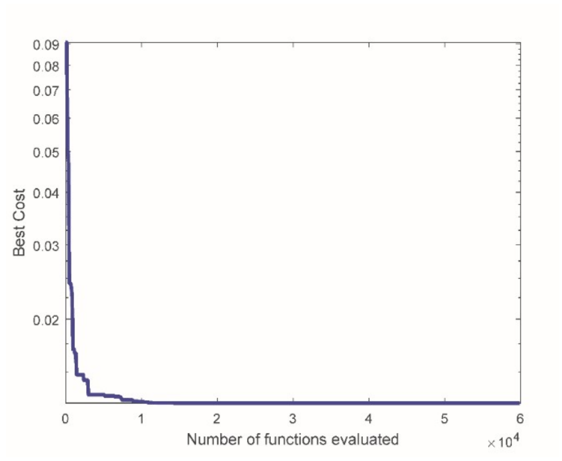


Figure 8: Convergence rate of the CHS for finding best solution of Tension/compression spring design optimization problem

In order to evaluate the performance of the algorithms in this problem, the value of NFE is considered 60,000. The comparative results in Table 6 show that the best performance is related to the CHS algorithm. The FA and TLBO algorithms have comparative results that show a better performance than HS and TLBO. The decision variables and the value of the objective function at the best solution are

$$X^* = (0.051840, 0.360373, 11.077829),$$

$$f(X^*) = 0.0126657.$$

Table 9: Comparative results of THE CHS with FA, HS, SFLA, and TLBO in Tension/compression spring design optimization problem.

Parameters	**FA**	**CHS**	**HS**	**SFLA**	**TLBO**
x_1	5.184988445E-02	5.184055530E-02	5.978313881E-02	5.743674882E-02	5.196149787E-02
x_2	3.605990697E-01	3.603732438E-01	5.845192287E-01	5.114076090E-01	3.633072944E-01
x_3	1.106499114E+01	1.107782927E+01	4.591474470	5.841058341	1.091284592E+01
g_1	5.184988445E-02	-2.220446049E-16	-5.553424742E-08	-2.220446049E-16	-5.536883352E-08
g_2	0	-2.220446049E-16	-5.749327642E-07	-3.330669074E-16	-1.695190543E-08
g_3	-4.061383701	-4.060945295	-4.352426075	-4.280629258	-4.066606162
g_4	-7.250340306E-01	-7.251908006E-01	-5.704650883E-01	-6.207704281E-01	-7.231541385E-01
Best Value	**1.266570321E-02**	**1.266565028E-02**	**1.377015419E-02**	**1.322883405E-02**	**1.266658115E-02**

The welded beam design problem The aim of this problem is to minimize the cost of the welded beam based on the shear stress τ, bending stress in the beam σ, buckling load on the bar $\mathrm{p_c}$, and end deflection on the beam δ. Based on (10), the decision variables are the weld thickness (x_1), the weld length (x_2), the height of the bar (x_3), and the thickness of the bar (x_4). In Figure 9, the welded beam design problem is shown schematically. Figure 10 shows the convergence rate of the CHS algorithm in finding the optimal solution to this problem. Equation (10) shows that this problem has two linear and five nonlinear unequal constraints as follows:

$$\begin{aligned}
\mathrm{Min} f(x) =& 1.10471x_1^2x_2 + 0.004811x_3x_4(14+x_2)\\
s.t.&\\
g_1(x) =& \tau_x - \tau_{max} \leq 0,\\
g_2(x) =& \sigma_x - \sigma_{max} \leq 0,\\
g_3(x) =& x_1 - x_4 \leq 0,\\
g_4(x) =& 1.10471x_1^2 + 0.04811x_3x_4(14+x_2) - 5 \leq 0,\\
g_5(x) =& 0.125 - x_1 \leq 0,\\
g_6(x) =& \delta_x - \delta_{max} \leq 0,\\
g_7(x) =& p - p_c(x) \leq 0,
\end{aligned} \tag{10}$$

where

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

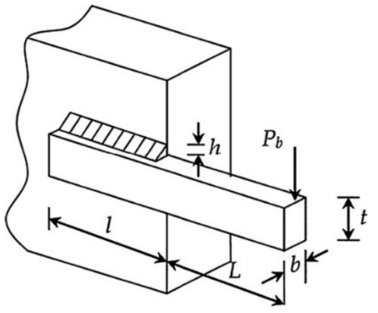


Figure 9: The welded beam design problem

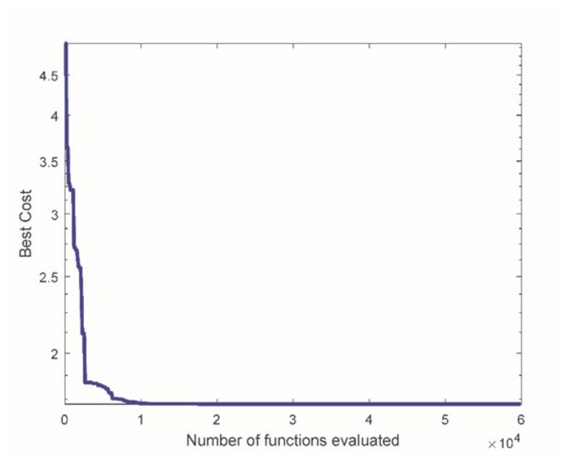


Figure 10: Convergence rate of the CHS for finding best solution of Welded beam design problem

Based on Table 7, with $NFE = 60,000$, each of the three algorithms, the FA, CHS, and TLBO algorithms, have had the same results in the values of decision variables, constraint values, and best solutions that have outperformed HS and SFLA. The decision variables and the value of the objective function at the best solution are

$$X^* = (0.2057296, 3.4704887, 9.0366239, 0.2057296)\,,$$

$$f(X^*) = 1.7248523.$$

Table 10: Comparative results of The CHS with FA, HS, SFLA, and TLBO in Welded beam design problem.

Parameters	**FA**	**CHS**	**HS**	**SFLA**	**TLBO**
x_1	2.057296398E-01	2.057296398E-01	2.350586911E-01	2.058573368E-01	2.057296398E-01
x_2	3.470488665	3.470488665	3.166035310	3.468820796	3.470488665
x_3	9.036623910	9.036623910	8.363477126	9.033820685	9.036623910
x_4	2.057296398E-01	2.057296398E-01	2.401792830E-01	2.058573368E-01	2.057296398E-01
g_1	0	0	-1.177816867E-02	-1.818989404E-12	0
g_2	-7.275957614E-12	-7.275957614E-12	-6.404832093E-04	0	-2.182787284E-11
g_3	6.695199950E-12	6.659200968E-12	-5.120591985E-03	8.307909916E-12	6.767586491E-12
g_4	-3.432983785	-3.432983785	-3.335285590	-3.432642630	-3.432983785
g_5	-8.072963979E-02	-8.072963979E-02	-1.100586911E-01	-8.085733680E-02	-8.072963979E-02
g_6	-2.355403226E-01	-2.355403226E-01	-2.343765145E-01	-2.355358357E-01	-2.355403226E-01
g_7	0	0	-3.061301599E+03	-9.953428131	0
Best Value	**1.724852309**	**1.724852309**	**1.852177649**	**1.725311383**	**1.724852309**

5 Conclusion

In this study, the CHS (crocodile optimization algorithm) was investigated as a swarm intelligence algorithm. The CHS algorithm was classified as the population-based algorithm that simulates the behavior of the crocodiles in finding food. Crocodiles have proper strategies while hunting. In this way, when hunting, the population is divided into two groups: the chasers and the ambushers. The chasers direct the prey to the attacking area without catching it, and the ambushers hide in the attacking area and attack fishes in the attacking area. Using this strategy, the prey was directed to the attacking area and eventually was hunted by the ambushers and chasers. In designing the CHS, two main equations were proposed for each group. The structure of the algorithm was designed in such a way that explorations are done in the first iterations, and exploitations are done in the last iterations. In order to evaluate the efficiency of the CHS algorithm, 20 classical test functions and four constrained engineering design problems were used. In classical benchmarks, 12 classic test functions were in two dimensions and eight classic test functions were investigated in 30 dimensions. In sum, it can be said that in the classical test functions, the CHS algorithm shows a better performance than the other algorithms in terms of convergence rate in less iteration and the exploration and exploitation capability. In addition, in constrained engineering design problems, the CHS algorithm was outperformed and able to produce good results. This performance showed that there is a good balance between exploration and exploitation in the CHS algorithm and that the equations are correctly designed.

References

1. Abdel-Basset, M., Abdel-Fatah, L. and Sangaiah, A.K. *Metaheuristic algorithms: A comprehensive review*, Computational intelligence for multimedia big data on the cloud with engineering applications, 2018, Elsevier,

185–231.

2. Akay, B. and Karaboga, D. *A modified artificial bee colony algorithm for real-parameter optimization,* Inf. Sci. 192 (2012), 120–142.

3. Alba, E. and Dorronsoro, B. *The exploration/exploitation tradeoff in dynamic cellular genetic algorithms,* IEEE Trans. Evol. Comput. 9(2) (2005), 126–142.

4. Askarzadeh, A. *Bird mating optimizer: an optimization algorithm inspired by bird mating strategies,* Commun. Nonlinear Sci. Numer. Simul. 19(4) (2014), 1213–1228.

5. Askarzadeh, A. *A novel metaheuristic method for solving constrained engineering optimization problems: crow search algorithm,* Comput. Struct. 169 (2016), 1–12.

6. Balavand, A. *A new feature clustering method based on crocodiles hunting strategy optimization algorithm for classification of MRI images,* Vis. Comput. (2021) 1–30.

7. Clerc, M., *Particle swarm optimization*, Vol. 93. John Wiley & Sons, 2010.

8. Coello, C.A.C., *Use of a self-adaptive penalty approach for engineering optimization problems,* Comput. Ind. 41(2) (2000), 113–127.

9. Coello, C.A.C., *Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art,* Comput. Methods Appl. Mech. Eng. 191(11-12) (2002), 1245–1287.

10. Cuevas, E., Cienfuegos, M., Zaldívar, D., and Pérez-Cisneros, M.*A swarm optimization algorithm inspired in the behavior of the social-spider,* Expert Syst. Appl. 40 (16) (2013), 6374–6384.

11. Dasgupta, D. *An overview of artificial immune systems and their applications, in Artificial immune systems and their applications,* Artificial immune systems and their applications (1993), 3–21.

12. Dinets, V. *Apparent coordination and collaboration in cooperatively hunting crocodilians,* Ethol. Ecol. Evol. 27(2) (2015), 244–250.

13. Dorigo, M. and Gambardella, L.M. *Ant colony system: a cooperative learning approach to the traveling salesman problem,* IEEE Trans. Evol. Comput. 1(1) (1997), 53–66.

14. Duman, E., Uysal, M. and Alkaya, A.F. *Migrating birds optimization: A new metaheuristic approach and its performance on quadratic assignment problem,* Inform. Sci. 217 (2012), 65–77.

15. Eskandar, H., Sadollah, A., Bahreininejad, A., and Hamdi, M. *Water cycle algorithm-A novel metaheuristic optimization method for solving constrained engineering optimization problems,* Comput. Struct. 110 (2012), 151–166.

16. Eusuff, M., Lansey, K. and Pasha, F. *Shuffled frog-leaping algorithm: a memetic meta-heuristic for discrete optimization,* Eng. Optim. 38(2) (2006), 129–154.

17. Faramarzi, A., Heidarinejad, M., Mirjalili, S., and Gandomi, A.H. *Marine predators algorithm: A nature-inspired Metaheuristic,* Expert Syst. Appl. 152 (2020) 113377.

18. Faris, H., Aljarah, I., Al-Betar, M. A., and Mirjalili, S. *Grey wolf optimizer: a review of recent variants and applications,* Neural Comput. Appl. 30(2) (2018), 413-435.

19. Heidari, A. A., Mirjalili, S., Faris, H., Aljarah, I., Mafarja, M., and Chen, H. *Harris hawks optimization: Algorithm and applications,* Future Gener. Comput. Syst. 97 (2019), 849–872.

20. Jain, M., Singh, V. and Rani, A. *A novel nature-inspired algorithm for optimization: Squirrel search algorithm,* Swarm Evol. Comput. 44 (2019), 148–175.

21. José-García, A. and Gómez-Flores, W. *Automatic clustering using nature-inspired metaheuristics: A survey,* Appl. Soft Comput. 41 (2016), 192–213.

22. Kashan, A.H. *An efficient algorithm for constrained global optimization and application to mechanical engineering design: League championship algorithm (LCA),* Comput Aided Des. 43(12) (2011), 1769–1792.

23. Kaveh, A. and Farhoudi, N. *A new optimization method: Dolphin echolocation,*Adv. Eng. Softw. 59 (2013), 53–70.

24. Kennedy, J. and Eberhart, R. *Particle swarm optimization,* in Proceedings of ICNN'95-international conference on neural networks. 1995.

25. Kiran, M.S., *TSA: Tree-seed algorithm for continuous optimization,* Expert Syst. Appl. 42(19) (2015), 6686–6698.

26. Kirkpatrick, S., Gelatt, C.D. and Vecchi, M.P. *Optimization by simulated annealing*, Science 220(4598) (1983), 671–680.

27. Li, J.Q., Sang, H.Y., Han, Y.Y., Wang, C.G. and Gao, K.Z. *Efficient multi-objective optimization algorithm for hybrid flow shop scheduling problems with setup energy consumptions,* J. Clean. Prod. 181 (2018), 584–598.

28. Lin, L. and Gen, M. *Auto-tuning strategy for evolutionary algorithms: balancing between exploration and exploitation,* Soft Comput. 13, (2) (2009), 157–168.

29. Martin, R. and Stephen, W. *Termite: A swarm intelligent routing algorithm for mobilewireless Ad-Hoc networks,* Stigmergic optimization. Springer, Berlin, Heidelberg, 2006. 155–184.

30. Michalewicz, Z. *A Survey of constraint handling techniques in evolutionary computation methods,* Evol Comput. 4 (1995), 135–155.

31. Mirjalili, S. *Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm,* Knowl. Based Syst. 89 (2015), 228–249.

32. Mirjalili, S. and Lewis, A. *The whale optimization algorithm,* Adv. Eng. Softw., 95 (2016), 51–67.

33. Mirjalili, S., Mirjalili, S.M. and Lewis, A. *Grey wolf optimizer,* Adv. Eng. Softw. 69 (2014), 46–61.

34. Mucherino, A. and Seref. O. *Monkey search: A novel metaheuristic search for global optimization*, AIP conference proceedings. Vol. 953. No. 1. American Institute of Physics, 2007.

35. Nezhad, A.M., Shandiz, R.A. and Jahromi, A.E. *A particle swarm-BFGS algorithm for nonlinear programming problems,* Comput. Oper. Res. 40(4) (2013), 963–972.

36. Olorunda, O. and Engelbrecht. A.P. *Measuring exploration/exploitation in particle swarms using swarm diversity,*In 2008 IEEE congress on evolutionary computation (IEEE world congress on computational intelligence), pp. 1128–1134. IEEE, 2008.

37. Pan, W.-T. *A new fruit fly optimization algorithm: taking the financial distress model as an example,* Knowl. Based Syst. 26 (2012), 69–74.

38. Ray, T. and Liew, K.M. *Society and civilization: An optimization algorithm based on the simulation of social behavior,* IEEE Trans. Evol. Comput. 7(4) (2003), 386–396.

39. Sadollah, A., Bahreininejad, A., Eskandar, H., and Hamdi, M. *Mine blast algorithm: A new population based algorithm for solving constrained engineering optimization problems,* Appl. Soft Comput. 13(5) (2013), 2592–2612.

40. Saremi, S., Mirjalili, S. and Lewis, A. *Grasshopper optimisation algorithm: theory and application,* Adv. Eng. Softw. 105 (2017), 30–47.

41. Yang, X.-S., *Firefly algorithm, stochastic test functions and design optimisation,*Int. J. Bio-Inspired Comput. 2(2) (2010), 78–84.

42. Yang, X.-S. and Deb. S. *Cuckoo search via Lévy flights,* In 2009 World congress on nature & biologically inspired computing (NaBIC), pp. 210–214. IEEE, 2009.

43. Yang, X.-S. and He, X. *Bat algorithm: literature review and applications,* Int. J. Bio-Inspired Comput. 5(3) (2013), 141–149.

44. Yapici, H. and Cetinkaya, N. *A new meta-heuristic optimizer: Pathfinder algorithm,* Appl. Soft Comput. 78 (2019), 545–568.

45. Zhang, J., Xiao, M., Gao, L. and Pan, Q. *Queuing search algorithm: A novel metaheuristic algorithm for solving engineering optimization problems,* Appl. Math. Model. 63 (2018), 464–490.

How to cite this article
A.R. Balavand Crocodile Hunting Strategy (CHS): A comparative study using benchmark functions. *Iranian Journal of Numerical Analysis and Optimization*, 2022; 12(2): 397-425. doi: DOI:10.22067/ijnao.2022.71418.1049.