Progress in Artificial Intelligence (2021) 10:349–374
https://doi.org/10.1007/s13748-021-00244-4

REGULAR PAPER

A chaotic and hybrid gray wolf-whale algorithm for solving continuous optimization problems

Kayvan Asghari[1] · Mohammad Masdari[1] · Farhad Soleimanian Gharehchopogh[1] · Rahim Saneifard[2]

Received: 17 September 2020 / Accepted: 3 April 2021 / Published online: 26 April 2021

Abstract
The gray wolf optimizer (GWO) and the whale optimization algorithm (WOA) are two esteemed optimization algorithms, and their various modified versions are proposed in recent years for different applications. The GWO and WOA simulate the hunting method of gray wolves and humpback whales, respectively. These algorithms have several operators for moving the search agents toward the optimum solution in the search space. But, the GWO and WOA encounter some problems such as falling in local optima and slow convergence. Various proposals have been presented so far to develop innovative and novel meta-heuristic optimization methods. Some of them are based on adding special evolutionary operators or local search steps to existing algorithms. Some others are established based on the combination of previous methods or applying the chaos theory in them. A novel hybrid method defined as chaotic GWO and WOA (CGWW) is proposed in this paper by modifying the WOA, merging it with GWO, and applying the chaotic maps. Also, the chaotic maps have been used in the CGWW algorithm to adjust the movement parameters and initialize the search agents. The combination of different operators of the mentioned algorithms and using the chaotic maps increases the exploration and exploitation power of the proposed algorithm and thus causes to obtain better results. Twenty-three mathematical benchmark functions are used to evaluate the CGWW algorithm. Besides, the proposed algorithm is applied for solving the feature selection problem in intrusion detection systems, which is intrinsically multi-objective. The proposed algorithm finds competitive results in contrast to other well-known meta-heuristic algorithms in most of the experiments. It can avoid local optima and find the global optimum in most cases using its balanced exploration and exploitation ability.

Keywords Gray wolf optimization · Whale optimization algorithm · Chaotic maps · Roulette wheel operator · Hybrid optimization method

1 Introduction

In recent years, expanding the range of complex industrial and theoretical problems has increased the need for advanced optimization algorithms. Some of these algorithms are called meta-heuristics, and part of them are inspired by the social behaviors of creatures or natural phenomena. Due to the random mechanisms of meta-heuristic algorithms, they can quickly escape from local optima and find the best solution.

1.1 Optimization algorithms

Typically, every meta-heuristic algorithm has two main phases of exploration and exploitation. In the exploration phase, the algorithm tries to probe the whole search space of the problem, and the movements of search agents are long

✉ Mohammad Masdari
m.masdari@iaurmia.ac.ir; dr.m.masdari@gmail.com

Kayvan Asghari
k.asghari@yahoo.com

Farhad Soleimanian Gharehchopogh
bonab.farhad@gmail.com

Rahim Saneifard
srsaneeifard@yahoo.com

[1] Department of Computer Engineering, Urmia Branch, Islamic Azad University, Urmia, Iran

[2] Department of Applied Mathematics, Urmia Branch, Islamic Azad University, Urmia, Iran

and random. But in the exploitation phase, the search agents are moving around the promising solutions with short steps [1]. For developing a new meta-heuristic algorithm, finding a balance between exploration and exploitation is a significant challenge[2, 3]. Various meta-heuristic optimization algorithms with high computational efficiency are introduced for global search [4]. One of the well-known nature-inspired optimization algorithms is the genetic algorithm (GA) [5], which imitates the Darwinian theory of evolution. Operators like selection, crossover, and mutation support the GA to generate new solutions and avoid local optima. Some of the other meta-heuristic algorithms that are applied for comparisons in this paper are as follows: the particle swarm optimizer (PSO) [6], ant lion optimizer (ALO) [7], artificial bee colony (ABC) [8], artificial electric field algorithm (AEFA)[9], salp swarm algorithm (SSA) [10], biogeography-based optimization [11], satin bower bird optimization (SBO)[12], poor and rich optimization method (PRO) [13], water evaporation optimization (WEO) [14], multi-verse optimizer (MVO) [15], moth search (MS) [16], grasshopper optimization algorithm (GOA) [17], barnacles mating optimizer (BMO) [18], farmland fertility [19], symbiotic organisms search (SOS) [20, 21], butterfly optimization algorithm (BOA) [22], Harris hawks optimizer (HHO) [23], interactive search algorithm (ISA) [24], gray wolf optimizer (GWO) [25], and whale optimization algorithm (WOA) [3].

The GWO algorithm [25] mimics the hierarchical leadership of gray wolves as a social behavior for hunting. Four types of wolves participate in this hierarchy called alpha, beta, delta, and omega. According to the experiments on unimodal benchmark functions, the GWO presents a superior performance in the exploitation phase. Also, the exploration power of the GWO is confirmed by the multimodal functions. The GWO has a high performance for engineering design and real problems with unknown search spaces. Mirjalili et al. proved the superiority of GWO to some other well-known optimization algorithms, which is related to the high search speed and precision along with the simplicity of the GWO [25]. All the mentioned strength points reveal the high research value of the GWO for applying as a part of new combinatorial optimization algorithms.

The WOA is another optimization algorithm that is inspired by the feeding behavior of humpback whales in nature. Three main activities in the WOA algorithm are searching for, encircling, and attacking the prey. The WOA algorithm presents a high performance in solving different kinds of problems like feature selection [26, 27] and data clustering [28, 29]. The GWO and WOA algorithms have some drawbacks like slow convergence and getting stuck in the local optimums.

As the No Free Lunch (NFL) theorem [30] says, most of the heuristic methods are suitable for solving specific problems, and they are not applicable for all fields. Various approaches have been proposed so far to develop effective and innovative meta-heuristics. Some of them are made based on improving the existing methods by adding special evolutionary operators or local search steps. Some others are made based on the combination of previous algorithms or applying the chaotic maps in them. The combinatorial algorithms usually benefit from one algorithm's high exploration and the other one's exploitation ability [31]. Numerous combinatory meta-heuristics are introduced so far which some of the recent ones are as follows: GSA-GA [32], PSO-GA [33], GA-GSA [34], PS-ABC [35], TVAC-GSA-PSO [36].

Applying random number generators is necessary for implementing most of the optimization algorithms. The chaos theory in mathematics is related to dynamic systems, which have infinite unstable and periodic variations. These systems are sensitive to the initial conditions, so a small change in the initial conditions may result in large variations in the outcomes. Chaos-based systems exhibit similar behavior to random systems. One of the applications of chaotic maps in the meta-heuristic optimization algorithms is adjusting the parameters, which increases the convergence speed and possibility of escaping from the local optima [37, 38]. Different types of chaotic maps are applied in the following algorithms: the PSO [39], harmony search (HS) [40], ABC [41], imperialist competitive algorithm (ICA) [42], firefly algorithm (FA) [43], Krill-Herd (KH) [44], butterfly optimization algorithm (BOA) [45], GWO [46], the WOA[47], and SSA [48]. The motivation of introducing a new hybrid algorithm in this paper was developing an algorithm to solve a more extensive scope of problems by combining and modifying the existing meta-heuristics methods.

1.2 Research objectives

This study aims to develop a combinatorial meta-heuristic algorithm for solving optimization problems. This algorithm, named chaotic GWO and WOA (CGWW), merges the GWO and modified WOA to get the advantage of both algorithms. The CGWW algorithm has the moving steps of the WOA and GWO algorithms simultaneously, so it has a perfect set of evolutionary operators. Accordingly, the hybrid algorithm can obtain more promising solutions for the problem. This algorithm uses a reliable strategy to initialize the search agents and guide them to achieve the near-optimal solutions. Several chaotic maps are applied for the initialization step and adjusting the parameters of the proposed algorithm. In the CGWW algorithm, the roulette wheel selection operator is employed to select the target search agents based on their fitness value. Therefore, the proposed algorithm preserves the elite search agents and tries to direct search agents toward optimal solutions. From another viewpoint, the hybrid algorithm has the properties of a multi-swarm algorithm with more diversity for population

members. The CGWW algorithm is evaluated using 23 benchmark functions and a real-world application for intrusion detection systems. Then the results are compared with the obtained results by several states of the art optimization methods.

The rest of this paper includes the following sections. A brief description of the applied and well-known meta-heuristic algorithms, the application of chaotic maps for optimization, and different kinds of chaotic maps are presented in Sect. 2. The structure of the proposed combinatory algorithm is presented in Sect. 3. In Sect. 4, the 23 mathematical benchmark problems, performed experiments, and the generated results are presented. Finally, in Sect. 5, the conclusion of the paper and future works are described.

2 Preliminary

Swarm-based meta-heuristic optimization methods are developed based on the evolution of a group of solutions for a problem in a search space. Solutions with high-quality will survive in the iterations of the optimization method to obtain the optimal solution. Followingly, the applied techniques and operation mechanisms of some of these meta-heuristic methods are summarized in Sect. 2.1. Using random number generators is necessary for implementing most of the optimization algorithms. Chaotic maps behave similarly to random number generators. In recent years, chaotic maps are applied to improve the exploration ability of the optimization algorithms. Instead of random generators, the chaotic maps can generate a sequence of numbers to visit more places in the search space. They can be used to initialize the population and adjust the parameters of meta-heuristic algorithms. Because of dynamic characteristics and better coverage of chaotic maps, meta-heuristic optimization algorithms can work with higher speeds [47, 49, 50]. Some of the meta-heuristic algorithms with the application of chaotic functions are discussed in Sect. 2.2. The GWO and WOA algorithms are briefly discussed in Sects. 2.3 and 2.4.

2.1 Meta-heuristic optimization algorithms and hybrid methods

One of the oldest optimization algorithms is the GA, which is inspired by the natural selection process [5]. The GA algorithm applies different kinds of operators like selection, mutation, and crossover. After some generations, solutions with the highest fitness value will survive. The DE algorithm is much like the GA but with the local search facilities [31, 51]. In the DE algorithm, all solutions have an equal chance to be selected for the next generation, but in GA, they are chosen based on their fitness. The moth-flame optimization algorithm (MFO) is created based on the routing mechanism of moths [52]. Moths fly in a straight line for long distances with a constant angle to the moon, but they sometimes entrap in a helix-shaped line around the artificial lights. These two straight-line and helix-shaped movements of moths implement the exploration and exploitation phases of the MFO algorithm. Group behavior of honeybees inspired the ABC algorithm [53, 54]. There are different groups of bees in the ABC, in which everyone in the group has a specific task. The final goal of all the activities is to produce honey, which is the solution to the investigated problem. Different theoretical and practical real-world problems have been tried to solve with the ABC algorithm [55, 56]. The ALO algorithm is inspired by the trapping and hunting method of the ant lions [7]. Ant lions make some traps in nature and hunt ants using them. Different stages of the ALO algorithm are designed based on this natural phenomenon. The AEFA is introduced by Anita et al. and is developed based on electrostatic forces [9]. In the AEFA, a group of particles establishes a population wherein each particle retains a charge value. This value indicates the fitness for a particle, and the electrostatic force of a particle with a high charge value attracts other particles with a lower charge. Hence, all particles try to move toward particles with higher fitness, which results in finding the optimum solutions. The SSA is presented by Mirjalili et al. and imitates the social behavior of salps [10]. It starts with some random salps, which will forage and move in the ocean water. The salps population tries to move toward the food sources, the salps with maximum fitness value. After some iterations, the near-optimal solutions in the search space are found.

In recent years, some hybrid meta-heuristic optimization algorithms have also been introduced. One of these algorithms is the GSA-GA algorithm, introduced by Garg, for solving the constrained optimization problems [32]. In the GSA-GA algorithm, the problem solutions are firstly conducted with the gravitational search method then, they are enhanced with the GA operators. The PSO-GA is another hybrid method again presented for solving the constrained optimization problems by Garg [33]. In the PSO-GA algorithm, the exploration and exploitation operation is performed with cooperating the GA operators and the PSO. A parameter-free penalty function handles the problem constraints in the PSO-GA. The GA-GSA is also a combinatorial optimization algorithm introduced by Garg for optimizing an industrial system [34]. The GA-GSA is used for maximizing the availability, reliability, and maintainability parameters as the objectives to increase the productivity and performance of the industrial system. The fuzzy sets are applied for resolving the conflicts between the problem objectives in [34]. The PS-ABC is another hybrid algorithm proposed by Li et al. for solving high-dimensional optimization problems [35]. This algorithm applies global search phases of the ABC algorithm with the local search phase of

the PSO to obtain the global optimum. The PS-ABC considers the aging degree of each search agent's best for deciding on the type of search. Beigvand et al. introduced the hybrid TVAC-GSA-PSO algorithm with time-varying acceleration coefficients for solving the large-scale Combined Heat and Power Economic Dispatch (CHPED) problems [36]. The TVAC-GSA-PSO algorithm is developed based on the self-adaptive learning strategy of swarm-based algorithms using various particle movements and the Newtonian laws for gravitation.

2.2 Chaotic maps and optimization algorithms

Seven kinds of ABC algorithms are proposed in [41]. In these algorithms, chaotic maps are applied to adjust the parameters' values to increase convergence speed and scape from local optima. For increasing the strength of the bat algorithm (BA) for global search, chaotic maps are applied in [49]. Thirteen chaotic maps are evaluated in [49], and four different types of BA are developed using chaos. A chaotic map named sinusoidal has been applied as the rate of pulse emission in the BA, which resulted in developing the CBA-IV algorithm with the best performance. A chaotic map called piecewise linear (PWLCM) is presented by Xiang et al. [57]. The PWLCM chaotic map is applied in the PSO to develop the PWLCPSO algorithm. Another PSO algorithm merged with chaotic maps is introduced in [58]. This algorithm uses the PSO and a new operator named adaptive inertia weight factor (AIWF) to explore the search space. Talatahari et al. presented an optimization method called chaotic imperialist and competitive algorithm (CICA) [59]. The ICA is inspired by the socio-political evolution process of the world's countries. Different kinds of chaotic maps are implemented and evaluated in CICA. The results indicated the superiority of the logistic and sinusoidal maps. The chaotic maps are applied to set the light and other absorption parameters of fireflies in the firefly algorithm (FA) to increase the global search capability [43]. The Gaussian map as the absorption coefficient is reported to have the best performance. The chaotic maps are also applied inside the KH algorithm to accelerate the general convergence [44]. Different types of krill movements are proposed using chaotic maps, in which the singer map has had the best performance.

2.3 The gray wolf optimization algorithm

Mirjalili suggested a meta-heuristic algorithm named gray wolf optimization (GWO), which imitates the leadership mechanism and hunting method of gray wolves in nature [25]. There are four types of wolves known as alpha, beta, delta, and omega in the GWO to simulate the leadership hierarchy. Moreover, the GWO algorithm has three fundamental steps of encircling, searching, and attacking the prey. From four types of wolves, alpha is the leader and manages the other types. The alpha wolves control the hunting steps and take the principal decisions. The beta wolves are in the second level of the hierarchy and return the feedback from others to the alpha type. The next level of the gray wolves hierarchy comprises delta wolves, which dominate the last one, containing omega wolves. Equation (1) calculates the distance of each wolf from alpha, beta, and delta wolves. X is the position of the current wolf in the search space.

$$D_{\text{alpha}} = \left| C_1 . X_{\text{alpha}} - X \right|, \quad D_{\text{beta}} = \left| C_2 . X_{\text{beta}} - X \right|, \\ D_{\text{delta}} = \left| C_3 . X_{\text{delta}} - X \right| \tag{1}$$

To obtain the next position of the current wolf, the *X1*, *X2*, and *X3* values, can be calculated by Eq. (2).

$$X_1 = X_{\text{alpha}} - A_1 . D_{\text{alpha}}, \quad X_2 = X_{\text{beta}} - A_2 . D_{\text{beta}}, \\ X_3 = X_{\text{delta}} - A_3 . D_{\text{delta}} \tag{2}$$

The A, a, and C are some parameters to control the algorithm calculated by Eq. (3). The r_1 and r_2 are two random values ranging from *0* to *1*. The C is a value to increase the exploration ability of the GWO algorithm. The value of 'a' changes linearly from *2* to *0* during the iterations of the algorithm.

$$A = 2a . r_1 - a, \quad C = 2 . r_2 \tag{3}$$

The next position of each wolf in the population is updated by Eq. (4).

$$X(t+1) = \left(X_1 + X_2 + X_3 \right) / 3 \tag{4}$$

2.4 The whale optimization algorithm

The whale optimization algorithm (WOA) [3] works based on a hunting method called bubble network, which belongs to humpback whales. In the WOA, for each iteration of the algorithm, the search agent with the best fitness (X^*) is selected. Other search agents try to get their location close to the X^*. The distance of each agent to the X^* is calculated by Eq. (5). In Eq. (5), C is a vector calculated by Eq. (6), X is the position vector, r is a random number between 0 and 1, and t is the iteration number.

$$\vec{D} = | \vec{C} . \vec{X}^*(t) - \vec{X}(t) | \tag{5}$$

$$\vec{C} = 2 . \vec{r} \tag{6}$$

The position of each agent in the next iteration is calculated by Eq. (7). The value of A is a vector computed by Eq. (8). $\vec{a}$ is also a coefficient that changes linearly from 2 to 0.

$$\vec{X}(t+1) = \vec{X}^{*}(t) - \vec{A} \cdot \vec{D} \tag{7}$$

$$\vec{A} = 2\vec{a} \cdot \vec{r} - \vec{a} \tag{8}$$

The exploitation phase of the bubble network hunting method contains two types of movements, named encircling-shrinking and spiral-shaped around the prey. For the exploration phase, random moves are applied to prey search. The encircling-shrinking is done by Eq. (7) when the value of a is decreasing from 2 to 1 during the iterations. Each search agent (whale) tries to get its position closer to the best one (prey) of the previous iteration in this movement. In the spiral-shaped move, the search agent goes through a spiral-shaped path around the best solution using Eq. (8). b is a constant which defines the shape of the spiral-shaped movement, and l is a random number between -1 and 1. Value of D' in Eq. (9) indicates the distance between the selected agent and the prey calculated by Eq. (10).

$$\vec{X}(t+1) = \vec{D}'.e^{bl}.\cos(2l) + \vec{X}^{*}(t) \tag{9}$$

$$\vec{D}' = |\vec{X}^{*}(t) - \vec{X}(t)| \tag{10}$$

Selecting encircling-shrinking or spiral-shaped movements is made by a random variable called p with a 50 percent chance for each one. In the exploration phase of the WOA, each population member moves toward a randomly selected search agent. This kind of movement is called the random search of the WOA, which increases the population diversity. This movement is implemented by Eqs. (11) and (12).

$$\vec{X}(t+1) = \vec{X}_{\text{rand}} - \vec{A} \cdot \vec{D} \tag{11}$$

$$\vec{D} = \left|\vec{C} \cdot \vec{X}_{\text{rand}} - \vec{X}\right| \tag{12}$$

X_{rand} in Eq. (11) is the position of the selected random search agent. 'A' is a number greater than one or less than minus one calculated by Eq. (8). To improve the diversity of solutions, the value of A enforces the whales to get away from each other. The WOA algorithm initializes a whales' population and updates the whales' positions with the three types of movements mentioned above.

The CWOA algorithm is the chaotic version of WOA [47]. In the CWOA, chaotic maps are applied to adjust the variables used to balance the exploration and exploitation abilities of WOA. Ten types of chaotic maps are used and evaluated for developing the CWOA algorithm. An improved version of the whale optimization algorithm, called IWOA, is also suggested to solve different scales of 0–1 knapsack problem with one and multi-dimensions [60]. A negative penalty value is considered inside the evaluation function to detect invalid solutions found by the IWOA. Moreover, to balance exploration and exploitation, a local search strategy and Lévy flight trajectories are implemented to develop the IWOA.

3 Chaotic and hybrid GWO and WOA algorithm

A chaotic and hybrid algorithm of GWO and WOA, called CGWW, is proposed in this section, which uses chaotic maps for initialization and adjusting its parameters.

3.1 Combination of the GWO and WOA algorithms

Exploration and exploitation are two fundamental phases of an optimization algorithm. In the exploration phase, the algorithm tries to search the entire search space of the problem with long steps. But in the exploitation, it strives to investigate the search space around the best solutions found in the exploration phase by small movements. The exploration phase of the WOA algorithm [3] is performed by the random search, where the exploitation phase is done by encircling-shrinking and spiral-shaped moves. But for some problems, the WOA gets stuck in the local optimum solutions.

In this paper, a modified version of the WOA algorithm is combined with the GWO to improve the exploration and exploitation operation. A random whale is selected in the random search of the WOA algorithm, as described by Eq. (11) in the previous section. As a modification, the whales' selection in this movement could perform by the roulette wheel operator to enhance the exploration mechanism of the WOA. In this way, the whales with better fitness values would be selected and became the target of other whales to move. The roulette wheel selection operator is one of the principal operators of GA [5].

Another defect of the WOA algorithm is fast convergence to solutions with low quality. Two approaches are proposed in this paper to solve this problem. The first approach is combining the GWO with the WOA to develop a multi-swarm algorithm with high exploration capability. The second one is applying chaotic maps to generate solutions with a high degree of diversity. Chaotic maps are used to initialize the population and adjust different parameters and coefficients of the CGWW algorithm. The chaotic maps that are explained in the next section are applied for developing the CGWW. Some of the most significant properties of the CGWW algorithm are listed below.

- The chaotic maps are applied to initialize the population of the CGWW algorithm.
- The CGWW algorithm is developed by combining the GWO and WOA, and it has more diverse movements for the search agents in the search space of the problem.
    - The initial population of the CGWW is divided into two subpopulations. The GWO works on the first one, and the WOA works on the second one.
    - After each iteration, the first and second subpopulations are merged in a total population. Then the fitness evaluation is performed, and the search agents of the total population are divided between the first and second subpopulations based on their fitness value. In this division, both subpopulations have approximately equal numbers of search agents with high fitness value.
    - The roulette wheel selection operator is applied in CGWW to select the search agents in random movements, so better ones from the fitness viewpoint would be the target whale instead of random ones.

The flowchart of the CGWW algorithm is presented in Fig. 1.

Algorithm 1 shows the pseudo-code of the proposed CGWW algorithm.

Algorithm 1. Pseudo-code of the hybrid CGWW algorithm

```
Initialize the total population using chaotic maps;
While (t < maximum number of iterations)
   Check if any search agent goes beyond the search space and amend it;
   Calculate the fitness of each search agent in the total population;
   Divide the total population to pop1(wolves of GWO, Gi (i=1,2, …, n/2)) and pop2 (whales, Wi (i = 1, 2, ..., n/2));
   Determine the Alpha, Beta, Delta, and Omega wolves in pop1;
   Determine the X* as the best search agent in the population of whales (pop2);
   For each search agent (wolf)  in pop1
      If (The search agent does not improve for the past five iterations)
         Initialize search agent again, using a chaotic map;
      End If
     Calculate A, C, a, X1, X2, and X3;
     Update the position of the search agents (wolves), using X1, X2, and X3;
   End For
   For each search agent in the whales' population
      If (The search agent does not improve for the past five iterations)
         Initialize search agent again, using a chaotic map;
      end if
      Update a and p;
      Update A, C, and l using chaotic maps instead of random numbers in the equations;
      If (p<0.5)
          If (|A|< 1)
             Update the position of the current search agent by the Eq. (7);
          Else If(|A|≥1)
             RandomAgent=Select a search agent by the roulette wheel;
             Update the position of the current search agent by the Eq. (11);
          End If
      Else If(p≥0.5)
             Update the position of the current search agent by the Eq. (9);
      End If
   End For
   Integrate pop1(wolves) and pop2 (whales) into the total population;
   Update the best-solution if there is a better one in the total population;
   t=t+1;
End While
Return the best-solution;
```

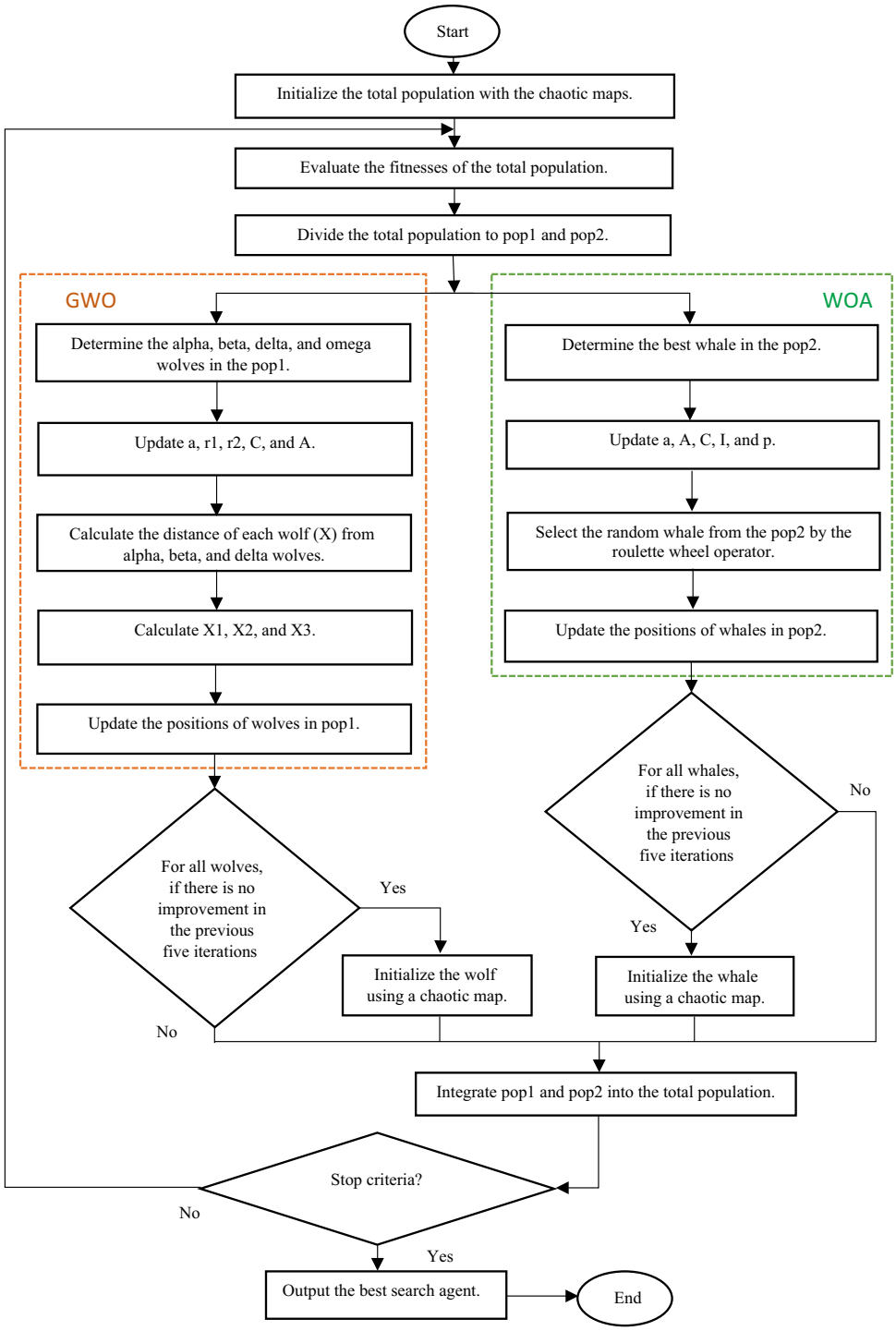

Fig. 1 Flowchart of the hybrid CGWW algorithm

3.2 Chaotic maps and the proposed hybrid algorithm

Random number generators are needed for implementing most of the optimization methods. In most optimization algorithms, randomness is attained by applying uniform or Gaussian probability distributions. The chaotic systems in mathematics are related to deterministic, non-linear, and dynamic systems, which have infinite unpredictable and periodic changes. They are sources of randomness, and they can perform explorations at higher speeds compared to random searches that principally rely on randomness [43]. Because of the similar behavior of chaotic maps and random systems, the chaotic maps are applied in most of the meta-heuristic optimization algorithms to improve the exploration phase. This improvement occurs by increasing the diversity

of population members using chaotic maps. In this way, the meta-heuristic algorithms can perform the iterative search faster than the standard random searches, which use the mentioned probability distributions. The chaotic systems are sensitive to the beginning conditions, and a minor variation in them may result in essential changes in the results [49, 50]. Due to the dynamic behavior and non-repetition characteristics of the chaotic maps, they are applied mainly for the initialization and adjusting the parameters of meta-heuristic methods to increase the convergence speed and scape from the local optima [37, 38]. Various chaotic maps are designed by mathematicians, physicians, and researchers which some of the most commonly used single-dimensional ones are presented below [61]. These maps are applied in this paper to implement the CGWW algorithm. In the following functions, k is the iteration number, and x_k is the kth chaotic value. All of the investigated chaotic maps are adjusted to generate values between 0 and 1.

(1) **Chebyshev map:** Eq. (13) defines this map [62].

$$x_{k+1} = \cos\left(k\cos^{-1}\left(x_k\right)\right) \tag{13}$$

(2) **Circle map:** This map is defined by Eq. (14), which produces a chaotic sequence between 0 and 1 for $a=0.2$ and $b=0.5$ [63].

$$x_{k+1} = x_k + b - \left(\frac{a}{2}\right)\sin\left(2x_k\right)\mathrm{mod}(1) \tag{14}$$

(3) **Gauss/Mouse map:** Equation (15) defines this map [61].

$$x_{k+1} = \begin{cases} 0 & x_k = 0 \\ \frac{1}{x_k \mathrm{mod}(1)} & \text{otherwise} \end{cases}, \quad \frac{1}{x_k \mathrm{mod}(1)} = \frac{1}{x_k} - \left[\frac{1}{x_k}\right] \tag{15}$$

(4) **Iterative map:** Iterative map with infinite falling points can be expressed by Eq. (16), as $a \in (0,1)$ [64].

$$x_{k+1} = \sin\left(\frac{a}{x_k}\right) \tag{16}$$

(5) **Sine map:** Eq. (17) expresses this map [65], which $0 < a \leq 4$.

$$x_{k+1} = \frac{a}{4}\sin\left(x_k\right) \tag{17}$$

(6) **Singer map:** Eq. (18) shows this one-dimensional map, as μ is a parameter between 0.9 and 1.08 [66].

$$x_{k+1} = \left(7.86x_k - 23.31x_k^2 + 28.75x_k^3 - 13.3x_k^4\right) \tag{18}$$

(7) **Sinusoidal map:** Eq. (19) expresses the sinusoidal chaotic map [67], that in a particular case, once $a=2.3$ and $X_0=0.7$, it can be simplified by Eq. (20).

$$x_{k+1} = ax_k^2\sin\left(x_k\right) \tag{19}$$

$$x_{k+1} = \sin\left(x_k\right) \tag{20}$$

(8) **Tent map:** This map is similar to the well-known logistic function that Eq. (21) defines [68].

$$x_{k+1} = \begin{cases} \frac{x_k}{0.7} & x_k < 0.7 \\ \frac{10}{3}\left(1 - x_k\right) & x_k > 0.7 \end{cases} \tag{21}$$

(9) **Sawtooth map:** It can be formulated by mod function by Eq. (22) [69].

$$x_{k+1} = 2x_k\mathrm{mod}(1) \tag{22}$$

(10) **Logistic map:** Eq. (23) expresses this map [67]. The x will be between 0 and 1 only if the initial value of x_0 is in (0, 1).

$$x_{k+1} = x_k\left(1 - x_k\right)x_0 \in (0.0, 0.25, 0.75, 0.5, 1.0) \tag{23}$$

3.3 Advantages of the proposed algorithm

The CGWW algorithm is developed to solve various kinds of continuous optimization problems with its diverse operators and procedures, which have been borrowed from the well-known GWO and WOA algorithms. In other words, theoretically, the strengths points of the GWO and WOA are integrated into the CGWW along with the chaotic maps to represent a high-performance meta-heuristic algorithm. Some advantages of the CGWW algorithm are as follows:

- The CGWW algorithm has a high power of exploration, and it can search the search space of the problem effectively to find the promising regions. This characteristic of the proposed algorithm is due to the search agents containing diverse solutions from the combined algorithms.
- Due to the dynamic behavior and application of chaotic maps in the optimization algorithms to better exploring the search space, they have been applied in the CGWW algorithm. The second reason for the increased exploration ability in the proposed algorithm is initializing the search agents with the chaotic maps.
- Also, during the iterations of the hybrid algorithm, the chaotic maps are applied to avoid stagnation in the solutions, which causes to find new promising regions in the search space.
- The proposed algorithm represents a high performance in the exploitation phase due to the combinatorial operators

given from the GWO and WOA. These operators search around the found promising solutions of the search space in the exploration phase. This ability is improved using the roulette wheel selection operator to determine the target solutions for the search agents of the WOA algorithm.

- About the local optima avoidance, the CGWW algorithm performs well with balancing the exploration and exploration ability. The chaotic maps also help in establishing this balance. In the first iterations of the proposed algorithm, the diversity of search agents is high due to generating the initial population by two strategies of the GWO and WOA, along with the chaotic maps. But in the ending iterations, the diversity decreases, and the population members converge to optimal solutions.
- Some mechanisms are applied to implement the elitism in the proposed algorithm. The best solutions of each iteration are saved as elite solutions and inserted in the subsequent populations. Besides, the stagnation of each search agent is examined in each iteration. If there is no improvement in the best position for the previous five iterations, the chaotic maps reinitialize the search agent.
- The last advantage of the proposed CGWW algorithm is its application for solving most of the continuous optimization problems without any information about the search space. In other words, the proposed algorithm looks at the given problem as a black box. Moreover, by some modifications in the fitness function, the proposed algorithm can also solve some kinds of multi-objective problems.

3.4 The complexity of the proposed algorithm

Considering the structure and implementation method of the proposed CGWW algorithm, the computational complexity is evaluated and discussed in this section. Most of the metaheuristic algorithms have limited memory usage, and the proposed algorithm is no exception. But in terms of runtime, various algorithms behave differently, and the algorithms with less computational complexity have a better performance. For the CGWW algorithm, the number of wolves in the GWO part, number of whales in the WOA part, maximum number of iterations, number of variables in the solution, applied chaotic map functions, sorting method of population, and the roulette wheel selection operator determines the computational complexity. The merged population, containing the wolves of the GWO and whales of the WOA, is sorted based on fitness value in each iteration of the CGWW algorithm. Whenever the sorting operation is required in the CGWW, the Quicksort algorithm with the complexity of O(n2) and O($n \log n$) for the worst and best case is used. As mentioned in the previous sections, to improve the WOA, the roulette wheel selection operator is applied for selecting the target whales during movements. The complexity of this operator, related to the implementation method, is $O(n)$ or $O(\log n)$. For every iteration of the proposed algorithm, if the search agents have no improvement in the last five iterations, the chaotic maps reinitialize them. But no function evaluations are required for these operations, and their cost is trivial. Equation (24) presents the computational complexity of the CGWW algorithm.

$$\begin{aligned} O(\text{CGWW}) = O(t \times O(&\text{Quick sort}) \\ &+ O(\text{positions update of the GWO}) \\ &+ O(\text{roulette wheel for the whales of the WOA}) \\ &+ O(\text{positions update of the WOA}))) \end{aligned} \tag{24}$$

$$\begin{aligned} O(\text{CGWW}) &= O(t \times n^2 + n/2d + n/2\log n + n/2 \times d)) \\ &= O\left(tn^2 + tnd + tn(\log n)/2\right) \end{aligned}$$

In Eq. (24), t is the maximum iterations, n is the number of search agents (wolves in the GWO plus whales in the WOA), and d is the dimension or number of variables.

3.5 Tips for converting the proposed algorithm to a multi-objective optimization method

The proposed CGWW algorithm is intrinsically a single-objective optimization algorithm but, with some modifications, it can be transformed into a multi-objective optimization algorithm. For solving a multi-objective problem, the final result is a set named Pareto optimal solutions. The currently proposed CGWW algorithm cannot be applied for solving multi-objective problems due to the following reasons. The first reason is that in each iteration of the CGWW, two solutions are found and saved as the best ones. One for the GWO and the other for the WOA. But for solving a multi-objective problem, the output is multiple non-dominated solutions. The second reason is that, for solving a multi-objective problem, a set of non-dominated solutions (Pareto set) are required for updating the positions of the remaining population members. Therefore, the first modification over the proposed algorithm may be having a repository of non-dominated solutions or the Pareto set. This repository is very similar to the archives in the multi-objective particle swarm optimization algorithm (MOPSO) [70, 71]. The obtained solutions of the GWO and WOA are compared against the repository members during the execution of the multi-objective algorithm using a dominance operator. If a solution dominates a member of the repository, the member will be replaced by it. During the optimization process, a solution may be non-dominated comparing with the repository members, which has to be added to the repository. By applying the mentioned rules, the repository always has the non-dominated solutions which have been found so far. The repository should have a limited capacity, so some

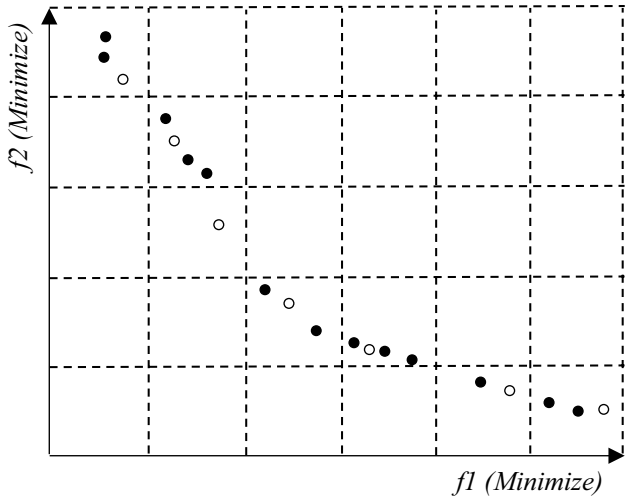

Fig. 2 The repository updating process when it is full (solid black circles are candidates to be removed)

members should be removed by adding new non-dominated solutions beyond capacity. The simplest way to remove the surplus solutions from the repository is the random method, but deleting the similar non-dominated solutions could be a better way. The proposed multi-objective algorithm should obtain the uniformly distributed Pareto optimal solutions. Therefore, removing some similar solutions from the populated regions of the repository is the best approach [71]. The segmentation of the repository for grouping similar solutions in each segment can be applied to do this. Then for each repository member, a rank is assigned based on the number of neighboring solutions. Solutions with a higher rank number are selected with the roulette wheel selection operator to remove from the repository. After some iterations of the multi-objective algorithm, the repository members will have improved distribution. In an ideal situation for the repository, there will be one solution per segment. Figure 2 illustrates an example for this repository updating process of the proposed multi-objective algorithm for solving a problem with two objective functions of f1 and f2. The solid black circles are candidates to be removed from the repository in the example of Fig. 2.

For the multi-objective CGWW algorithm, there should be a single repository for the GWO and WOA parts. In each iteration of the multi-objective CGWW, by using the dominance operator, the GWO algorithm compares its alpha wolf with the repository members, and the WOA does the same for the best-found whale.

There is more than one best solution in a multi-objective search space. Therefore, as mentioned previously, the selection method of targets in the GWO and WOA for updating the positions of the remaining population members in each iteration is the second issue for developing the multi-objective version of the CGWW. This issue can be solved with the random selection of the targets. But a wiser way is to apply the same ranking method and roulette wheel operator for removing the surplus solutions from the repository. However, the non-dominated solutions with a lower rank, which has a less populated neighborhood, have a higher chance to be selected as the targets. For example, the non-dominated solution in the third row and second column in Fig. 2 with no neighbor has the highest probability for selection. Algorithm 2 presents the pseudo-code of the proposed multi-objective CGWW algorithm.

Table 1 Unimodal benchmark evaluation functions

Name	Function	Var-number	Range	$f_{\min}$
Sphere	$F_1(x) = \sum_{i=1}^{n} x_i^2$	30	[−100, 100]	0
Schwefel 2.22	$F_2(x) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	30	[−10, 10]	0
Schwefel 1.2	$F_3(x) = \sum_{i=1}^{n} (\sum_{j-1}^{i} x_j)^2$	30	[−100, 100]	0
Schwefel 2.21	$F_4(x) = max_i\{\lvert x_i \rvert, 1 \le i \le n\}$	30	[−100, 100]	0
Rosenbrock	$F_5(x) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	30	[−30, 30]	0
Step	$F_6(x) = \sum_{i=1}^{n} ([x_i + 0.5])^2$	30	[−100, 100]	0
Quartic	$F_7(X) = \sum_{i=1}^{n} i x_i^4 + random[0,1)$	30	[−1.28, 1.28]	0

Algorithm 2 Pseudo-code of the multi-objective CGWW algorithm

```
Initialize the total population using chaotic maps;
While (t < maximum number of iterations)
  Check if any search agent goes beyond the search space and amend it;
  Calculate the fitness of each search agent in the total population;
  Determine the non-dominated search agents in the total population;
  Update the repository, considering the obtained non-dominated search agents;
  Rank the repository members based on the number of neighboring solutions;
  If (The repository is full)
      Remove the surplus repository members using the roulette wheel based on their ranks (Maximum rank);
  End If
  Divide the total population to pop1(wolves of GWO, Gi (i=1,2, …, n/2)) and pop2 (whales, Wi (i = 1, 2, ..., n/2));
  Determine the Alpha using the roulette wheel from the repository (Minimum rank);
  Determine the Beta, Delta, and Omega wolves in pop1, using the obtained alpha wolf;
  Determine the X* (Best whale) using the roulette wheel from the repository (Minimum rank);
  For each search agent (wolf)  in pop1
      If (The search agent does not improve for the past five iterations)
          Initialize search agent again, using a chaotic map;
      End If
    Calculate A, C, a, X1, X2, and X3;
    Update the position of the search agents in pop1 (wolves), using X1, X2, and X3;
  End For
  For each search agent in the whales' population
      If (The search agent does not improve for the past five iterations)
          Initialize search agent again, using a chaotic map;
      end if
      Update a and p;
      Update A, C, and l using chaotic maps instead of random numbers in the equations;
      If (p<0.5)
          If (|A|< 1)
            Update the position of the current search agent by the Eq. (7);
          Else If(|A|≥1)
            RandomAgent=Select a search agent by the roulette wheel;
            Update the position of the current search agent by the Eq. (11);
          End If
      Else If(p≥0.5)
            Update the position of the current search agent by the Eq. (9);
      End If
  End For
  Integrate pop1(wolves) and pop2 (whales) into the total population;
  t=t+1;
End While
Return repository;
```

Table 2 Multimodal benchmark evaluation functions

Name	Function	Var-number	Range	f_{min}
Schwefel	$F_8(x) = \sum_{i=1}^{n} -x_i \sin(\sqrt{\lvert x_i \rvert})$	30	[−500, 500]	$-418.9829 \times Dim(30)$
Rastrigin	$F_9(x) = \sum_{i=1}^{n} [x_i^2 - 10\cos(2x_i) + 10]$	30	[−5.12, 5.12]	0
Ackley	$F_{10}(x) = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2x_i)\right) + 20 + e$	30	[−32, 32]	0
Griewank	$F_{11}(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	30	[−600, 600]	0
Penalty 1	$F_{12}(x) = \frac{}{n}\left\{10\sin(y_1) + \sum_{i=1}^{n-1}(y_i - 1)^2\left[1 + 10sin^2(y_{i+1})\right] + (y_n - 1)^2\right\} + \sum_{i=1}^{n} u(x_i, 10,100,4)$	30	[−50, 50]	0
	$y_i = 1 + \frac{x_i+1}{4}u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m x_i > a \\ 0 - a < x_i < -a \\ k(-x_i - a)^m x_i < -a \end{cases}$			
Penalty 2	$F_{13}(x) = 0.1\{sin^2(3\pi x_1) + \sum_{i=1}^{n}(x_i - 1)^2\left[1 + sin^2(3\pi x_i + 1)\right] + (x_n - 1)^2\left[1 + sin^2(2\pi x_n)\right]\} + \sum_{i=1}^{n} u(x_i, 5,100,4)$	30	[−50, 50]	0

Table 3 Composite benchmark evaluation functions

Name	Function	Var- Number	Range	f_{min}
Foxholes	$F_{14}(x) = \left(\frac{1}{500} + \sum_{j=1}^{25} \frac{1}{j+\sum_{i=1}^{2}(x_i - a_{ij})^6}\right)^{-1}$	2	[−65, 65]	1
Kowalik	$F_{15}(x) = \sum_{i=1}^{11}\left[a_i - \frac{x_1(b_i^2+b_i x_2)}{b_i^2+b_i x_3+x_4}\right]^2$	4	[−5, 5]	0.00030
Six-Hump Camel-Back	$F_{16}(x) = 4X_1^2 - 2.1X_1^4 + \frac{1}{3}X_1^6 + X_1X_2 - 4X_2^2 + 4X_2^4$	2	[−5, 5]	−1.0316
Branin	$F_{17}(x) = (X_2 - \frac{5.1}{4^2}X_1^2 + \frac{5}{X_1} - 6)^2 + 10\left(1 - \frac{1}{8}\right)COS(X_1) + 10$	2	[−5, 5]	0.398
Goldstein-Price	$F\_18(x) = [1 + (x\_1 + x\_2 + 1)^2(19 - 14x\_1 + 3x\_1^2 - 14x\_2 + 6x\_1x\_2 + 3x\_2^2)]$ $\times[30 + (2x\_1 - 3x\_2)^2 \times (18 - 32x\_1 + 12x\_1^2 + 48x\_2 - 36x\_1x\_2 + 27x\_2^2)]$	2	[−2, 2]	3
Hartman 3	$F_{19}(x) = -\sum_{i=1}^{4} c_i exp(-\sum_{j=1}^{3} a_{ij}(x_j - p_{ij})^2)$	3	[1, 3]	−3.86
Hartman 6	$F_{20}(x) = -\sum_{i=1}^{4} c_i exp(-\sum_{j=1}^{6} a_{ij}(x_j - p_{ij})^2)$	6	[0, 1]	−3,32
Shekel 5	$F_{21}(x) = -\sum_{i=1}^{5}[(X - a_i)(X - a_i)^T + C_i]^{-1}$	4	[0, 10]	−10.1532
Shekel 7	$F_{22}(x) = -\sum_{i=1}^{7}[(X - a_i)(X - a_i)^T + C_i]^{-1}$	4	[0, 10]	−10.4028
Shekel 10	$F_{23}(x) = -\sum_{i=1}^{10}[(X - a_i)(X - a_i)^T + C_i]^{-1}$	4	[0, 10]	−10.5363

Algorithm 2 indicates that the multi-objective CGWW algorithm first initializes the total population using the chaotic maps. This algorithm then calculates the fitness of each solution and obtains the non-dominated ones. The repository is updated considering the new non-dominated solutions. If the repository becomes full, some solutions with a populated neighborhood are selected using the roulette wheel operator and then removed from the repository. In the next step, the alpha wolf and the best whale are chosen from the repository, again with the roulette wheel, which both of them have the least populated neighborhood. The rest of algorithm 2 is the same as algorithm 1 for the single objective CGWW.

In this section, some suggestions are presented to transform the CGWW algorithm into a multi-objective approach. But, as mentioned earlier, the CGWW algorithm is intrinsically single-objective in its current form. However, with some modifications to the objective functions, the multi-objective problems can be solved by the single-objective optimization algorithm to some extent. For instance, the GWO algorithm is applied for solving the feature selection problem in anomaly-based intrusion detection systems in [72]. In this study, a weighted sum fitness function has been proposed using the objective of the problem. Therefore, to further evaluate the CGWW algorithm, a kind of feature selection problem in intrusion detection systems has been tried to solve with the proposed CGWW algorithm in Sect. 4.6.

4 Results of experiments and discussion

Several experiments are conducted to evaluate the CGWW algorithm with some benchmark functions and a real-world application for intrusion detection systems. In the following sections, the details of the performed experiments and their conditions and results are presented.

4.1 Benchmark functions and the optimization algorithms

The CGWW algorithm is evaluated using unimodal, multimodal, and composite benchmark functions [73–79]. The first group of benchmark functions, called unimodal, includes seven members (F1 to F7), where each one has one optimum solution. The unimodal benchmark functions presented in Table 1 can evaluate the convergence power and exploitation ability of the optimization algorithms. The second group, called multimodal (F8 to F13), has large dimensions and is presented in Table 2. Multimodal benchmark functions have many local and one global optimum, so they are more challenging than unimodal functions. Multimodal benchmarks can evaluate the exploration and local optima avoidance abilities of optimization algorithms. The optimum

Table 4 Parameters values of the compared algorithms

Algorithm	Parameters	Values
PSO	Inertial value	[0.4, 0.9]
	Acceleration rate (c_1 and c_2)	2
	Maximum velocity	8
MFO	Convergence constant	[−2, −1]
ABC	Abandonment limit	0.6*Dimension*Colony size
GA	Crossover rate	0.8
	Mutation rate	8
	Selection pressure	0.1
AEFA	K_0	500
	a	30
SSA	c_1	2*exp(−(4* Current iteration / Maximum iterations)^2)
	c_2	Random numbers between 0 and 1
	c_3	Random numbers between 0 and 1
GWO	Control parameter (a)	[2, 0]
WOA	*a*	[2, 0]
	a2	[1, 2]
	r_1, r_2	Random numbers between 0 and 1
CGWW	*a*	[2, 0]
	a2	[1, 2]
	r_1, r_2	Chaotic numbers between 0 and 1

value of the F8 in Table 2 is a coefficient of problem dimension, which is 30 in the experiments. For the other benchmarks of Tables 1 and 2, the optimum value is zero. The last group of functions is composite or multimodal with fixed or small dimensions (F14 to F23). Composite benchmark functions, shown in Table 3, are rotated, biased, shifted, and combined versions of unimodal or multimodal ones [80, 81]. Most of the composite benchmarks have complicated shapes and many local optimums, so they are more like real-world problems. For the composite benchmark functions, finding a balance between exploration and exploitation operations is essential to find the global optimum.

The proposed CGWW algorithm is compared with eight well-known optimization algorithms to evaluate the performance of it. The algorithms are the GA [5, 82], PSO [6], MFO [52], ABC [8, 54], SSA [10], AEFA [9], GWO [25] and WOA [3].

4.2 Experimental setup

The proposed CGWW and other compared algorithms have some execution parameters. The initial values for the significant parameters are adjusted to evaluate the CGWW algorithm. In Table 4, the initial values of the parameters of the compared algorithms are presented. These values are adjusted according to the values stated in the literature. The same initialization method has been applied for all the algorithms to compare them equivalently. The number of search agents and iterations for all algorithms is considered 30 and 500, respectively. Parameter values of the proposed CGWW algorithm are a combination of values for the GWO and WOA algorithms. The experiments and algorithms are implemented in Matlab R2018a using a system with a Core i5 Intel processor, 2.8 GHz clock speed, 8 GB of RAM size, and Microsoft Windows 10 64-bit OS.

The parameters of Table 4 for the PSO are adjusted according to the values recommended in [6, 83–85]. The inertial value between 0.4 and 0.9 adjusts the exploration and exploitation phases, while the acceleration rates of $c_1 = c_2 = 2$ control the moving distance of particles. For the MFO algorithm, the convergence parameter linearly decreases from −1 to −2, according to [52], while the abandonment limit parameter of ABC is determined based on [8]. The GA algorithm has three parameters of crossover rate, mutation rate, and selection pressure. The crossover is the essential operator of the GA, and *0.7* is a reasonable rate for it. A high probability for this operator will combine different solutions to generate new ones. The mutation rate should be low since it will increase the diversity more than enough, and this will remove the high-quality solutions during the generations [5]. The K_0 and *'a'* are two main parameters of the AEFA algorithm, which are used to set the Coulomb's constant according to [9]. The most significant parameter of the SSA is c_1, which establishes a balance between the exploration and exploitation mechanisms. The other parameters of the SSA are c_2 and c_3, random numbers between *0*

Table 5 Results of compared algorithms on the unimodal benchmark functions

F		GA	PSO	ABC	MFO	SSA	AEFA	GWO	WOA	CGWW
1	Best	22.2014	3.70E−05	10.3477	0.66762	2.93E−08	0.00020107	1.38E−29	4.94E−84	**4.30E−221**
	Worst	64.1458	0.0008444	50.0275	10,000.9141	3.47E−07	16.604	5.19E−26	3.41E−75	**2.48E−179**
	Aver	37.5605	0.00032939	25.0387	2004.2166	1.37E−07	3.1811	6.06E−27	6.54E−76	**2.77E−180**
	Stdd	13.9768	0.00028676	12.1122	4214.5541	1.03E−07	5.2947	1.62E−26	1.38E−75	**0**
2	Best	0.59135	0.021652	0.22036	10.1181	1.08	12.1174	1.76E−17	1.06E−56	**3.01E−123**
	Worst	2.0649	30.0945	0.46986	70.0051	5.0871	45.5724	1.19E−16	3.36E−47	**4.52E−100**
	Aver	1.1733	10.0444	0.33566	34.1064	2.3394	21.0975	7.09E−17	3.36E−48	**5.05E−101**
	Stdd	0.44339	9.44	0.091427	18.2993	1.145	9.818	3.51E−17	1.06E−47	**1.42E−100**
3	Best	2231.7304	57.2928	25,399.548	8199.3378	507.3865	1570.9143	1.70E−08	33,922.2505	**5.56E−156**
	Worst	4961.8458	180.7681	44,914.4413	41,971.0847	4742.1429	3965.5814	6.46E−06	68,087.9477	**2.39E−31**
	Aver	3689.2329	101.8678	35,100.8536	25,205.7984	1705.5503	2592.5253	1.16E−06	50,313.5754	**2.39E−32**
	Stdd	936.0154	38.5628	5588.6961	11,263.3919	1227.5846	870.3249	2.09E−06	11,847.3727	**7.56E−32**
4	Best	6.5736	1.2184	58.1881	46.1332	3.7487	3.0868	6.84E−08	0.12557	**7.13E−89**
	Worst	12.717	1.9079	66.8287	81.2321	16.683	8.6916	1.81E−06	84.9739	**2.71E−64**
	Aver	8.4476	1.5406	61.945	66.871	10.0647	5.3573	7.68E−07	47.2322	**3.66E−65**
	Stdd	1.7648	0.21167	2.7193	10.4563	3.626	1.7431	5.19E−07	31.5726	**8.41E−65**
5	Best	574.4635	**21.3694**	75,410.9051	240.4725	26.3205	596.5247	26.1099	27.1857	25.7637
	Worst	3469.4291	604.3909	285,425.531	90,242.872	1282.9922	11,579.1998	27.9393	28.7278	**26.3556**
	Aver	1890.875	117.2056	164,579	13,461.3532	345.5362	5023.5947	26.9936	27.9263	**25.9996**
	Stdd	953.0738	174.8223	84,848.7836	28,971.5555	429.1635	3811.8084	0.66428	0.48877	**0.20006**
6	Best	19.8057	6.76E−06	11.7883	0.92733	2.04E−08	**3.78E−23**	0.24228	0.061205	0.00012259
	Worst	53.6208	0.002217	46.5119	20,201.1074	**2.41E−05**	74.5242	1.5062	0.54408	0.000224
	Aver	34.165	0.00039875	20.8553	7012.3351	**2.54E−06**	10.3218	0.87557	0.28524	0.00015043
	Stdd	12.9069	0.00066162	11.5099	8236.5063	**7.59E−06**	23.3306	0.41897	0.15791	2.99E−05
7	Best	0.029668	0.099	0.25216	0.14864	0.081967	0.12042	0.00074658	0.00013975	**2.29E−05**
	Worst	0.068514	24.2649	0.50086	35.0882	0.3628	7.4237	0.0051373	0.0087893	**0.00044815**
	Aver	0.044213	6.8317	0.37721	7.7682	0.18583	1.6134	0.0022485	0.0024748	**0.00019148**
	Stdd	0.012955	9.0343	0.092499	10.6399	0.082081	2.2478	0.0012851	0.0029317	**0.00015532**

and *1*, according to [10]. The GWO and WOA algorithms have a control parameter named *'a'*, which balances the exploration and exploitation and changes from *2* to *0* according to [25]. The *'a2'* parameter of the WOA and the hybrid proposed algorithm varies between *1* and *2*, while r_1 and r_2 are two random values as stated in [3]. The chaotic maps are used to set the r_1 and r_2 values in this paper.

For implementing the CGWW algorithm, the mentioned chaotic maps of Sect. 3.2 have been used and compared. The comparison results showed that the Iterative, Circle, and Sine maps present better performance than the others. All of the experiments of this paper have been repeated 100 times to get more accurate results. For all compared algorithms, the average, standard deviation, worst, and best values are computed and reported as the performance metrics using the obtained results. The first two criteria show the stability of the algorithm in solving the investigated problems. For each test function, the best-obtained solution is displayed by the bold-face in the tables.

4.3 Results of experiments on the benchmark functions

In the following sections, the comparison results of CGWW with eight popular meta-heuristic algorithms for solving the unimodal, multimodal, and composite benchmark test functions are presented to evaluate the proposed algorithm.

4.3.1 Results for solving the unimodal benchmark evaluation functions

The first group of test functions, named unimodal, has only one optimum solution. Accordingly, the unimodal functions (F1 to F7) are applicable for evaluating the convergence speed of optimization algorithms. A high convergence speed could improve the exploitation phase of the algorithms. Table 5 presents the execution results of the compared algorithms for the unimodal benchmark functions. The results indicate that the CGWW algorithm could find the best solutions on 6 out of 7 unimodal test functions.

Table 6 Results of compared algorithms on the multimodal benchmark functions

F		GA	PSO	ABC	MFO	SSA	AEFA	GWO	WOA	CGWW
8	Best	−11,344.4073	−6943.301	−5268.4464	−10,226.6115	−8207.2115	−2676.0121	−6921.0634	−12,569.4474	**−12,556.914**
	Worst	**−9724.438**	−3598.6637	−4546.3108	−7267.5241	−6580.2456	−1821.8973	−4937.1445	−8481.8849	−6574.1193
	Aver	**−10,492.8704**	−6138.2173	−4996.9335	−8407.9911	−7518.2401	−2196.1506	−5933.1667	−10,454.4268	−9441.969
	Stdd	418.6733	956.0528	235.1372	895.6639	595.7159	315.5463	567.0973	1904.6725	2124.3916
9	Best	19.893	79.8104	215.1665	115.4351	29.8487	29.5316	5.68E−14	0	**0**
	Worst	53.2736	197.7717	253.3559	221.1784	65.6671	51.7466	8.494	0	**0**
	Aver	37.6934	139.6562	236.6185	177.8986	43.5791	38.2044	2.648	0	**0**
	Stdd	11.8384	39.4131	12.467	39.0635	12.3453	7.6203	3.0897	0	**0**
10	Best	1.844	0.0050213	3.1874	3.0792	1.5017	0.0054556	7.55E−14	8.88E−16	**8.88E−16**
	Worst	2.9187	1.6476	5.1843	19.9531	3.4042	2.4077	1.15E−13	7.99E−15	**8.88E−16**
	Aver	2.6348	0.37188	4.0129	17.2683	2.1807	1.3801	9.11E−14	5.15E−15	**8.88E−16**
	Stdd	0.38085	0.5832	0.53952	5.0937	0.61096	0.67805	1.37E−14	2.80E−15	**0**
11	Best	1.1626	9.21E−07	1.2644	0.66634	0.0003893	6.5506	0	0	**0**
	Worst	1.5046	0.019699	1.935	91.0322	0.023885	15.1046	0.020587	0.41795	**0**
	Aver	1.3537	0.0059311	1.5692	9.9442	0.013011	10.6147	0.0052174	0.041795	**0**
	Stdd	0.10395	0.0069755	0.18663	28.4919	0.0091304	2.4228	0.0087235	0.13217	**0**
12	Best	0.016931	**2.70E−07**	25,343.4171	4.5307	3.25	1.9855	0.026159	0.0084505	8.39E−06
	Worst	0.81983	0.00036585	788,912.7727	199.7127	12.711	6.53	0.12311	0.045092	**4.19E−05**
	Aver	0.31876	5.46E−05	210,414.1492	25.6359	7.1681	3.9173	0.061956	0.018617	**2.10E−05**
	Stdd	0.31438	0.00011728	233,428.9647	61.1931	2.6647	1.3389	0.032775	0.011753	**1.01E−05**
13	Best	0.83187	**1.26E−05**	46,024.7852	5.4385	0.0063786	15.4749	0.29919	0.31085	0.0001618
	Worst	3.7114	0.011522	1,111,359.569	83.4894	35.0118	55.7666	1.0973	0.92355	**0.011175**
	Aver	1.8032	0.0056008	634,054.9312	28.2289	15.2215	30.3806	0.66445	0.60965	**0.0024956**
	Stdd	0.87863	0.0058451	374,347.4635	23.4247	12.4785	12.2515	0.27943	0.2471	**0.0045749**

In the GWO algorithm, there is a hierarchy of wolves based on the solution quality. One of the responsibilities of this hierarchy is focusing on the high-quality solutions found so far. In this way, the exploitation ability of the algorithm will improve. About the WOA, it has different motion operators for the whales. The helix-shaped movement and the movement around the current best solution participate in the exploitation phase. Besides, the roulette wheel operator, which forces the whales to move toward high-quality solutions, improves the exploitation operation. Integration of the above operators and movements in the hybrid CGWW algorithm results in high performance for the exploitation best solutions to find the global optimum. The best, worst, average, and standard deviation results in Table 5 indicate that the proposed CGWW algorithm obtained the minimums for the F1, F2, F3, F4, and F7 functions. About the F5, the minimum of best solution belongs to the PSO algorithm, but for other evaluation parameters, the CGWW has found the minimum. About the F6, the proposed algorithm does not have an acceptable performance where the best answer belongs to the AEFA, and the minimum of worst, average, and standard deviation are obtained by the SSA. Therefore the CGWW has a high capability for finding the global optimum solution of the unimodal functions and presents a high performance in the exploitation phase with extra precision.

4.3.2 Results for solving the multimodal benchmark evaluation functions

The second group of test functions, named multimodal (F8 to F13), have one global optimum and many local optima. By increasing the problem dimensions, the number of local optima increases exponentially in the multimodal functions. These functions can evaluate the strength of optimization algorithms to escape from local optima. Therefore, the multimodal functions can estimate the power of the optimization algorithms in the exploration phase. Table 6 presents the execution results of the compared algorithms for the multimodal benchmark functions. The results indicate that the CGWW algorithm could find the best solutions on 5 out of 6 multimodal benchmark functions. The exploration power of the CGWW algorithm comes from its several features. First of all, the GWO algorithm has remarkable proficiency in exploring the search space and avoiding local optima as a part of the CGWW. Secondly, the WOA, which has various moving strategies, can avoid local optima by its random movement in most cases and explore the search space thoroughly. Thirdly, the chaotic maps are

Table 7 Results of compared algorithms on the composite benchmark functions

F		GA	PSO	ABC	MFO	SSA	AEFA	GWO	WOA	CGWW
14	Best	**0.998**	**0.998**	**0.998**	**0.998**	**0.998**	1.0107	**0.998**	**0.998**	**0.998**
	Worst	5.9288	7.874	1.0053	5.9288	**0.998**	10.7763	12.6705	10.7632	**0.998**
	Aver	1.9877	3.2674	0.99883	2.5767	**0.998**	4.4665	5.5902	2.7682	**0.998**
	Stdd	1.5427	2.4552	0.0022993	2.3335	1.66E−16	3.3362	4.9911	2.9771	**0**
15	Best	0.00062025	0.00065566	0.00062008	0.00080801	0.00061086	0.0015347	0.00030752	0.00032392	**0.00030755**
	Worst	0.020714	0.0083337	0.00086265	0.0083337	0.020365	0.019672	0.0015948	0.0022519	**0.00053222**
	Aver	0.0039725	0.001771	0.00075548	0.0021187	0.0049533	0.010245	0.00049982	0.00081913	**0.00037017**
	Stdd	0.0065172	0.0023313	8.95E−05	0.002201	0.0081286	0.0061943	0.00039149	0.00061805	**8.63E−05**
16	Best	**−1.0316**	**−1.0316**	**−1.0316**	**−1.0316**	**−1.0316**	**−1.0316**	**−1.0316**	**−1.0316**	**−1.0316**
	Worst	**−1.0316**	**−1.0316**	**−1.0316**	**−1.0316**	**−1.0316**	**−1.0316**	**−1.0316**	**−1.0316**	**−1.0316**
	Aver	**−1.0316**	**−1.0316**	**−1.0316**	**−1.0316**	**−1.0316**	**−1.0316**	**−1.0316**	**−1.0316**	**−1.0316**
	Stdd	7.25E−08	**1.05E−16**	2.21E−07	0	1.27E−14	0	3.54E−08	2.82E−10	1.16E−08
17	Best	**0.39789**	**0.39789**	**0.39789**	**0.39789**	**0.39789**	**0.39789**	**0.39789**	**0.39789**	**0.39789**
	Worst	**0.39789**	**0.39789**	**0.39789**	**0.39789**	**0.39789**	**0.39789**	**0.39789**	0.39791	**0.39789**
	Aver	**0.39789**	**0.39789**	**0.39789**	**0.39789**	**0.39789**	**0.39789**	**0.39789**	0.3979	**0.39789**
	Stdd	1.50E−07	**0**	1.18E−09	**0**	8.32E−15	**0**	3.97E−07	9.01E−06	1.49E−06
18	Best	**3**	**3**	**3**	**3**	**3**	**3**	**3**	**3**	**3**
	Worst	**3**	**3**	**3**	**3**	**3**	**3**	3.0001	3.0002	**3**
	Aver	**3**	**3**	**3**	**3**	**3**	**3**	**3**	**3**	**3**
	Stdd	6.95E−06	**1.36E−15**	4.55E−06	1.63E−15	5.44E−14	1.57E−15	2.79E−05	6.29E−05	5.93E−06
19	Best	**−3.8628**	**−3.8628**	**−3.8628**	**−3.8628**	**−3.8628**	**−3.8628**	**−3.8628**	−3.8627	**−3.8628**
	Worst	**−3.8628**	−3.8613	**−3.8628**	**−3.8628**	**−3.8628**	−3.8626	−3.8556	−3.8049	**−3.8628**
	Aver	**−3.8628**	−3.8625	**−3.8628**	**−3.8628**	**−3.8628**	−3.8627	−3.8617	−3.8552	**−3.8628**
	Stdd	2.90E−08	0.00048677	7.49E−10	9.36E−16	1.72E−12	7.37E−05	0.0023177	0.017911	**9.00E−16**
20	Best	**−3.322**	**−3.322**	**−3.322**	**−3.322**	**−3.322**	**−3.322**	**−3.322**	−3.3216	**−3.322**
	Worst	−3.2031	−3.1345	**−3.322**	−3.2031	−3.1337	−3.2029	−3.1375	−3.147	**−3.322**
	Aver	−3.2863	−3.2484	**−3.322**	−3.2507	−3.238	−3.3101	−3.2794	−3.2717	**−3.322**
	Stdd	0.057431	0.081061	9.39E−07	0.061396	0.074563	0.037656	0.07079	0.072752	**4.68E−16**
21	Best	**−10.1532**	**−10.1532**	**−10.1532**	**−10.1532**	**−10.1532**	**−10.1532**	−10.1525	−10.1487	−10.151
	Worst	−2.6305	−2.6829	**−9.4575**	−2.6305	−2.6305	−5.0552	−3.0846	−5.0549	−5.0552
	Aver	−5.1605	−7.7346	**−10.0532**	−6.1441	−7.9069	−5.565	−8.9398	−9.6296	−9.5831
	Stdd	3.5258	3.1961	**0.2234**	3.5765	3.617	1.6121	2.5986	1.6074	1.5922
22	Best	**−10.4029**	**−10.4029**	**−10.4029**	**−10.4029**	**−10.4029**	**−10.4029**	−10.4022	−10.3975	−10.4024
	Worst	−2.7519	−1.8376	**−10.4029**	−2.7519	−2.7659	−5.0877	−2.5854	−3.7215	−10.3079
	Aver	−7.1027	−7.2922	**−10.4029**	−7.5384	−9.1118	−8.2768	−9.6195	−6.5257	−10.385
	Stdd	3.5227	3.4171	**1.09E−05**	3.7121	2.7783	2.7448	2.4715	2.6683	0.030991
23	Best	**−10.5364**	**−10.5364**	−10.5363	**−10.5364**	**−10.5364**	**−10.5364**	−10.5356	−10.5322	**−10.5364**
	Worst	−2.4217	−2.4217	−5.124	−2.4217	−5.1756	−2.4273	−10.533	−2.411	**−10.5364**
	Aver	−6.7109	−7.8413	−9.9263	−7.6114	−10.0003	−9.3083	−10.5343	−7.8163	**−10.5364**
	Stdd	4.0507	3.5984	1.6892	3.852	1.6952	2.7503	0.00096393	3.5949	**1.53E−05**

applied in the initialization phase and during the iterations of the hybrid algorithm for solving the stagnation problem of search agents. So, the proposed algorithm is prevented from getting stuck in local optima, and its exploration ability is improved. The obtained results in Table 6 indicate that the CGWW algorithm finds the minimum values for the F9, F10, and F11. For the F8 function, the minimum value for the best solution is found by the proposed algorithm. But, the minimum value of worst and average solutions is obtained by the GA. For F12 and F13 problems, the CGWW algorithm finds the minimum for the average, worst, and standard deviation. But, the minimum of the best answer is found by the PSO algorithm. Therefore, the CGWW presents a high performance in the exploration phase and can explore the search space to obtain more promising areas and

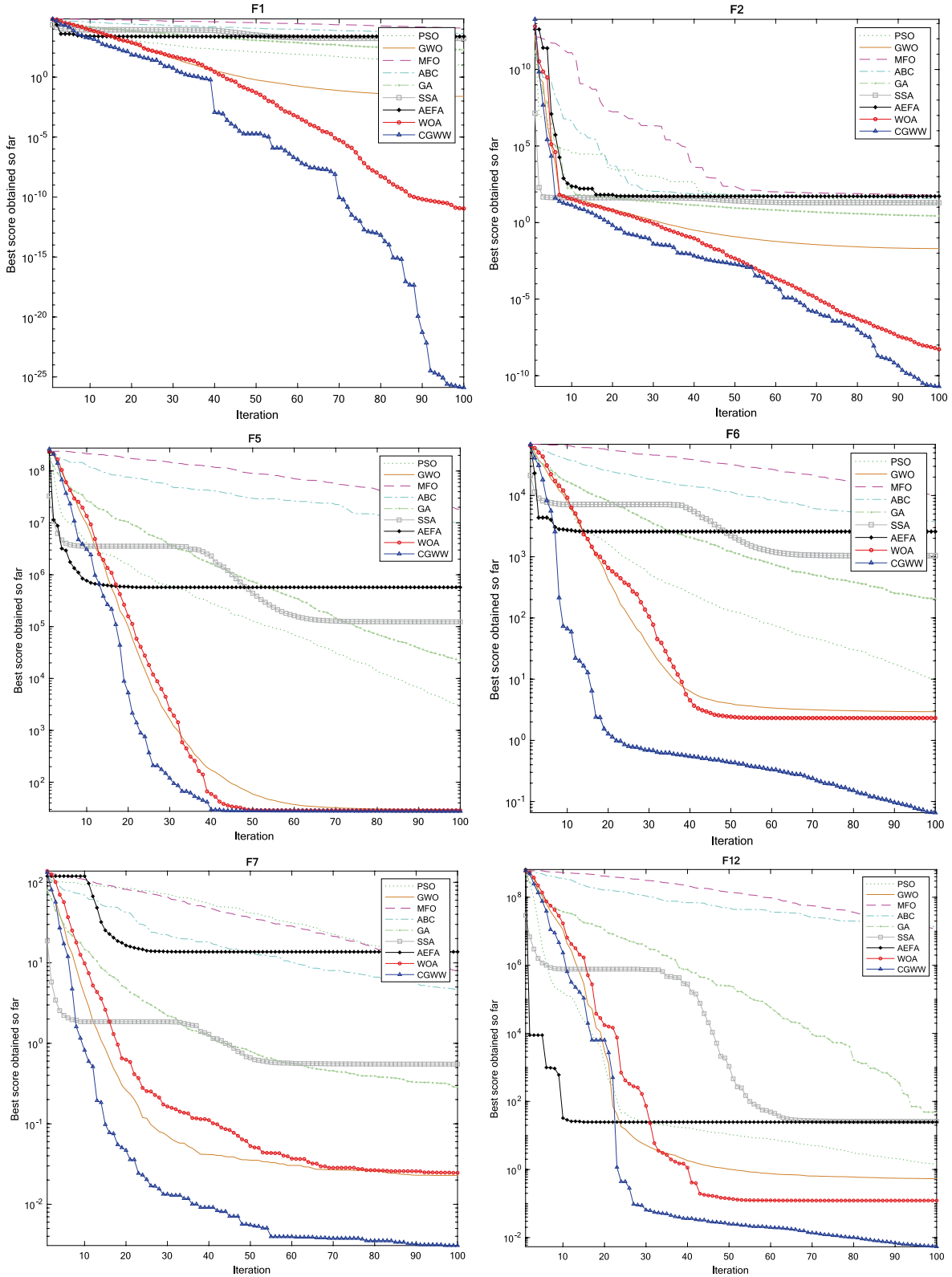

Fig. 3 Convergence curves of compared optimization algorithms

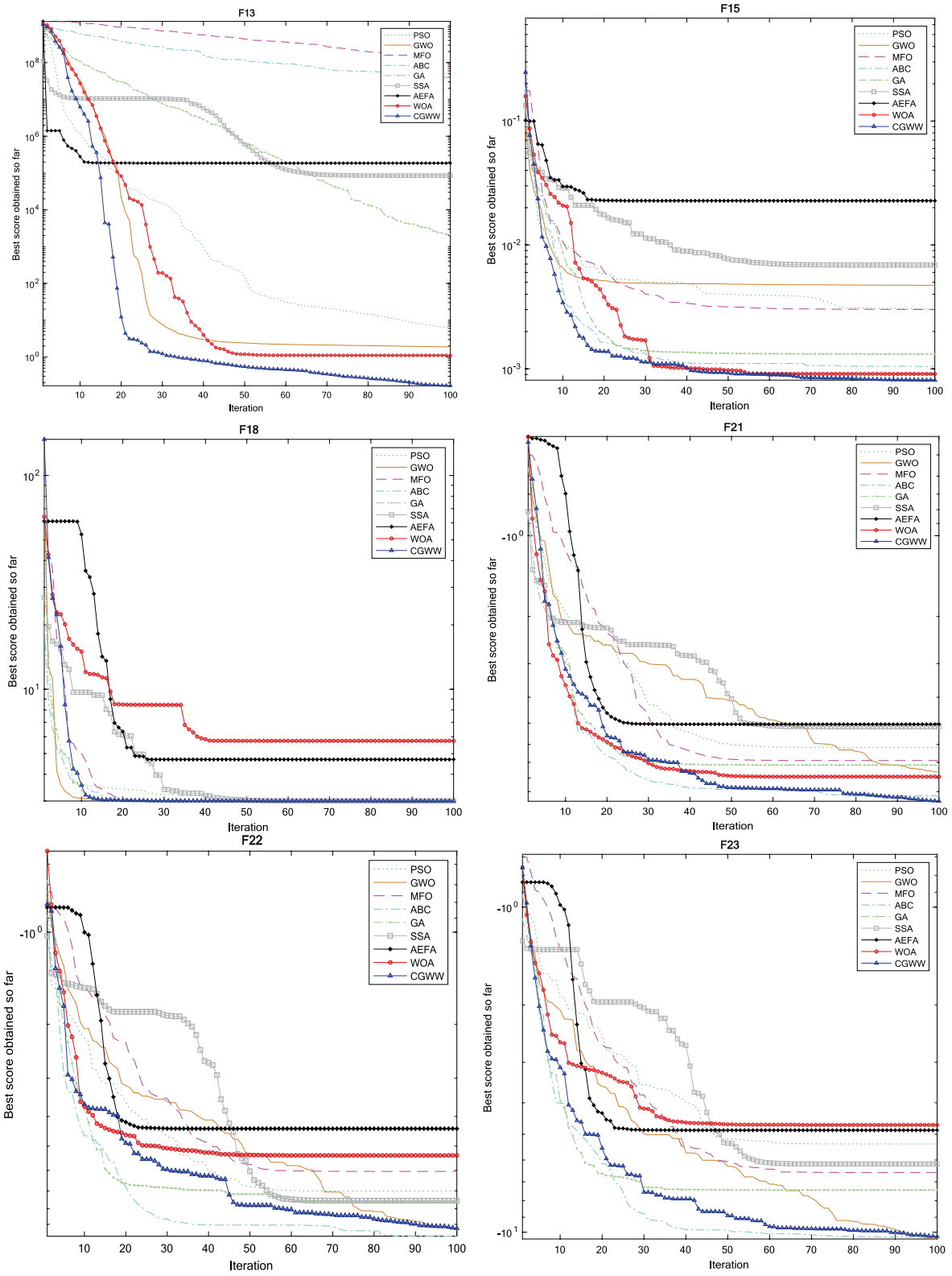

Fig. 3 (continued)

the global optimum for most of the multimodal benchmark functions by avoiding all local optima.

4.3.3 Results for solving the composite benchmark evaluation functions

The third group of test functions, named composite (F14 to F23), have fewer local optima and dimensions than the multimodal group. For solving the composite benchmark functions, which is an exceptionally challenging job, the optimization algorithms should establish a balance between the exploration and exploitation operations. Table 7 presents the execution results of the compared algorithms on the composite benchmark functions. According to the obtained results, the CGWW algorithm can find the best solutions on 8 out of 10 composite benchmark functions. The GWO and WOA algorithms have acceptable performance for balancing the exploration and exploitation operations. The hybrid CGWW algorithm inherits this skill from both algorithms. The chaotic maps and roulette wheel operator also reinforce the hybrid algorithm to perform the exploration and exploitation phases more accurately. Therefore, the hybrid algorithm focuses on the exploration and does the long and sudden movements in the early iterations. Then the proposed algorithm considers short and gradual movements to perform the exploitation in the last repetitions. For the F14, F15, F19, F20, and F23 composite benchmark functions, the CGWW algorithm finds the minimum best, worst, average, and standard deviation. The compared algorithms have similar behavior for finding the minimum best, worst, and average values for the F16, F17, and F18 functions. But, about the standard deviation value, the PSO is better than the other algorithms. Finally, the ABC is superior to the others on the F21 and F22 functions in finding the minimum value for the best, worst, average, and standard deviations. The results of Table 7 confirm that the CGWW is capable of solving complicated optimization problems because the search spaces of the composite test functions are very similar to the search space of real-world problems. Therefore, the CGWW presents a high performance to balance the exploration and exploitation phases to solve problems having a complicated search space.

4.4 Convergence analysis of proposed algorithm

In this section, the convergence speed of the proposed algorithm is compared with the other optimization methods. The purpose of convergence analysis is to demonstrate the exploration and exploitation mechanisms of the CGWW algorithm. The convergence curves of GA, PSO, ABC, MFO, GWO, WOA, SSA, AEFA, and CGWW algorithms are demonstrated for some of the benchmark test functions in Fig. 3. a and b. For each iteration of the mentioned algorithms, the average of the best solution's value for 100 times of executions is plotted as the convergence curve in Fig. 3a and b. The search agents investigate the promising areas of the search space during the iterations of the CGWW and perform the exploitation operations. These search agents move with a high velocity in the beginning steps and then gradually converge at a lower speed to a near-optimal solution. Figure 3 demonstrates that the convergence performance of the CGWW algorithm For F1, F2, F6, F7, F12, and F13 benchmark functions is much better than other algorithms. The convergence curve of the proposed algorithm is very similar to the WOA and GWO for the F5. Competition between the WOA and CGWW is very intense but, after iteration 65, the CGWW is the winner. The same conditions exist about the F21 function between the ABC and CGWW algorithms. For the F18, the GWO algorithm has the best convergence curve between the others. Finally, for the F22 and F23, the ABC algorithm has the best performance about the convergence curve. The investigation results imply that the CGWW presents the fastest convergence for most of the benchmark functions. For the remaining benchmarks, the obtained results of the CGWW are competitive with those of the other compared meta-heuristic optimization methods.

4.5 Results of the proposed algorithm and the Literature

In this section, the results of the CGWW are compared with the existing ones in the literature. The experimental conditions in the literature are applied for the execution process of the proposed algorithm. Table 8 presents the adopted results from [3, 25] for the WOA, PSO, DE, and GWO, along with the results of the CGWW, using the same experimental conditions, for solving 23 benchmark test functions. The number of iterations and search agents for all the algorithms are considered 500 and 30, respectively. The results in Table 8 indicate that the CGWW algorithm could find the best solutions in 14 out of 23 benchmark functions. The results of applying the GA, ABC, SBO, and ALO algorithms for solving 13 unimodal and multimodal benchmark functions, extracted from [12], are also presented in Table 9. According to experimental conditions in [12], the number of iterations for the proposed algorithm is assumed to be 1000. The results in Table 9 indicates that the CGWW algorithm could find the best solutions in 12 out of 13 benchmark functions. The results of Tables 8 and 9 show that the CGWW algorithm presents a remarkable performance for solving unimodal, multimodal, and composite benchmarks by establishing a proper balance between the exploration and exploitation stages of the proposed algorithms.

Table 8 Results of the CGWW algorithm in comparison with the results in [3, 25]

Function number		DE	PSO	GWO	WOA	CGWW
1	Aver	8.2E−14	0.000136	6.59E−28	1.41E−30	**7.65E−155**
	Stdd	5.9E−14	0.000202	6.34E−05	4.91E−30	**2.42E−154**
2	Aver	1.5E−09	0.042144	7.18E−17	1.06E−21	**1.46E−87**
	Stdd	9.9E−10	0.045421	0.029014	2.39E−21	**4.63E−87**
3	Aver	6.8E−11	70.12562	3.29E−06	5.39E−07	**2.39E−32**
	Stdd	7.4E−11	22.11924	79.14958	2.93E−06	**7.56E−32**
4	Aver	**0**	1.086481	5.61E−07	0.072581	4.50E−89
	Stdd	**0**	0.317039	1.315088	0.39747	1.42E−88
5	Aver	**0**	96.71832	26.81258	27.86558	23.7855
	Stdd	**0**	60.11559	69.90499	0.763626	8.2609
6	Aver	**0**	0.000102	0.816579	3.116266	0.00015043
	Stdd	**0**	8.28E−05	0.000126	0.532429	2.99E−05
7	Aver	0.00463	0.122854	0.002213	0.001425	**0.00096686**
	Stdd	0.0012	0.044957	0.100286	0.001149	**0.00083983**
8	Aver	−11,080.1	−4841.29	−6123.1	−5080.76	**−12,502.2565**
	Stdd	574.7	1152.814	−4087.44	695.7968	144.8139
9	Aver	69.2	46.70423	0.310521	**0**	**0**
	Stdd	38.8	11.62938	47.35612	**0**	**0**
10	Aver	9.7E−08	0.276015	1.06E−13	7.4043	**8.88E−16**
	Stdd	4.2E−08	0.50901	0.077835	9.897572	**0**
11	Aver	**0**	0.009215	0.004485	0.000289	**0**
	Stdd	**0**	0.007724	0.006659	0.001586	**0**
12	Aver	**7.9E−15**	0.006917	0.053438	0.339676	2.25E−05
	Stdd	**8E−15**	0.026301	0.020734	0.214864	8.16E−06
13	Aver	**5.1E−14**	0.006675	0.654464	1.889015	0.0024956
	Stdd	**4.8E−14**	0.008907	0.004474	0.266088	0.0045749
14	Aver	0.998004	3.627168	4.042493	2.111973	**0.998**
	Stdd	3.3E−16	2.560828	4.252799	2.498594	**9.84E−11**
15	Aver	**4.5E−14**	0.000577	0.000337	0.000572	0.00037017
	Stdd	0.00033	0.000222	0.000625	0.000324	**9.63E−05**
16	Aver	−1.03163	−1.03163	−1.03163	−1.03163	**−1.0316**
	Stdd	3.1E−13	6.25E−16	−1.03163	4.2E−07	**3.62E−08**
17	Aver	0.397887	**0.397887**	0.397889	0.397914	0.39789
	Stdd	9.9E−09	**0**	0.397887	2.7E−05	6.97E−07
18	Aver	**3**	**3**	3.000028	**3**	**3**
	Stdd	2E−15	**1.33E−15**	3	4.22E−15	**5.93E−06**
19	Aver	N/A	**−3.86278**	−3.86263	−3.85616	−3.8625
	Stdd	N/A	**2.58E−15**	−3.86278	0.002706	0.00048677
20	Aver	N/A	−3.26634	−3.28654	−2.98105	**−3.3101**
	Stdd	N/A	0.060516	−3.25056	0.376653	**0.037656**
21	Aver	**−10.1532**	−6.8651	−10.1514	−7.04918	−10.1204
	Stdd	**0.0000025**	3.019644	−9.14015	3.629551	0.080657
22	aver	−10.4029	−8.45653	−10.4015	−8.18178	**−10.375**
	Stdd	3.9E−07	3.087094	−8.58441	3.829202	**0.071551**
23	Aver	**−10.5364**	−9.95291	−10.5343	−9.34238	−10.5352
	Stdd	**1.9E−07**	1.782786	−8.55899	2.414737	0.0014377

Table 9 Results of CGWW algorithm in comparison with the results in [12]

Function number		ABC	GA	SBO	ALO	CGWW
1	Aver	4.44	0.146	0.00337	4.60E−7	**0**
	Stdd	3.2672	0.089	0.0016	4.60E−7	**0**
2	Aver	9.68E+1	0.077	0.0146	34.56	**2.84E−194**
	Stdd	22.600	0.014	0.0037	47.42	**0**
3	Aver	5.93E+4	1.85E+3	2.09E+2	2.67E+2	**2.0257E−36**
	Stdd	9.51E+3	740.60	55.33	116.84	**6.1568E−36**
4	Aver	55.84	2.06	0.262	7.84	**1.0941E−177**
	Stdd	4.68	0.25	0.0395	3.94	**0**
5	Aver	1.25E+6	99.89	85.44	273.75	**21.1825**
	Stdd	7.33E+5	39.65	67.39	443.74	**9.7944**
6	Aver	3.505	0.132	0.0028	6.69E−7	**3.5702E−05**
	Stdd	2.168	0.071	0.0012	4.30E−7	**1.6529E−05**
7	Aver	0.377	0.036	0.0037	0.0465	**0.00040644**
	Stdd	0.1147	0.010	8.116E−4	0.0183	**0.00046511**
8	Aver	−6.30E+3	−6.19e+3	−8.83E+3	−5.48E+3	**−12,560.8569**
	Stdd	978.86	1.178e+3	290.09	57.89	**20.9493**
9	Aver	2.24E+2	1.30e+2	16.96	69.58	**0**
	Stdd	13.93	67.64	3.315	25.53	**0**
10	Aver	3.014	0.078	0.0134	1.79	**8.88E−16**
	Stdd	0.684	0.021	0.0025	0.918	**0**
11	Aver	0.958	0.16	0.0082	0.010	**0**
	Stdd	0.138	0.061	0.0059	0.0095	**0**
12	Aver	3.15E+6	7.89E−4	1.89E−5	8.65	**5.862E−06**
	Stdd	1.46E+6	9.49E−4	3.39E−5	3.495	**1.7076E−06**
13	Aver	4.41E+6	0.004	**6.57E−6**	1.16E−2	8.67E−05
	Stdd	2.27E+6	0.004	**5.74E−6**	0.013	4.63E−05

4.6 Application of the proposed algorithm for intrusion detection

4.6.1 Feature selection and intrusion detection systems

Intrusion detection systems are classified into two main groups of anomaly detection and misuse or signature−based detection [72, 86]. Anomaly detection systems build profiles of normal network events. Network events that do not match with these profiles are classified as attacks. In misuse detection systems, attacks are detected based on their signatures and according to known attack methods. These systems cannot identify new network attacks because they decide based on the previously known attack types. But anomaly detection systems can detect new attacks if they do not conform to normal profiles. But the drawback of these systems is more false alarms due to the failure to identify previously unknown normal events. Hybrid models of the anomaly and misuse detection provide the ability to detect attacks more efficiently [86]. Due to the existing large number of features in each network event in intrusion detection systems, feature reduction is a significant problem to speed up attack detection. For feature selection, by removing related and duplicate features, an optimal subset of features is selected that represents the entire dataset. The optimal set of features is used to construct a classifier with high detection accuracy. The job function of a classifier is to classify network events and identify normal events from the attacks. Several classifiers are applied in intrusion detection systems. In this experiment, the proposed CGWW algorithm is used to select the optimal set from the set of available features for network intrusion detection. Other mentioned optimization algorithms are compared in this experiment with the proposed CGWW algorithm for solving the feature selection problem.

One of the simplest and most widely used forms of Bayesian networks is the Naive Bayesian network (NBN). NBN is a data mining tool used in various fields, such as a classifier in intrusion detection systems. The reason for the simplicity of this classifier is the independence of its constituent features from each other[87]. The focus of this experiment is more on the feature selection step of the intrusion detection systems. Therefore the NBN is applied to use the selected features in the previous step as a simple classifier.

4.6.2 The NSL KDD99 dataset

The NSL KDD99 is adopted as the intrusion detection dataset in the feature selection experiments. It is an updated

Table 10 Different types of attacks in NSL KDD99

Class of attack	Description	Attack name in the train and test set
DOS	Denial of Service	Teardrop, Smurf, and Neptune
Probe	Probing attack	Satan, Portsweep, and Saint
U2R	User to Root	Rootkit, Buffer_overflow, and Loadmodule
R2L	Remote to Local	Xsnoop, Httptunnel, and Password

Table 11 The intrusion detection and false-positive rate of compared algorithms for $k=5$, 10, 15 and 20

Algorithm	$k=5$		$k=10$		$k=15$		$k=20$	
	DR	FPR	DR	FPR	DR	FPR	DR	FPR
CGWW	**95.7999**	27.5049	**94.9505**	22.6032	92.3946	17.4029	90.8829	22.7886
GA	95.3479	27.1857	94.6077	22.9431	89.9088	16.3732	90.1582	24.6525
PSO	95.231	27.3195	89.5893	**16.3114**	82.4982	**14.7565**	89.3634	17.6192
GWO	92.0284	17.0116	94.3427	26.2692	88.5763	18.1032	89.7062	22.0781
ABC	91.9504	16.9807	94.1791	20.4716	**93.3453**	17.1764	**94.6544**	22.0781
SSA	88.8958	13.356	89.6205	21.1101	88.74	17.7737	93.8284	25.7234
AEFA	86.449	**13.2942**	85.4438	20.5231	87.688	24.6113	83.3476	**16.1981**
WOA	91.9504	16.9807	90.5868	16.9807	88.9737	21.2131	87.3841	20.8115

Table 12 The accuracy rate of compared algorithms for $k=5$, 10, 15 and 20

Algorithm	AR for $k=5$	AR for $k=10$	AR for $k=15$	AR for $k=20$
CGWW	85.7612	87.3891	88.1742	84.9938
GA	85.6414	87.0476	87.2028	83.7784
PSO	85.5172	87.0476	83.6808	86.3556
GWO	**88.1343**	85.464	85.6991	84.6301
ABC	88.1033	**87.8682**	**88.813**	**87.4468**
SSA	87.9258	84.9982	85.9342	85.4063
AEFA	86.5596	82.8735	82.39	83.5433
WOA	88.1033	87.327	84.5857	83.8538

version of the KDD99 dataset which most of the defects are fixed [88]. There are no duplicate records in the train and test set of NSL KDD99. This dataset contains 125,973 and 22,544 records for train and test sets, respectively. Each record in the NSL KDD dataset includes 41 features, which have three types of nominal, binary, and numeric. There are normal and anomaly records in the dataset. Anomaly records are categorized according to Table 10.

4.6.3 Evaluation criteria and fitness function for the feature selection problem

In this experiment, three evaluation criteria are used to compare different optimization algorithms for feature selection [89]. The first criterion is Detection Rate (DR), which refers to the percentage of the anomaly events which are correctly classified. The second criterion is False Positive Rate percentage (FPR) which relates to the normal network events identified as attacks. The Accuracy Rate (AR) is the third criterion which is the percentage of correctly classified events. The DR and AR are expected to be high, but low values are awaited for the FPR. The DR and FPR are two inconsistent objectives for the feature selection problem, so, similar to the presented approach in [72], a weighted sum value of the DR and FPR is applied as the fitness function for this experiment. However, the number of selected features and AR are the objectives of the proposed approach of [72]. The fitness function to evaluate the feature subsets, obtained with optimization algorithms, is presented in Eq. (25).

$$\text{Fitness of each subset} = m * DR + n * (1 - \text{FPR}) \quad (25)$$

Because the DR value is more important to identify the attacks, we assumed $m=0.7$, $n=0.3$. Another important measure to deal with the feature selection problem is the number of selected features subset. For the simplicity of the problem, we assumed the number of features to be $k=5$, 10, 15, and 20. But the best way to consider all measures for solving the problem is by applying a multi-objective algorithm with an objective function for selected features count.

4.6.4 Solution representation

For the feature selection problem, each search agent of all the algorithms contains a solution. This solution is in an array form consists of 41 floating-point numbers for 41 features. Each number in the array represents the importance of the related feature in the feature set. After some iterations of each algorithm, the k most significant features in candidate

solutions are selected to perform classification over the test set. For example, consider the solution represented in Eq. (26) for ten numbers related to ten features. If $k=5$, the features 3, 4, 6, 8, and 10 have the five highest values in the array, so they are selected for the classification.

A sample solution for 10 features

$$= [12.3, 15, 16.7, 32, 11, 87.9, 11.2, 17.2, 15.7, 42] \quad (26)$$

4.6.5 Comparison of optimization algorithms for feature selection

The proposed CGWW algorithm is compared with the GA, PSO, GWO, ABC, SSA, and AEFA for solving the feature selection problem of intrusion detection systems. Table 11 shows the DR and FPR values resulted from compared algorithms for NSLKDD 99 dataset.

Table 11 shows that the CGWW presents better performance than the other algorithms for $k=5$ and $k=10$. For $k=15$, the DR value for CGWW is lower than ABC and higher than the others. For $k=20$, the DR value for CGWW is lower than ABC and SSA and higher than the remaining algorithms. Table 11 shows that 10 is a reasonable number for the number of selected features. Table 11 also shows that the proposed CGWW algorithm has high FPR for $k=5$ and $k=20$. If the number of selected features is 10 and 15, the FPR value for the CGWW algorithm is lower than 2 and 4 compared algorithms, respectively. Table 12 shows the AR values resulted from compared algorithms for the NSLKDD 99 dataset.

Results in Table 12 presents that the proposed CGWW algorithm has better AR than 6 of the compared algorithms for $k=10$ and 15 selected features. But for $k=5$ and 20, the AR value of the CGWW algorithm is not so noticeable. According to the results of Tables 10, 11, and 12 the CGWW algorithm can find the second-best solutions in most cases for the feature selection problem in the intrusion detection systems.

The conducted experiments indicate that the proposed CGWW algorithm presents competitive performance compared with the other optimization algorithms for solving the benchmark and real-world applications.

5 Conclusion

In the present paper, a chaotic and hybrid optimization algorithm has been presented named CGWW. The CGWW algorithm has been developed by the composition of the GWO and improved WOA algorithms to utilize the strength points of both algorithms. Applying the chaotic maps instead of random generators in the initialization and controlling parameters of the CGWW algorithm is also led to an improvement in the exploration mechanism. The main modification of the WOA is the utilization of the roulette wheel selection method to choose the search agents with better scores and consequently improve the exploitation mechanism of the CGWW. The second advantage of the proposed algorithm is its multi-swarm characteristic. A group of whales from WOA are cooperating with the gray wolves of GWO in the proposed algorithm to establish a more diverse population. For evaluating the proposed algorithm, 23 benchmark problems are employed. Furthermore, the CGWW algorithm is evaluated in solving the feature selection problem in intrusion detection systems. The experimental results presented that the CGWW algorithm can offer remarkably competitive outcomes compared with the other optimization algorithms such as PSO, MFO, ABC, GA, SSA, SBO, ALO, AEFA, GWO, and WOA.

There are some limitations in comparing the CGWW with some of the new optimization approaches. For some algorithms, the source code was inaccessible, or precise implementation was impossible. For the others, the investigated benchmark functions were different from those applied in this paper. Therefore, the CGWW algorithm may be compared with other state-of-the-art meta-heuristics in future works. The proposed algorithm also has some disadvantages. Firstly, it cannot find the optimum solution for one of the unimodal, one for multimodal, and two of the composite benchmark functions. Therefore, by more investigation on some parameters of the hybrid algorithm, obtaining better results is possible. The first set of parameters are those for balancing exploration and exploration. The second one is the member count of sub-populations. Secondly, as the computational complexity of the CGWW algorithm indicates, by adding the roulette wheel selection operator and sorting the total population of the hybrid algorithm to balance high-quality solutions between sub-populations, the running time of the algorithm increased. Using the chaotic maps consumes more computational time than the random generator. Nevertheless, the complexity order of the CGWW algorithm has no increase in comparison with other compared meta-heuristics. As the last point, developing a binary and multi-objective version of the CGWW algorithm for solving practical optimization problems in different areas could be future works.

Funding The authors received no financial support for the research.

Availability of data and material Data sharing is not applicable to this article as no new data were created or analyzed in this study.

Declarations

Conflict of interest The authors declare no conflicts of interest.

References

1. Alba, E., Dorronsoro, B.: The exploration/exploitation tradeoff in dynamic cellular genetic algorithms. IEEE Trans. Evol. Comput. **9**(2), 126–142 (2005). https://doi.org/10.1109/TEVC.2005.843751
2. Song, H., Triguero, I., Özcan, E.: A review on the self and dual interactions between machine learning and optimisation. Progress Artificial Intell. **8**(2), 143–165 (2019). https://doi.org/10.1007/s13748-019-00185-z
3. Mirjalili, S., Lewis, A.: The Whale optimization algorithm. Adv. Eng. Softw. **95**, 51–67 (2016). https://doi.org/10.1016/j.advengsoft.2016.01.008
4. Gandomi, A.H., Alavi, A.H.: Krill herd: A new bio-inspired optimization algorithm. Commun. Nonlinear Sci. Numer. Simul. **17**(12), 4831–4845 (2012). https://doi.org/10.1016/j.cnsns.2012.05.010
5. Holland, J.H.: Genetic algorithms. Sci. Am. **267**(1), 66–73 (1992)
6. Eberhart, R., Kennedy, J.: Particle swarm optimization, proceeding of IEEE International Conference on Neural Network. Perth, Australia, 1942–1948 (1995). https://doi.org/10.1109/ICNN.1995.488968
7. Mirjalili, S.: The ant lion optimizer. Adv. Eng. Softw. **83**, 80–98 (2015). https://doi.org/10.1016/j.advengsoft.2015.01.010
8. Karaboga, D., Gorkemli, B., Ozturk, C., Karaboga, N.: A comprehensive survey: artificial bee colony (ABC) algorithm and applications. Artif. Intell. Rev. **42**(1), 21–57 (2014). https://doi.org/10.1007/s10462-012-9328-0
9. Anita, Yadav, A.: AEFA: Artificial electric field algorithm for global optimization. Swarm and Evolutionary Computation **48**, 93–108 (2019). https://doi.org/10.1016/j.swevo.2019.03.013
10. Mirjalili, S., Gandomi, A.H., Mirjalili, S.Z., Saremi, S., Faris, H., Mirjalili, S.M.: Salp Swarm Algorithm: A bio-inspired optimizer for engineering design problems. Adv. Eng. Softw. **114**, 163–191 (2017). https://doi.org/10.1016/j.advengsoft.2017.07.002
11. Garg, H.: An efficient biogeography based optimization algorithm for solving reliability optimization problems. Swarm Evol. Comput. **24**, 1–10 (2015). https://doi.org/10.1016/j.swevo.2015.05.001
12. Samareh Moosavi, S.H., Khatibi Bardsiri, V.: Satin bowerbird optimizer: A new optimization algorithm to optimize ANFIS for software development effort estimation. Eng. Appl. Artif. Intell. **60**, 1–15 (2017). https://doi.org/10.1016/j.engappai.2017.01.006
13. Samareh Moosavi, S.H., Bardsiri, V.K.: Poor and rich optimization algorithm: A new human-based and multi populations algorithm. Eng. Appl. Artif. Intell. **86**, 165–181 (2019). https://doi.org/10.1016/j.engappai.2019.08.025
14. Kaveh, A., Bakhshpoori, T.: Water evaporation optimization: A novel physically inspired optimization algorithm. Comput. Struct. **167**, 69–85 (2016). https://doi.org/10.1016/j.compstruc.2016.01.008
15. Mirjalili, S., Mirjalili, S.M., Hatamlou, A.: Multi-verse optimizer: a nature-inspired algorithm for global optimization. Neural Comput. Appl. **27**(2), 495–513 (2016). https://doi.org/10.1007/s00521-015-1870-7
16. Wang, G.-G.: Moth search algorithm: a bio-inspired metaheuristic algorithm for global optimization problems. Memetic Comput.**10**(2), 151–164 (2018). https://doi.org/10.1007/s12293-016-0212-3
17. Saremi, S., Mirjalili, S., Lewis, A.: Grasshopper optimisation algorithm: Theory and application. Adv. Eng. Softw. **105**, 30–47 (2017). https://doi.org/10.1016/j.advengsoft.2017.01.004
18. Sulaiman, M.H., Mustaffa, Z., Saari, M.M., Daniyal, H.: Barnacles mating optimizer: A new bio-inspired algorithm for solving engineering optimization problems. Eng. Appl. Artif. Intell. **87**, 103330 (2020). https://doi.org/10.1016/j.engappai.2019.103330
19. Shayanfar, H., Gharehchopogh, F.S.: Farmland fertility: A new metaheuristic algorithm for solving continuous optimization problems. Appl. Soft Comput. **71**, 728–746 (2018). https://doi.org/10.1016/j.asoc.2018.07.033
20. Gharehchopogh, F.S., Shayanfar, H., Gholizadeh, H.: A comprehensive survey on symbiotic organisms search algorithms. Artif. Intell. Rev. **53**(3), 2265–2312 (2020). https://doi.org/10.1007/s10462-019-09733-4
21. Cheng, M.-Y., Prayogo, D.: Symbiotic organisms search: A new metaheuristic optimization algorithm. Comput. Struct. **139**, 98–112 (2014). https://doi.org/10.1016/j.compstruc.2014.03.007
22. Arora, S., Singh, S.: Butterfly optimization algorithm: a novel approach for global optimization. Soft. Comput. **23**(3), 715–734 (2019). https://doi.org/10.1007/s00500-018-3102-4
23. Heidari, A.A., Mirjalili, S., Faris, H., Aljarah, I., Mafarja, M., Chen, H.: Harris hawks optimization: Algorithm and applications. Futur. Gener. Comput. Syst. **97**, 849–872 (2019). https://doi.org/10.1016/j.future.2019.02.028
24. Mortazavi, A., Toğan, V., Nuhoğlu, A.: Interactive search algorithm: A new hybrid metaheuristic optimization algorithm. Eng. Appl. Artif. Intell. **71**, 275–292 (2018). https://doi.org/10.1016/j.engappai.2018.03.003
25. Mirjalili, S., Mirjalili, S.M., Lewis, A.: Grey wolf optimizer. Adv. Eng. Softw. **69**, 46–61 (2014). https://doi.org/10.1016/j.advengsoft.2013.12.007
26. Mafarja, M.M., Mirjalili, S.: Hybrid Whale optimization algorithm with simulated annealing for feature selection. Neurocomputing **260**, 302–312 (2017). https://doi.org/10.1016/j.neucom.2017.04.053
27. Mohammadzadeh, H., Gharehchopogh, F.S.: A novel hybrid whale optimization algorithm with flower pollination algorithm for feature selection: Case study Email spam detection. Computational Intelligence **n/a**(n/a) (2020). https://doi.org/10.1111/coin.12397
28. Jadhav, A.N., Gomathi, N.: WGC: Hybridization of exponential grey wolf optimizer with whale optimization for data clustering. Alex. Eng. J. (2017). https://doi.org/10.1016/j.aej.2017.04.013
29. Rahnema, N., Gharehchopogh, F.S.: An improved artificial bee colony algorithm based on whale optimization algorithm for data clustering. Multimedia Tools Appl. (2020). https://doi.org/10.1007/s11042-020-09639-2
30. Wolpert, D.H., Macready, W.G.: No free lunch theorems for optimization. IEEE Trans. Evol. Comput. **1**(1), 67–82 (1997). https://doi.org/10.1109/4235.585893
31. Nenavath, H., Jatoth, R.K.: Hybridizing sine cosine algorithm with differential evolution for global optimization and object tracking. Appl. Soft Comput. **62**, 1019–1043 (2018). https://doi.org/10.1016/j.asoc.2017.09.039
32. Garg, H.: A hybrid GSA-GA algorithm for constrained optimization problems. Inf. Sci. **478**, 499–523 (2019). https://doi.org/10.1016/j.ins.2018.11.041
33. Garg, H.: A hybrid PSO-GA algorithm for constrained optimization problems. Appl. Math. Comput. **274**, 292–305 (2016). https://doi.org/10.1016/j.amc.2015.11.001
34. Harish, G.: A Hybrid GA-GSA algorithm for optimizing the performance of an industrial system by utilizing uncertain data. In: Pandian, V. (ed.) Handbook of Research on Artificial Intelligence Techniques and Algorithms, pp. 620–654. IGI Global, Hershey, PA, USA (2015)

35. Li, Z., Wang, W., Yan, Y., Li, Z.: PS–ABC: A hybrid algorithm based on particle swarm and artificial bee colony for high-dimensional optimization problems. Expert Syst. Appl. **42**(22), 8881–8895 (2015). https://doi.org/10.1016/j.eswa.2015.07.043
36. Beigvand, S.D., Abdi, H., La Scala, M.: Hybrid gravitational search algorithm-particle swarm optimization with time vvarying acceleration coefficients for large scale CHPED problem. Energy **126**, 841–853 (2017). https://doi.org/10.1016/j.energy.2017.03.054
37. Kellert, S.H.: In the wake of chaos: Unpredictable order in dynamical systems. University of Chicago press, (1994)
38. Yang, D., Li, G., Cheng, G.: On the efficiency of chaos optimization algorithms for global optimization. Chaos, Solitons Fractals **34**(4), 1366–1375 (2007). https://doi.org/10.1016/j.chaos.2006.04.057
39. Alatas, B., Akin, E., Ozer, A.B.: Chaos embedded particle swarm optimization algorithms. Chaos, Solitons Fractals **40**(4), 1715–1734 (2009). https://doi.org/10.1016/j.chaos.2007.09.063
40. Alatas, B.: Chaotic harmony search algorithms. Appl. Math. Comput. **216**(9), 2687–2699 (2010). https://doi.org/10.1016/j.amc.2010.03.114
41. Alatas, B.: Chaotic bee colony algorithms for global numerical optimization. Expert Syst. Appl. **37**(8), 5682–5687 (2010). https://doi.org/10.1016/j.eswa.2010.02.042
42. Talatahari, S., Farahmand Azar, B., Sheikholeslami, R., Gandomi, A.H.: Imperialist competitive algorithm combined with chaos for global optimization. Commun. Nonlinear Sci. Numer. Simul. **17**(3), 1312–1319 (2012). https://doi.org/10.1016/j.cnsns.2011.08.021
43. Gandomi, A.H., Yang, X.-S., Talatahari, S., Alavi, A.H.: Firefly algorithm with chaos. Commun. Nonlinear Sci. Numer. Simul. **18**(1), 89–98 (2013). https://doi.org/10.1016/j.cnsns.2012.06.009
44. Wang, G.-G., Guo, L., Gandomi, A.H., Hao, G.-S., Wang, H.: Chaotic krill herd algorithm. Inf. Sci. **274**, 17–34 (2014). https://doi.org/10.1016/j.ins.2014.02.123
45. Arora, S., Singh, S.: An improved butterfly optimization algorithm with chaos. J. Intell. Fuzzy Syst. **32**(1), 1079–1088 (2017). https://doi.org/10.3233/JIFS-16798
46. Kohli, M., Arora, S.: Chaotic grey wolf optimization algorithm for constrained optimization problems. J. Comput. Des. Eng. (2017). https://doi.org/10.1016/j.jcde.2017.02.005
47. Kaur, G., Arora, S.: Chaotic Whale optimization algorithm. J. Comput. Des. Eng. (2018). https://doi.org/10.1016/j.jcde.2017.12.006
48. Majhi, S.K., Mishra, A., Pradhan, R.: A chaotic salp swarm algorithm based on quadratic integrate and fire neural model for function optimization. Progress Artif. Intell. **8**(3), 343–358 (2019). https://doi.org/10.1007/s13748-019-00184-0
49. Gandomi, A.H., Yang, X.-S.: Chaotic bat algorithm. J. Comput. Sci. **5**(2), 224–232 (2014). https://doi.org/10.1016/j.jocs.2013.10.002
50. Coelho, L.d.S., Mariani, V.C.: Use of chaotic sequences in a biologically inspired algorithm for engineering design optimization. Expert Syst. Appl. **34**(3), 1905–1913 (2008). https://doi.org/10.1016/j.eswa.2007.02.002
51. Das, S., Mullick, S.S., Suganthan, P.N.: Recent advances in differential evolution—an updated survey. Swarm Evol. Comput. **27**, 1–30 (2016). https://doi.org/10.1016/j.swevo.2016.01.004
52. Mirjalili, S.: Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm. Knowl. Based Syst. **89**, 228–249 (2015). https://doi.org/10.1016/j.knosys.2015.07.006
53. Karaboga, D.: An Idea Based on Honey Bee Swarm for Numerical Optimization, Technical Report - TR06. (2005).
54. Karaboga, D., Basturk, B.: A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. J. Global Optim. **39**(3), 459–471 (2007). https://doi.org/10.1007/s10898-007-9149-x
55. Garg, H.: Solving structural engineering design optimization problems using an artificial bee colony algorithm. J. Ind. Manage. Optim. **10**(3), 777–794 (2014). https://doi.org/10.3934/jimo.2014.10.777
56. Shah, H., Tairan, N., Garg, H., Ghazali, R.: Global Gbest guided-artificial bee colony algorithm for numerical function optimization. Computers **7**(4), 69 (2018). https://doi.org/10.3390/computers7040069
57. Xiang, T., Liao, X., Wong, K.-w.: An improved particle swarm optimization algorithm combined with piecewise linear chaotic map. Applied Mathematics and Computation **190**(2), 1637–1645 (2007). https://doi.org/10.1016/j.amc.2007.02.103
58. Liu, B., Wang, L., Jin, Y.-H., Tang, F., Huang, D.-X.: Improved particle swarm optimization combined with chaos. Chaos, Solitons Fractals **25**(5), 1261–1271 (2005). https://doi.org/10.1016/j.chaos.2004.11.095
59. Talatahari, S., Azar, B.F., Sheikholeslami, R., Gandomi, A.: Imperialist competitive algorithm combined with chaos for global optimization. Commun. Nonlinear Sci. Numer. Simul. **17**(3), 1312–1319 (2012)
60. Abdel-Basset, M., El-Shahat, D., Sangaiah, A.K.: A modified nature inspired meta-heuristic whale optimization algorithm for solving 0–1 knapsack problem. International Journal of Machine Learning and Cybernetics, 1–20 (2017). https://doi.org/10.1007/s13042-017-0731-3
61. He, D., He, C., Jiang, L.-G., Zhu, H.-w., Hu, G.-r.: Chaotic characteristics of a one-dimensional iterative map with infinite collapses. IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications **48**(7), 900–906 (2001). https://doi.org/10.1109/81.933333
62. Tavazoei, M.S., Haeri, M.: Comparison of different one-dimensional maps as chaotic search pattern in chaos optimization algorithms. Appl. Math. Comput. **187**(2), 1076–1085 (2007). https://doi.org/10.1016/j.amc.2006.09.087
63. Hilborn, R.C.: Chaos and nonlinear dynamics: an introduction for scientists and engineers. Oxford University Press on Demand, (2000)
64. May, R.M.: Simple mathematical models with very complicated dynamics. In: The Theory of Chaotic Attractors. pp. 85–93. Springer, (2004)
65. Takens, F.: An introduction to chaotic dynamical systems. In. Springer, (1988)
66. Peitgen, H.-O., Jürgens, H., Saupe, D.: Chaos and fractals: new frontiers of science. Springer Science & Business Media, (2006)
67. Li, Y., Deng, S., Xiao, D.: A novel Hash algorithm construction based on chaotic neural network. Neural Comput. Appl. **20**(1), 133–141 (2011). https://doi.org/10.1007/s00521-010-0432-2
68. Ott, E.: Chaos in dynamical systems. Cambridge university press, (2002)
69. Wolf, A.: Quantifying chaos with Lyapunov exponents. Chaos **16**, 285–317 (1986)
70. Coello, C.A.C., Lechuga, M.S.: MOPSO: a proposal for multiple objective particle swarm optimization. In: Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No.02TH8600), 12–17 May 2002, pp. 1051–1056 vol.1052
71. Coello, C.A.C., Pulido, G.T., Lechuga, M.S.: Handling multiple objectives with particle swarm optimization. IEEE Trans. Evol. Comput. **8**(3), 256–279 (2004). https://doi.org/10.1109/TEVC.2004.826067
72. Alamiedy, T.A., Anbar, M., Alqattan, Z.N.M., Alzubi, Q.M.: Anomaly-based intrusion detection system using multi-objective grey wolf optimisation algorithm. J. Ambient. Intell. Humaniz. Comput. **11**(9), 3735–3756 (2020). https://doi.org/10.1007/s12652-019-01569-8

73. Yao, X., Liu, Y., Lin, G.: Evolutionary programming made faster. IEEE Trans. Evol. Comput. **3**(2), 82–102 (1999). https://doi.org/10.1109/4235.771163
74. Digalakis, J.G., Margaritis, K.G.: On benchmarking functions for genetic algorithms. Int. J. Comput. Math. **77**(4), 481–506 (2001). https://doi.org/10.1080/00207160108805080
75. Molga, M., Smutnicki, C.: Test functions for optimization needs. (2005)
76. Yang, X.-S.: Test Problems in Optimization. (2010). arXiv preprint arXiv:1008.0549
77. Yang, X.-S.: Firefly algorithm. Stochastic Test Funct. Des. Optim **2** (2010). https://doi.org/10.1504/IJBIC.2010.032124
78. Jamil, M., Yang, X.-S.: A literature survey of benchmark functions for global optimisation problems. Int. J. Math. Modell. Numer. Optim. **4**(2), 150–194 (2013). https://doi.org/10.1504/IJMMNO.2013.055204
79. Zhu, G.-Y., Zhang, W.-B.: Optimal foraging algorithm for global optimization. Appl. Soft Comput. **51**, 294–313 (2017). https://doi.org/10.1016/j.asoc.2016.11.047
80. Liang, J., Suganthan, P., Deb, K.: Novel composition test functions for numerical global optimization. **2005**, 68–75 (2005). https://doi.org/10.1109/SIS.2005.1501604
81. Suganthan, P., Hansen, N., Liang, J., Deb, K., Chen, Y.-p., Auger, A., Tiwari, S.: Problem Definitions and Evaluation Criteria for the CEC 2005 Special Session on Real-Parameter Optimization. **341–357** (2005).
82. Dhiman, G., Kumar, V.: Spotted hyena optimizer: A novel bio-inspired based metaheuristic technique for engineering applications. Adv. Eng. Softw. **114**, 48–70 (2017). https://doi.org/10.1016/j.advengsoft.2017.05.014
83. Shi, Y., Eberhart, R.C.: Empirical study of particle swarm optimization. In: Proceedings of the 1999 Congress on Evolutionary Computation-CEC99, 6–9 July 1999, pp. 1945–1950 Vol. 1943
84. Eberhart, R.C., Shi, Y.: Comparing inertia weights and constriction factors in particle swarm optimization. In: Proceedings of the 2000 Congress on Evolutionary Computation. CEC00 (Cat. No.00TH8512), 16–19 July 2000 pp. 84–88 vol.81
85. Shi, Y., Eberhart, R.C.: Parameter selection in particle swarm optimization. In, Berlin, Heidelberg 1998. Evolutionary Programming VII, pp. 591–600. Springer Berlin Heidelberg
86. Depren, O., Topallar, M., Anarim, E., Ciliz, M.K.: An intelligent intrusion detection system (IDS) for anomaly and misuse detection in computer networks. Expert Syst. Appl. **29**(4), 713–722 (2005). https://doi.org/10.1016/j.eswa.2005.05.002
87. Koc, L., Mazzuchi, T.A., Sarkani, S.J.E.S.w.A.: A network intrusion detection system based on a Hidden Naïve Bayes multiclass classifier. **39**(18), 13492–13500 (2012). https://doi.org/10.1016/j.eswa.2012.07.009
88. UNB ISCX, NSL-KDD. In. Information security Centre of Excellence (ISCX), Univ. New Brunswick, (2015)
89. Chen, R., Cheng, K., Chen, Y., Hsieh, C.: Using Rough Set and Support Vector Machine for Network Intrusion Detection System. In: 2009 First Asian Conference on Intelligent Information and Database Systems, 1–3 April 2009, pp. 465–470