A Chaotic Grey Wolf Optimizer Applied to Condition-Based Maintenance Optimization

Leonardo Ramos Rodrigues
Electronics Engineering Division
Aeronautics Institute of Technology - ITA
São José dos Campos-SP, Brazil, 12228-900
Email: leonardolrr2@fab.mil.br

João Paulo Pordeus Gomes
Department of Computer Science
Federal University of Ceará - UFC
Fortaleza-CE, Brazil, 60440-554
Email: jpaulo@dc.ufc.br

***Abstract*—The Grey Wolf Optimizer (GWO) algorithm is a nature-inspired population-based metaheuristic that simulates the social behavior observed in a grey wolf pack. GWO has been successfully applied to different optimization problems. In this paper, we propose a chaotic version of GWO, denoted by CGWO, that uses a chaotic variable to define the number of wolves in the pack that will act as leaders, i.e. the number of wolves that guide the hunting process in each iteration of the algorithm. The proposed algorithm is used to find the optimal maintenance scope for a series-parallel system. We assume that a Prognostics and Health Management (PHM) system is available and provides the degradation level and the Remaining Useful Life (RUL) prediction for each component. The goal is to find the maintenance scope that minimizes the expected total cost per cycle until the next maintenance intervention. The performance of the proposed model is compared with the performance of the original GWO and the well-studied Ant Colony Optimization algorithm (ACO). Different chaotic maps were tested. The results show that the proposed model presented a competitive performance.**

***Keywords*—Grey Wolf Optimizer; Chaotic Maps; Condition-Based Maintenance; Maintenance Optimization**

I. Introduction

Bio-inspired optimization algorithms have become very popular due to their capability of finding close-to-optimal solutions in an acceptable amount of time, even for difficult optimization problems. Methods such as Particle Swarm Optimization (PSO) [1], Genetic Algorithms (GA) [2] and Ant Colony Optimization (ACO) [3] are among the most common alternatives to classical gradient-based optimization methods. Although such methods reached remarkable results, no method was able to outperform the existing ones in all optimization problems [4]. As a result, proposing new metaheuristic algorithms and developing improvements for the existing ones is still an active research topic [5]. Also, the large number of existing metaheuristic algorithms allows researchers to develop hybrid methods that take into account the advantages of different metaheuristics to build solutions for complex problems [6], [7].

In [8], the authors introduced a new bio-inspired technique, named Grey Wolf Optimizer (GWO). This algorithm is based on the leadership hierarchy and the hunting mechanisms observed in grey wolves. The GWO algorithm is a competitive alternative for the solution of different optimization problems such as optimal power flow problems [9], feature selection problems [10], human recognition [11], among others.

Although the GWO algorithm is a recently proposed metaheuristic, many variants have been proposed aiming at improving the performance of the algorithm in different aspects. In [10], the authors propose a binary version of GWO. The performance of the proposed variant is assessed using classification benchmark problems from a public data repository. The binary version presented good results when compared to the results obtained with a Particle Swarm Optimization (PSO) and a Genetic Algorithm (GA). In [12], the authors propose a modified version of GWO in which a statistical mean parameter is used to modify the original mathematical formulation of GWO to improve the balance between the intensification and the diversification capabilities of the algorithm.

Aiming at improving the convergence rate of GWO, a hybrid version of GWO using an elite opposition-based learning strategy was proposed in [13]. In [14], the authors proposed a hybrid PSO-GWO algorithm to balance the local and global search capabilities of GWO. Another hybrid version of GWO was presented in [15], where the authors combined GWO with the Cuckoo Search (CS) algorithm in order to improve the performance of GWO in high-dimensional problems.

Most of the evolutionary algorithms include in their mathematical formulation a parameter that is randomly changed during the execution of the method. A study on the impact of using chaotic parameters instead of random parameters in evolutionary algorithms is presented in [16]. Many works have been published reporting the successful use of chaotic mechanisms in different algorithms such as Symbiotic Organisms Search (SOS) [17], Firefly algorithm [18], Whale Optimization [19], and Teaching-Learning Based Optimization (TLBO) [20].

Chaotic versions of GWO have also been investigated recently. The first variant of GWO that incorporated a chaotic mechanism was proposed in [21], where the authors introduced a Chaotic Local Search (CLS) that is conducted at the end of each iteration of the algorithm. A search neighborhood centered at the current position of the wolf with the best fitness value is considered, and the size of the search neighborhood is narrowed during the execution of the algorithm. Tests using twelve chaotic maps and several benchmark functions were conducted, and the authors concluded that the use of chaotic mechanisms improved the balance between the diversification and intensification capabilities of GWO, improving its performance.

In [22], the authors used a different strategy to include a chaotic mechanism in GWO. Aiming at improving the algorithm capability to escape from local optima, they replaced the random procedure used to define the control parameters in the original GWO with a chaotic sequence. The authors concluded that, due to the pseudo-randomness and the ergodicity of chaotic variables, this strategy allows the algorithm to alternate between diversification and intensification, resulting in a better performance.

In this paper, we propose a new strategy to introduce a chaotic mechanism into the GWO algorithm, denoted by CGWO. In the proposed version, instead of using a fixed number of leaders during the execution of the algorithm, a chaotic variable is used in order to define, in each iteration, the number of wolves that are considered as leaders in the pack. The proposed model can be used with other modified versions of GWO in a straightforward way.

The remaining sections of this paper are organized as follows. Section II presents the original version of GWO. Section III describes the proposed version of GWO. Section IV presents the maintenance scope definition problem. Section V illustrates the application of the proposed version of GWO in a numerical example of the maintenance scope definition problem. Concluding remarks and opportunities for future research are presented in section VI.

II. Grey Wolf Optimizer

The Grey Wolf Optimizer (GWO) algorithm is a population-based metaheuristic inspired by the behavior of grey wolves, originally proposed in [8]. The algorithm simulates both the leadership hierarchy and the hunting mechanisms observed in grey wolves in nature. Grey wolves often prefer to live in a packs with a well-defined hierarchy, as presented in Fig. 1.

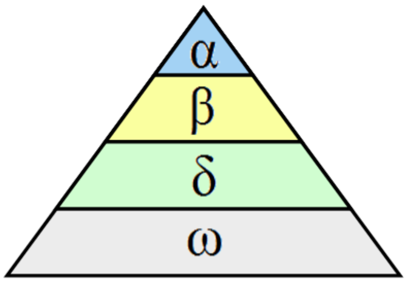

Figure 1. Hierarchy of a grey wolf pack [8].

The social hierarchy in a pack contains four levels that are described as follows:

- The alpha wolves (α) are at the top of the hierarchy. They are the leaders of the pack and are responsible for making decisions that are followed by all other wolves. These decisions are related to hunting time, prey selection, sleeping place, etc.
- Beta wolves (β) are in the second level in the pack hierarchy. They coordinate the lower-level wolves, acting as advisors of the alphas and helping them in their activities.
- Delta wolves (δ) are in the third level of the hierarchy. Their main function in the pack is to execute the decisions of the leaders. They are responsible for performing different tasks such as watching the territory, helping the alphas and betas during a hunt, taking care of weak wolves, among others.
- In the lowest hierarchy level are the omega wolves (ω). Omega wolves submit to all the other dominant wolves.

The GWO algorithm model considers not only the social hierarchy observed in a pack, but also the group hunting behavior. The grey wolf hunting procedure can be divided into the following three main phases [23], [24]:

- Tracking, chasing, and approaching the prey.
- Pursuing, encircling, and harassing the prey.
- Attacking the prey.

The social hierarchy and the hunting behavior described above are modeled to optimize the GWO algorithm. First, a group of candidate solutions for the optimization problem under consideration is randomly generated in the search space. Then, the best, the second best, and the third best solutions in the initial population are identified as the α, β, and δ wolves, respectively. Then, the positions of the wolves are updated as a function of their distances to the α, β, and δ wolves, aiming at getting closer to the prey and encircle it, as explained in the following sections.

A. *GWO Mathematical Model*

Let M be the number of wolves in the pack. Also, let $X_i = [x_1, x_2, \ldots, x_K]$ be the vector that defines the position of the i-th wolf. The model used to update the position of each wolf based on the encircling behavior is presented in Eqs. (1) and (2).

$$D = |C \cdot X_p(t) - X(t)| \tag{1}$$
$$X(t+1) = X_p(t) - A \cdot D \tag{2}$$

where t is the current iteration, $X(t)$ is the current position of the wolf, and $X_p(t)$ is the position of the prey. A and C are vectors computed according to Eqs. (3) and (4), respectively.

$$A = 2a \cdot r_1 - a \tag{3}$$
$$C = 2 \cdot r_2 \tag{4}$$

where r_1 and r_2 are random vectors in [0,1], and a is a vector whose components decrease linearly from 2 to 0 during the execution of the algorithm.

Figure 2 illustrates the encircling behavior of the wolves while hunting in a 2D scenario. A wolf updates its position based on the estimated position of the prey. A wolf may move to different new positions based on vectors A and C. In the example presented in Fig. 2, the position of a grey wolf that is originally in point (x_W, y_W) is updated considering the estimated position of the prey, denoted by (x_P, y_P). Different values for vectors A and C allows the wolf to move to different new locations. Vectors r_1 and r_2 allow the wolf to move to any point in the indicated area.

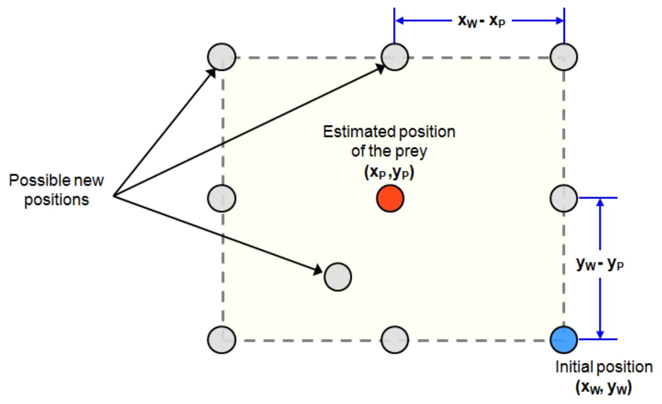

Figure 2. Encircling behavior of a pack.

Equations (1) and (2) indicate that the positions of the wolves are updated based on the position of the prey. However, the exact position of the prey (i.e. the optimal solution of the problem under consideration) is unknown. As mentioned earlier, the hunting procedure is guided by the α, β and δ wolves. In order to model this characteristic, GWO assumes that the α, β and δ wolves, which are the candidate solutions with the best fitness values, have more accurate knowledge on the location of the prey. Thus, the positions of the wolves are updated according to the position of the α, β and δ wolves, as shown in Eqs. (5) to (11). Figure 3 illustrates the procedure to update the position of the wolves. A pseudo-code to implement the GWO algorithm is presented in Algorithm 1.

$$
\begin{aligned}
D_\alpha &= |C_1 \cdot X_\alpha - X| && (5)\\
D_\beta &= |C_2 \cdot X_\beta - X| && (6)\\
D_\delta &= |C_3 \cdot X_\delta - X| && (7)\\
X_1 &= X_\alpha - A_1 \cdot D_\alpha && (8)\\
X_2 &= X_\beta - A_2 \cdot D_\beta && (9)\\
X_3 &= X_\delta - A_3 \cdot D_\delta && (10)\\
X(t+1) &= \frac{X_1 + X_2 + X_3}{3} && (11)
\end{aligned}
$$

III. Chaotic GWO

As mentioned earlier, in the original GWO algorithm the hunting process is always guided by the three best wolves in the pack. However, when this strategy is adopted, if the three best wolves are close to each other in the first iterations, the exploration capability of the algorithm is reduced and its capability of escaping from local optima is affected.

In the proposed algorithm, for each iteration t, the number of wolves that are considered as leaders, denoted by $n(t)$, is computed according to Eq. (12).

$$n(t) = \text{round}\left[\frac{M \cdot g(t)}{2}\right] \qquad (12)$$

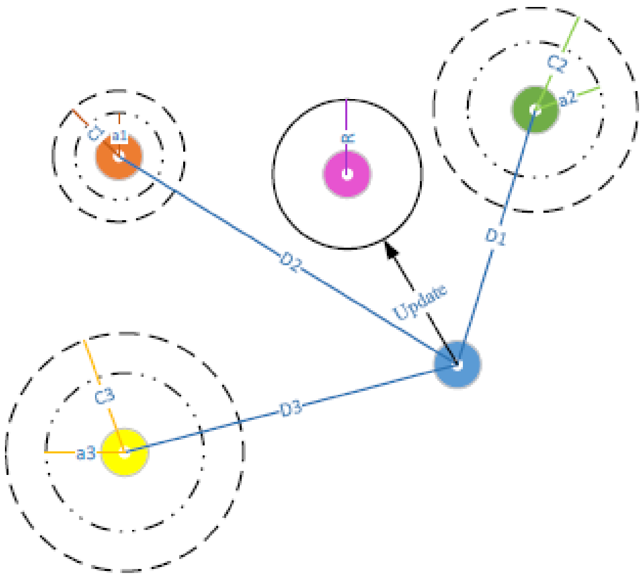

Figure 3. Position update in the GWO algorithm [24].

Algorithm 1: A pseudo code to implement the GWO algorithm.

```
 1: Initialize the population $X_i$, for $i = 1, \ldots, M$
 2: Initialize $a = 2$
 3: Initialize vectors $A$ and $C$
 4: Compute the fitness value of each wolf
 5: $X_{alpha}$ = the wolf with the best fitness value
 6: $X_{beta}$ = the wolf with the second best fitness value
 7: $X_{delta}$ = the wolf with the third best fitness value
 8: **While** $t <$ max. number of iterations **do**
 9:     **For** each wolf $i$ **do**
10:         Update its position according to Eq. (11)
11:     **End For**
12:     Update $a$, $A$, and $C$
13:     Compute the new fitness value of each wolf
14:     Update $X_\alpha$, $X_\beta$, and $X_\delta$
15:     increment the iteration number ($t = t + 1$)
16: **End While**
17: Return $X_\alpha$ =0
```

where M is the number of wolves in the pack, and $g(t)$ is a chaotic variable in the range [0,1].

Chaotic mathematical sequences are ruled by iterated functions that return an output value in each iteration. The sequence of values generated by a chaotic function is called an orbit. An orbit is represented by a chaotic map that maps an input value in output value. Chaotic maps have the following characteristics [16]:

- The rule of generating the sequence of numbers is deterministic;
- The orbits are aperiodic;
- The orbits are bounded (the chaotic variables assume a value between an upper and a lower limit); and
- The sequence has a sensitive dependence on the initial condition.

In the proposed model, the position of each wolf is not updated according to Eq. (11). Instead, it is updated according to Eq. (13).

$$X(t+1) = \frac{\sum_{v=1}^{n(t)} X_v}{n(t)} \tag{13}$$

where

$$X_v = X_j(t) - A_j \cdot D_j \tag{14}$$

where $X_j(t)$ is the position of the wolf with the j-th best fitness value in iteration t, A_j is a random vector, and D_j is computed according to Eq. (15).

$$D_j = |C_j \cdot X_j(t) - X| \tag{15}$$

where C_j is a random vector, and X is the current position of the wolf.

IV. Maintenance Scope Optimization

Consider a series-parallel system composed of a set of N_P components. We assume that the system has N_S subsystems connected in series. Each subsystem i has n_i identical components connected in parallel. A system level failure occurs when all the components belonging to a subsystem fail simultaneously.

Let $S = [s_1, \ldots, s_{N_P}]$ be a maintenance scope vector. Each element s_i of the maintenance scope vector, with $i = \{1, \ldots, N_P\}$, is a binary variable that assumes value 1 if component i is maintained and zero otherwise.

A fixed maintenance cost C_F is incurred whenever at least one component is maintained. A preventive maintenance cost $C_P^{(i)}$ is incurred whenever a preventive maintenance task is performed in component i. A corrective maintenance cost $C_C^{(i)}$ is incurred whenever a corrective maintenance task is performed in component i.

The following assumptions are also made:

- The system operates in cycles with a fixed interval. Maintenance activities are carried out only at the end of each cycle.
- Maintenance activities always bring the component to the "as good as new" condition.
- A component fails whenever its degradation level reaches a known failure threshold level FT.
- A preventive maintenance intervention is required if the probability of the system finishing the next cycle without failing is lower than a safety level SL.

The problem consists in finding the maintenance scope S that minimizes the expected total cost per cycle, $TC(S)$, until the next maintenance activity is performed. The steps to compute $TC(S)$ are presented in section IV-B.

A. Degradation Process

Let $z_i(t)$ be the expected degradation level of component i at the end of cycle t. The increment in the degradation level z_i of each component i in each cycle is a random variable that follows a gamma distribution. The PDF (probability density function) of the gamma distribution is presented in Eq. (16).

$$f(x|w,\theta) = \frac{x^{w-1} \cdot exp^{\frac{-x}{\theta}}}{\Gamma(w) \cdot \theta^{w}}; x, w, \theta > 0 \tag{16}$$

where w is the shape parameter and θ is the scale parameter of the gamma distribution. $\Gamma(w)$ is the Gamma function evaluated at w, as presented in Eq. (17).

$$\Gamma(w) = \int_{x=0}^{\infty} x^{w-1} \cdot exp^{-x} dx \tag{17}$$

B. Maintenance Cost Model

As mentioned earlier, the optimization problem consists in finding the maintenance scope S that minimizes the expected total cost per cycle until the next maintenance intervention. It is assumed that the next maintenance intervention is performed when the system-level failure probability reaches the value $1 - SL$. The expected total cost per cycle, $TC(S)$, is computed according to Eq. (18).

$$TC(S) = \frac{M(S)}{L(S)} \tag{18}$$

where $L(S)$ is the expected number of cycles the system will operate until it reaches the safety level SL if a maintenance activity with scope S is carried out, and $M(S)$ is the total maintenance cost associated with the maintenance scope S.

1) Computation of $M(S)$: The total maintenance cost associated with a maintenance activity with scope S is computed according to Eq. (19).

$$M(S) = C_F + \sum_{i=1}^{N_P} s_i \cdot \left[\delta_i \cdot C_C^{(i)} + (1-\delta_i) \cdot C_P^{(i)}\right] \tag{19}$$

where δ_i is a binary variable that assumes value 1 if a corrective maintenance intervention is performed in component i and zero otherwise.

2) Computation of $L(S)$: In order to compute the expected number of cycles the system will run until it reaches the safety level if a maintenance activity with scope S is carried out, the first step is to update the degradation levels of maintained components. It is assumed that maintenance activities always bring the component to the "as good as new" condition. So, the degradation levels of maintained components are set to zero. The degradation levels of the remaining components are not affected.

Then, the RUL distribution for each component is computed. Let $z_i(0)$ be the current degradation level of component i. Let $p_i(j)$ be the probability that component i will fail at the end of the j-th cycle from now. The RUL distribution of component i is given by Eq. (20) [25].

$$p_i(j+1) = p\{F(x|w,\theta) \geq FT - z_i(j)|z_i(j) < FT\} \quad (20)$$

where $F(x|w,\theta)$ is the CDF (cumulative distribution function) of the gamma distribution, $p_i(j)$ is the probability that a failure in component i will be detected at the end of the j-th cycle from now, $z_i(j)$ is the expected degradation level of component i at the end of the j-th cycle from now, and FT is the failure threshold.

If component i fails, then $p_i(0) = 1$ and $p_i(j) = 0 \; \forall j > 0$. If component i is functioning, then $p_i(0) = 0$ and $p_i(j) \; \forall j > 0$ can be recursively computed using Eq. (21).

$$p_i(j+1) = \left[1 - \sum_{v=0}^{j} p_i(v)\right] \cdot \Omega \quad (21)$$

where

$$\Omega = p\left[F(x|j \cdot w, \theta) \geq FT - z_i(j)\right] \quad (22)$$

The next step is to compute the RUL distribution for each subsystem. Let $p_k(j)$ be the probability that subsystem k will fail at the end of the j-th cycle from now. Subsystem fails when all components belonging to it fail simultaneously. The failure probability of subsystem k can be obtained using Eq. (23).

$$p_k(j) = \sum_{v \in V} \left[\prod_{i \in v} p_i(j)\right] \quad (23)$$

Let $p_s(j)$ be the probability that a system-level failure will occur at the end of the j-th cycle from now. Once the failure probability of each subsystem is known, the RUL distribution of the whole system can be obtained using Eq. (24).

$$p_s(j) = 1 - \prod_{k=1}^{N_S} \left[1 - p_k(j)\right] \quad (24)$$

Finally, the expected number of cycles that the system will operate until it reaches the safety level SL if a maintenance activity with scope S is carried out can be obtained according to Eq. (25).

$$L(S) = \min j | \left[p_s(j) > 1 - SL\right] - 1 \quad (25)$$

V. Numerical Experiment

This section presents a numerical example to illustrate the application of the proposed chaotic version of GWO to the maintenance scope definition problem. In this example, we consider a series-parallel system with 20 components. We assume that components in the same subsystem are identical, and components in different subsystems have different characteristics. The increment in the degradation level z_i of each component i during each cycle is a random variable that follows a gamma distribution. Table I shows the parameters of the gamma distributions and the preventive maintenance cost C_P of each component. Figure 4 shows the block diagram for the system under consideration.

Table I
System Data.

Subsystem	1	2	3	4	5
n_i	6	2	5	3	4
w	3.5	3.0	2.5	2.5	3.0
θ	2.5	2.0	2.0	1.5	1.5
C_P	30	40	50	55	60

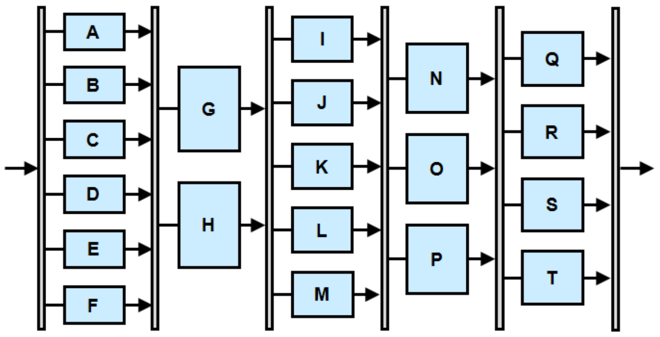

Figure 4. System block diagram.

A fixed maintenance cost C_F of 20 is used. The corrective maintenance cost C_C is assumed to be 2.5 times the corresponding preventive maintenance cost C_P of each component. The failure threshold level FT and the safety level SL are 100 and 0.95, respectively.

Table II shows the current degradation level of each component. It can be seen that three components (B, D, and J) have failed. However, the system as a whole is still functioning since all subsystems have at least one functional component. In this situation, maintenance interventions are mandatory for the failed components, while preventive maintenance interventions can be performed in the remaining components to reduce the expected maintenance cost per cycle.

Table II
Current degradation level.

Component	Degradation	Component	Degradation
A	9	K	33
B	100	L	51
C	23	M	38
D	100	N	43
E	41	O	58
F	56	P	62
G	25	Q	41
H	55	R	44
I	25	S	23
J	100	T	42

A. Chaotic Maps

Different chaotic maps are reported in the literature. Since different maps may lead to different results, a set of chaotic maps must be investigated to find the best one for the problem under consideration. In this paper, nine different chaotic maps are considered. These maps are presented in Table III.

Table III
CHAOTIC MAPS

Number	Name	Equation
1	Logistic map	$z(t+1) = 4z(t) \cdot (1 - z(t))$
2	PWLCM	$z(t+1) = \begin{cases} z(t)/0.7 & ,0 < z(t) \leq 0.7 \\ (1 - z(t)) \cdot (1 - 0.7) & ,0.7 < z(t) \leq 1 \end{cases}$
3	Sine map	$z(t+1) = \sin(\pi z(t))$
4	Tent map	$z(t+1) = \begin{cases} z(t)/0.4 & ,0 < z(t) \leq 0.4 \\ (1 - z(t))/0.6 & ,0.4 < z(t) \leq 1 \end{cases}$
5	Bernoulli map	$z(t+1) = \begin{cases} z(t)/0.6 & ,0 < z(t) \leq 0.6 \\ (z(t) - 0.6)/0.4 & ,0.6 < z(t) < 1 \end{cases}$
6	Chebyshev map	$z(t+1) = \cos(0.5\cos^{-1} z(t))$
7	ICMIC	$z(t+1) = \sin(70/z(t))$
8	Cubic map	$z(t+1) = 2.59z(t) \cdot (1 - z^2(t))$
9	Singer map	$z(t+1) = 1.073[7.86z(t) - 23.31z^2(t) + 28.75z^3(t) - 13302875z^4(t)]$

B. Simulation Results

The results obtained with the proposed CGWO using the different chaotic maps are compared with the performance obtained with the original GWO algorithm, which is used as a reference baseline. The performance obtained with the Ant Colony Optimization (ACO) is also considered. The ACO algorithm is a well-known metaheuristic that has been used in combinatorial optimization problems with a competitive performance [26], [13]. In this paper, the performance comparison is made in terms of the quality of solutions provided by each algorithm.

For each algorithm, a Monte Carlo simulation with 30 runs was carried out. Based on experimental observations, a population size of 50 wolves and a maximum number of 200 iterations were considered for GWO and CGWO. For the ACO algorithm, a colony with 50 ants and a pheromone evaporation rate of 5% were considered. A maximum number of 200 iterations were used for the ACO algorithm so that the number of objective function evaluations is the same for all the methods. The average result, the variance, the best result, and the average simulation time observed for each algorithm are presented in Table IV.

Table IV
SIMULATION RESULTS.

Algorithm	Average	Variance	Best	Simulation Time (s)
ACO	516.3	2.24	512.8	3.96
GWO	515.5	2.16	508.8	3.61
CGWO (Logistic)	516.7	2.17	510.0	3.70
CGWO (PWLCM)	514.1	2.00	**507.4**	3.63
CGWO (Sine)	514.3	4.92	**507.4**	3.65
CGWO (Tent)	515.7	2.45	508.8	3.54
CGWO (Bernoulli)	511.2	2.37	**507.4**	3.62
CGWO (Chebyshev)	**510.4**	3.08	**507.4**	3.57
CGWO (ICMIC)	516.0	2.38	**507.4**	**3.51**
CGWO (Cubic)	515.1	3.08	508.8	3.57
CGWO (Singer)	515.8	2.46	**507.4**	3.58

The original GWO and most of the proposed variants outperformed the ACO in terms of average response. The only exception was CGWO with the Logistic chaotic map. This result validates the use of GWO as a good alternative for the problem under consideration. The best result in terms of average response was obtained with the Chebyshev chaotic map in CGWO.

The average response obtained with the original GWO was better than 4 out of the 9 CGWO variants. However, in terms of best result, most of the variants outperformed the original GWO or presented the same performance. Again, the only exception was CGWO with the Logistic chaotic map. The ACO presented the worst performance in terms of best result.

In terms of average simulation time, the GWO and all the CGWO variants outperformed the ACO algorithm. Five variants of CGWO presented a smaller average simulation time in comparison with the original GWO. However, the differences in simulation time observed in the different CGWO versions were small.

C. Statistical Test

This section presents the results of a statistical test conduct to check whether the performance presented by each chaotic variant of GWO is statistically different from the performance obtained with the original version of GWO.

Let X and Y be two normally distributed random variables. The mean and the variance of X are μ_X and σ_X^2, respectively. Similarly, the mean and the variance of Y are μ_Y and σ_Y^2, respectively. Also, assume that $\sigma_X^2 \geq \sigma_Y^2$.

In order to test whether X and Y have different means, the first step is to conduct an F-Test to test whether X and Y can be assumed to have the same variance. The null hypothesis H_0 for the F-Test is that $\sigma_X = \sigma_Y$. The statistics F_{calc} is computed according to Eq. (26).

$$F_{calc} = \frac{\sigma_X^2}{\sigma_Y^2} \tag{26}$$

The statistics F_{calc} is compared with a critical value F_{crit}. The critical value F_{crit} is computed according to Eq. (27). The null hypothesis H_0 can be rejected if $F_{calc} > F_{crit}$.

$$F(F_{crit}; d_X, d_Y) = 1 - \alpha \tag{27}$$

where $F(\cdot)$ is the cumulative Fisher-Snedecor distribution, α is the confidence level for the test, d_X is the number of degrees of freedom associated to X, and d_Y is the number of degrees of freedom associated to Y.

A T-Test is then conducted to test whether X and Y can be assumed to have different means. The result of the F-Test is used to define how to compute the statistics for the T-Test. If the null hypothesis of the F-Test is rejected (i.e. X and Y have different variances), then the statistics T_{calc} for the T-Test is computed according to Eq. (28). Otherwise, T_{calc} is computed according to Eq. (29).

$$T_{calc} = \frac{\mu_X - \mu_Y}{\sqrt{\frac{\sigma_X^2}{n_X} + \frac{\sigma_Y^2}{n_Y}}} \tag{28}$$

$$T_{calc} = \frac{\mu_X - \mu_Y}{\sqrt{\sigma^2 \left(\frac{1}{n_X} + \frac{1}{n_Y}\right)}} \tag{29}$$

where n_X and n_Y are the number of samples used to compute the means and variances of X and Y, respectively, and σ^2 is computed according to Eq. (30).

$$\sigma^2 = \frac{(n_X - 1)\sigma_X^2 + (n_Y - 1)\sigma_Y^2}{n_X + n_Y - 2} \tag{30}$$

Again, the statistics T_{calc} is compared with a critical value T_{crit}. The critical value T_{crit} is computed according to Eq. (31). The null hypothesis H_0 can be rejected if $T_{calc} > T_{crit}$.

$$T(T_{crit}; n) = 1 - alpha/2 \tag{31}$$

where $T(\cdot)$ is the cumulative Student's t distribution, and n is the number of degrees of freedom computed using Eq. (32).

$$n = \frac{\left[\left(\frac{\sigma_X^2}{n_X} + \frac{\sigma_Y^2}{n_Y}\right)\right]^2}{\frac{\left(\frac{\sigma_X^2}{n_X}\right)^2}{n_X - 1} + \frac{\left(\frac{\sigma_Y^2}{n_Y}\right)^2}{n_Y - 1}} \tag{32}$$

Table V shows the results obtained from the T-Test for each CGWO algorithm. A confidence level α of 5% was used in every test. The average result obtained with the original GWO and with each chaotic version are presented again for clarity purposes.

Table V
SIMULATION RESULTS.

Algorithm	Average	T-Test Result
GWO	515.5	N/A
CGWO (Logistic)	516.7	H_0 rejected
CGWO (PWLCM)	514.1	H_0 rejected
CGWO (Sine)	514.3	H_0 rejected
CGWO (Tent)	515.7	H_0 not rejected
CGWO (Bernoulli)	511.2	H_0 rejected
CGWO (Chebyshev)	510.4	H_0 rejected
CGWO (ICMIC)	516.0	H_0 not rejected
CGWO (Cubic)	515.1	H_0 not rejected
CGWO (Singer)	515.8	H_0 not rejected

The null hypothesis of the T-Test could not be rejected in four tests (Tent map, ICMIC map, Cubic map, and Singer map). It means that these maps had a performance close to the original GWO algorithm. The remaining five CGWO versions provided a significant difference in comparison with GWO. Four of them (PWLCM map, Sine map, Bernoulli map, and Chebyshev map) presented a significantly better performance. The Logistic map was the only chaotic map that presented a significantly worse performance.

VI. CONCLUSION

This paper presented a modified version of the Grey Wolf Optimizer (GWO) algorithm, denoted by CGWO. The proposed method adds a chaotic variable to the mathematical formulation of GWO to vary the number of wolves that guides the hunting process during the execution of the algorithm. This strategy differs from most models proposed in the literature. With a dynamic number of leaders in each iteration of the algorithm, wolves with poor solutions may influence other wolves.

A numerical example of the application of the proposed CGWO to solve the maintenance scope definition problem is presented. Nine different chaotic maps were considered. The performance of CGWO was compared with the performance of the original GWO and the Ant Colony Optimization algorithm (ACO) in terms of the average solution. Four chaotic versions provided a significantly better performance in comparison with the original GWO. The Logistic map was the only chaotic version that was outperformed by the original GWO with a significant difference.

The algorithm proposed in this paper differs from previously modified versions of GWO found in the literature because it does not add a chaotic behavior in the control parameters. Instead, it changes the original GWO algorithm by changing the number of leaders in the pack during the execution of the algorithm. The proposed method can be combined with other modified versions already proposed in a straightforward way.

Future research may extend the scope of this paper by investigating the performance of the CGWO algorithm in multi-objective optimization problems. Another way to extend the scope of this paper is to investigate the performance of CGWO with other chaotic maps. The performance of the proposed CGWO combined with other GWO variants proposed in the literature can also be investigated.

ACKNOWLEDGMENTS

The authors would like to thank the Brazilian National Council for Scientific and Technological Development - CNPq (grants 423023/2018-7 and 305048/2016-3).

REFERENCES

[1] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks*, 1995, pp. 1942–1948.

[2] J. H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. Cambridge, MA, USA: MIT Press, 1992.

[3] M. Dorigo, "Optimization, Learning and Natural Algorithms," Ph.D. dissertation, Politecnico di Milano, Italy, 1992.

[4] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 67–82, 1997.

[5] M.-Y. Cheng and D. Prayogo, “Symbiotic organisms search: A new metaheuristic optimization algorithm,” *Computers and Structures*, vol. 139, pp. 98–112, 2014.

[6] T. Tometzki and S. Engell, “Systematic initialization techniques for hybrid evolutionary algorithms for solving two-stage stochastic mixed-integer programs,” *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 2, pp. 196–214, 2011.

[7] Q. Lin, J. Chen, Z. Zhan, W. Chen, C. A. C. Coello, Y. Yin, C. Lin, and J. Zhang, “A hybrid evolutionary immune algorithm for multiobjective optimization problems,” *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 5, pp. 711–729, 2016.

[8] S. Mirjalili, S. M. Mirjalili, and A. Lewis, “Grey wolf optimizer,” *Advances in Engineering Software*, vol. 69, pp. 46–61, 2014.

[9] H. A. Hassan and M. Zellagui, “Application of grey wolf optimizer algorithm for optimal power flow of two-terminal HVDC transmission system,” *Advances in Electrical and Electronic Engineering*, vol. 15, no. 5, pp. 701–712, 2017.

[10] E. Emary, H. M. Zawbaa, and A. E. Hassanien, “Binary grey wolf optimization approaches for feature selection,” *Neurocomputing*, vol. 172, pp. 371–381, 2016.

[11] D. Sánchez, P. Melin, and O. Castillo, “A grey wolf optimizer for modular granular neural networks for human recognition,” *Computational Intelligence and Neuroscience*, vol. 2017, pp. 1–26, 2017.

[12] N. Singh and S. Singh, “A modified mean gray wolf optimization approach for benchmark and biomedical problems,” *Evolutionary Bioinformatics*, vol. 13, pp. 1–28, 2017.

[13] Z. Zhang and K. Zou, “Simple ant colony algorithm for combinatorial optimization problems,” in *2017 36th Chinese Control Conference (CCC)*, July 2017, pp. 9835–9840.

[14] Z.-j. Teng, J.-l. Lv, and L.-w. Guo, “An improved hybrid grey wolf optimization algorithm,” *Soft Computing*, vol. 23, no. 15, pp. 6617–6631, 2019.

[15] H. Xu, X. Liu, and J. Su, “An improved grey wolf optimizer algorithm integrated with cuckoo search,” in *2017 9th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS)*, vol. 1, Sep. 2017, pp. 490–493.

[16] E. Emary and H. M. Zawbaa, “Impact of chaos functions on modern swarm optimizers,” *Plos One*, vol. 11, no. 7, pp. 1–26, 2016.

[17] M. Z. M. Khairuzzaman, I. Musirin, S. S. Izwan, and T. Bouktir, “Chaos embedded symbiotic organisms search technique for optimal FACTS device allocation for voltage profile and security improvement,” *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 8, no. 1, pp. 146–153, 10 2017.

[18] I. Brajević and P. Stanimirović, “An improved chaotic firefly algorithm for global numerical optimization,” *International Journal of Computational Intelligence Systems*, vol. 12, pp. 131–148, 2018.

[19] G. Kaur and S. Arora, “Chaotic whale optimization algorithm,” *Journal of Computational Design and Engineering*, vol. 5, no. 3, pp. 275–284, 2018.

[20] A. Farah, T. Guesmi, H. H. Abdallah, and A. Ouali, “A novel chaotic teaching-learning-based optimization algorithm for multi-machine power system stabilizers design problem,” *International Journal of Electrical Power & Energy Systems*, vol. 77, pp. 197–209, 2016.

[21] H. Yu, Y. Yu, Y. Liu, Y. Wang, and S. Gao, “Chaotic grey wolf optimization,” in *2016 International Conference on Progress in Informatics and Computing (PIC)*, Dec 2016, pp. 103–113.

[22] H. Mehrotra and S. K. Pal, “Using chaos in grey wolf optimizer and application to prime factorization,” in *International Conference on Soft Computing for Problem Solving (SocProS)*, 2018, pp. 25–43.

[23] C. Muro, R. Escobedo, L. Spector, and R. P. Coppinger, “Wolf-pack (canis lupus) hunting strategies emerge from simple rules in computational simulations,” *Behavioural Processes*, vol. 88, no. 3, pp. 192–197, 2011.

[24] S. Dai, D. Niu, and Y. Li, “Daily peak load forecasting based on complete ensemble empirical mode decomposition with adaptive noise and support vector machine optimized by modified grey wolf optimization algorithm,” *Energies*, vol. 11, no. 1, pp. 1–25, 2018.

[25] L. R. Rodrigues, J. P. P. Gomes, F. A. S. Ferri, I. P. Medeiros, R. K. H. Galvão, and C. L. Nascimento Júnior, “Use of PHM information and system architecture for optimized aircraft maintenance planning,” *IEEE Systems Journal*, vol. 9, no. 4, pp. 1197–1207, 2015.

[26] J. Yang and Y. Zhuang, “An improved ant colony optimization algorithm for solving a complex combinatorial optimization problem,” *Applied Soft Computing*, vol. 10, no. 2, pp. 653–660, 2010.