

## THE CHAOS-BASED APPROACHES FOR ACTUAL METAHEURISTIC ALGORITHMS

*Yiğit Çağatay KUYU\**  
*Fahri VATANSEVER\*\**

Received: 02.05.2018; revised:15.10.2018; accepted: 17.10.2018

**Abstract:** Along with rapid developments in computational technologies, evolutionary/heuristic/metaheuristic algorithms have frequently used in many applications to solve optimization problems. Nowadays, new algorithms are being developed and improvements are being made to existing algorithms. In this study, chaos-based modifications have been proposed for recently developed metaheuristic algorithms: Backtracking Search (BS), Grey Wolf Optimizer (GWO) and Vortex Search (VS), and the algorithms have been analyzed by detailed comparisons. The proposed approaches are based on generating new values through chaos maps, rather than some random numbers normally used in the algorithms, to improve their solutions. In addition, some modifications are performed to the structural operations of the algorithms used in the optimization process by taking advantage of chaos-based values. The performances of the algorithms are evaluated by considering two metrics: convergence rates and statistical results. Experiments demonstrated that the performance of the algorithms with the proposed modifications based on the chaos approach, are better than, or at least comparable to, the original algorithms.

**Keywords:** Metaheuristic algorithms, Chaotic maps, Chaotic sequences.

### Güncel Metasezgisel Algoritmalar İçin Kaos Tabanlı Yaklaşımlar

**Öz:** Hesaplama teknolojilerindeki hızlı gelişmelerle orantılı olarak, optimizasyon problemlerinin çözümünde evrimsel/sezgisel/metasezgisel algoritmalarından birçok alandaki uygulamalarda sıklıkla faydalanılmaktadır. Günümüzde, yeni algoritmalar geliştirilmekte ve mevcut algoritmalara yenilikler uygulanmaya devam edilmektedir. Bu çalışmada, son zamanlarda geliştirilmiş olan metasezgisel algoritmalarından olan: Geri İzleme Arama (BS), Gri Kurt Optimizasyon (GWO) ve Girdap Arama (VS) algoritmalarına kaos tabanlı modifikasyonlar önerilmiş ve algoritmaların, kıyaslamalarla detaylı analizleri gerçekleştirilmiştir. Önerilen yaklaşımlar, algoritmaların çözümlerini geliştirmek için işlemlerinde kullandıkları bazı rassal değişkenler yerine, kaos haritalarına dayanan yeni değişkenlerin üretilmesi temeline dayanmaktadır. Bunun yanında, kaos tabanlı bu değişkenler kullanılarak algoritmaların optimizasyon sürecinde kullandıkları yapısal işlemlerinde modifikasyonlar gerçekleştirilmektedir. Algoritmaların performansları; istatistiksel ve yakınsama hızları açısından, iki yönlü olarak analiz edilmektedir. Kaotik haritalara dayanan yaklaşımların, orijinal algoritmalar üzerinde daha iyi veya en azından karşılaştırılabilir sonuçlar ürettiği, gerçekleştirilen deneylerde gösterilmiştir.

**Anahtar Kelimeler:** Metasezgisel algoritmalar, Kaos haritaları, Kaotik diziler.

## 1. INTRODUCTION

The optimization process is based on the principle of defining the vectors representing the possible solutions of the problem within the search space, maximizing or minimizing the objective function, and achieving the most appropriate solutions. Nowadays, existing metaheuristic algorithms generally focus on such optimization problems. In general, these

\* Uludağ University, Faculty of Engineering, Electrical-Electronics Eng. Dept., 16059 Bursa/Turkey

\*\* Uludağ University, Faculty of Engineering, Electrical-Electronics Eng. Dept., 16059 Bursa/Turkey

Corresponding author: Fahri Vatansever (fahriv@uludag.edu.tr)

optimization approaches can be categorized according to whether the processes they are based upon are random or deterministic. Gradient information is also an important notion in deterministic approaches, which can achieve similar solutions to the same problem if they have the same initial starting points. Alternatively, metaheuristic algorithms using random approaches can produce different solutions for the same problem without any repetitions, due to their random-like behavior.

In general, metaheuristic algorithms are based on the initial population of individuals representing possible solutions to the problem. Afterwards, the algorithms improve the solutions within their populations using their own special operations in order to reach the best solutions in the search space. One of the main factors that affects the performance of algorithms is how to obtain a good trade-off between exploration and exploitation capabilities. Generally speaking, exploration reveals the characteristics of the whole search space, while exploitation is the ability to improve the solutions defined within the search space. If exploration is too robust, the algorithm can visit undesirable regions of the search space, whereas if exploitation is much stronger than exploration, it causes premature convergence problems (Zaharie, 2003). Therefore, maintaining a good balance between these two factors is vital for the performance of metaheuristic algorithms.

The main aim of metaheuristic algorithms is to provide a proper balance between exploration and exploitation, which directly affects the performance of the algorithms. These algorithms, which are often inspired by natural evolution, can produce solutions without using gradient information (Yang, 2010, Goldberg, 1989). Over the last few decades, a lot of metaheuristics based on a random approach have been designed, such as particle swarm optimization (PSO) (Kennedy and Eberhart, 1995) differential evolution (DE) (Storn, 1995), cuckoo search (CS) (Yang and Deb, 2009), harmony search (HS) (Geem et al., 2001), and so on. In addition, according to the “No Free Lunch Theorem (Wolpert and Macready, 1997)”, it cannot be said that a single algorithm can perform well on all problems. This theory leaves the door open for algorithm developers, challenging them to develop more powerful and efficient algorithms (Civicioglu, 2013, Mirjalili et. al., 2014, Dogan and Olmez, 2015).

Random numbers are frequently used by the algorithms. On the other hand, using these random numbers may give good results on one type of problem while they are not applicable to another. Besides, the solutions achieved by metaheuristics cannot always give the desired solution due to the nature of random search strategies. Chaos approach can be helpful to ensure the performance stability of the algorithms and it has been successfully applied to metaheuristics in balancing exploration and exploration in published literature (Wang et. al., 2014, Zhenyu et. al., 2006, Wang et. al., 2014, Gandomi et. al., 2013, Li et. al., 2011). In this study, chaotic values are substituted for some of the random numbers generated by the actual metaheuristic algorithms: Backtracking Search (BS) (Civicioglu, 2013), Grey Wolf Optimizer (GWO) (Mirjalili et. al., 2014) and Vortex Search (VS) (Dogan and Olmez, 2015), and the modifications are proposed by utilizing six different one-dimensional chaotic maps. The experimental results show that using chaotic sequences generated by the maps can be a better alternative to random numbers for improving the performance of the algorithms.

The rest of this paper is organized as follows: Metaheuristic algorithms and the chaotic maps are briefly described in Section 2. The proposed modifications are presented in Section 3. The experimental results and the performance comparisons are given in Section 4 and finally, the study conclusions are detailed in Section 5.

## **2. METAHEURISTIC ALGORITHMS AND CHAOTIC MAPS**

### **2.1. Metaheuristic Algorithms**

Nowadays, metaheuristics, such as evolutionary algorithms (EAs), inspired by ideas of natural evolution have attracted much interest and have been frequently used by engineers, researchers, etc. for solving optimization problems. These algorithms have shown effective

performances and have gradually been improved to solve various complex problems. In this section, the recently developed metaheuristic algorithms: BS, GWO and VS, are briefly described, and the flowcharts and pseudo-codes for these algorithms are given in Table 1.

The BS algorithm introduced by Civicioglu (2013), is an EA for solving optimization problems, which comprises five main processes: initialization, selection-I, mutation, crossover and selection-II. BS initializes the population by using a uniform distribution in the  $n$ -dimensional search space and then it applies the selection-I process to calculate the search direction of the current population. In the mutation process, BS generates the initial mutant population at the beginning of this process and afterwards the crossover process is used to achieve the final form of the trial population. In the selection-II process, BS chooses the best individual according to the fitness values by comparing the trial population with the current population. GWO was presented by Mirjalili et. al. (2014) and took inspiration from how gray wolves hunt by using the social hierarchy between them. This hierarchy can be defined according to dominance and is divided into four categories: alpha ( $\alpha$ ), beta ( $\beta$ ), delta ( $\delta$ ) and omega ( $\omega$ ). The first level of wolves is called the alpha wolf, which is mainly responsible for guiding the search, whereas the fourth level of wolves is called the omega wolf. The VS algorithm, developed by Dogan and Olmez (2015) is one of the most recent metaheuristic algorithms based on vortex patterns created by the vortical flow of stirred fluids. VS uses the vortex pattern methodology by modeling a number of nested circles and decreases the radius of the circle gradually to achieve its final solution.

## 2.2. Chaotic Maps

Chaos theory is a deterministic approach and this theory can be used in a wide variety of applications in engineering, such as non-linear systems (Schuster, 2006, Pecora and Carroll, 1990). Although this approach seems like a random behavior, there is not always a need to be random in order for the systems to exhibit chaotic behavior (Kellert, 1993). Recently, chaos-based approaches have been adopted in many studies to enhance the performance of algorithms. Alatas et al. used this approach with PSO algorithm by applying 12 different modifications based on chaotic maps. The results of this study showed that the performance of the algorithms can be increased by benefiting from these modifications (Alatas et. al., 2009). Wang and Zhong (2015) redefined the scaling factor and fraction probability by using chaotic maps for the CS algorithm. The performances of the algorithms were analyzed on 20 test functions and were compared statistically. In the published literature, there exist a wide variety of chaotic versions of metaheuristics, such as genetic, firefly and krill herd algorithms (Yao et. al., 2001, Gandomi et. al., 2013, Saremi et. al., 2014).

The visualization and formulation of the different chaotic maps used in this study are given in Table 2 (Li-Jiang, Y and Tian-Lun, 2002, Zaharie, 2003, Pecora and Carroll, 1990, Yao et. al., 2001, Saremi et. al., 2014). All maps are in the interval [0,1].

## 3. PROPOSED MODIFICATIONS

### 3.1. Proposed Modifications for BS

The historical population  $OldP$  is updated according to random numbers  $a$  and  $b$  in the BS algorithm, as shown in Equation 1 (Civicioglu, 2013):

$$\text{If } a < b \text{ then } OldP = P \quad (1)$$

In the proposed approach to the BS algorithm, the probability of updating is determined by comparing the value generated by a chaos map with a random number in the selection-I phase, and  $OldP$  is redefined according to this probability. In other words, chaotic values are substituted for random values to manipulate the selection probability. As a result of the experiments with six different chaos maps, the most successful results are found on map 6, called the Sine map, and this map is used in the experiments with the modified BS algorithm.

**Table 1. Flowcharts and pseudo-codes of the algorithms**

	Flowchart	Pseudo-code
Backtracking search (BS)		<ol style="list-style-type: none"> <li>i. Initialize the algorithm parameters</li> <li>ii. Generate the initial population</li> <li>iii. Calculate the fitness of the initial population</li> <li>iv. While termination criteria not met             <ol style="list-style-type: none"> <li>a. Apply the selection-I process</li> <li>b. Apply the mutation process</li> <li>c. Apply the crossover process</li> <li>d. Apply the selection-II process</li> </ol> </li> <li>End While</li> <li>v. Return the best individual found so far</li> </ol>
Grey wolf optimizer (GWO)		<ol style="list-style-type: none"> <li>i. Initialize the algorithm parameters</li> <li>ii. Generate the population of grey wolves</li> <li>iii. Calculate the fitness of the population</li> <li>iv. Determine alpha, beta and delta wolves</li> <li>v. While termination criteria not met             <ol style="list-style-type: none"> <li>a. Update the position of each wolf</li> <li>b. Calculate the fitness of the population</li> <li>c. Update alpha, beta and delta wolves</li> </ol> </li> <li>End While</li> <li>vi. Return the grey wolf with the smallest fitness value</li> </ol>
Vortex search (VS)		<ol style="list-style-type: none"> <li>i. Initialize the algorithm parameters</li> <li>ii. Calculate the initial center and radius</li> <li>iii. Generate the candidate solutions</li> <li>iv. Calculate the fitnesses of the candidate solutions</li> <li>v. While termination criteria not met             <ol style="list-style-type: none"> <li>a. Apply boundary control mechanism to the solutions</li> <li>b. Memorize the best solution</li> <li>c. Update the center and reduce the radius</li> <li>d. Calculate the fitnesses of the solutions</li> <li>e. Update the solutions</li> </ol> </li> <li>End While</li> <li>vi. Return the best solution found so far</li> </ol>

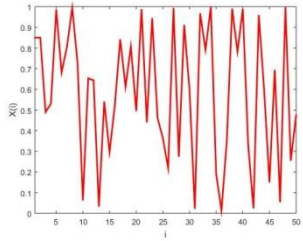
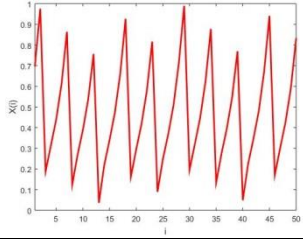
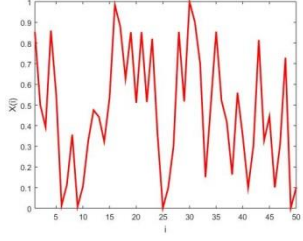
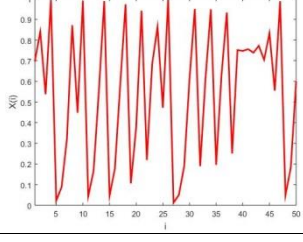
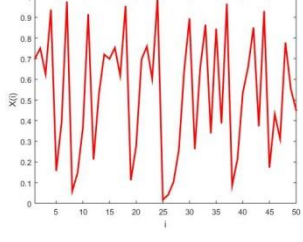
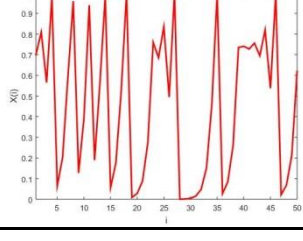
Through this chaos map, the chaotic sequence  $C_i$  is generated with the maximum number of iterations at the beginning of the optimization process, and the selection probability is chaotically changed, based on comparisons between the values at each iteration, from the first to the last. The formula for  $C_i$  is given in Equation 2:

$$C_i = Map_6(i) \tag{2}$$

where  $i$  is the iteration index and  $Map_6$  denotes the chaotic values of map 6. In the proposed approach, Equation 1 is redefined as in Equation 3:

$$If C_i < b \text{ then } OldP = P \tag{3}$$

**Table 2. The chaotic maps used**

	Expression	Map
1 → Chebyshev	$X_{i+1} = \text{Cos}(i \text{Cos}^{-1}(x_i))$	
2 → Circle	$X_{i+1} = \text{mod}\left(x_i + b - \frac{a}{\pi} \text{Sin}(2\pi x_i), 1\right), a = 0.5, b = 0.2$	
3 → Iterative	$X_{i+1} = \text{Sin}\left(\frac{a\pi}{X_i}\right), a = 0.7$	
4 → Logistic	$X_{i+1} = aX_i(1 - X_i), a = 4$	
5 → Piecewise	$f(x) = \begin{cases} \frac{X_i}{P} & 0 \leq X_i < P, \quad P = 0.4 \\ \frac{X_i - P}{0.5 - P} & P \leq X_i < 0.5 \\ \frac{0.5 - P}{1 - P - X_i} & 0.5 \leq X_i < 1 - P \\ \frac{0.5 - P}{1 - X_i} & 1 - P \leq X_i < 1 \\ \frac{1 - X_i}{P} & \end{cases}$	
6 → Sine	$X_{i+1} = \frac{a}{4} \text{Sin}(i), a = 4$	

The pseudo-code for the modifications applied is given in Figure 1.

---

**Algorithm 1**

---

```

Generate chaotic sequence with respect to the Sine map equation.
if  $C_i < b$  then
    | Redefine historical population  $OldP$  by using current population  $P$ .
end
    
```

---

**Figure 1:**  
Pseudo-code for the modifications of BS

### 3.2. Proposed Modifications for GWO

In the GWO, the positions of  $\alpha$ ,  $\beta$  and  $\delta$  wolves are used to determine the new positions of the wolves representing the possible solution set. The values of  $C_i$  used in Equation 4 are generated randomly when updating the positions of the wolves (Mirjalili et. al., 2014). Six different chaotic maps have been utilized to get rid of the uncertainty of these random values and to exploit the advantages of the chaotic maps for the modifications.

$$D_\alpha = |C_1 * X_\alpha - X|, D_\beta = |C_2 * X_\beta - X|, D_\delta = |C_3 * X_\delta - X| \quad (4)$$

Instead of using random values, maps 1 and 2 are used for  $C_1$ , maps 3 and 4 are used for  $C_2$  and maps 5 and 6 are used for  $C_3$  to generate the new chaos-based values. The value of  $C_i$  given in Equation 4 can be redefined as follows:

$$C_i = Map_k Map_l \quad (5)$$

where  $k$  and  $l$  are the indexes of the maps used for updating the positions of  $\alpha$ ,  $\beta$  and  $\delta$  wolves. New  $C_i$  values involve the multiplication of two different chaotic sequences. In Equation 4, the current wolf population  $X$  is utilized to recognize the location of prey, and the hunt is often guided by  $\alpha$  wolves. To increase convergence towards  $\alpha$  wolves, the distances of  $\beta$  and  $\delta$  to  $\alpha$  are used along with the chaotic values in Equation 6. Additionally, in order for  $\alpha$  wolves to go to possible different prey, the position of  $X$  is used to calculate the next position of  $\alpha$  wolves, which can put them in any location between prey and other wolves. Therefore, the hunting process of the wolves is completely changed by the chaos-based modifications made. Equation 6 shows a mathematical model of the processes mentioned above as follows:

$$D_\alpha = |C_1 * X_\alpha - X|, D_\beta = |C_2 * X_\beta - X_\alpha|, D_\delta = |C_3 * X_\delta - X_\alpha| \quad (6)$$

where  $C_1$ ,  $C_2$  and  $C_3$  indicate the multiplications of chaotic values generated by the chaotic maps. The pseudo-code of the modifications used is given in Figure. 2.

---

**Algorithm 2**

---

```

Generate chaotic sequences with respect to six chaos map equations.
while Termination criterion not met do
    | Calculate  $C_i$  values according to Equation 5.
    | Update the positions of  $\alpha$ ,  $\beta$  ve  $\delta$  wolves by using Equation 6.
end
    
```

---

**Figure 2:**  
Pseudo-code for the modifications of GWO

### 3.3. Proposed Modifications for VS

The boundary control mechanism of VS is given in Equation 7 (Dogan and Olmez, 2015):

$$S_k^i = \begin{cases} rand \cdot (upLim^i - lowLim^i) + lowLim^i, & \text{if } S_k^i < lowLim^i \text{ or } S_k^i > upLim^i \\ S_k^i, & \text{otherwise} \end{cases} \quad (7)$$

As can be seen in Equation 7, VS puts solutions into an acceptable range when solutions exceed the predefined boundary constraints during the evolving process using random values. Candidate solutions can be located far from the desired solutions due to the unbalanced structure of random values and this negatively affects the search process for an optimal solution. Here, utilizing the chaos approach, candidate solutions that exceed boundaries are produced through chaotic values in the search space, which can ensure stability between the boundaries. The new boundary control mechanism of VS is redefined via map 6 in the following equation:

$$S = \begin{cases} Map_6(i). (upLim - lowLim) + lowLim, & S < lowLim \\ S & lowLim \leq S \leq upLim \\ Map_6(i). (upLim - lowLim) + lowLim, & S > upLim \end{cases} \quad (8)$$

where  $S$  denotes the current candidate solution,  $Map_6(i)$  is the value generated by map 6 at  $i$ th iteration,  $upLim$  and  $lowLim$  are the upper and lower boundary constraints of the problem. The pseudo-code of the modifications for VS is given in Figure 3.

---

**Algorithm 3**

---

```

Generate chaotic sequence with respect to the Sine map equation.
if Candidate solutions exceed the boundaries of problem then
    | Move the solutions into the boundaries according to Equation 8.
end
    
```

---

**Figure 3:**  
Pseudo-code for the modifications of VS

#### 4. EXPERIMENTAL RESULTS

The parameter values for the algorithms used are given in Table 3. The benchmark functions used in this study are chosen from CEC2013 test functions (Liang et. al., 2013) and the properties of the functions are given in Table 4. The detailed description of the functions can be found in (Liang et. al., 2013). The 30 dimensional problems are used in this study for all the experiments, and the mean and standard deviation values of the solutions are calculated over 20 runs by using the function error value ( $F_{best} - F_{opt}$ ), where  $F_{best}$  is the best value found by the algorithms and  $F_{opt}$  is the global minimum of the related function. The results found by the algorithms are given in Tables 5 to 7 and the best results are marked in bold. To analyze the convergence performance of the algorithms, six functions are selected to show their convergence characteristics, and the convergence graphs of the related algorithms are given in Figures 4 to 6.

**Table 3. Control parameters of the algorithms**

BS	GWO	VS
Iteration cycles= 500 Population size= 60 mixrate= 0.9	Iteration cycles = 500 Population size= 60 $\alpha_0=0.9$	Iteration cycles = 500 Number of candidate sol.= 60 $\chi= 0.9$

**Table 4. Properties of the CEC2013 benchmark functions**

Test Func.	Function Names	Main Equations	Global Minimum	Range	Types
F6	Rotated Rosenbrock	$f(x) = \sum_{i=1}^{D-1} (100(z_i^2 - z_{i+1})^2) + (z_i - 1)^2$	-900	$[-100, 100]^D$	Multi-modal, Non-separable
F7	Rotated Schaffers	$f(x) = \left( \frac{1}{D-1} \sum_{i=1}^{D-1} (\sqrt{z_i} + \sqrt{z_i \sin^2(50z_i^{0.2})}) \right)^2$	-800	$[-100, 100]^D$	Multi-modal, Non-separable, Asymmetrical
F9	Rotated Weierstrass	$f(x) = \sum_{i=1}^D \left( \sum_{k=0}^{kmax} [a^k \cos(2\pi b^k (z_i + 0.5))] \right) - D \sum_{k=0}^{kmax} [a^k \cos(2\pi b^k \cdot 0.5)]$	-600	$[-100, 100]^D$	Multi-modal, Non-separable, Asymmetrical
F11	Rastrigin	$f(x) = \sum_{i=1}^D (z_i^2 - 10 \cos(2\pi z_i) + 10)$	-400	$[-100, 100]^D$	Multi-modal, Separable, Asymmetrical
F12	Rotated Rastrigin	$f(x) = \sum_{i=1}^D (z_i^2 - 10 \cos(2\pi z_i) + 10)$	-300	$[-100, 100]^D$	Multi-modal, Non-separable, Asymmetrical
F14	Schwefel	$f(z) = 418.9829 \times D - \sum_{i=1}^D g(z_i)$	-100	$[-100, 100]^D$	Multi-modal, Non-separable, Asymmetrical
F15	Rotated Schwefel	$f(z) = 418.9829 \times D - \sum_{i=1}^D g(z_i)$	100	$[-100, 100]^D$	Multi-modal, Non-separable, Asymmetrical
F16	Rotated Katsuura	$f(x) = \frac{10}{D^2} \prod_{i=1}^D \left( 1 + i \sum_{j=1}^{32} \frac{ 2^j z_i - \text{round}(2^j z_i) }{2^j} \right) - \frac{10}{D^2}$	200	$[-100, 100]^D$	Multi-modal, Non-separable, Asymmetrical
F17	Lunacek BiRastrigin	$f(x) = \min \left\{ \sum_{i=1}^D (\hat{x}_i - \mu_0)^2, dD + s \sum_{i=1}^D (\hat{x}_i - \mu_1)^2 \right\} + 10 \left( D - \sum_{i=1}^D \cos(2\pi \hat{z}_i) \right)$	300	$[-100, 100]^D$	Multi-modal, Non-separable, Asymmetrical
F23	Composition Function 3	$f(x) = \sum_{i=1}^n \{w_i * [\lambda_i g_i(x) + bias_i]\}, n = 3, bias = [0, 100, 200], \lambda = [1, 1, 1], g_{1-3} = F15$	900	$[-100, 100]^D$	Multi-modal, Non-separable, Asymmetrical
F25	Composition Function 5	$f(x) = \sum_{i=1}^n \{w_i * [\lambda_i g_i(x) + bias_i]\}, n = 3, bias = [0, 100, 200], \lambda = [0.25, 1, 2.5], g_1 = F15, g_2 = F12, g_3 = F9$	1100	$[-100, 100]^D$	Multi-modal, Non-separable, Asymmetrical
F26	Composition Function 6	$f(x) = \sum_{i=1}^n \{w_i * [\lambda_i g_i(x) + bias_i]\}, n = 5, bias = [0, 100, 200, 300, 400], \lambda = [0.25, 1, 1e-7, 2.5, 10], g_1 = F15, g_2 = F12, g_3 = F2, g_4 = F9, g_5 = F10$	1200	$[-100, 100]^D$	Multi-modal, Non-separable, Asymmetrical

**4.1. Performance Analysis of Modified BS and BS**

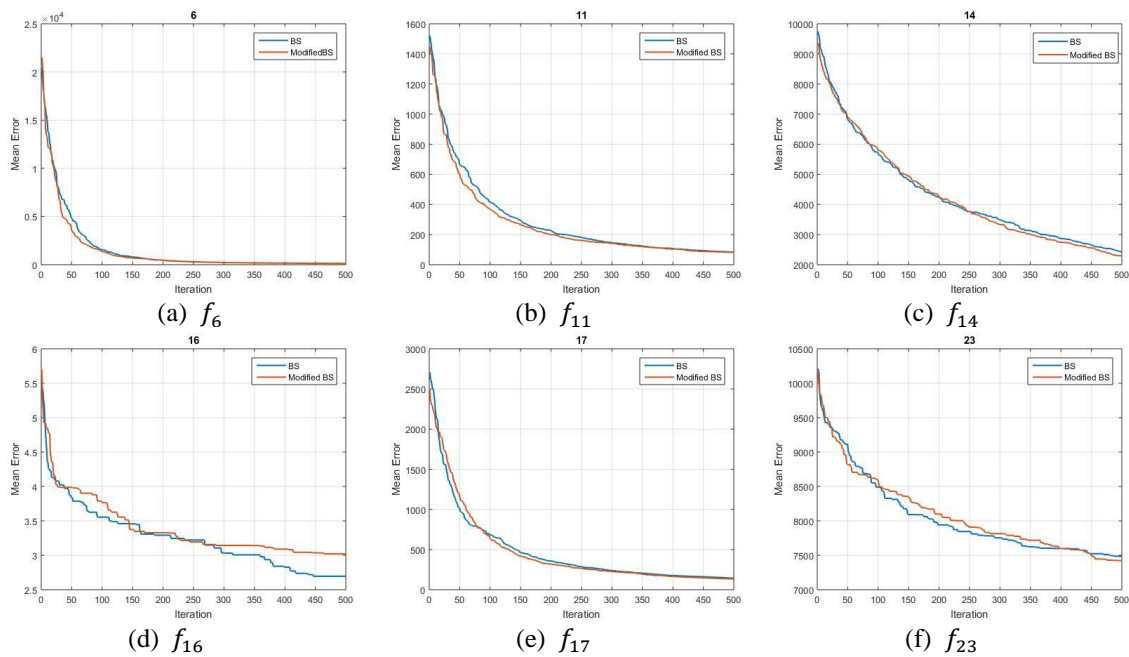
As can be seen in Table 5, according to the mean error and standard deviation results, the modifications applied to the original BS algorithm increase the performance of the algorithm. The modified BS algorithm produced an inferior performance only for functions F12 and F16 out of the 12 functions when compared to the original algorithm, achieving better results for the rest of the functions. This also demonstrates that the proposal is more capable of jumping out of local minima. The convergence graphs of the related algorithms are given in Figure 4. It can be observed from this figure that the proposed approach to the BS can show faster convergence in



the first half of iterations for functions F6, F11 and F17, but there is no significant difference observed between the convergence speeds of the algorithms for function F14. For function F16, the BS algorithm approaches a final solution faster than the proposal. Despite this, the performance of the modified algorithm is still comparable to the BS.

**Table 5. Experimental results for 20 runs of 12 test functions for Modified BS and BS algorithms**

Test Functions from CEC2013	BS (Mean Error ± Std. Dev.)	Modified BS (Mean Error ± Std. Dev.)
F6	1.31e+02±1.87e+01	<b>1.28e+02±1.40e+01</b>
F7	1.41e+02±1.65e+01	<b>1.38e+02±1.27e+01</b>
F9	3.44e+01±1.08e+00	<b>3.41e+01±9.48e-01</b>
F11	8.41e+01±1.25e+01	8.13e+01±1.08e+01
F12	<b>2.16e+02±1.07e+01</b>	2.32e+02±1.37e+01
F14	2.43e+03±2.40e+02	<b>2.29e+03±2.20e+02</b>
F15	6.87e+03±3.40e+02	<b>6.53e+03±3.80e+02</b>
F16	<b>2.69e+00±3.34e-01</b>	3.02e+00±2.11e-01
F17	1.44e+02±2.02e+01	<b>1.34e+02±1.16e+01</b>
F23	7.49e+03±3.10e+02	<b>7.42e+03±2.16e+02</b>
F25	3.08e+02±3.99e+00	<b>3.06e+02±5.50e+00</b>
F26	2.09e+02±2.16e+00	<b>2.08e+02±3.05e+00</b>



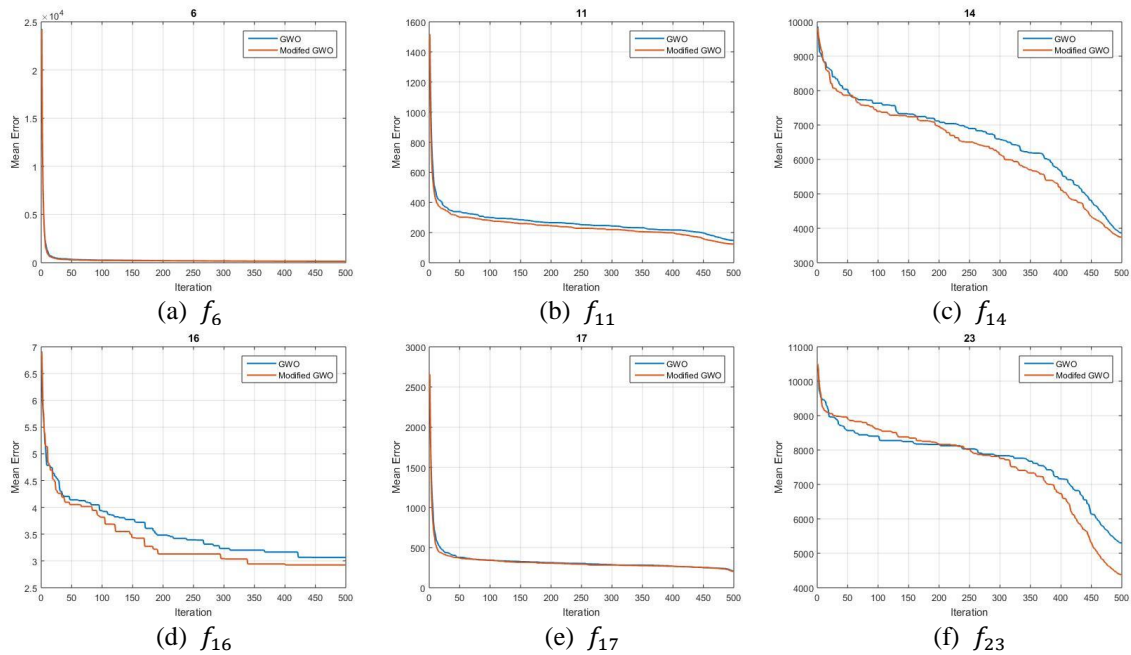
**Figure 4:**  
Convergence graphs of  $f_6$ ,  $f_{11}$ ,  $f_{14}$ ,  $f_{16}$ ,  $f_{17}$  and  $f_{23}$  for BS and modified BS

#### 4.2. Performance Analysis of Modified GWO and GWO

The comparative results of GWO and the modified GWO algorithms are shown in Table 6. As can be seen from this table, the proposed approach outperforms GWO in 8 out of 12 functions, whereas it does not show any improvement over the original algorithm for functions F6, F15, F25 and F26. The convergence graphs of the algorithms are shown in Figure 5. It can be observed that our proposal has a good convergence behavior for functions F14 and F16, but there are very few improvements for functions F11 and F17, meaning that the convergence behaviors are very close to each other. In function F23, we can obviously see that the convergence speed of the proposed algorithm is much faster than the original GWO.

**Table 6. Experimental results for 20 runs of 12 test functions for Modified GWO and GWO algorithms**

Test Functions from CEC2013	GWO (Mean Error ± Std. Dev.)	Modified GWO (Mean Error ± Std. Dev.)
F6	<b>1.43e+02±2.08e+01</b>	1.48e+02±4.81e+01
F7	8.39e+01±1.56e+01	<b>8.30e+01±1.82e+01</b>
F9	2.12e+01±2.54e+00	<b>2.03e+01±1.94e+00</b>
F11	1.48e+02±4.22e+01	<b>1.24e+02±3.04e+01</b>
F12	1.67e+02±6.07e+01	<b>1.67e+02±7.19e+01</b>
F14	3.86e+03±1.53e+03	<b>3.74e+03±1.29e+03</b>
F15	<b>3.55e+03±5.13e+02</b>	3.76e+03±1.48e+03
F16	3.07e+00±3.40e-01	<b>2.93e+00±4.19e-01</b>
F17	2.08e+02±5.97e+01	<b>2.02e+02±6.89e+01</b>
F23	5.30e+03±1.73e+03	<b>4.37e+03±1.49e+03</b>
F25	<b>2.75e+02±7.23e+00</b>	2.81e+02±6.90e+00
F26	<b>2.89e+02±7.65e+01</b>	3.39e+02±4.82e+01



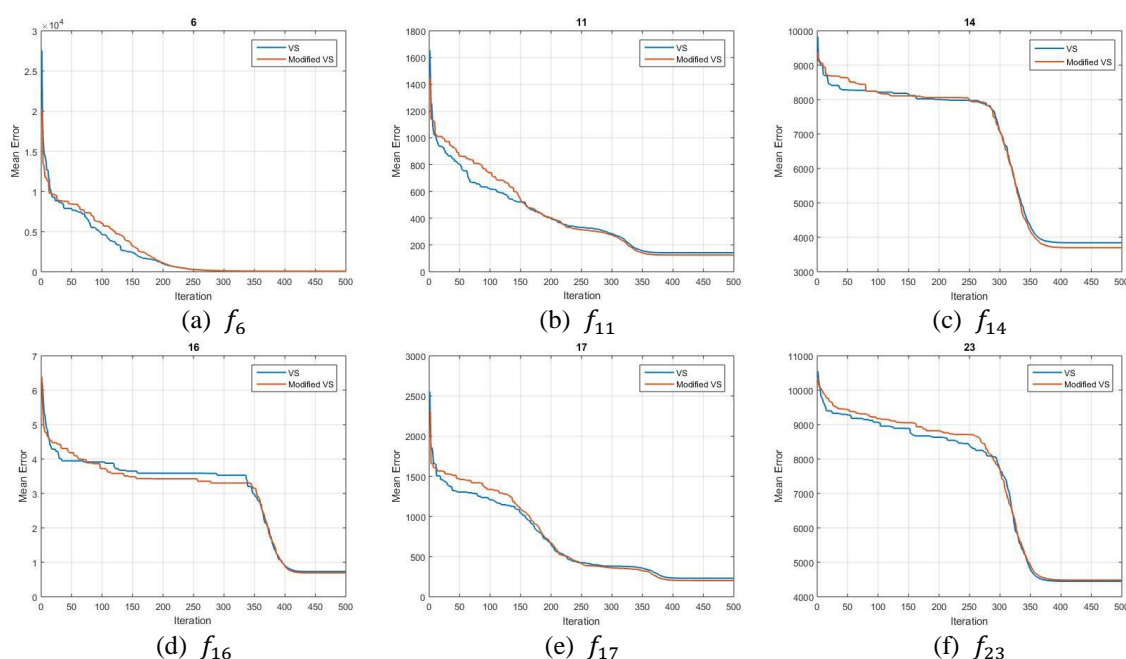
**Figure 5:**  
Convergence graphs for  $f_6$ ,  $f_{11}$ ,  $f_{14}$ ,  $f_{16}$ ,  $f_{17}$  and  $f_{23}$  for GWO and modified GWO

### 4.3. Performance Analysis of Modified VS and VS

The comparative results of the VS and the modified VS algorithm are presented in Table 7. Considering the performance of the proposed approach on the related functions, experimental results demonstrate that the proposed approach achieves better solutions in 9 out of 12 functions. The modified VS algorithm only performs worse than the original algorithm in functions F7, F9 and F23. As shown in Figure 6, analyzing the convergence graphs of the algorithms, the VS improves solutions better at the beginning of the evolutionary stage, whereas the modified VS produces a better solution during the latter half of iterations. In addition, we noticed that the proposed approach usually shows better performance than its original version in the later evolutionary stage.

**Table 7. Experimental results for 20 runs of 12 test functions for Modified VS and VS algorithms**

Test Functions from CEC2013	VS (Mean Error ± Std. Dev.)	Modified VS (Mean Error ± Std. Dev.)
F6	6.72e+01±2.04e+01	<b>6.29e+01±3.44e+01</b>
F7	<b>7.94e+01±2.44e+01</b>	8.64e+01±2.55e+01
F9	<b>2.08e+01±2.91e+00</b>	2.39e+01±3.07e+00
F11	1.42e+02±3.11e+01	<b>1.25e+02±3.08e+01</b>
F12	1.23e+02±4.30e+01	<b>1.21e+02±2.67e+01</b>
F14	3.84e+03±3.85e+02	<b>3.70e+03±6.71e+02</b>
F15	3.96e+03±1.68e+02	<b>3.56e+03±3.75e+02</b>
F16	7.29e-01±4.99e-01	<b>6.93e-01±4.62e-01</b>
F17	2.30e+02±6.24e+01	<b>2.02e+02±5.15e+01</b>
F23	<b>4.45e+03±6.12e+02</b>	4.48e+03±8.26e+02
F25	3.03e+02±1.71e+01	<b>2.96e+02±1.58e+01</b>
F26	2.01e+02±2.38e-01	<b>2.01e+02±1.89e-01</b>



**Figure 6:**  
Convergence graphs for  $f_6$ ,  $f_{11}$ ,  $f_{14}$ ,  $f_{16}$ ,  $f_{17}$  and  $f_{23}$  for VS and modified VS

## 5. CONCLUSION

This study presents modifications to the following actual metaheuristics: BS, GWO and VS, using a chaotic-based approach. The modifications are intended to take advantage of using chaotic values generated by chaos maps instead of some random variables used by algorithms in their operations. Using six different chaos maps in total, comparative performance analyzes were performed on 12 functions used in published literature. The performances of the algorithms, using experimental results, have been evaluated with respect to two different metrics: convergence rates and statistical results. Experimental results on 12 benchmark functions from CEC2013 problems demonstrated that the modifications applied to the algorithms were able to enhance the performance of the original versions of the algorithms. The aim of future work is to investigate the use of the proposed algorithms to solve other optimization problems.

## REFERENCES

1. Alatas, B., Akin, E. and Ozer, A. B. (2009) Chaos embedded particle swarm optimization algorithms, *Chaos, Solitons Fractals*, 40(4), 1715-1734. doi: [10.1016/j.chaos.2007.09.063](https://doi.org/10.1016/j.chaos.2007.09.063)
2. Civicioglu, P. (2013) Backtracking search optimization algorithm for numerical optimization problems, *Applied Mathematics and Computation*, 219(15), 8121-8144, 2013. doi: [10.1016/j.amc.2013.02.017](https://doi.org/10.1016/j.amc.2013.02.017)
3. Dogan, B. and Olmez, T. A. (2015) A new metaheuristic for numerical function optimization: vortex search algorithm, *Information Sciences*, 293, 125-145. doi: [10.1016/j.ins.2014.08.053](https://doi.org/10.1016/j.ins.2014.08.053)
4. Gandomi, A., Yang, X-S., Talatahari, S. and Alavi, A. (2013) Firefly algorithm with chaos, *Communications in Nonlinear Science and Numerical Simulation.*, 18(1), 89-98. doi: [10.1016/j.cnsns.2012.06.009](https://doi.org/10.1016/j.cnsns.2012.06.009)
5. Geem, Z., Kim, J. and Loganathan, G. (2001) A new heuristic optimization algorithm: harmony search, *Simulation*, 76(2), 60-68. doi: [10.1177/003754970107600201](https://doi.org/10.1177/003754970107600201)
6. Goldberg D. E. (1989) *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley Longman Publishing, USA.
7. Kellert, S. (1993) *In the Wake of Chaos: Unpredictable Order in Dynamical Systems*, University of Chicago Press, USA.
8. Kennedy J. and Eberhart R. (1995) Particle swarm optimization, *IEEE International Conference on Neural Networks*, 1942-1948. doi:10.1109/ICNN.1995.488968
9. Li, Y., Deng, S. and Xiao, D. (2011) A novel hash algorithm construction based on chaotic neural network, *Neural Computing and Applications*, 20(1), 133-141. doi: 10.1007/s00521-010-0432-2
10. Li-Jiang, Y. and Tian-Lun, C. (2002) Application of chaos in genetic algorithms, *Communications in Theoretical Physics*, 38(2), 168. doi: 10.1088/0253-6102/38/2/168
11. Liang, J. J., Suganthan, P. N. and Hernandez-Diaz, A. G. (2013) Problem definitions and evaluation criteria for the CEC 2013 special session on real-parameter optimization, *Zhengzhou University and Nanyang Technological University*, 3-18. Technical report:201212
12. Mirjalili, S., Mirjalili, S. M. and Lewis, A. (2014) Grey wolf optimizer, *Advances in Engineering Software*, 69, 46-61. doi: [10.1016/j.advengsoft.2013.12.007](https://doi.org/10.1016/j.advengsoft.2013.12.007)
13. Pecora, L. and Carroll, T. (1990) Synchronization in chaotic system, *Physical Review Letters*, 64(8), 821-824. doi: 10.1103/PhysRevLett.64.821
14. Saremi, S., Mirjalili, S. M. and Mirjalili, S. (2014) Chaotic krill herd optimization algorithm, *Procedia Techno*, 12, 180-185. doi: [10.1016/j.protcy.2013.12.473](https://doi.org/10.1016/j.protcy.2013.12.473)
15. Schuster, H. G. and Just, W. (2006) *Deterministic chaos: an introduction*, John Wiley & Sons, Germany.
16. Storn, R. and Price K. (1995) Differential evolution: A simple and efficient adaptive scheme for global optimization over continuous spaces, *International Computer Science Institute*, 1-12. Technical report:TR-95-012
17. Wang, N., Liu, L. M. and L. L. Liu. (2001) Genetic algorithm in chaos, *Or Transactions*, 5(5), 1-10.

18. Wang, L. and Zhong, Y. (2015) Cuckoo search algorithm with chaotic maps, *Mathematical Problems in Engineering*, 6(6), 546-554. doi: 10.1155/2015/715635
19. Wang, G.-G., Guo, L., Gandomi, A., Hao, G.-S. and Wang, H. (2014) Chaotic krill herd algorithm, *Information Sciences*, 274, 17-34. doi: [10.1016/j.ins.2014.02.123](https://doi.org/10.1016/j.ins.2014.02.123)
20. Wolpert, D. H. and Macready, W. G. (1997) No free lunch theorems for optimization, *IEEE Transactions on Evolutionary Computation*, 1(1), 67-82. doi: [10.1109/4235.585893](https://doi.org/10.1109/4235.585893)
21. Yang, X-S. (2010) *Nature-inspired Metaheuristic Algorithms*, Luniver Press, UK.
22. Yang, X-S. and Deb, S. (2009) Cuckoo search via L'evy flights, *Proceeding of World Congress on Nature Biologically Inspired Computing*, 210-214, doi: 10.1109/NABIC.2009.5393690
23. Yao, J. F., Mei, C., Peng, X. Q., Hu, Z. K. and Hu, J (2001) A new optimization approach-chaos genetic algorithm, *Systems Engineering*, 1, 1-5.
24. Zaharie, D. (2003) Control of population diversity and adaptation in differential evolution algorithms, *Mendel 9th International Conference on Soft Computing*, Brno, 41-46.
25. Zhenyu, G. Bo, Y. C. and Min, C. B. (2006) Self-adaptive chaos differential evolution, *International Conference on Natural Computation*, 972-975. doi: 10.1007/11881070\_128

