Chaotic Fitness Dependent Optimizer for Planning and Engineering Design

Hardi M. Mohammed (✉ hardi.mohammed@charmouniversity.org)

Charmo University   https://orcid.org/0000-0002-9766-9100

Tarik A. Rashid

University of Kurdistan Hewler

Chaotic Fitness Dependent Optimizer for Planning and Engineering Design

Hardi M. Mohammed[1,2], Tarik A. Rashid[3]

[1]Applied Computer Department, College of Medicals and Applied Sciences, Charmo University, Sulaimani, Chamchamal, KRG, Iraq.

[2]Technical College of Informatics, Sulaimani Polytechnic University, Sulaimani, KRG, Iraq

[3]Computer Science and Engineering, University of Kurdistan Hewler (UKH), Erbil, KRG, Iraq.

Corresponding author: Hardi M. Mohammed (e-mail: hardi.mohammed@charmouniversity.org).

ORCID: 0000-0002-9766-9100

ABSTRACT Fitness Dependent Optimizer (FDO) is a recent metaheuristic algorithm that mimics the reproduction behavior of the bee swarm in finding better hives. This algorithm is similar to Particle Swarm Optimization (PSO) but it works differently. The algorithm is very powerful and has better results compared to other common metaheuristic algorithms. This paper aims at improving the performance of FDO, thus, the chaotic theory is used inside FDO to propose Chaotic FDO (CFDO). Ten chaotic maps are used in the CFDO to consider which of them are performing well to avoid local optima and finding global optima. New technic is used to conduct population in specific limitation since FDO technic has a problem to amend population. The proposed CFDO is evaluated by using 10 benchmark functions from CEC2019. Finally, the results show that the ability of CFDO is improved. Singer map has a great impact on improving CFDO while the Tent map is the worst. Results show that CFDO is superior to GA, FDO, and CSO. Both CEC2013 and CEC2005 are used to evaluate CFDO. Finally, the proposed CFDO is applied to classical engineering problems, such as pressure vessel design and the result shows that CFDO can handle the problem better than WOA, GWO, FDO, and CGWO. Besides, CFDO is applied to solve the task assignment problem and then compared to the original FDO. The results prove that CFDO has better capability to solve the problem.

Keywords Fitness Dependent Optimizer, Chaotic Maps, Chaotic Fitness Dependent Optimizer, Benchmark Functions, Pressure Vessel Design Problem, Task Assignment Problem

1. INTRODUCTION

Finding an optimal solution is the aim of solving optimization problems while they have various constraints. The optimal solution should be found in a feasible search space and in a reasonable amount of time. Usually, traditional algorithms have been used to solve optimization problems. These algorithms require a lot of time to achieve an optimal solution because they are searching from a single point and attempting to collect much information [1]. Traditional algorithms, such as Hill climbing, Simulated Annealing (SA), and Random Search, and then, metaheuristic algorithms have been used to solve complex optimization problems [2]. However, these algorithms also have some deficiencies in terms of finding global optima due to having different constraints in real-world applications, namely, engineering design problems [3], task planning problems, and economical problems [4].

Metaheuristic algorithms are used to solve real-world problems either by taking physical or biological phenomenon [5], [6]. Depending on the stochastic technics, metaheuristic algorithms could avoid local optima and obtain global optima while they have different values and convergence speeds [7], [8]. Recent metaheuristic algorithms, such as Particle Swarm Optimization (PSO) [9], Bat Optimization Algorithm (BA) [10], Grey Wolf Optimization Algorithm (GWO) [11], and Whale Optimization Algorithm (WOA) [12] are popular because of two factors: Firstly, they are very simple, efficient, robust, and adaptable. Secondly, in terms of implementation, they can be implemented easily in any programming language. Because of these two factors, they are used to solve various real-world problems [13], [14], [15].

Even though finding local optima and global optima depend on the search mechanism inside the algorithm, furthermore, balancing between exploration and exploitation is another crucial point in metaheuristic algorithms because it depends on stochastic numbers. Achieving better performance can be done by having a proper balance between exploration and exploitation [13], [16]. Exploitation uses information that has been collected so far to achieve a solution better than the previous one in the next generation [17], [18]. In contrast, exploration helps the algorithm avoid local optima and obtaining a new solution that can be far from the current solution [19], [20]. Therefore, avoiding local optima is the advantage of exploration while having slow convergence is the drawback of it. However, the advantage of exploitation is high convergence while it may lead to stuck in local optima [13], [21].

Despite of having similarities among developed algorithms, new metaheuristic algorithms will be developed by researchers. However, there are different algorithms that are based on social insects and their biological behaviors are hard to understand by expertise in optimization algorithm [22]. So, researchers should use optimization algorithm terms when they develop new metaheuristic due to simplicity [23]. Also, researchers called their developed algorithms as a novel metaheuristic algorithm

while the researchers of paper [24] mentioned that BA, GWO, Firefly are not new algorithm while they are just a repetition of the idea of PSO,

FDO is a nature metaheuristic algorithm that is based on the reproductive behavior of bee swarm while the bees are trying for finding new hives. FDO is based on PSO but it uses a different mechanism to update agent position. Scout bees search for positions randomly, while the better position is found, the previous position is neglected. If the next position is not better than the previous position, then the FDO uses the previous position to find a better solution [25]. FDO has a very competitive result compared to other metaheuristic algorithms and it has also been used to solve some real-world problems, such as frequency-modulated sound waves, aperiodic antenna array designs, pedestrian evacuation models, etc. [10] [26].

Trapping into the local optima is a vital matter due to the random initialization of variables and population. Hence, to circumvent the above problem chaos theory is used in FDO. Consequently, in this paper, FDO and CFDO are applied to the classical engineering problem, which is called pressure vessel design.

There are three ways to use chaos theory inside metaheuristic algorithms. Firstly, chaos can be used to initialize random numbers in the algorithm. Using chaos as the operation is the second way of using chaos, and finally, it can be used to initialize the population [4], [27]. Using these techniques can improve algorithms in terms of convergence speed and avoiding local optima. Chaos theory has been used by many metaheuristic algorithms, such as PSO [28], Artificial Bee Colony (ABC) [29], Bat Algorithm (BA) [14], Grey Wolf Optimization (GWO) [16], and Whale Optimization Algorithm (WOA) [1]. Therefore, the performance of these algorithms is improved by using chaos theory. Chaos theory is selected to improve FDO because chaotic maps have unpredictable dynamic behavior which improves the performance of metaheuristic algorithms to jump out from local optima [30].

The main contribution of this paper is proposing a Chaotic Fitness Dependent Optimizer and then using it to enhance the performance of FDO and to solve the pressure vessel design problem and task assignment problem. This contribution is done by using ten chaotic maps. The following are the main parts of the contribution in FDO:

1) Initializing population using chaotic map

2) Updating speed of current bee by multiplying its position with a chaotic random value.

3) Changing amending mechanism of position if it is beyond the search space. When the new position is calculated. A mechanism is used to check whether the new position is inside the search space or not.

Both of the following points are used at the same time to improve the FDO so that it: 1) avoids falling into local optima and 2) enhances the search capability of it. CEC2019 benchmark test functions are used to evaluate the result of CFDO compared

to the standard FDO and also Wilcoxon Rank Sum test is used to evaluate the result statistically. CEC2013 and CEC2005 benchmark functions are also used to evaluate the performance of CFDO.

The rest of the sections are organized as follows: FDO is described in detail in Section 2, the next, chaotic maps are discussed in Section 3. Then, in Section 4, CFDO is described. In section 5, the results are presented and evaluated. Then, solving the pressure vessel design problem and task assignment problem are presented in Section 6. In the final section, the conclusion and future work are presented.

2. FDO

FDO is proposed by Jaza M. Abdullah and Tarik A. Rashid in 2019. This algorithm is based on the searching mechanism of the bee swarm and the reproduction process of finding better hives. Scout bees are responsible for searching among many potential hives and then they decide on which hive is better than the rest.

FDO starts by initializing the scout bee population randomly inside the search space. Discovering the hive depends on the position and fitness function of each scout bee. Finding the better hive (new solution) is the main purpose of the scout bee. If the next solution is better than the previous solution, then the scout bee ignores the previous one. However, if a better solution cannot be found, then the scout bee relies on the previous solution to change its position. Scout bee can be represented as X_i *(i=1, 2,...,n)*.

Fitness weight and random walk methods are used by scout bees to search inside the search space randomly. As can be seen in Equation (1), the scout bee movement depends on the pace of changing the position of the current scout. By adding the pace, the scout tries to explore a new solution that is better than the previous one.

$$X_{i,t+1} = X_{i,t} + pace \tag{1}$$

Where *X* represents the artificial scout bee, *i* denotes the current scout bee (search agent) and *t* is the current iteration. the movement of the scout that can be detected is represented by the *pace*, while it depends on the fitness weight *fw*. Although, randomization is the base of the *pace*. Therefore, *fw* equation can be represented as follow:

$$fw = \left| \frac{X^*_{i,t\,fitness}}{X_{i,t\,fitness}} \right| - wf \tag{2}$$

$X^*_{i,t}$ *fitness* represents the best solution that was found so far by scout bee and $X_{i,t}$ *fitness* is the best solution of the current scout bee. The weight factor is represented by *wf* and it has the range value in [0,1]; FDO has *r* variable, which is a random number in the range of [-1, 1], this random number is initialized by using the Levy flight mechanism. Therefore, FDO has

simplicity in its calculation for finding objective functions because each agent requires fitness weight and a random number to be calculated.

After initializing the scout bee randomly on the search space by using the upper and lower boundaries. Then, a global solution is obtained. Next, *fw* is calculated, which is depending on the following conditions through the below equations:

If *fw* value is equal to 1, then the *pace* can be calculated by Equation (3) if one of the three conditions is true: *fw=0 fw=1* or $X_{i,t\ fitness} = 0$.

$$pace = X_{i,t} * random \tag{3}$$

When *fw* greater than zero and less than 1, then the random number *r* is generated in the range [1, -1]. After that, if *r* is less than zero Equation (4) is used to calculate the *pace*. Otherwise, the *pace* is found by Equation (5).

$$pace = distance_{best_{bee}} * random \tag{4}$$

$$pace = distance_{best_{bee}} * fw \tag{5}$$

Where *distance_best_bee* can be calculated by differentiating the current scout bee from the best scout bee as can be seen from Equation (6).

$$distance_{best_{bee}} = X^* - X_{i,t} \tag{6}$$

3. Chaotic Maps

To improve the performance of metaheuristic algorithms, different mathematical approaches are used. One of these approaches is the chaos theory, which is a nonlinear system [31], which has dynamic behaviour. Because of its effectiveness, it has been used in different areas by researchers: research optimization, synchronization [32], and chaos control [33].

Chaotic maps are bounded by a nonlinear system that has random properties. These maps are very sensitive to initial values [34]. The chaos theory was discovered by Lorenz when he found that a little change in the initialization makes a radical change in results [27]. Furthermore, chaotic maps can be initialized by choosing any value between 0 and 1; consequently, 0.7 was used in this study as the initial point of the chaotic maps [1]. Therefore, ten common chaotic maps were used for the initialization of random variables and the population of the scout bee with FDO. Table 1 shows ten chaotic maps.

Chaotic maps are described with different NIAs in this section. Researchers have used chaotic maps to improve the performance of many algorithms. For example, chaotic GA (CGA) was proposed to overcome premature convergence problems to maximize the number of iterations and reach the global optima. Two chaotic maps are used to improve the

performance of GA: tent map and logistic map. These maps were used to generate random numbers in GA. The results from five benchmark functions proved that the diversity of population from CGA can jump out of local optima and it was better than the original GA and PSO. CGA also improved GA in terms of reducing the number of iterations required to solve problems [35].

Chaotic Grasshopper Optimization Algorithm (CGOA) was proposed to consider the convergence speed problem. Ten chaotic maps were used to verify which of them has higher performance with GOA. The results revealed that the circle map had better results when it was tested on 13 benchmark functions. The aim of using a chaotic map was to improve a balance between the exploration and exploitation phases. The results showed that CGOA obtained better results by using proper balance and avoiding local optima against GOA [36].

The investigation into a proper balance between exploration and exploitation challenges researchers because metaheuristic algorithms are based on stochastic behaviour. Thus, the exploration phase uses random behaviour to find better search space, while achieving solutions in a fast way is the aim in the exploitation phase. Therefore, a chaotic GSA was proposed to overcome the problems. Adaptive normalization was also used inside GSA to transit between the exploration phase and the exploitation phase. Then, 12 benchmark functions were used to test the performance of GSA. The statistical results showed that the sinusoidal map achieved significant results compared with the original GSA [37].

Fruit fly optimization algorithm (FOA) was combined with chaos methods to overcome stagnating into local optima, and the proposed algorithm was called chaotic FOA (CFOA). Therefore, a logistic chaotic map was used to investigate the optimum best fruit fly, and the new location was based on the best location of the fruit fly and the best chaotic position. Balancing between both phases depended on a chaotic map or fruit fly independently. The simulation results based on testing six benchmark functions showed that CFOA achieved better solutions against PSO, ABC, FOA [38].

CS has difficulties in terms of convergence speed and solution quality because it is based on a random walk with a scaling factor. Chaotic CS (CCS) was proposed to avoid local optima and improve the solution quality. Consequently, the chaotic map used a variable for both scaling factor and fractional probabilities. Controlling dimension expectation was based on the fraction probability. Therefore, 20 benchmark test functions were used to test CCS to ensure that it works well when it dealt with the high dimensionality of problems [39].

According to the study used seven chaotic maps to improve ABC because ABC has problems relating to local optima and convergence speed. Chaotic maps were used to generate a random number that was required by the original ABC. Three benchmark functions were used to verify the performance of CABC. The results indicated that CABC performs well against the original ABC [40].

4. Chaotic Fitness Dependent Optimizer (CFDO)

Original FDO has achieved better performance compared to DA, WOA, and SSA when tested on ten CEC2019 benchmark test functions. By looking at the results of FDO in [25], it can be said that FDO still requires enhancement due to avoiding local optima and increase the convergence rate. In general, chaos can be called a source of randomness. Chaotic derives from a chaotic map, thus, chaos is the feature of the complex system and the map is the relation of the chaos behavior by using some parameters inside the algorithm [14]. Consequently, chaotic maps have been used by researchers in the field of optimization algorithms because of their nonlinear and dynamic behavior [16].

Therefore, to introduce CFDO, 10 chaotic maps are used in FDO while each of them has a different mathematical equation. As a result, CFDO is introduced in this paper to tackle this problem and improve the performance of FDO concerning the convergence speed and local optima. The pseudocode of CFDO is illustrated in Algorithm 1.

Chaotic maps are embedded in FDO in two different ways:

1) Initialization of the random variable r, which is generated by using a chaotic map instead of using the Levy flight method. This variable is used to update *pace* by multiplying position. New technic is also used to amend the population if they are outside the limitation.

2) Initializing the scout bee population using a chaotic map instead of the stochastic mechanism.

Algorithm 1 Pseudocode of the CFDO

Initialize the population of scout bee by using chaotic map $X_{i,t}$ (i=1,2,…, n)
while (t< max_iteration)

    for each scout bee $X_{i,t}$

        Select best scout bee $X^*_{i,t}$

        If ($X_{i,t}$ *fitness*==0)

            fitness weight =0

        else

        find *fitness weight* using Equation (2)

        end if

        if (*fitness weight=0 // fitness weight=1 // $X_{i,t}$ fitness==0*)

            find *pace* by using Equation *(3)*

        else

            r generated in range [-1, 1] by using chaotic map

            if (r>=0)

             calculate the *pace* using Equation (5)

            else

             calculate *pace* using Equation (4)

            end if

        end if

$$\textit{determine } X_{i,\,t+1} \text{ using Equation (1)}$$

if $(X_{i,\,t+1} < X_{i,t}$ *fitness*)

update position of scout and select *pace*

else

calculate $X_{i,\,t+1}$ using Equation (1)

if $(X_{i,\,t+1} < X_{i,t}$ *fitness*)

update position of scout and select *pace*

else

update current position without any movement

end if

end if

end for

end while

---

5. EXPERIMENTAL RESULTS AND DISCUSSION

According to [1], [14] ten chaotic maps have been used to initialize a random variable and population of scout bee in the proposed chaotic algorithms and then compared against the FDO. As a result, the ten chaotic maps are used separately inside the FDO for both initializing the random value and initializing the scout bee population.

Therefore, to ensure that the performance of any optimization algorithms is efficient, they must be tested by common benchmark test functions. As a result, different benchmark functions, such as CEC2005, CEC2013, and CEC2019 are used to evaluate the proposed CFDO.

5.1 CEC2019 Benchmark Functions

Parameters are set as 30 is taken as a population size for CFDO, GA, PSO, CSO, FDO. Each algorithm performed 100 iterations to find the optimum solution. Then, ten chaotic maps are used inside CFDO. As a result, the Singer map performed superior compared to other maps. This algorithm runs 30 times and then the average results of the 30 times are taken for CEC2019 benchmark functions.

As a result, CEC2019 benchmark functions are proposed in [41] and used in [42], and then, the implementation is conducted using Matlab code to verify the performance of our proposed chaotic FDO. CEC2019 consists of ten benchmark functions. All the functions in CEC2019 are multimodal. Therefore, multimodal functions have more than one optimum value, accordingly, it is a challenging matter for the CFDO, because it has to find one global optimum and avoid others, which is called local optima. Table 2 presents CEC2019 functions, which show the functions name, dimension, and the range of the search space for each function.

5.1.1 Comparing CFDO Vs. FDO using different chaotic maps

Results are conducted from various chaotic maps by taking 50 iterations and 30 population sizes. The average results are taken from 30 independent runs. Ten chaotic maps are applied to the DFO, namely, Chebyshev, Circle, Iterative, Logistic, Piecewise, Sine, Singer, Sinusoidal, and Tent. These maps are presented in Table 1 as CFDO1 to CFDO 10 respectively.

Table 3, 4, 5, 6, and 7 show the results of each chaotic map against FDO. Each table includes the results of two chaotic maps. Table 5 shows that CFDO5 has superior results compared to FDO and other chaotic FDOs. CFDO 6 and CFDO 7 have the second and third best averages respectively. Therefore, CFDO 1 to CFDO 9 has improved the performance of FDO. In contrast, Table 7 illustrates that CFDO 10 has produced the worst result against FDO. It can be said that Chebyshev, Circle, Iterative, Logistic, Piecewise, Sine, Singer, and Sinusoidal could improve the performance of FDO. The p-value showed that CFDO 8 has six significant values and it is the best one and followed by CFDO 5 and CFDO 6, which their results are significant in 5 benchmark functions out of ten. Overall, the Tent map shows the worse results in all ten benchmark functions. However, the Singer map enhances the performance of CFDO and recorded the first best result.

5.1.2 Statistical Results

Evaluating performance based on the average and standard deviation is not accurate enough, hence, the statistical test must be used. Depends on the statistical test, the proposed chaotic algorithm can be evaluated against other metaheuristic algorithms. Therefore, Wilcoxon rand-sum test is conducted in this paper to evaluate the mean and standard deviation to ensure that either the result is significant or not. In general, the p-value is calculated using the Wilcoxon rank-sum test and must be less than 0.05, then the result is significant. Table 3, 4, 5, 6, and 7 presents the p-value for the results and the significant results are underlined.

It can be seen that the Singer map shows the best significant results in 6 out of 10 functions while the Circle map and Tent map provide the worst results. Furthermore, the Logistic map and Piecewise map have the second-best significant value in the 5 functions. Sine map also provides significant value in 4 functions while Gauss/ Mouse, Iterative and Sinusoidal map has 3 significant values. However, the second-worst result is the Circle map, which has only one single significant result.

Overall, Singer, Logistic, and Piecewise maps are better chaotic maps that could enhance the performance of CFDO. The reason behind this improvement is that these three chaotic maps can improve the exploration capability of CFDO and avoid local optima and also find the best global optima among various solutions.

As it is clear that from Table 8 Singer map is the best chaotic map that improves the performance of FDO. Consequently, CFDO is evaluated on the CEC2019 benchmark functions and then compared to CSO, GA, PSO, and FDO. It can be seen that

from Table 8 that CFDO has outperformed FDO in six functions. However, CFDO and FDO have the same result in function three. CFDO is also better than GA and CSO in six functions while it outperforms well in one function against PSO. Overall, CFDO performance is better than FDO, GA, and CSO. However, PSO achieved the best results compared to CFDO.

 The reason behind these results is that using the chaotic singer map for the initializing population and the random number is very efficient. Finally, results show that the proposed CFDO is very efficient and can be applied to solve real-world problems.

5.2 EC2013 Benchmark Functions

The population size is set to 100 search agents with 100 dimensions for each test function. The algorithm runs 5000 iterations for each time. The results of the compared algorithms are taken from 51 run times. Then, CFDO is compared against the algorithms.

Another common benchmark function that is used to evaluate CFDO is the CEC2013 benchmark function. CEC2013 consists of 28 benchmark functions which are: F1-F5 unimodal functions, F6-F20 multimodal functions, and F21-F28 composite functions. Table 9 presents the benchmark functions of CEC2013.

CFDO is compared with the different common algorithms, such as Gravitation Search Algorithm (GSA), Chaotic (CJADE), Success History-based parameter Adaptation for Differential Evolution (SHADE), Increasing Population covariance matrix adaptation evolution strategy (IPOP-CMA-ES), and NBIPOP-CMA-ES. The results of the algorithms are taken from [43], [44], [45], [46].

Table 10 shows the comparison of CFDO against CJADE, IPOP-CMA-ES, NIPOP-CMA-ES, SHADE, and GSA. Results proved that CFDO has the worst results compared to the other algorithms. However, it has the same result as GSA and CJADE in function eight. Despite having poor results, CFDO achieved better results in function nine against SHADE, NBIPOP-CMA-ES, and IPOP-CMA-ES. As it can be seen that CFDO did not have better results compared to the other algorithm because of the complexity of CEC2013 benchmark functions. Also, it can be said that chaotic map techniques did not perform well to solve CEC2013 benchmark functions.

5.3 CEC2005 Benchmark Functions

To evaluate the performance of CFDO, 14 benchmark functions were used CFDO. The benchmark functions are unimodal, multimodal, and expanded benchmark functions. F1-F5 are unimodal functions while F6-F12 are multimodal functions. Both functions 13 and 14 are expanded benchmark functions. Table 11 shows the CEC2005 benchmark functions. The population

size using CEC2005 benchmark functions is set to 100 with 30 dimensions. Each algorithm runs 30 times with 5000 iterations. The results of the comparison are taken from the average of the 30 runs. Table 12 presents that CFDO compared against BSO and CBSO. The results of BSO and CBSO were taken from [30]. As it can be seen from the table CFDO outperforms well compared to BSO and CBSO in 7 functions out of 14 functions. CFDO is better than CBSO in three unimodal functions which means that it performs well in terms of exploration. However, CBSO is superior regarding the exploitation phase because CFDO is better in only 2 functions out of 7. The reason behind the improvement of CFDO in exploration is that using a chaotic map improved the randomization of the population. Also using the chaotic technique instead of the Levy Random Number technique improved the performance of FDO because the random number is responsible for updating the *pace* value in FDO.

6. Solving Application Problems by CFDO

In this section, CFDO is used to solve two problems, which are pressure vessel design and task assignment problem.

6.1 Pressure Vessel Design Problem

Classical engineering problem like pressure vessel design is a problem. The cost of three sections of the cylindrical pressure vessel is required to be optimized, so, minimizing the cost is the aim of optimizing this problem. Forming, material, and welding should be minimized. The head of the vessel has a hemispherical shape while the end of both sides of the vessel is crapped. Four variables have to be optimized. These variables are the head thickness T_h , shell thickness T_s, inner radius R and cylindrical length section without counting the head L. As a result, the problem has to be minimized depends on four constraints. The following equations describe the constraints of the problem.

$$n = 1,2, \ldots., 4$$

$$\vec{x} = [x_1 x_2 x_3 x_4] = [T_s T_h\ R\ L],$$

$$f(\vec{x}) = 0.6224x_1x_3x_4 + 1.7781x_2x_3^2 + 3.1661x_1^2x_4 + 19.84x_1^2x_3, \tag{7}$$

Variable limitation
$$0 \le x_1 \le 99,$$

$$0 \le x_2 \le 99,$$

$$10 \le x_3 \le 200,$$

$$10 \le x_4 \le 200,$$

These are subjected to

$$g_1(\vec{x}) = -x_1 + 0.0193x_3\ \le 0 \tag{8}$$

$$g_2(\vec{x}) = -x_3 + 0.00954x_3 \le 0 \qquad (9)$$

$$g_3(\vec{x}) = -\pi x_3^2 x_4 - \frac{4}{3}\pi x_3^3 + 1{,}296{,}000 \le 0 \qquad (7)$$

$$g_4(\vec{x}) = x_4 + 240 \le 0 \qquad (8)$$

Different algorithms have been used to solve this problem, such as GA, WOA, PSO, and GWO. Therefore, the authors applied FDO and CFDO to solve the problem. Table 13 shows that CFDO outperforms well compared to other algorithms while CGWO has the second-best result after CFDO. Finally, CFDO obtained these results 1.54, 6.10, 30.58, 73.29 for $T_s, T_h, R \ and \ L$ respectively.

6.2 Solving Task Assignment Problem

The task assignment problem is an NP problem and it is the base step to ensure the ability of exploitation in parallel or distributed computing systems. It is working on diving tasks among different processors [47] [48]. The aim of optimizing this problem is to minimize the execution cost for each task. Task assignment problem has the following equations:

$$f(x) = \sum_{i=1}^{n} \sum_{j=1}^{n} C_{ij} X_{ij} \qquad (12)$$

Subject to:

$$X_{ij} = \begin{cases} 1 \ if \ the \ task \ is \ assigned \ to \ specific \ processor, \\ 0, \qquad otherwise \end{cases}$$

Where C_{ij} is the cost of each task while X_{ij} is the indication for the task whether it is assigned for a specific processor or not. In this paper, the authors have a task assignment problems related to a department. The problem is that we have five employees with five tasks. Each employee can execute the task at different times. The execution time for all the tasks has a lower limit and upper limit, which are 10 and 99 respectively. Table 14 illustrates the tasks and employees as an example. Therefore, CFDO and FDO are used to solve this problem to show the efficiency of the proposed CFDO compared to FDO.

CFDO and FDO algorithms have been run for 15 iterations because they could solve the problem in several iterations. As it can be seen that from Table 15, CFDO outperforms well and its performance is much better than FDO due to using chaotic maps inside FDO.

7. Result Analysis

To analyse the results of CFDO, few points should be taken into consideration. Initializing the population of metaheuristic algorithms is a critical point by researchers. So, different technic of randomizations is used to have proper search space. However, in this paper using the chaotic randomization technique improved the performance of FDO because chaotic maps have non-repetition and ergodicity features which help the algorithm to explore the search space at a higher speed.

The *pace* vector has a great role inside FDO to update the position to define the new solution. A random variable, which is called *r* is a key parameter that can be tuned to change the value of *pace* while the new position is updated based on the *pace* value. In addition, using a chaotic map improved the exploration of FDO because of having great dynamic randomization behavior of the chaotic maps. As a result, the main reason behind the great achievement of CFDO against the compared algorithm is using chaotic maps to initialize the population of the scout bee and initializing the random variable (*r*).

8. Conclusion and Future Work

In this paper chaotic theory is embedded inside the FDO algorithm, thus, the CFDO is proposed. The CFDO is presented to enhance the performance of CFDO. To make sure that the chaotic maps have a great impact on the performance of CFDO, chaotic maps were used in two different ways at the same time. The chaotic map is used to initialize the random variable and to initialize the scout bee population. Ten different chaotic maps were used in the CFDO algorithm, then, the CFDO was evaluated using CEC2019 benchmark functions, which consisted of ten benchmark functions. Wilcoxon's rand-sum test was also calculated to ensure that the results were either significant or not. In general, chaotic maps improved the ability of CFDO for searching for global optima and avoiding local optima because chaotic maps could control the random variable and population of scout bee for finding the optimum solution and improving the convergence of CFDO. The results showed that the Singer map had the best results compared to the original FDO. Although, the Tent map achieved the worst results.

CEC2019 benchmark functions were used to evaluate the performance of CFDO against FDO, GA, CSO, and PSO. Results indicated that CFDO could achieve higher performance compared to FDO, GA, and CSO. However, PSO performed well against CFDO. CFDO was used to solve CEC2013 benchmark functions and the results proved that CFDO did not achieve the best results compared to the other five algorithms. However, it obtained high performance when it was used to solve CEC2005 functions. Therefore, CFDO achieves seven best results out of 14 compared to BSO and CBSO. FDO and CFDO applied to pressure vessel design as an engineering problem and then compared to WOA, GWO, and CGWO. Results proved that CFDO

improved the ability of FDO to solve the problem. Then, the task assignment problem has been solved by CFDO and FDO. Results proved that CFDO was better at solving the problem.

Chaotic maps could improve the performance of FDO and avoid local optima because of their dynamic and superlative way of generating random numbers. Using new technic to amend the bee population if they are outside the limitation is another reason which enhanced the capacity of FDO. Since using its old technic to solve task assignment problems, FDO has problems with amending the population. Also, CFDO can be further extended to solve mixed-type problems.

Compliance with Ethical Standards

Conflict of Interest: The authors declare that they have no conflict of interest.

Ethical approval: This article does not contain any studies with human participants or animals performed by any of the authors.

References

[1] G. Kaur and S. Arora, "Chaotic whale optimization algorithm," *J. Comput. Des. Eng.*, vol. 5, no. 3, pp. 275–284, 2018.

[2] V. Granadeiro, L. Pina, J. P. Duarte, J. R. Correia, and V. M. S. Leal, "A general indirect representation for optimization of generative design systems by genetic algorithms: Application to a shape grammar-based design system," *Autom. Constr.*, vol. 35, pp. 374–382, Nov. 2013.

[3] R. Khatibi, M. A. Ghorbani, and F. A. Pourhosseini, "Stream flow predictions using nature-inspired Firefly Algorithms and a Multiple Model strategy – Directions of innovation towards next generation practices," *Adv. Eng. Informatics*, vol. 34, pp. 80–89, Oct. 2017.

[4] H. Lu, X. Wang, Z. Fei, and M. Qiu, "The effects of using chaotic map on improving the performance of multiobjective evolutionary algorithms," *Math. Probl. Eng.*, 2014.

[5] M.-Y. Cheng, D. Prayogo, Y.-W. Wu, and M. M. Lukito, "A Hybrid Harmony Search algorithm for discrete sizing optimization of truss structure," *Autom. Constr.*, vol. 69, pp. 21–33, Sep. 2016.

[6] D. Tien Bui, N.-D. Hoang, H. Nguyen, and X.-L. Tran, "Spatial prediction of shallow landslide using Bat algorithm optimized machine learning approach: A case study in Lang Son Province, Vietnam," *Adv. Eng. Informatics*, vol. 42, p. 100978, Oct. 2019.

[7] M. S. El-Abbasy, A. Elazouni, and T. Zayed, "MOSCOPEA: Multi-objective construction scheduling optimization using elitist non-dominated sorting genetic algorithm," *Autom. Constr.*, vol. 71, pp. 153–170, Nov. 2016.

[8] M. Guerrero, F. G. Montoya, R. Baños, A. Alcayde, and C. Gil, "Community detection in national-scale high voltage transmission networks using genetic algorithms," *Adv. Eng. Informatics*, vol. 38, pp. 232–241, Oct. 2018.

[9] R. Poli, J. Kennedy, and T. Blackwell, "Particle swarm optimization," *Swarm Intell.*, vol. 1, no. 1, pp. 33–57, Oct. 2007.

[10] X.-S. Yang, "A New Metaheuristic Bat-Inspired Algorithm," in *Studies in Computational Intelligence*, 2010, pp. 65–74.

[11] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey Wolf Optimizer," *Adv. Eng. Softw.*, vol. 69, pp. 46–61, 2014.

[12] S. Mirjalili and A. Lewis, "The Whale Optimization Algorithm," *Adv. Eng. Softw.*, vol. 95, pp. 51–67, 2016.

[13] X.-S. Yang and X. He, "Nature-Inspired Optimization Algorithms in Engineering: Overview and Applications," in *Studies in Computational Intelligence*, X.-S. Yang, Ed. Switzerland: Springer International Publishing Switzerland 2016, 2016, pp. 1–20.

[14]   A. H. Gandomi and X. S. Yang, "Chaotic bat algorithm," *J. Comput. Sci.*, vol. 5, no. 2, pp. 224–232, 2014.

[15]   H. Zhang, L. Tang, C. Yang, and S. Lan, "Locating electric vehicle charging stations with service capacity using the improved whale optimization algorithm," *Adv. Eng. Informatics*, vol. 41, p. 100901, Aug. 2019.

[16]   M. Kohli and S. Arora, "Chaotic grey wolf optimization algorithm for constrained optimization problems," *J. Comput. Des. Eng.*, vol. 5, no. 4, pp. 458–472, 2018.

[17]   H. M. Mohammed, S. U. Umar, and T. A. Rashid, "A Systematic and Meta-Analysis Survey of Whale Optimization Algorithm," *Comput. Intell. Neurosci.*, vol. 2019, pp. 1–25, Apr. 2019.

[18]   H. Mohammed and T. Rashid, "A novel hybrid GWO with WOA for global numerical optimization and solving pressure vessel design," *Neural Comput. Appl.*, Mar. 2020.

[19]   B. A. Hassan and T. A. Rashid, "Operational framework for recent advances in backtracking search optimisation algorithm: A systematic review and performance evaluation," *Appl. Math. Comput.*, vol. 370, p. 124919, Apr. 2020.

[20]   A. M. Ahmed, T. A. Rashid, and S. A. M. Saeed, "Cat Swarm Optimization Algorithm: A Survey and Performance Evaluation," *Computational Intelligence and Neuroscience*. 2020.

[21]   C. M. Rahman and T. A. Rashid, "Dragonfly Algorithm and Its Applications in Applied Science Survey," *Comput. Intell. Neurosci.*, vol. 2019, pp. 1–21, Dec. 2019.

[22]   K. Sörensen, "Metaheuristics-the metaphor exposed," *Int. Trans. Oper. Res.*, vol. 22, no. 1, pp. 3–18, Jan. 2015.

[23]   C. García-Martínez, P. D. Gutiérrez, D. Molina, M. Lozano, and F. Herrera, "Since CEC 2005 competition on real-parameter optimisation: a decade of research, progress and comparative analysis's weakness," *Soft Comput.*, vol. 21, no. 19, pp. 5573–5583, 2017.

[24]   C. L. Camacho Villalón, T. Stützle, and M. Dorigo, "Grey Wolf, Firefly and Bat Algorithms: Three Widespread Algorithms that Do Not Contain Any Novelty," in *Grey Wolf, Firefly and Bat Algorithms: Three Widespread Algorithms that Do Not Contain Any Novelty*, Springer, Cham, 2020, pp. 121–133.

[25]   J. M. Abdullah and T. Ahmed, "Fitness Dependent Optimizer: Inspired by the Bee Swarming Reproductive Process," *IEEE Access*, vol. 7, pp. 43473–43486, 2019.

[26]   D. A. Muhammed, S. A. Saeed, and T. A. Rashid, "Improved Fitness-Dependent Optimizer Algorithm," *IEEE Access*, vol. XX, pp. 1–1, 2020.

[27]   E. N. Lorenz, "Deterministic nonperiodic flow," in *Universality in Chaos, Second Edition*, 2017, pp. 130–141.

[28]   Y. Feng, G. F. Teng, A. X. Wang, and Y. M. Yao, "Chaotic inertia weight in particle swarm optimization," in *Second International Conference on Innovative Computing, Information and Control, ICICIC 2007*, 2008, pp. 475–

475.

[29]   B. Wu and S. H. Fan, "Improved artificial bee colony algorithm with chaos," in *Communications in Computer and Information Science*, 2011, pp. 51–56.

[30]   Y. Yu, S. Gao, S. Cheng, Y. Wang, S. Song, and F. Yuan, "CBSO: a memetic brain storm optimization with chaotic local search," *Memetic Comput.*, vol. 10, no. 4, pp. 353–367, Dec. 2018.

[31]   G. I. Sayed, G. Khoriba, and M. H. Haggag, "A novel chaotic salp swarm algorithm for global optimization and feature selection," *Appl. Intell.*, vol. 48, no. 10, pp. 3462–3481, Oct. 2018.

[32]   Z. Zhiliang, L. Shuping, and Y. Hai, "A new approach to generalized chaos synchronization based on the stability of the error system," in *Kybernetika*, 2008, pp. 492–500.

[33]   L. Zhang and C. Zhang, "Hopf bifurcation analysis of some hyperchaotic systems with time-delay controllers," *Kybernetika*, vol. 44, no. 1, pp. 35–42, 2008.

[34]   L. Ding, H. Wu, Y. Yao, and R. Ma, "Chaotic artificial bee colony algorithm for the system identification of a small-scale unmanned helicopter," *Zhendong Ceshi Yu Zhenduan/Journal Vib. Meas. Diagnosis*, vol. 2015, no. Article ID 801874, p. 11 pages, 2016.

[35]   M. Javidi and R. Hosseinpourfard, "Chaos genetic algorithm instead genetic algorithm," *Int. Arab J. Inf. Technol.*, 2015.

[36]   S. Arora and P. Anand, "Chaotic grasshopper optimization algorithm for global optimization," *Neural Comput. Appl.*, vol. 31, no. 8, pp. 4385–4405, Aug. 2019.

[37]   S. Mirjalili and A. H. Gandomi, "Chaotic gravitational constants for the gravitational search algorithm," *Appl. Soft Comput.*, vol. 53, pp. 407–419, Apr. 2017.

[38]   M. Mitić, N. Vuković, M. Petrović, and Z. Miljković, "Chaotic fruit fly optimization algorithm," *Knowledge-Based Syst.*, vol. 89, pp. 446–458, Nov. 2015.

[39]   L. Wang and Y. Zhong, "Cuckoo Search Algorithm with Chaotic Maps," *Math. Probl. Eng.*, vol. 2015, pp. 1–14, 2015.

[40]   B. Alatas, "Chaotic bee colony algorithms for global numerical optimization," *Expert Syst. Appl.*, vol. 37, no. 8, pp. 5682–5687, Aug. 2010.

[41]   P. N. S. K. V. Price, N. H. Awad, M. Z. Ali, "Problem Definitions and Evaluation Criteria for the 100-Digit Challenge Special Session and Competition on Single Objective Numerical Optimization," no. November, p. 22, 2018.

[42] H. M. Mohammed, S. U. Umar, and T. A. Rashid, "A Systematic and Meta-Analysis Survey of Whale Optimization Algorithm," *Comput. Intell. Neurosci.*, vol. 2019, pp. 1–25, Apr. 2019.

[43] T. Liao and T. Stutzle, "Benchmark results for a simple hybrid algorithm on the CEC 2013 benchmark set for real-parameter optimization," in *2013 IEEE Congress on Evolutionary Computation*, 2013, pp. 1938–1944.

[44] I. Loshchilov, "CMA-ES with restarts for solving CEC 2013 benchmark problems," in *2013 IEEE Congress on Evolutionary Computation*, 2013, pp. 369–376.

[45] R. Tanabe and A. Fukunaga, "Success-history based parameter adaptation for Differential Evolution," in *2013 IEEE Congress on Evolutionary Computation*, 2013, pp. 71–78.

[46] S. Gao, Y. Yu, Y. Wang, J. Wang, J. Cheng, and M. Zhou, "Chaotic Local Search-Based Differential Evolution Algorithms for Optimization," *IEEE Trans. Syst. Man, Cybern. Syst.*, vol. 51, no. 6, pp. 3954–3967, Jun. 2021.

[47] A. Salman, I. Ahmad, and S. Al-Madani, "Particle swarm optimization for task assignment problem," *Microprocess. Microsyst.*, vol. 26, no. 8, pp. 363–371, 2002.

[48] H. Hu, Y. Wu, and T. Wang, "A Metaheuristic Method for the Task Assignment Problem in Continuous-Casting Production," *Discret. Dyn. Nat. Soc.*, vol. 2018, no. Article ID 8073648, pp. 1–12, Oct. 2018.

Supplementary Files

This is a list of supplementary files associated with this preprint. Click to download.

- Tables.docx
- Results.docx