FOX: A Fox-inspired Optimization Algorithm

Hardi M. Mohammed (hardi.mohammed@charmouniversity.org)
Charmo University https://orcid.org/0000-0002-9766-9100
Tarik A. Rashid (tarik.ahmed@ukh.edu.krd)
University of Kurdistan Hewler https://orcid.org/0000-0002-8661-258X

Research Article

Keywords: FOX Optimizer, Benchmark Test Functions, Pressure Vessel Design Problem, Economic Load Dispatch Problem

Posted Date: August 9th, 2022

DOI: https://doi.org/10.21203/rs.3.rs-1939478/v1

Cite as: Mohammed, H., Rashid, T. FOX: a FOX-inspired optimization algorithm. *Appl Intell* (2022). https://doi.org/10.1007/s10489-022-03533-0

FOX: A Fox-inspired Optimization Algorithm

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

[*] *Corresponding author email: hardi.mohammed@charmouniversity.org; ORCID of the corresponding author: 0000-0002-9766-9100*

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

[2] *Computer Science and Engineering Department, University of Kurdistan Hewler, Erbil, KRG, Iraq.*

Abstract

This paper proposes a novel nature-inspired optimization algorithm called the Fox optimizer (FOX) which mimics the foraging behavior of foxes in nature when hunting prey. The algorithm is based on techniques for measuring the distance between the fox and its prey to execute an efficient jump. After presenting the mathematical models and the algorithm of FOX, five classical benchmark functions and CEC2019 benchmark test functions are used to evaluate its performance. The FOX algorithm is also compared against the Dragonfly optimization Algorithm (DA), Particle Swarm Optimization (PSO), Fitness Dependent Optimizer (FDO), Grey Wolf Optimization (GWO), Whale Optimization Algorithm (WOA), Chimp Optimization Algorithm (ChOA), Butterfly Optimization Algorithm (BOA) and Genetic Algorithm (GA). The results indicate that FOX outperforms the above-mentioned algorithms. Subsequently, the Wilcoxon rank-sum test is used to ensure that FOX is better than the comparative algorithms in s statistically significant manner. Additionally, parameter sensitivity analysis is conducted to show different exploratory and exploitative behaviors in FOX. The paper also employs FOX to solve engineering problems, such as pressure vessel design, and it is also used to solve electrical power generation: economic load dispatch problems. The FOX has achieved better results in terms of optimizing the problems against GWO, PSO, WOA, and FDO.

Keywords:

FOX Optimizer, Benchmark Test Functions, Pressure Vessel Design Problem, Economic Load Dispatch Problem.

Acknowledgments

The authors wish to thank Charmo University, Sulaimani Polytechnic University, and the University of Kurdistan Hewler (UKH).

Declarations:

Declarations of interest: none

Funding: This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

Code availability: the code of the paper will be published at https://github.com/Hardi-Mohammed/FOX

Cite as: Mohammed, H., Rashid, T. FOX: a FOX-inspired optimization algorithm. *Appl Intell* (2022). https://doi.org/10.1007/s10489-022-03533-0

1. Introduction

Currently, meta-heuristic optimization algorithms are widely used to solve real-world application problems [1]. These algorithms are predominantly inspired by biological or physical behavior. Algorithms that are inspired by physical behavior include space gravitational optimization (SGO) [2] and the gravitational search algorithm (GSA) [3]. These algorithms are derived from the theory of relativity, the law of gravity, and interactions between objects.

Consequently, different metaheuristic algorithms have been proposed by researchers since 1948, when Alan Turing broke the code of the Enigma cipher machine. After the idea of Turing's heuristic method, the genetic algorithm (GA)[4] was proposed, which mimics natural evolution. After the GA was proposed, many algorithms have been developed, such as simulated annealing (SA) [5], tabu search [6], ant colony optimization (ACO) [7], particle swarm optimization (PSO) [8], and differential evolution (DE) [9]. Some of the recent algorithms in this family are dragonfly optimization algorithm (DA) [10], whale optimization algorithm (WOA) [1], grey wolf optimization (GWO) [11], butterfly optimization algorithm (BOA) [12], cat swarm optimization (CSO) [13], the fitness-dependent optimizer (FDO) [14], [15], the donkey and smuggler optimization algorithm (DSO) [16] and the learner performance-based behavior algorithm (LPB) [17], K-means GWO [18], chimp optimization algorithm (ChOA) [19]. Each algorithm has different inspirations and operators. For example, GA uses three basic operators: selection, crossover, and mutation [20]. The GA and its operators are used by researchers to solve optimization problems. Genetic algorithm is quite efficient but the problem presentation is always a challenge for researchers due to the use of discrete chromosomes. GA also has premature convergence and it has many parameters to set, such as population size, crossover rate, mutation rate, and the selection method [21][22].

PSO [23] was proposed by Kennedy in 1995 that mimics the behavior of swarm particles. Each particle represents a solution in the search space [24]. PSO uses the global best position of the particle and the local best position of particles to calculate new velocity. The new velocity is based on the memory of the previous particles and the neighbor's experience. Then, the new position of the particle is found based on the previous position and the velocity [25]. Despite using PSO for solving problems in different areas including neural network training, estimation, and face recognition, it has disadvantages of falling into local optima and having low convergence when dealing with high dimensional problems as discussed in [26], [27], [28].

DA is a meta-heuristic algorithm inspired by the static and dynamic swarming behavior of dragonflies in nature [29]. DA is using these behaviors either for migration or hunting. DA has three basic operators to explore and exploit food sources. These operators are separation, alignment, and cohesion [10]. DA had been used in different areas to solve real-world applications: image processing, machine learning, and networking. Results proved that DA outperformed against PSO for solving multidimensional functions. DA is also superior to PSO while it is evaluated on CEC2019 benchmark functions. According to [10], DA achieved competitive results against common algorithms. DA also has drawbacks due to trapping into local optima and premature convergence. Using levy flight also makes more interruption in the search space due to the implementation of big steps. It also uses more time in the exploitation phase.

After DA, another algorithm was proposed which is called WOA. Humpback whales are the source of inspiration for this algorithm proposed in 2016 [1]. WOA consists of different methods to catch prey. Firstly, it uses an encircling mechanism to create a search space around the prey. Then, it uses the bubble net attacking method to shrink the circle and update whales' position toward the prey. The exploratory method is activated due to discovering better prey based on a random walk and best positions [30]. WOA had been used to solve many applications in different fields, such as electronic engineering, economic scheduling, civil engineering, fuel and energy, and medical engineering. Such diverse application areas demonstrated that WOA achieved high performance in terms of exploration and exploitation. WOA also has a high chance of escaping local optima by balancing both the exploration and exploitation phases. WOA has limitations regarding convergence speed and achieving optimum results. One of the problems is the randomization methods in WOA, these methods increase the computational time when it deals with a problem that has many dimensions. Having poor convergence is another disadvantage of WOA because both exploration and exploitation phases depend on a single variable. Besides, WOA tends to be trapped into local optima in large-scale problems because it uses an encircling mechanism[30].

BOA is inspired by the butterflies foraging behavior. This behavior is responsible for moving the butterflies towards the food position. BOA was proposed in 2018 [12]. Butterflies discover the direction of the food source via their senses and analyze the smell of the food in the air. BOA consists of three phases which are the initialization phase, iteration phase, and final phase. The searching process is done in the iteration phase while the best position is found the final face is terminated [12]. BOA has been used to solve different applications: spring design, gear train design, and welded beam design problems. It is also used to solve the green vehicle routing problem. Results proved that BOA obtained high performance against PSO, Artificial Bee Colony (ABC), and Differential Evolution (DE) [12]. Despite having better performance, BOA has difficulties due to low convergence [31].

FDO is a recent meta-heuristic algorithm that was inspired by the reproduction process of bees while searching for better hives [14]. The main aim of FDO is how scout bees try to find better food sources among various hives. FDO works based on two parts which are the pace and positions of the scout bees. The pace is the velocity of the scout bees toward the food source. Therefore, the updating positions of the scout bees are based on the new velocity and the previous position of the scout [32]. FDO is a state-of-the-art algorithm used to solve different applications: antenna design, economic load dispatch problems, and power systems [33]. It achieved superior results compared to other common algorithms: GA, PSO, DA, ChOA, WOA. FDO has an $O(n)$ time and space complexities. Despite achieving these results, there are still chances for improving FDO in terms of improving solutions [32], [15]. In addition, slow convergence and not having a proper balance between exploration and exploitation are disadvantages of FDO [34]. Therefore, it can be said that active nature-inspired algorithms entail being proposed by researchers to solve the issues of the state-of-the-art algorithm.

Chimp Optimization Algorithm (ChOA) is a recent metaheuristic algorithm that mimics the behavior of chimps while they are hunting in a group proposed in 2020 [19]. It works based on individual intelligence and the sexual behavior of chimps. Therefore, four types of chimps are used in ChOA which are attacker, barrier, chaser, and driver. These types of chimps are responsible for exhibiting diverse intelligence. ChOA also has four main steps to hunt preys which are driving, chasing, blocking, and attacking [35]. ChOA has been used to solve many applications. According to [19] ChOA was used to solve ten design problems. ChOA achieved the best results against GA, PSO, and GWO for solving benchmark functions. However, it has limitations due to the transition between the exploration and exploitation phases [36]. It also has issues solving binary problems [37] and the solution requires more enhancement [35].

Lion Optimization Algorithm (LOA) was inspired by lion's behavior and social interaction. LOA use social mechanism and encircling mechanism to hunt prey [38] while TIGER algorithm uses eight different technics to catch prey such as birth and breed, migration of children, migration, death, suicide, prowl, mating, and signing [39].

These metaheuristic algorithms mentioned above are used to optimize real applications in various fields, such as structural optimization [18], [40], scheduling and routing [41], software testing [42], image processing [43], and data mining [44]. However, metaheuristic algorithms may have issues related to their poor search capability and falling into local optima [45], [46].

For example, the WOA falls into local optima when confronting high-dimensionality problems because of having only an encircling mechanism in the exploration phase [47]. Therefore, this problem has been solved by using a new method, which is used to update the position and enhance the capability of the exploration phase [48]. In [47], chaotic local search is combined with the WOA to enhance the capacity of the WOA to overcome local optima.

GWO is another example that falls into local optima when solving high-dimensional problems and complex multimodal problems [49], [50]. However, this problem is solved by using weight variables for different types of wolves in GWO. By using the weight variables and changing the control parameter a, GWO can achieve the maximum admissible error (MAE), which is required in engineering optimization problems [49].

The bat algorithm (BA) also has difficulties with some multidimensional functions because it can be trapped in local optima. Thus, this problem is solved by modifying the exploitation phase by changing the dimensional size to a dynamic dimension and adding an inertia weight [50].

Additionally, having an improper balance between the exploration and exploitation phases is problematic in WOA [51], [52]. As a result, an improved WOA is proposed in [53] and hybridized with DE's mutation operator. The modified WOA is called improved WOA (IWOA). The mutation operator is integrated inside the exploration phase of the WOA, and the exploitation phase of the IWOA is the same as that of the WOA. Besides, in [51], researchers solved this issue by hybridizing GWO with the WOA to improve the performance of the WOA and achieve a better balance between the two phases. Like WOA, GWO has the same problem. Consequently, researchers proposed improved GWO combined with DE to achieve a proper balance between the phases [54]. Therefore, the position of the wolves is updated based on the principle of the survival of the fittest (SOF). After adding the operation of DE, the position is updated in GWO. Similar to GWO, the BA also has this problem. Thus, an inertia weight is added to update the velocity of the bat to improve the efficiency of the BA in terms of balancing both phases [55], [51]. The inertia weight dynamically decreases throughout the iterations. It has a high value, in the beginning, to allow the BA to perform a global search. Decreasing the value allows the BA to have better local searchability.

It should be noted that there are other fox-inspired algorithms in the literature. However, none of them used the solo hunting strategy modeled in this work. To the best of our knowledge, there are two papers published in the literature that attempted to propose an optimization algorithm inspired by fox's behaviours. One of them was published in 2016 under the name fox hunting algorithm (FHA) [56] and the other was published in 2021, which is called the red fox optimization algorithm (RFO) [57]. FHA mimics the hunting behavior of the hunters who tries to hunt the red fox by using horses and hounds. Therefore, the FHA is related to the way the hunter wants to catch the fox. The RFO also uses the behavior of searching and hunting mechanism of red fox. However, RFO uses the reproduction mechanism and leaves behind the herd by red fox. RFO also uses a sorting mechanism inside the exploration phase while FOX's exploration is based on the best position and random search. Despite proposing both FHA and RFO, FOX is different from them because it uses the behavior of red fox for searching and hunting in the snow environment. It also mimics the jumping technic in order to dive into the snow to hunt the prey. In the exploitation phase, distance of red fox is measured by the prey and then it estimates the jump technic in order to dive and hunt the prey. FOX also requires an artificial fox to jump in northeast direction and its opposite. Therefore, the new position is found based on the distance of the prey, jump value, and the direction range value.

The main aim of this paper is to overcome the stagnated local optima problem and not have the proper balance between exploration and exploitation phases. Therefore, the proposed FOX algorithm effectively balances the exploration and exploitation phases better than its alternatives. FOX is evaluated using CEC2019 benchmark functions. It can also be used to solve various optimization problems. Additionally, decreasing a specific parameter provides support for avoiding local optima. Using the exploration phase based on the best position and average time variable has a great impact on not trap into local optima.

Therefore, the main contributions of this paper are:

1. Proposing the FOX optimization algorithm based on fox hunting behavior.
2. Solving the pressure vessel design problem and economic load dispatch problem.
3. Using a unique technique in the exploration phase that can be adapted to propose new algorithms or modify algorithms.

The rest of this paper is organized as follows: Section 2 describes fox life, and fox behavior, and then, the FOX is described in detail in Section 3. Section 3 describes the results and gives a discussion of the FOX, and then the pressure vessel design problem and economic load dispatch problem are explained and solved by the FOX. Finally, the conclusion and future work are presented.

2. Inspirations, mathematical models, and the algorithm of FOX

In this paper, a novel red fox in the snow optimization algorithm is proposed which is inspired by the hunting behavior of red fox. The algorithm uses both the searching and hunting techniques of the red fox. The following subsections present red fox life, red fox behavior in the snow, and the FOX algorithm.

2.1 Fox life

A fox can live in a species-poor environment that has low production [58]. Foxes have different colors, but the most common colors are white and red. Two common types of foxes are the red fox and arctic fox. The red fox is the most widespread animal, and it has colonized urban areas in the USA, Europe, Canada, Japan, and Australia [39, 40]. The arctic fox has a strong ability to hunt its prey either from above or below. Fox diets include geese and lemmings in North America. However, a large part of the fox diet is lemmings in Fennoscandia, Siberia. The fox also lives on islands such as the Pribilof Island and Svalbard. On these islands, the fox focuses on resources in the sea, such as fish and birds [59]. According to [59], foxes that live on these islands have acquired a mouth that opens wider to catch more prey. The main difference between these two foxes is that Arctic foxes live at lower temperatures compared to red foxes. Arctic foxes have difficulties living in their habitat because of human interference [60].

The red fox is an example of a fox that eats mammals and plants. It lives between approximately 2 to 4 years, and the weight of this animal is between 6.5 and 24 pounds [61]. It lives in diverse locations, such as grassland, forest, desert, and mountains [62].

By being an omnivore, the fox can adapt itself to its habitat in terms of food resources. Consequently, the fox eats vegetables, fruit, frogs, and fish [63]. However, the arctic fox eats lemmings and geese [64]. Foxes use intelligent methods to hunt their prey [65].

2.2 Red fox behavior in snow

The FOX algorithm mimics the hunting behavior of a red fox when it dives into the snow to hunt its prey. The main techniques are based on the red fox trying to hunt the best prey [64], [66]. The steps are described below:

1. When snow covers the ground, the snow blocks the sight of the prey. Red fox tries to search for prey randomly.
2. Consequently, the red fox finds the prey by hearing ultrasound from the prey. Then, it takes time to get to close the prey.
3. By listening to the sound of the prey and the time difference, the red fox can determine the distance between itself and the prey.
4. After determining the distance, the red fox estimates the jump that is needed to catch the prey.
5. Walking is performed randomly according to the minimum time and best position.

In the beginning, the red fox moves randomly in the search space to discover prey. It can find prey via hearing ultrasound of the prey. This random walk is what we inspired and use for providing exploratory behavior in FOX. While the red fox searches, it may hear the sound of the prey. After hearing the sound, the red fox is in the exploitation phase. The red fox can hear ultrasound, so the sound of the prey takes time to reach the red fox. The distance sound travels can be calculated by multiplying it by the speed of sound travel in the air, which is 343 meters per second. However, because of having static value, a different method is used to investigate the speed of sound travel in the air. The red fox tries to move forward the prey decides to jump against the prey. Therefore, the red fox tries to jump depending on the time the sound takes to catch prey. A study showed that a fox prefers to jump northeast based on the magnetic alignment effect [66]. If it jumps in the northeastern direction, then the chance of killing its prey is 82%. However, if it jumps in the opposite direction, then the chance of catching the prey is 18%. Consequently, it can be said that there are two to catch the prey. Fig. 1 shows the hunting behavior of the red fox.

Cite as: Mohammed, H., Rashid, T. FOX: a FOX-inspired optimization algorithm. *Appl Intell* (2022). https://doi.org/10.1007/s10489-022-03533-0

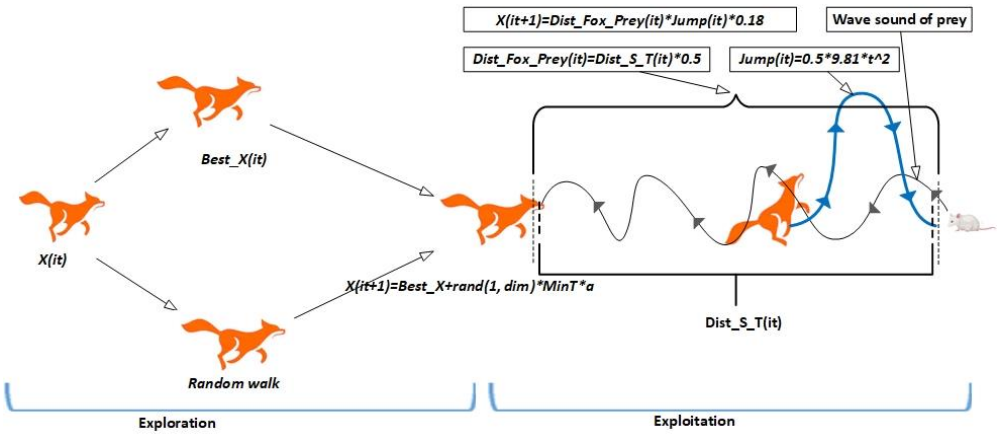


Fig. 1 Red Fox Hunting Behavior: Exploration and Exploitation

2.3 The FOX optimization algorithm

In the beginning, FOX initializes the population, which is called the *X* matrix. An *X* is the position of red foxes. Then, the *fitness* of each search agent is calculated by using standard benchmark functions in each iteration. To investigate the *BestFitness* and best position (*BestX)*, the *fitness* value of each search agent (each row in an *X* matrix) is compared to the fitness of other agents (other rows). *BestFitness* and *BestX* are conducted using a condition that is used to compare the fitness of the new row ($fitness_{i+1}$) and fitness of the previous row($fitness_i$) throughout iterations are returned.

Then, to balance the exploration and exploitation phases, a condition is used with a random variable. This variable aims to divide the phases equally in terms of the number of iterations. Using a random variable called *r, we assign 50% probably to do either exploration or exploitation in FOX.* In other words, nearly half of the iterations are used for exploration and the other half for exploitation. The proposed algorithm is essential in terms of balancing and avoiding local optima. Thus, a condition statement is used to divide the iterations equally for both exploration and exploitation. The variable *a* is used to decrease the search performance according to the *BestX*, and after each iteration, the value is decreased, which means that the agent pursues the prey better in every iteration. In addition to providing the condition to update the position, the fitness value affects the search agents to avoid local optima, because if the new position does not change, the exploration phase deactivates to allow other phases to be activated. The following two sections explain exploitation and exploration.

2.3.1 Exploitation

In the exploitation phase, we have a condition relating to the chance of killing the prey as mentioned in Section 2.2. The value of the random variable *p* is in the range of [0,1]. Therefore, if the random number *p* is greater than 0.18, the new position of the red fox needs to be found. To find a new position, the distance sound travels $Dist\_S\_T_{it}$, the distance of the red fox from the prey $Dist\_Fox\_Prey_{it}$ and jumping value $Jump_{it}$ must be calculated. Consequently, we generate a random number ranging from 0 to 1 for the sound travel time $Time\_S\_T_{it}$. The distance of the sound from the red fox is found by multiplying the speed of sound in the air Sp_S with time sound travels time $Time\_S\_T_{it}$ [67]:

$$Dist\_S\_T_{it} = Sp\_S * Time\_S\_T_{it}, \quad (1)$$

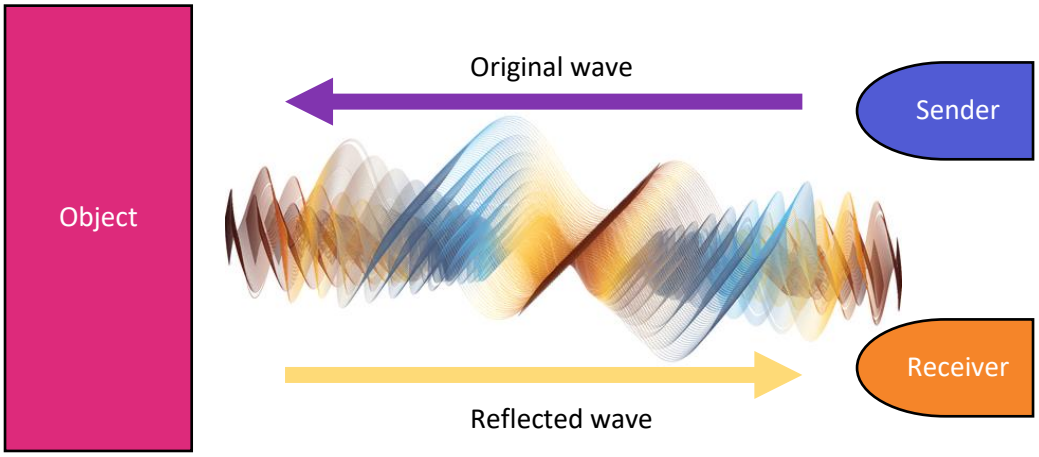
where the speed of sound in the medium Sp_S is equal to 343 in the air and $Time\_S\_T_{it}$ is a random number in the range [0, 1]. Then, *it* is the number of iterations, which ranges from 1 to 500. However, another equation is constructed to find Sp_S, which is based on the best position found so far by dividing the time sound takes to travel between the fox and prey. $Time\_S\_T_{it}$ is a random and $BestPosition_{it}$ is the best search agent in the population. Equation (2) shows how to find the speed of the sound Sp_S based on the best position.

Cite as: Mohammed, H., Rashid, T. FOX: a FOX-inspired optimization algorithm. *Appl Intell* (2022). https://doi.org/10.1007/s10489-022-03533-0

$$Sp\_S = \frac{BestPosition_{it}}{Time\_S\_T_{it}} \tag{2}$$

To find the distance sound travels, equation (1) is used, which means that distance ($Dist\_S\_T_{it}$)= velocity (Sp_S) ∗ time ($Time\_S\_T_{it}$). Consequently, the distance of the fox from the prey; $Dist\_Fox\_Prey_{it}$ can be calculated by halving $Dist\_S\_T_{it}$. Therefore, to calculate the distance between sensor and object in physics, the distance of sound travels is divided by 2 since the distance between the sensor and the object is one-half of the distance that has been found by the sound wave [67]. Fig. 2 presents the sensor, which sends a sound wave signal to an object and then receives the signal. Therefore, the sensor multiplies the distance of the sound travels time by 0.5 or ½. Both multiplying by 0.5 or dividing by ½ are the same because half of the sound wave is considered [67]. So equation (3), is taken from techniques of sensors and objects to find the distance from Fig. 2:

$$Dist\_Fox\_Prey_{it} = Dist\_S\_T_{it} * 0.5 \tag{3}$$



s

Fig. 2 Calculating distance by ultrasound sensor [67]

After finding the distance between a fox and the prey, the red fox needs to find a new position so that the red fox requires to jump to catch the prey. Therefore, the fox needs to calculate the jump height $Jump_{it}$. Thus, $Jump_{it}$ can be calculated by the following equation:

$$Jump_{it} = 0.5 * 9.81 * t^2 \tag{4}$$

where 9.81 is the acceleration due to gravity and t is equal to the average time that sound takes to travel and it is squared because of the up and down steps in the jump. The time transition tt value is calculated by dividing the sum of the $Time\_S\_T_{it}$ to dimensions. Equation (7) shows the tt and $MinT$ calculation. Then average time t is found by dividing tt by 2. The average time and gravity are multiplied by 0.5 because the jump value takes two different times to go up and down. As a result, both gravity and average time must be multiplied by 0.5. Then, the *Jump* value is multiplied by $Dist\_Fox\_Prey_{it}$ and c_1. The variable $c_1$has a range, which is within [0, 0.18] when the red fox jumps

to the northeast direction. The equation below shows the calculation of the new position of the red fox if the *p,* which random number in the range [0,1], is greater than 0.18.

$$X_{(it+1)} = Dist\_Fox\_Prey_{it} * Jump_{it} * c_1 \quad (5)$$

Both Equations (5) and (6) are used to find a new location for the red fox. Only one of them is executed in each iteration because of using the *p* condition. However, the only difference is in the second part of the *p* condition in Equation (5). If the else condition of the *p* is different, the equation (5) is multiplied by c_2 instead of c_1 if the value of *p* is less than or equal to 0.18. Therefore, if the *p* is greater than 0.18, then the new position is calculated by equation (5). However, if the value is less than 0.18, then the new position is calculated by equation (6). The range of c_2 is within [0.19, 1].

$$X_{(it+1)} = Dist\_Fox\_Prey_{it} * Jump_{it} * c_2 \quad (6)$$

The value of c_1 *and* c_2 are 0.18 and 0.82 respectively. These values are based on the jump movement of a red fox, which either jumps to the northeast or the opposite. Therefore, if the *p* value is greater than 0.18, it means that the red fox jumps to the northeast direction. Consequently, to find a new position both $Dist\_Fox\_Prey_{it}$ and $Jump_{it}$ are multiplied by c_1. Accordingly, chances of exploiting a new position are great and the red fox goes toward global optima. However, the red fox jumps in the opposite northeast direction, if the *p* value is less than 0.18, this means that the chances of killing prey are low (%18). Hence, both $Dist\_Fox\_Prey_{it}$ and $Jump_{it}$ are multiplied by c_2.

2.3.2 Exploration

To control the random walk, the fox searches randomly in this phase according to the best position of the fox that has been found so far. The fox does not have a jumping technique in this phase because it has to walk randomly to explore prey in the search area. To ensure that the fox walks randomly toward the best position, a minimum time variable $MinT$ and the variable *a* are used to control the search. Equations (7) and (8) show the calculation of the $MinT$ and variables. *MinT* is calculated by finding the minimum of *tt*.

$$tt = \frac{sum\left(Time_{S\_T_{it}}(i,:)\right)}{dimension}, \qquad MinT = Min(tt) \quad (7)$$

Summation of $Time\_S\_T_{it}\ (i,:)$ is divided by the dimension of the problem to find the minimum time average tt.

$$a = 2 * (it - \left(\frac{1}{Max_{it}}\right)) \quad (8)$$

where Max_{it} , is the maximum iterations. Calculating both $MinT$ and *a* variable has a vital effect on the search phase to move toward the solution that is close to the best solution. Using $rand(1, dimension)$ to ensure that the fox walks stochastically to explore the prey. However, to strengthen the searchability of FOX, both $MinT$ and *a* variable are used. Variable *r,* which is a random number, is also used to balance the exploration and exploitation phases. The best solution $BestX_{it}$ that has been found so far has a great impact on the exploration phase. Equation (9) shows the exploration technique of the fox in searching for a new position in the search space $X_{(it+1)}$. The equations in this phase

can be adapted to existing algorithms to enhance their performance. They can also be used to propose new metaheuristic algorithms.

$$X_{(it+1)} = BestX_{it} * rand(1, dimension) * MinT * a \quad (9)$$

The equations in both phases do not need any modification rather than adaptation to a specific problem while FOX is used to solve the multidimensional space problem. The details of the FOX can be seen in Algorithm 1 and Figure 3.

Algorithm 1 FOX optimization algorithm

1: Initialize the red fox population $\boldsymbol{X}$ *i (i=1, 2, ..., n)*
2: **While it<Maxit**
3: Initialize $\boldsymbol{Dist\_S\_T}$, $\boldsymbol{Sp\_S}$, $\boldsymbol{Time\_S\_T}$, $\boldsymbol{BestX}$, $\boldsymbol{Dist\_Fox\_Prey}$, $\boldsymbol{Jump}$, $\boldsymbol{MinT}$, *a, BestFitness.*
4: Calculate the *fitness* of each search agent
5: Select $\boldsymbol{BestX\ and\ BestFitness}$ **among the fox population** (*X)* in each iteration.
6: **If1** $\boldsymbol{fitness_i > fitness_{i+1}}$
7: *BestFitness=*$\boldsymbol{fitness_{i+1}}$
8: *BestX=X(i, :)*
9: **Endif1**
10: **If2** *r >= 0.5*
11: **If3** *p* >0.18
12: Initialize time randomly;
13: Calculate Distance_Sound_travels using Eq. (1)
14: Calculate *Sp_S* from Eq. (2)
15: Calculate distance from fox to prey using Eq. (3)
16: Tt= average time;
17: T=Tt/2;
18: Calculate jump using Eq. (4)
19: Find $\boldsymbol{X_{(it+1)}}$ using Eq. (5);
20: **Elseif** *p <=0.18*
21: Initialize time randomly;
22: Calculate Distance_Sound_travels using Eq. (1)
23: Calculate *Sp_S* from Eq. (2)
24: Calculate distance from fox to prey using Eq. (3)
25: Tt= average time;
26: T=Tt/2;
27: Calculate jump using Eq. (4)
28: Find $\boldsymbol{X_{(it+1)}}$ using Eq. (6);
29: **EndIf3**
30: **else**
31: Find MinT using Eq. (7)
32: Explore $\boldsymbol{X_{(it+1)}}$ using Eq. (9)
33:
34: **EndIf2**
35: Check and amend the position if it goes beyond the limits
36: Evaluate search agents by their fitness
37: Update *BestX*
38: *it=it+1*
39: **End while**
40: return *BestX & BestFitness*

Cite as: Mohammed, H., Rashid, T. FOX: a FOX-inspired optimization algorithm. *Appl Intell* (2022). https://doi.org/10.1007/s10489-022-03533-0

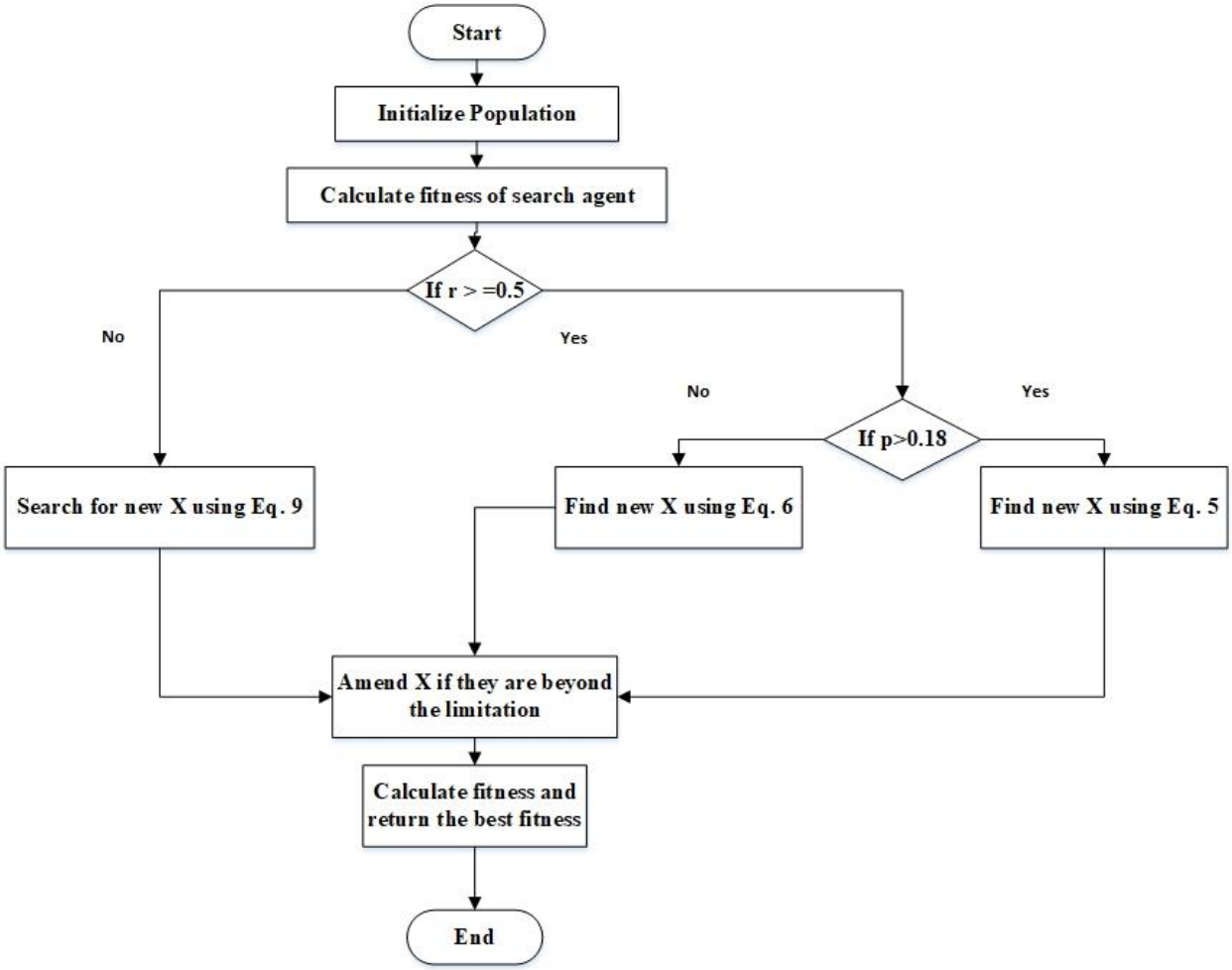


Fig. 3 Flowchart of FOX

Overall, FOX begins by initializing the population of the red fox randomly. Then, in the first iteration, the population is checked whether the position of each red fox is inside the boundary of the benchmark function or not. After that, the fitness value of the benchmark function is calculated based on the row of the population. Following these steps, the *BestFitness* value and the best position (*BestX)* are selected. Then, a condition is started by comparing the random number *r.* If it is greater than or equal to 0.5, then the exploitation phase is activated. Also, a *p* condition exists inside the exploitation phase. If the *p* value is greater than 0.18, then the new position of the red fox is found based on Equations (1), (2), (3), (4), and (5). However, if *p* value is less than or equal to 0.18, then the new position is calculated based on Equations (1), (2), (3), (4), and (6). If *r is* less than 0.5 in the else condition, the exploration phase is activated. Therefore, the new position is found based on the best position, random number, and multiplying *MinT* variable by variable *a*. Thus, the *BestFitness* is returned in the first iteration. After modifying the population, the same steps are done repeatedly in the second iteration to find the best fitness and best position.

Regarding the computational complexity of FOX: each iteration, has a time complexity of *O(SearchAgents * D * it)* where *SearchAgents* is the population size, *D* is the dimension of the problem, and *it* is the number of iterations. Therefore, it can be said that FOX has an $O(n^2)$ as time complexity. Also, the space complexity of FOX is calculated based on the vectors and matrices that can be seen in Algorithm 1. So, FOX has an $O(n^2)$ space complexity for each iteration.

Cite as: Mohammed, H., Rashid, T. FOX: a FOX-inspired optimization algorithm. *Appl Intell* (2022). https://doi.org/10.1007/s10489-022-03533-0

3. Results and Discussion

To ensure that the proposed algorithm works well, it must be tested by using different benchmark test functions. Five classical benchmark functions and the recent benchmark test functions are used to evaluate FOX performance. Besides, the results of the proposed algorithm are compared to Butterfly Optimization Algorithm (BOA), DA, ChOA, FDO, WOA, and GA. Then, these results are statistically compared to each other to determine whether the result is significant. Therefore, to ensure that the FOX works well to solve real-world applications, the FOX is used to solve the pressure vessel design problem and economic load dispatch problem.

3.1 Classical benchmark functions

Five different classical benchmarks were chosen to evaluate the performance of FOX [68]. These benchmark functions are divided into unimodal and multimodal functions. Unimodal functions are better for evaluating the exploitation phase because they have one local optimum. However, multimodal functions are used for the exploration phase due to having more local optima [69]. Table 1 illustrates both unimodal and multimodal functions.

Table 1 Five Classical Benchmark Functions.

No.	Function	Dimension	Search range	*fmin*
Unimodal Benchmark Functions				
1	$f_1(x) = \sum_{i=1}^{n} x_i^2$	30	[-100, 100]	0
2	$f_5(x) = \sum_{i-1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	30	[-30, 30]	0
Multimodal Benchmark functions				
3	$f_8(x) = \sum_{i=1}^{n} -x_i \sin\left(\sqrt{\lvert x_i \rvert}\right)$	30	[-500, 500]	-418.9829×5
4	$f_9(x) = \sum_{i=1}^{n} [x_i^2 - 10\cos(2\pi x_i) + 10]$	30	[-5.12, 5.12]	0
5	$f_{11}(x) = \frac{1}{4000} \sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	30	[-600, 600]	0

FOX algorithms were executed 40 times with 100 iterations and then compared to five different algorithms, such as DA, FDO, and WOA. Table 2 presents results that are conducted from papers [13], [14] and the achieving results of FOX. As a result, FOX obtained better exploitation ability in the first unimodal function while WOA achieved optimum results in the second unimodal function. FOX also outperformed well in functions 3 and 4. However, FDO achieved the third rank which has a superior result in only function 5. Regarding the three multimodal functions, FOX also outperformed the other algorithms in two functions and it means that FOX has greater exploration ability because of dividing nearly half of the iteration for the exploration phase and the other half for the exploitation phase.

Table 2 Comparison Results of Five Classical Benchmark Functions for FOX.

F	BOA		DA		ChOA		FDO		WOA		FOX	
	Average	**Std**	**Average**	**Std**	**Average**	**Std**	**Average**	**Std**	**Average**	**Std**	**Average**	**Std**
F1	1.01E−11	1.66E−12	2.85E-18	7.16E-18	6.8573E-49	0.00000003	7.47E-21	7.26E-19	1.41E−30	4.91E−30	**0**	**0**
F2	8.9355	0.0215	7.6005	6.7864	27.1546	0.0016241	23.5010	59.7883	**0.072581**	0.39747	38.4337	0.082471
F3	28.68	20.178	16.0188	9.4791	5.6843E-14	0.0012031	14.5654	5.2022	**0**	**0**	**0**	**0**
F4	1.35E−13	6.27E−14	0.1933	0.0734	**0**	0	0.5687	0.1042	0.000289	0.001586	**0**	0
F5	NA	NA	-2857	383.64	−3628.8022	5.1249	**-2285207**	206684	−5080.76	695.7968	-6097.8	387.2942

3.2 CEC2019 benchmark test functions

This modern benchmark suite includes 10 functions that were developed at the CEC conference[70]. These ten functions were invented for an annual optimization competition called "The 100-Digit Challenge". The dimensionalities of these functions are different, and all of them are scalable. Functions CEC01 to CEC03 have different dimensionalities [14]. However, the other functions have the same dimensionality, which is [-100,100]. Table 3 presents the CEC2019 functions.

Table 3 CEC2019 Benchmark Test Functions with Their Dimensions and Search Ranges: CEC2019 consists of Ten Benchmark Functions.

No.	*Functions*	$F_i^* = F_i(x^*)$	*D*	*Search Range*
1	*Storn's Chebyshev Polynomial Fitting Problem*	1	9	[-8192, 8192]
2	*Inverse Hilbert Matrix Problem*	1	16	[-16384, 16384]
3	*Lennard-Joes Minimum Energy Cluster*	1	18	[-4, 4]
4	*Rastrigin's Function*	1	10	[-100, 100]
5	*Griewangk's Function*	1	10	[-100, 100]
6	*Weierstrass Function*	1	10	[-100, 100]
7	*Modified Schwefel's Function*	1	10	[-100, 100]
8	*Expand Schaffer's F6 function*	1	10	[-100, 100]
9	*Happy Cat Function*	1	10	[-100, 100]
10	*Ackley Function*	1	10	[-100, 100]

The FOX is compared with GWO, ChoA, GA, and FDO [13]. These algorithms are selected because they are very competitive and are commonly used to solve benchmark functions and real-world problems. Therefore, the FOX is

implemented in MATLAB code, which can be found at https://github.com/Hardi-Mohammed/FOX, and 30 agents are used by each algorithm to search in the landscape. This search is performed using 500 iterations. The results are determined after running each algorithm 30 times, and then the average results are used for comparison. The results of other algorithms such as FDO, BOA, and ChoA are conducted from [13], [71], [19]. The algorithms that are used in this paper have used the following parameters as can be seen in Table 4.

Table 4 Parameter Setting.

Parameters	**Numbers**
Iterations	500
Run of Algorithms	30
Search agents	30
Dimension	Based on Table 3

Table 5 shows that the FOX outperforms all other algorithms on four out of 10 functions except FDO. However, the GA and GWO have the second rank among the six algorithms, both GA and GWO have the best results in only three functions. ChoA and BOA do not have great results against FOX, GA, and GWO. Therefore, ChOA and BOA have the worst result compared to other algorithms.

In addition, FOX has superior results in 9 out of 10 functions compared to BOA, while it outperforms well in 8 functions against ChOA. FOX also achieved the best results compared to GA in 7 functions. However, GWO has competitive results, so FOX is great in only five functions.

FDO is a new metaheuristic algorithm and it achieved the best results. Therefore, FOX was also compared with FDO in Table 5, results proved that FOX obtained well in only two functions while FDO achieved optimum results in 7 functions. Thus, FOX can explore efficiently compared with other algorithms and avoid local optima. It can also better exploit the best solution than the other algorithms in four functions.

Table 5 Comparison of the GWO, ChOA, GA, BOA, FDO, and the FOX using 10 Functions from the CEC2019 Functions. Results were conducted using 500 iterations with 30 runs.

F	GWO		ChOA		GA		BOA		FDO		FOX	
	Avg.	**STD**	**Avg.**	**STD**	**Avg.**	**STD**	**Avg.**	**STD**	**Avg.**	**STD**	**Avg.**	**STD**
1	2.13 E+08	3.07 E+08	4.24E+ 09	9.67E+ 09	5.32 E+04	7.04 E+04	5.89E+ 04	1.14E+ 04	1.66E+ 09	1.54E+ 09	**2.58E+ 04**	22491. 27
2	18.343 9	0.0003 04	18.408 3	0.0185 8	17.35 02	17.349 1	1.89E+ 01	2.91E- 01	**17.342 9**	1.07E- 14	18.344 2	0.0004 25
3	13.702 4	7.23 E-15	13.702 4	7.11E- 06	**12.70 24**	13.702 4	1.37E+ 01	6.17E- 04	12.702 4	5.33E- 15	13.702 5	7.11 E-15
4	300.98 15	686.81 53	5932.6 2	2855.2 07	6.23 E+04	61986. 61	2.09E+ 04	7.71E+ 03	**44.489 9**	21.525	1.06E+ 03	835.24 78
5	2.4313	0.2516 07	4.2094	0.8873	7.539 6	7.2765	6.18E+ 00	7.08E- 01	**1.1916**	0.1669 77	5.3148	0.8000 19
6	11.939 4	0.7307 45	12.154 4	0.6826	7.400 5	6.6877	1.18E+ 01	7.71E- 01	10.703 3	1.0345 59	**5.0325**	1.4994 43
7	534.76 55	292.02 04	1007.1 3	179.01 66	791.7 42	697.89 64	1.04E+ 03	2.15E+ 02	**294.04 8**	171.95 19	306.79 4	141.85 88
8	**5.4021**	0.9939 56	6.7846	0.1562	6.100 4	5.8228	6.34E+ 00	3.59E- 01	**5.1731**	0.6685 29	5.4621	0.3819 07
9	14.732 8	49.951 42	449.27	245.49	5.31 E+03	5.29 E+03	2.27E+ 03	8.11E+ 02	**2.5578**	0.1645 43	3.7959	0.4408 76
1 0	21.497 3	0.0685 13	21.498 5	0.0719 5	**20.10 59**	20.023 6	2.15E+ 01	7.95E- 02	**19.337**	3.5913 28	20.981 8	0.0110 96

Cite as: Mohammed, H., Rashid, T. FOX: a FOX-inspired optimization algorithm. *Appl Intell* (2022). https://doi.org/10.1007/s10489-022-03533-0

3.3 Statistical analysis

To evaluate the performance of the algorithm, statistical analysis is required to determine whether it is significant or not. Therefore, the Wilcoxon rank-sum test is used to check the *p*-value of the results. Table 6 shows that FOX is very competitive against all the algorithms. The FOX has a significant value in 7 functions against the GA and it has 9 significant values against the BOA while it has 8 best results in terms of significance against the ChOA. However, it outperforms the GWO in only 6 functions. Overall, the FOX has superior results compared to the GWO, BOA, ChOA, and GA. These results prove that FOX is very competitive against common algorithms.

Table 5 Statistical comparison (*p-value*) of the FOX against GWO, FDO, BOA, ChOA, and the GA using the Wilcoxon rank-sum test.

F	FOX vs. GWO	FOX vs. FDO	FOX vs. BOA	FOX vs. ChOA	FOX vs. GA
1	0.000	0.06	0.000	0.005	0.000
2	0.008	0.102	0.000	0.003	N/A
3	1.000	0.233	0.000	0.000	N/A
4	N/A	0.008	0.000	0.004	0.000
5	N/A	0.432	N/A	N/A	0.000
6	0.000	0.000	0.000	0.000	0.000
7	0.002	0.06	0.000	0.000	0.000
8	N/A	0.075	0.003	0.032	0.000
9	0.000	0.62	0.000	0.001	0.000
10	0.000	0.991	0.000	0.005	N/A

3.4 Parameter sensitivity analysis and search mechanisms of FOX

Both c_1 *and* c_2 variables have a different range as described in the exploitation phase. c_1 range value is [0, 0.18] while c_2 range value is [0.19, 1]. Therefore, to use a better threshold due to achieving better performance by FOX, six random numbers were chosen to evaluate the performance of FOX. The c_1 random numbers are {0.16, 0.03, 0.16, 0.02, 0.06, 0.18} and c_2 random numbers are {0.49, 0.65, 0.53, 0.37, 0.96, 0.82}. By choosing each random number from both c_1 *and* c_1 respectively, the performance of FOX is evaluated. Fig. 4 presents that FOX obtained higher performance when the value of $c_1 = 0.18$ and $c_2 = 0.82$. FOX with both random values applied on all CEC2019 functions. However, significant results can be seen in F4, F7, and F10. As a result, it can be said that both c_1 *and* c_1 has a value of 0.18 and 0.82 respectively. Both c_1 *and* c_1 shows as rand6 in Fig. 4 which shows that FOX achieved better performance with rand6.

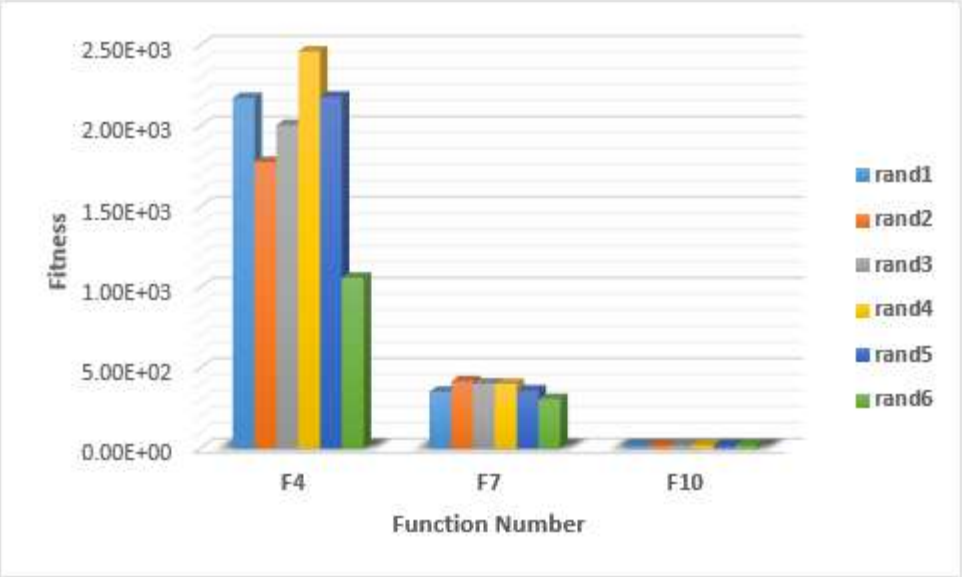


Fig. 4 Evaluating Six Random Numbers

Using the equations inside FOX and parameters guaranteed that the FOX achieves a better solution, as can be seen in Table 4, which includes five algorithms. Therefore, FOX is very competitive against the other algorithms.

The variable and equations in FOX can be changed because the distance of the fox from the prey can be found by different equations. Additionally, the *Jump* variable can be found differently: *Jump=0.5*9.81* t^2,* which is used to find the jump value. However, it can also be used to subtract the current position from the best position. Therefore, by changing the jump equation, the results are changed. These parameters can be tuned as optimization parameters like *a* value, changing the equation for finding the distance of the fox from the prey. The value of *a* is one of the vital values that affect the exploration and exploitation phases. Therefore, by increasing the value, the result changes. Consequently, this variable can be used to tune the FOX.

After implementing the FOX, three different versions of the FOX are implemented by changing the *a* value and jump equation. These modifications are based on the results in Table 4, which shows that FOX is not competitive against FDO. It means that FOX has limitations based on search capability and exploitation performance. Hence, the FOX exploration phase can be enhanced to be more competitive against other algorithms. The exploitation phase of FOX also can be improved based on changing the equations that are used to find the distance of red fox from prey and the jump equation. The convergence of the fitness value can be improved based on changing exploration and exploitation equations. As it can be seen from Fig. 6 FOX (*a val+jump*) version improves the FOX original. However, the results are required further improvements but it can be said that these parts can be considered as the limitation and improvement part for future work on the FOX.

The FOX versions are the FOX with *a* val=4, the FOX using different jump equations, and both types of FOX combined (*a* val+jump equations). The results in the previous sections are related to the original FOX because the other versions do not improve against GWO, the WOA, CSO, and the GA. However, the original FOX can be enhanced based on the previously mentioned versions because they achieve better results than the original FOX. The results of these versions can be seen in Table 6.

Table 6 Comparison of Four Different Versions of the FOX: FOX original, FOX (*a* val=4), FOX(jump), and FOX (*a* val+jump).

Func.	FOX original	FOX *a* val=4	FOX jump	FOX *a* val+jump
1	2.58 E+04	3.17 E+04	2.12 E+04	**1.53 E+04**

2	18.3442	18.3442	**18.3437**	18.3443
3	13.7025	13.7025	**13.7024**	**13.7024**
4	1.06 E+03	189.7129	1.38 E+03	**1.74 E+02**
5	6.295	**3.4661**	4.0394	3.4793
6	**5.0325**	6.6608	5.4181	6.0872
7	456.3214	371.0237	**364.0703**	423.5878
8	5.6778	5.7067	5.7042	**5.6309**
9	3.7959	3.3712	3.594	**3.3702**
10	**20.9878**	20.9975	20.9994	20.9929

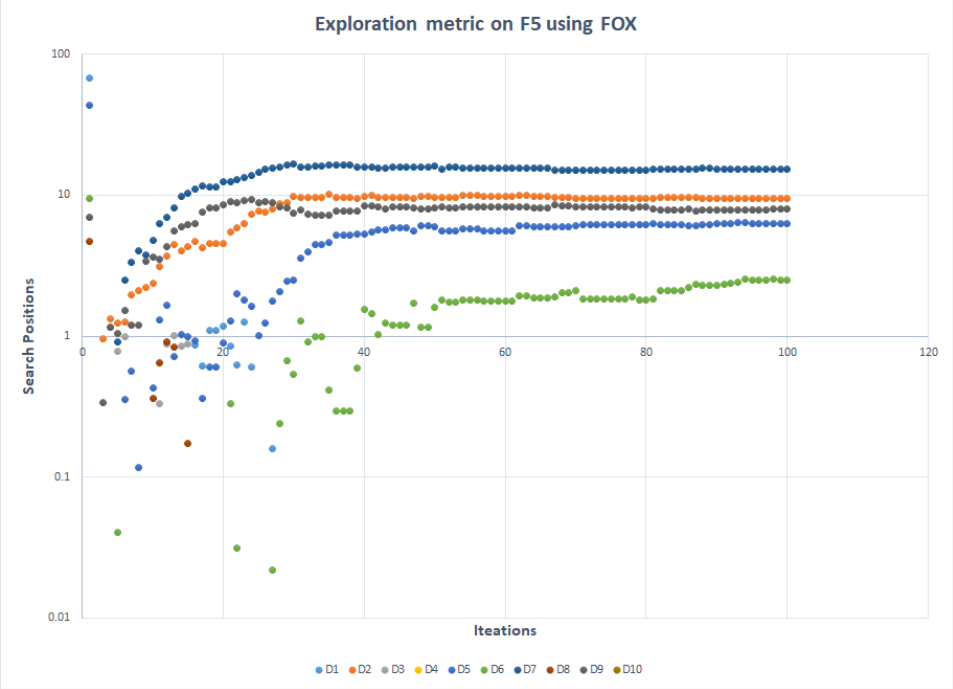
To analyze the FOX versions in detail, two quantitative metrics were used. These two metrics are used to show the results as a graph to present the exploration and convergence rate of the search agents. Figures 5 and 6 show the results of these two metrics.

The CECE2019 benchmark functions are also used in both metrics. Function 5 is used for the first metric, and then three functions were chosen for analysis randomly by the convergence metric, which are functions 1, 5, and 9. The FOX used 10 search agents using 100 iterations. Then, the results were obtained and put on a graph.

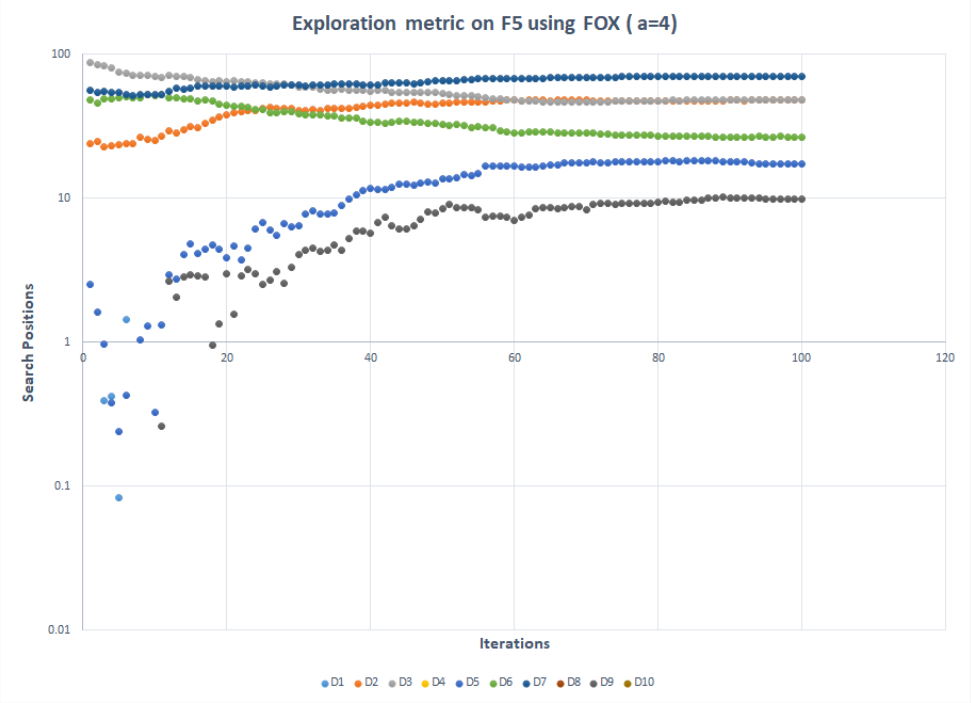
The first experiment is used to illustrate the exploration capability of three different versions of the fox search agents and how they search through the search space. These versions are called *a* val, jump, and *a* val+jump. The version of the original FOX represented in this paper with *a* val has a value of this variable that is changed to 4. The second version is jump, which is the FOX with different jump functions. Finally, *a* val+jump is a combination of the previous two. In this technique, the best position of each iteration is recorded regarding its dimension. Therefore, the results of function 5 are as shown in Fig. 5. It can be seen that the foxes try to cover the landscape and then move toward optimality. Fig. 5 shows that FOX *a* val and FOX *a* val+jump achieve better results than FOX original and FOX jump. This shows the efficiency of search agents using a=4 because they search in a large area compared to the other types. Besides, the FOX and FOX jump do not explore efficiently compared to FOX *a* val.

The second experiment is conducted by recording the fitness function of the three functions (functions 1, 5, and 9). Fig. 6 shows that the FOX versions start from a high fitness and then gradually decrease the fitness value until it reaches the optimum value.

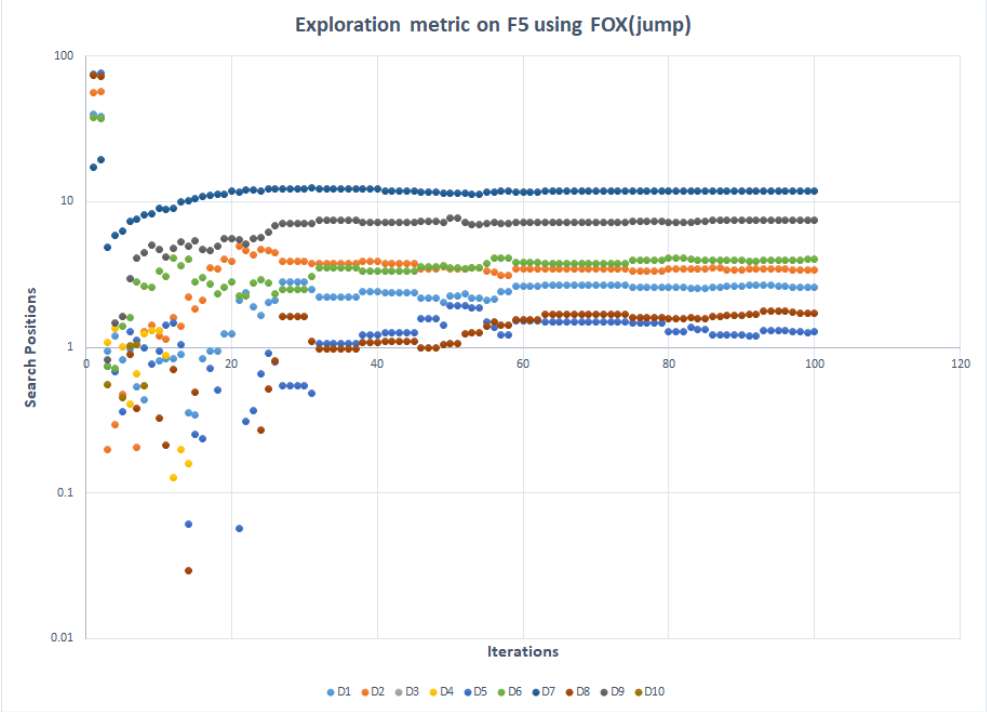
Cite as: Mohammed, H., Rashid, T. FOX: a FOX-inspired optimization algorithm. *Appl Intell* (2022). https://doi.org/10.1007/s10489-022-03533-0



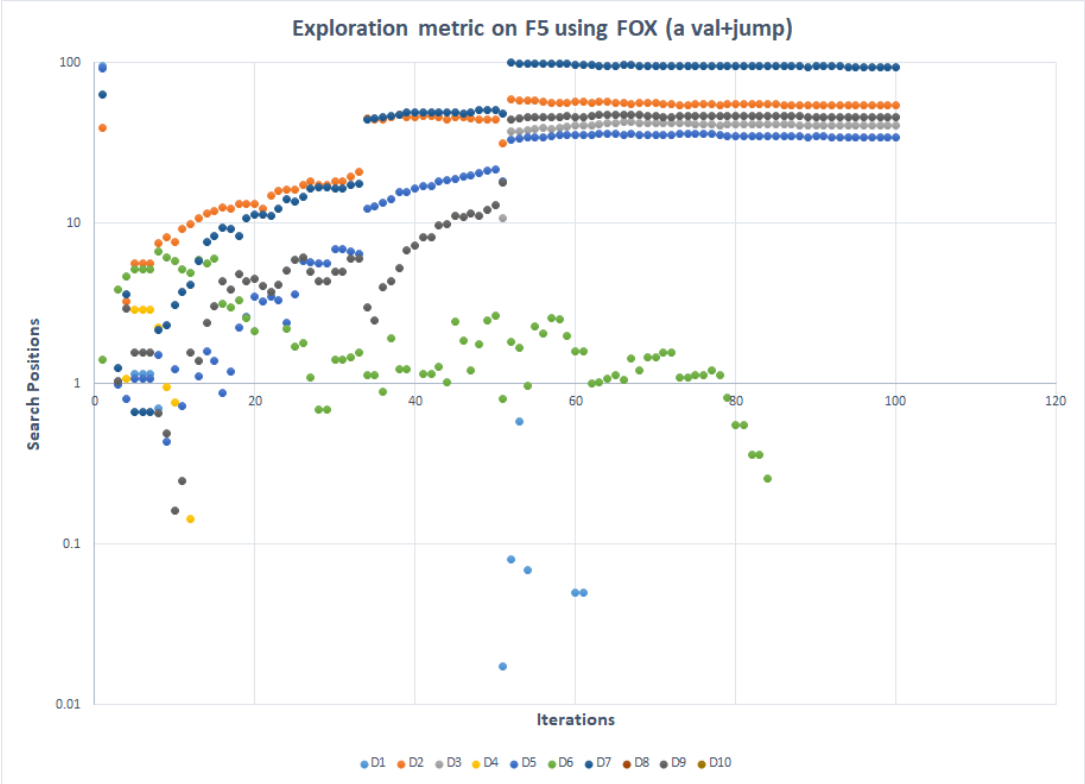
(a) FOX original



(b) FOX (a val=4)



(c) FOX (jump)

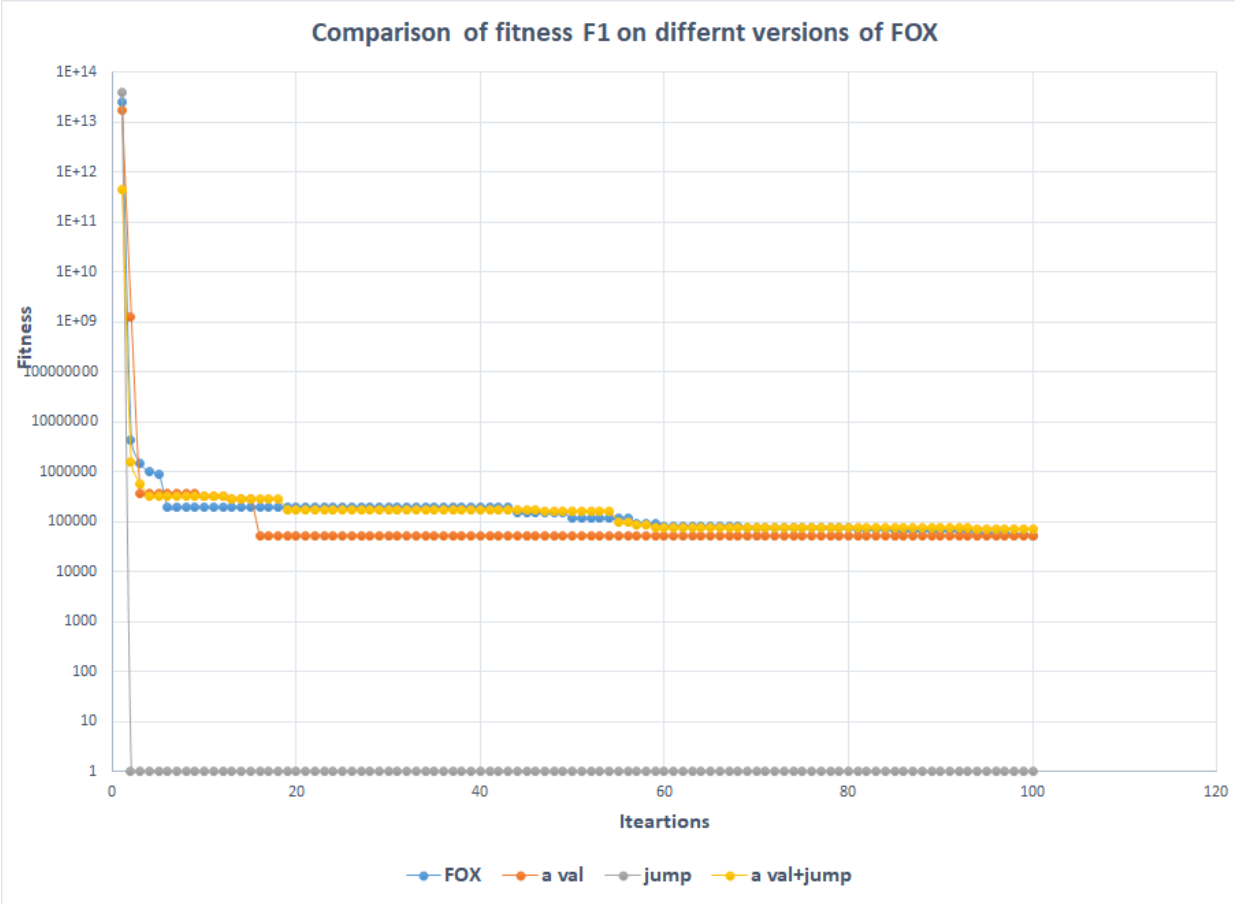


(d) FOX (a val+jump)

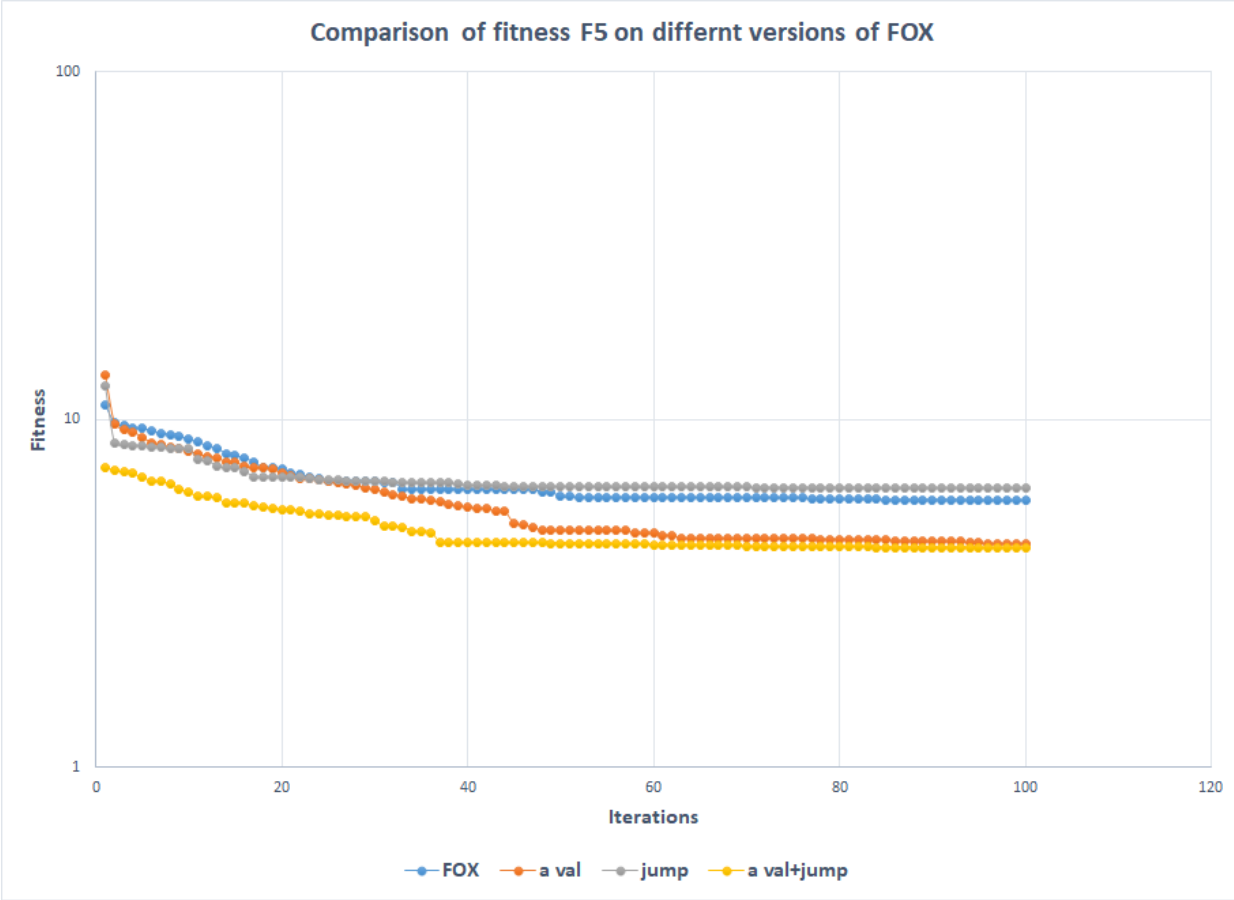
Fig. 5 FOX Capability of exploration using Functions 5

Cite as: Mohammed, H., Rashid, T. FOX: a FOX-inspired optimization algorithm. *Appl Intell* (2022). https://doi.org/10.1007/s10489-022-03533-0

Fig. 6 also shows that all four different versions of the FOX dramatically decrease and then stabilize during the iterations. However, the *a* val+jump version of the FOX performs well and achieves minimum results against the other versions in F5. Furthermore, the *a* val+jump version of the FOX has higher optimum results in F9, and the second-best version is the original FOX.

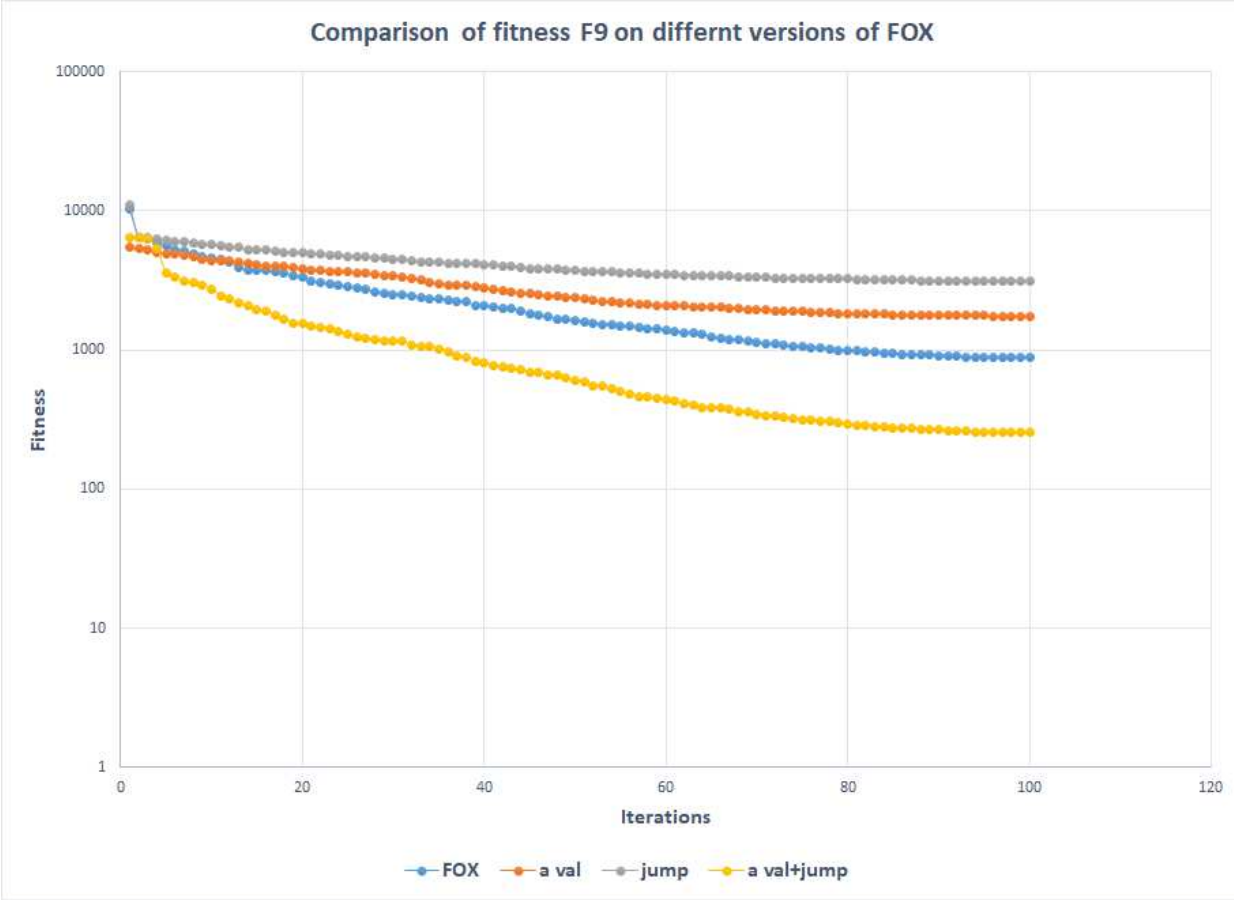


(a) Benchmark Function 1



(b) Benchmark Function 5

Cite as: Mohammed, H., Rashid, T. FOX: a FOX-inspired optimization algorithm. *Appl Intell* (2022). https://doi.org/10.1007/s10489-022-03533-0



(c) Benchmark Function 9

Fig. 6 Convergence of FOX versions

Overall, these improvements in the exploration capability and fitness value are related to the accuracy of the detecting techniques in finding the distance between the fox and the prey. Therefore, a better position is detected for the next iteration. As a result, the FOX is capable of searching in a landscape and can avoid local optima with efficient convergence toward optimality. FOX is better than the compared algorithms because it obtained a better average value when tested on five classical benchmark functions and CEC2019 benchmark functions. FOX is statistically effective because it has more significant values compared to the other algorithms. This algorithm is interesting because it has two different ways of exploitation based on the jump direction. It is also interested because FOX is inspired by a new idea which is the hunting behavior of the red fox.

4. Engineering Case Studies to test FOX:

In this section, two real-world applications are presented to be optimized by using FOX. The problems are pressure vessel design and economic load dispatch problems.

4.1 Solving a Pressure Vessel Design Problem

Pressure vessel design is a classical engineering problem that has been solved by researchers using different algorithms [1], [72], [73]. Finding the optimal solution is the aim of this problem for three aspects of a cylindrical pressure vessel. These aspects are material, welding, and forming, and they should be optimized. This problem has four variables that can be optimized: the shell thickness T_s, inner radius R, cylindrical length section without counting the head L and head thickness T_h. Figure 7 illustrates the variables of the problem that can be optimized. However, the four constraints must be tested before optimizing the variables. The following shows the problem equation and constraints.

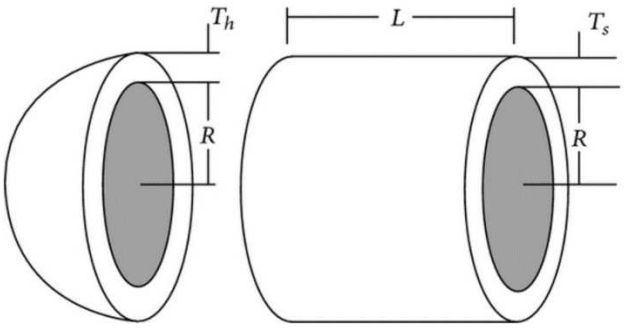


Fig. 7 Pressure vessel design problem [74]

$$n = 1,2,3,4$$

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

$$f(\vec{x}) = 0.6224x_1x_3x_4 + 1.7781x_2x_3^2 + 3.1661x_1^2x_4 + 19.84x_1^2x_3, \tag{10}$$

The variable limits are

$$0 \le x_1 \le 99,$$

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

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

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

These are subject to the following constraints:

$$g_1(\vec{x}) = -x_1 + 0.0193x_3 \ \le 0 \tag{11}$$

$$g_2(\vec{x}) = -x_3 + 0.00954x_3 \ \le 0 \tag{12}$$

$$g_3(\vec{x}) = -\pi x_3^2 x_4 - \frac{4}{3}\pi x_3^3 + 1{,}296{,}000 \ \le 0 \tag{13}$$

$$g_4(\vec{x}) = x_4 + 240 \ \le 0 \tag{14}$$

The WOA, WOAGWO, GWO, and FDO were used to solve this problem [51]. To prove that FOX has better performance against other algorithms, FOX is used to solve this problem. Table 7 shows that the FOX has high efficiency against the other algorithms. Overall, solving this problem is another proof that FOX is very competitive.

Table 7 Solving the Pressure Vessel Design Problem

Run No.	WOA	WOAGWO	FDO	FOX
1	13382.3	16697.97	377819.2	6866.846
2	8106.258	13691.68	101800.8	7062.53
3	13210.93	11391.75	57267.63	6535.226
4	8369.843	11718.12	7556.644	7390.026
5	8398.161	9536.122	6873.656	9511.966
6	7438.549	11639.14	6777.363	14667.82
7	19753.58	11127.94	19710.33	20734.75
8	8205.785	15935.19	38812.92	24518.77
9	8439.767	13165.6	8933.924	6346.123
10	7966.724	10444.29	18258.82	6604.371
11	8404.567	11303.79	116160.9	8786.426
12	11396.46	16382.91	51808.24	17404.11
13	10763.95	10050.45	83474.28	10341.99
14	7714.56	11063.28	7538.922	8306.025
15	159506.7	10520.38	47368.41	25313.13
Avg.	20070.54	12311.24	63344.13	**12026.01**
Std.	37401.74	2261.521	90960.63	6544.212

4.2 Solving Economic Load Dispatch Problem

The economic load dispatch problem is an optimization problem in electrical power. The main aim of optimizing this problem is to minimize the energy production cost while considering the load demand within various equality and inequality constraints [75]. Therefore, minimum fuel should be used to optimize the power generation unit to reduce the operation cost of generating energy [76]. Equation (15) shows the function of ELD.

$$C_t = \sum_{i=1}^{n} C_i(P_i) \tag{15}$$

Where C_t is the total cost of fuel, P_i is the actual power generation by generator *i,* C_i is the required cost by generator and n is generator numbers. Therefore, the quadratic function used to represent C_i as follow and the equation (16) has to be optimized:

$$C_i = \sum_{i=1}^{n} a_i P_i^2 + b_i P_i + c_i \tag{16}$$

where each of a_i, b_i, $and\ c_i$ are used as coefficient cost for generator i. The above equation works under two constraints: the sum of the total power generator must be equal to the power demand with the power loss and the power generator must be inside the limitation of the power generator [77], [78].

To solve the economic load dispatch problem with three power generators with a power demand of 150 MW. FOX was applied to solve this problem by using 1000 iterations with 50 independent runs. Table 9 shows the data by three different generators. Then, FOX results were compared against GWO, PSO, WOA, and FDO. All these algorithms are implemented to compare them against the FOX algorithm. Results proved that FOX achieved an optimum result in terms of generating powers while GWO, WOA, FDO, and FOX used nearly the same cost. Table 10 presents that FOX can obtain the required power at the minimum cost compared to other algorithms.

Table 8 Power Limits and Cost Coefficients for Three Generators ELD.

Generating Unit	P_{min}	P_{max}	a($/MW2)	b($/MW2)	c($/MW2)
P1	10	85	0.008	7	200
P2	10	80	0.009	6.3	180
P3	10	70	0.007	6.8	140

Table 9 Comparison of FOX with other Algorithms for Three Generators ELD Problem when PD = 150 MW.

Unit/Algorithm	GWO	PSO	WOA	FDO	FOX
P1 (MW)	31.94	60.0345	31.938	32.665	31.937
P2 (MW)	67.284	25.6626	67.284	65.489	67.277
P3 (MW)	50.777	67.2313	50.778	51.846	50.785
Total Generated Power (MW)	150.001	152.9285	150	149.999	152.6089
Cost ($/hr)	1579.698	1637.084	1579.699	1579.87	**1579.699**

5. Result analysis

The FOX algorithm demonstrated better exploration and exploitation when solving five classical benchmark functions compared to BOA, DA, ChOA, and WOA. It can be said that FOX has a great characteristic in terms of exploitation because it has optimum results compared to other algorithms in unimodal functions. Table 2 presents the exploitation capability of FOX for solving unimodal functions. FOX also benefits from superior exploratory behavior because it achieved the best results in solving multimodal functions in Tables 2 and 5. Therefore, the reason behind this efficiency is that the FOX avoids local optima because it uses variable *a* that affects the exploration phase to go toward the best solution. Dividing the iterations so that nearly a half is used for exploration and the other half is used for exploitation by using a condition with a random variable has a great impact on achieving a better balance between the two phases. Having an exploration phase based on best position, variable *a,* and *MinT* variables would have a great impact to explore solutions that are going forward with the global optima. Furthermore, finding the distance between fox and prey with a fox jump method is a vital method that is used to catch the prey in the exploitation phase. As a result, FOX achieved better performance against FDO, BOA, GA, GWO, and ChOA.

FOX optimizer uses a physical method to find its distance from prey by retrieving the sound of the prey under the snow and then trying to use a jump mechanism to catch prey. Finding distance from prey and jump method has benefits over LOA, Tigers, and GWO because these algorithms use social hierarchy and group hunting mechanisms. Despite these technics, the FOX optimizer achieved better performance compared to GWO[11], Tigers [39], LOA [38]in solving benchmark functions.

Three different versions of FOX were compared in Table 6 to show that FOX can be improved by changing the value of variable *a* and *jump* equation. These versions can improve FOX but they are not statistically significant. Therefore, it can be used in terms of modifications of FOX. The reason is that the equations of jump can be identified by different methods in the future and also the value of *a* variable has a great impact on the exploration of FOX, so by increasing the value of *a,* the performance of FOX is improved.

FOX is also used to solve pressure vessel design problems and economic ELD problems. Then, it achieved better performance against WOA, WOAGWO, GWO, PSO, and FDO. However, FDO performance is better than FOX in solving CEC2019 benchmark functions. These results prove that an algorithm can work well to solve problem A while it does not work well for problem B. The reason is that FOX has better exploration capability to avoid local optima and divide the iterations into both exploration and exploitation phases.

Overall, it can be said that FOX can avoid local optima and it also has proper balancing between exploration and exploitation by using random variable *r*. FOX also mimics the behavior of the red fox by using physical technics to create different equations from the red fox behavior. Finding the distance between the red fox and prey by taking advantage of the physical experiment in Fig. 2 improves the performance of FOX.

6. Conclusion

To conclude, this paper presented a new metaheuristic algorithm inspired by foxes' hunting behavior in nature called FOX. It uses a random walk based on the best position in the exploration phase, while it uses a distance measurement method to find the distance between fox and prey. It also uses jump action to catch the prey in the exploitation phase.

The FOX was tested by using five classical benchmark functions and CEC2019 benchmark functions, then compared to the GWO, ChOA, GA, PSO, BOA, and GA. Then, statistical results showed that FOX has the first rank and is better than the compared algorithms. The reason behind the performance of FOX is that FOX uses jump technics to catch prey. The other reason is that FOX searches based on best position, *a* variable, and *MinT*.

To provide parameter sensitivity analysis and discover more insights into the search mechanisms of FOX, exploration metric and convergence capability metric were used to present the performance of FOX. Three different versions of FOX are used by the metrics and the results showed that the FOX *a* val and FOX (*a* val+jump) have a great ability to explore new positions against the FOX original. As a result, both of these versions perform well in terms of exploration. The convergence metric is implemented by using Functions 1, 5, and 9. Jump+*a* val FOX versions showed excellent convergence and achieved better results than the other versions in functions 5 and 9. Therefore, FOX requires further research improvement.

The FOX was also applied to solve a pressure vessel design problem and economic load dispatch problem. The FOX achieved results better than WOA, WOAGWO, GWO, KMGWO, and FDO. FOX generated optimum power results because it has efficient exploration and exploitation capabilities. The reason behind the high performance of the FOX is that the FOX uses a proper balance technique between the exploration and exploitation phases to avoid being trapped in local optima. Decreasing the value of a specific variable throughout the iterations enhanced the exploitation phase.

This algorithm can be further extended since the FOX works for single-objective optimization; multi-objective optimization will be studied in the future. The FOX can also be used to solve optimization problems in planning and medical problems.

References

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

[2] Ying-Tung Hsiao, Cheng-Long Chuang, Joe-Air Jiang, and Cheng-Chih Chien, "A Novel Optimization Algorithm: Space Gravitational Optimization," in *2005 IEEE International Conference on Systems, Man and Cybernetics*, 2005, vol. 3, pp. 2323–2328.

[3] E. Rashedi, H. Nezamabadi-pour, and S. Saryazdi, "GSA: A Gravitational Search Algorithm," *Inf. Sci. (Ny).*, vol. 179, no. 13, pp. 2232–2248, Jun. 2009.

[4] S. Mirjalili, "Genetic Algorithm," in *Studies in Computational Intelligence*, 2019, pp. 43–55.

[5] B. Chopard and M. Tomassini, "Simulated annealing," in *Natural Computing Series*, 2018.

[6] M. Laguna, "Tabu search," in *Handbook of Heuristics*, 2018.

[7] C. Blum and M. López-Ibáñez, "Ant colony optimization," in *Intelligent Systems*, 2016.

[8] X. S. Yang, *Nature-Inspired Optimization Algorithms*. Elsevier, 2014.

[9] A. Bhattacharya and P. K. Chattopadhyay, "Hybrid differential evolution with biogeography-based optimization for solution of economic load dispatch," *IEEE Trans. Power Syst.*, vol. 25, no. 4, pp. 1955–1964, 2010.

[10] 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.

[11] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey Wolf Optimizer," *Adv. Eng. Softw.*, 2014.

[12] S. Arora and S. Singh, "Butterfly optimization algorithm: a novel approach for global optimization," *Soft Comput.*, vol. 23, no. 3, pp. 715–734, Feb. 2019.

[13] 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.

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

[15] D. A. Muhammed, S. A. M. Saeed, and T. A. Rashid, "Improved Fitness-Dependent Optimizer Algorithm," *IEEE Access*, vol. 8, pp. 19074–19088, 2020.

[16] A. S. Shamsaldin, T. A. Rashid, R. A. Al-Rashid Agha, N. K. Al-Salihi, and M. Mohammadi, "Donkey and smuggler optimization algorithm: A collaborative working approach to path finding," *J. Comput. Des. Eng.*, vol. 6, no. 4, pp. 562–583, Oct. 2019.

[17] C. M. Rahman and T. A. Rashid, "A new evolutionary algorithm: Learner performance based behavior algorithm," *Egypt. Informatics J.*, Sep. 2020.

[18] H. M. Mohammed, Z. K. Abdul, T. A. Rashid, A. Alsadoon, and N. Bacanin, "A new K-means grey wolf algorithm for engineering problems," *World J. Eng.*, vol. ahead-of-p, no. ahead-of-print, Mar. 2021.

[19] M. Khishe and M. R. Mosavi, "Chimp optimization algorithm," *Expert Syst. Appl.*, vol. 149, p. 113338, Jul. 2020.

[20] M. Srinivas and L. M. Patnaik, "Genetic Algorithms: A Survey," *Computer (Long. Beach. Calif).*,

vol. 27, no. 6, pp. 17–26, 1994.

[21] Z. Michalewicz, C. Z. Janikow, and J. B. Krawczyk, "A modified genetic algorithm for optimal control problems," *Comput. Math. with Appl.*, vol. 23, no. 12, pp. 83–94, Jun. 1992.

[22] D. Jude Hemanth and J. Anitha, "Modified Genetic Algorithm approaches for classification of abnormal Magnetic Resonance Brain tumour images," *Appl. Soft Comput.*, vol. 75, pp. 21–28, Feb. 2019.

[23] S. Kalyani and K. S. Swarup, "Particle swarm optimization based K-means clustering approach for security assessment in power systems," *Expert Syst. Appl.*, vol. 38, no. 9, pp. 10839–10846, Sep. 2011.

[24] K. Hussain, M. N. Mohd Salleh, S. Cheng, and Y. Shi, "Metaheuristic research: a comprehensive survey," *Artif. Intell. Rev.*, vol. 52, no. 4, pp. 2191–2233, Dec. 2019.

[25] Ankita and S. K. Sahana, "Ba-PSO: A Balanced PSO to solve multi-objective grid scheduling problem," *Appl. Intell.*, Jul. 2021.

[26] M. Li, W. Du, and F. Nian, "An Adaptive Particle Swarm Optimization Algorithm Based on Directed Weighted Complex Network," *Math. Probl. Eng.*, vol. 2014, pp. 1–7, 2014.

[27] A. Pradhan, S. K. Bisoy, and A. Das, "A survey on PSO based meta-heuristic scheduling mechanism in cloud computing environment," *J. King Saud Univ. - Comput. Inf. Sci.*, Jan. 2021.

[28] Y. Zhang, S. Wang, and G. Ji, "A Comprehensive Survey on Particle Swarm Optimization Algorithm and Its Applications," *Math. Probl. Eng.*, vol. 2015, pp. 1–38, 2015.

[29] S. Mirjalili, "Dragonfly algorithm: a new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems," *Neural Comput. Appl.*, vol. 27, no. 4, pp. 1053–1073, May 2016.

[30] 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.

[31] M. Zhang, D. Long, T. Qin, and J. Yang, "A Chaotic Hybrid Butterfly Optimization Algorithm with Particle Swarm Optimization for High-Dimensional Optimization Problems," *Symmetry (Basel).*, vol. 12, no. 11, p. 1800, Oct. 2020.

[32] H. M. Mohammed and T. A. Rashid, "Chaotic fitness-dependent optimizer for planning and engineering design," *Soft Comput.*, Aug. 2021.

[33] A. Daraz, S. A. Malik, I. U. Haq, K. B. Khan, G. F. Laghari, and F. Zafar, "Modified PID controller for automatic generation control of multi-source interconnected power system using fitness dependent optimizer algorithm," *PLoS One*, vol. 15, no. 11, p. e0242428, Nov. 2020.

[34] K. K. Chiu, Po Chan; Selamat, Ali; Krejcar, Ondrej; Kuok, "Hybrid Sine Cosine and Fitness Dependent Optimizer for Global Optimization," *Swinburne Res. Bank*, vol. 9, pp. 128601–128622, 2021.

[35] H. Jia, K. Sun, W. Zhang, and X. Leng, "An enhanced chimp optimization algorithm for continuous optimization domains," *Complex Intell. Syst.*, Apr. 2021.

[36] W. Kaidi, M. Khishe, and M. Mohammadi, "Dynamic Levy Flight Chimp Optimization," *Knowledge-Based Syst.*, vol. 235, p. 107625, Jan. 2022.

[37] J. Wang, M. Khishe, M. Kaveh, and H. Mohammadi, "Binary Chimp Optimization Algorithm

(BChOA): a New Binary Meta-heuristic for Solving Optimization Problems," *Cognit. Comput.*, vol. 13, no. 5, pp. 1297–1316, Sep. 2021.

[38] M. Yazdani and F. Jolai, "Lion Optimization Algorithm (LOA): A nature-inspired metaheuristic algorithm," *J. Comput. Des. Eng.*, vol. 3, no. 1, pp. 24–36, Jan. 2016.

[39] S. D. Kermany, "TIGER Algorithm," in *2020 International Conference on High Performance Big Data and Intelligent Systems (HPBD&IS)*, 2020, pp. 1–8.

[40] 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.

[41] M. K. Marichelvam, T. Prabaharan, and X. S. Yang, "Improved cuckoo search algorithm for hybrid flow shop scheduling problems to minimize makespan," *Appl. Soft Comput.*, vol. 19, pp. 93–101, Jun. 2014.

[42] P. R. Srivatsava, B. Mallikarjun, and X. S. Yang, "Optimal test sequence generation using firefly algorithm," *Swarm Evol. Comput.*, 2013.

[43] S. Nandy, X. S. Yang, P. P. Sarkar, and A. Das, "Color Image Segmentation By Cuckoo Search," *Intell. Autom. Soft Comput.*, 2015.

[44] L. M. Abualigah, A. T. Khader, and E. S. Hanandeh, "A new feature selection method to improve the document clustering using particle swarm optimization algorithm," *J. Comput. Sci.*, vol. 25, pp. 456–466, Mar. 2018.

[45] A. N. Jadhav and N. Gomathi, "WGC: Hybridization of exponential grey wolf optimizer with whale optimization for data clustering," *Alexandria Eng. J.*, vol. 57, no. 3, pp. 1569–1584, Sep. 2018.

[46] M. A. Tawhid and A. M. Ibrahim, "A hybridization of grey wolf optimizer and differential evolution for solving nonlinear systems," *Evol. Syst.*, pp. 1–23, Jul. 2019.

[47] Z. Xu, Y. Yu, H. Yachi, J. Ji, Y. Todo, and S. Gao, "A novel memetic whale optimization algorithm for optimization," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2018.

[48] Y. Sun, T. Yang, and Z. Liu, "A whale optimization algorithm based on quadratic interpolation for high-dimensional global optimization problems," *Appl. Soft Comput.*, vol. 85, p. 105744, Dec. 2019.

[49] Z.-M. Gao and J. Zhao, "An Improved Grey Wolf Optimization Algorithm with Variable Weights," *Comput. Intell. Neurosci.*, vol. 2019, pp. 1–13, Jun. 2019.

[50] M. R. Ramli, Z. A. Abas, M. I. Desa, Z. Z. Abidin, and M. B. Alazzam, "Enhanced convergence of Bat Algorithm based on dimensional and inertia weight factor," *J. King Saud Univ. - Comput. Inf. Sci.*, vol. 31, no. 4, pp. 452–458, Oct. 2019.

[51] 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.

[52] N. Saha and S. Panda, "Cosine adapted modified whale optimization algorithm for control of switched reluctance motor," *Comput. Intell.*, Apr. 2020.

[53] S. Mostafa Bozorgi and S. Yazdani, "IWOA: An improved whale optimization algorithm for optimization problems," *J. Comput. Des. Eng.*, vol. 6, no. 3, pp. 243–259, Jul. 2019.

[54] J.-S. Wang and S.-X. Li, "An Improved Grey Wolf Optimizer Based on Differential Evolution and Elimination Mechanism," *Sci. Rep.*, vol. 9, no. 1, p. 7181, Dec. 2019.

[55] Li, Li, Liu, and Ruan, "An Improved Bat Algorithm Based on Lévy Flights and Adjustment Factors," *Symmetry (Basel).*, vol. 11, no. 7, p. 925, Jul. 2019.

[56] M. Onay, "A New and Fast Optimization Algorithm: Fox Hunting Algorithm (FHA)," in *Proceedings of the 2016 International Conference on Applied Mathematics, Simulation and Modelling*, 2016, no. Amsm, pp. 153–156.

[57] D. Połap and M. Woźniak, "Red fox optimization algorithm," *Expert Syst. Appl.*, vol. 166, p. 114107, Mar. 2021.

[58] R. S. McDonald, J. D. Roth, and F. B. Baldwin, "Goose persistence in fall strongly influences Arctic fox diet, but not reproductive success, in the southern Arctic," *Polar Res.*, vol. 36, no. sup1, p. 5, Sep. 2017.

[59] O. Nanova and M. Prôa, "Cranial features of mainland and Commander Islands (Russia) Arctic foxes (Vulpes lagopus) reflect their diverging foraging strategies," *Polar Res.*, vol. 36, no. sup1, p. 7, Sep. 2017.

[60] D. Gallant, B. G. Slough, D. G. Reid, and D. Berteaux, "Arctic fox versus red fox in the warming Arctic: four decades of den surveys in north Yukon," *Polar Biol.*, vol. 35, no. 9, pp. 1421–1431, Sep. 2012.

[61] M. Lucherini and S. Lovari, "Habitat richness affects home range size in the red fox Vulpes vulpes," *Behav. Processes*, vol. 36, no. 1, pp. 103–105, Feb. 1996.

[62] E. Young, "Red Fox | National Geographic," 2011. [Online]. Available: https://www.nationalgeographic.com/animals/mammals/r/red-fox/. [Accessed: 24-Mar-2020].

[63] C. D. Soulsbury, P. J. Baker, G. Iossa, and S. Harris, "Fitness costs of dispersal in red foxes (Vulpes vulpes)," *Behav. Ecol. Sociobiol.*, vol. 62, no. 8, pp. 1289–1298, Apr. 2008.

[64] K. Frafjord, "Do arctic and red foxes compete for food?," *Zeitschrift fur Saugetierkd.*, 2000.

[65] L. Barth, A. Angerbjörn, and M. Tannerfeldt, "Are Norwegian lemmings Lemmus lemmus avoided by arctic Alopex lagopus or red foxes Vulpes vulpes ? A feeding experiment," *Wildlife Biol.*, vol. 6, no. 1, pp. 101–109, Jan. 2000.

[66] J. Červený, S. Begall, P. Koubek, P. Nováková, and H. Burda, "Directional preference may enhance hunting accuracy in foraging foxes," *Biol. Lett.*, vol. 7, no. 3, pp. 355–357, Jun. 2011.

[67] Irina Igel, "Hands-on Activity: Measuring Distance with Sound Waves," *Polytechnic Institute of New York University*, 2012. [Online]. Available: https://www.teachengineering.org/activities/view/nyu_soundwaves_activity1. [Accessed: 10-Oct-2020].

[68] K. K. Mishra, S. Tiwari, and A. K. Misra, "A bio inspired algorithm for solving optimization problems," in *2011 2nd International Conference on Computer and Communication Technology (ICCCT-2011)*, 2011, pp. 653–659.

[69] K. K. Mishra, S. Tiwari, and A. K. Misra, "Improved Environmental Adaption Method for Solving Optimization Problems," in *Communications in Computer and Information Science*, vol. 316 CCIS, 2012, pp. 300–313.

[70] 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.

[71] S. Arora and S. Singh, "An improved butterfly optimization algorithm with chaos," *J. Intell. Fuzzy Syst.*, vol. 32, no. 1, pp. 1079–1088, Jan. 2017.

[72] H. Joshi and S. Arora, "Enhanced Grey Wolf Optimization Algorithm for Global Optimization," *Fundam. Informaticae*, vol. 153, no. 3, pp. 235–264, Jun. 2017.

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

[74] H. Zheng and Y. Zhou, "A Cooperative Coevolutionary Cuckoo Search Algorithm for Optimization Problem," *J. Appl. Math.*, vol. 2013, pp. 1–9, 2013.

[75] V. K. Kamboj, S. K. Bath, and J. S. Dhillon, "Solution of non-convex economic load dispatch problem using Grey Wolf Optimizer," *Neural Comput. Appl.*, vol. 27, no. 5, pp. 1301–1316, Jul. 2016.

[76] M. M. Nischal and S. Mehta, "Optimal Load Dispatch Using Ant Lion Optimization," *Int. J. Eng. Res. Appl.*, vol. 5, no. 8, pp. 10–19, 2015.

[77] B. Sharma, R. Prakash, S. Tiwari, and K. K. Mishra, "A variant of environmental adaptation method with real parameter encoding and its application in economic load dispatch problem," *Appl. Intell.*, vol. 47, no. 2, pp. 409–429, 2017.

[78] M. Pradhan, P. K. Roy, and T. Pal, "Oppositional based grey wolf optimization algorithm for economic dispatch problem of power system," *Ain Shams Eng. J.*, vol. 9, no. 4, pp. 2015–2025, Dec. 2018.