A Systematic Study of Krill Herd and FOX Algorithms

Rebwar Khalid Hamad[1](✉) and Tarik A. Rashid[2]

[1] Department of Information Systems Engineering, Erbil Technical Engineering College, Erbil Polytechnic University, Erbil, Iraq
Rebwar.khalid@epu.edu.iq

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

Abstract. In 2012, Amir Hossein Gandomi and Amir Hossein Alavi presented the Krill Herd algorithm (KH), a revolutionary biologically inspired method for addressing optimization tasks. On the other hand, another new and powerful metaheuristic algorithm called FOX was proposed by Hardi Mohammed and Tarik Rashid in 2022, to address engineering difficulties, such as pressure vessel design and electrical power generating tasks, for example, economic load dispatch. The Dragonfly optimization algorithm, Particle Swarm Optimization algorithm, Fitness Dependent Optimizer algorithm, Grey Wolf Optimization algorithm, Whale optimization algorithm, Chimp optimization algorithm, Butterfly optimization algorithm, and Genetic Algorithm are also evaluated against the FOX algorithm. This paper demonstrates how the KH and Fox Algorithms are implemented, and it uses them as a model in a case study to minimize a fitness function. As a consequence, the KH and FOX algorithms successfully enhanced the original population and found the best option.

Keywords: Metaheuristic · Krill Herd · FOX Algorithm · Optimization

1 Introduction

Metaheuristic optimization techniques have recently become popular for tackling complicated optimization issues. These algorithms are more powerful than traditional approaches, which are based on formal logic or mathematical programming (Yang, 2010). The metaheuristic algorithms' two key characteristics are intensification and diversity (Gandomi et al., 2013). The most common advantages of using these algorithms are to perform complex real-world problems in the shortest amount of computational time. Optimization algorithms are produced from natural systems, biological, chemical, and physical systems, such as Krill Herd Algorithm, Bacterial Foraging Algorithm (BFA), Ant Colony Optimization (ACO), and Gravitational Search Algorithms (GSA).

Consequently, a Krill Herd algorithm is proposed for solving many real-world problems in different areas of our lives (Gandomi & Alavi, 2012). The KH algorithm is based on a simulation of krill individual herding behavior. The objective function for krill movement is the smallest distance between each krill from food and the maximum

N. Bacanin and H. Shaker (Eds.): ICIITB 2022, ACSR 104, pp. 168–186, 2023.
https://doi.org/10.2991/978-94-6463-110-4_13

density of the herd. Recently, a FOX algorithm has been developed for tackling engineering difficulties (Mohammed & Rashid, 2022). The FOX simulates the natural foraging behavior of foxes when pursuing prey. To execute an effective leap, the algorithm is based on approaches for estimating the distance between the fox and its prey.

As a result, since 1948, when Alan Turing cracked the code of the Enigma encryption machine, researchers have devised a plethora of metaheuristic algorithms. Turing's heuristic method inspired the development of the Genetic Algorithm (GA) (Sumida, 1990), which simulates natural evolution. Since the GA was proposed, many approaches have been developed, including Tabu search (Glover, 1989), Ant Colony Optimization (ACO) (M., 1992), simulated annealing (Bertsimas & Tsitsiklis, 1993), Bacterial Foraging Algorithm (BFA) (Passino, 2002), Gravitational Search Algorithms (Rashedi et al., 2009) and Fitness Dependent Optimizer (Abdullah & Ahmed, 2019). If the reader is interested in learning more about meta-search algorithms, more information will be provided in the future (Jovanovic et al., 2022), (Zivkovic et al., 2021), (Bacanin, Antonijevic, Bezdan, et al., 2022), (Zivkovic et al., 2022), (Bacanin, Zivkovic, Bezdan, et al., 2022), (Salb et al., 2023), (Zivkovic et al., 2023), (Bacanin, Zivkovic, Al-Turjman, et al., 2022), (Bacanin, Zivkovic, Sarac, et al., 2022), (Bacanin, Zivkovic, Jovanovic, et al., 2022), (Bacanin, Arnaut, Zivkovic, et al., 2022).

The primary contribution of the research is to utilize KH and FOX as case studies to manually optimize and obtain optimal solutions. As a result, simple step-by-step instructions are provided. Researchers can also utilize the paper to expand, enhance, or hybridize these algorithms with others.

The structure of the paper is divided into some parts. First, the introduction to the KH algorithm, pseudocode, and flowchart is given in Sect. 2. The mathematical implementation of the KH algorithm is illustrated in Sect. 3. Presenting the FOX algorithm is shown in Sect. 4. A case study of the FOX algorithm is explained in Sect. 5, and finally, the conclusion was outlined.

2 Krill Herd

The KH algorithm is based on a simulation of krill individual herding behavior. The aim function for krill movement is the smallest distance between each krill's food and the maximum density of the herd. As an initial stage in this algorithm, they define a search space and a group of individuals is chosen from the population. Afterward, evaluate the fitness function and examine the best krill, worst krill, and best position. Figure 1 shows the work of the KH algorithm in action. The depiction of the KH algorithm's search process is shown in Algorithm 1.

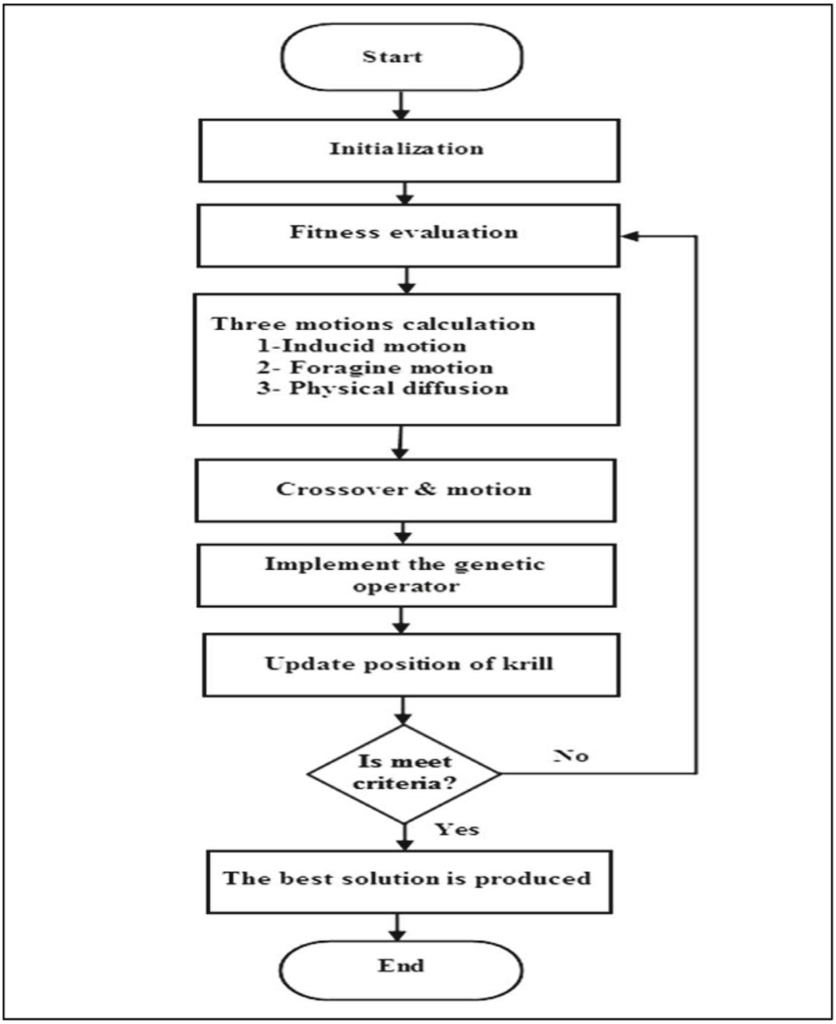

Fig. 1. The Krill Herd algorithm's flowchart

Algorithm 1 Krill Herd Algorithm

```
1:  Initialization of krill parameters: Vf, RDmax, Θmax, CR, MR, and np.
2:  for j = 1 to np do
3:      for i = 1 to d do
4:          xij = LBi + (UBi − LBi) × U(1, d) {Initialization of krill population}
5:  end for
6:  Compute fj {Evaluate each krill}
7:  end for
8:  Sort the krill and find xbest, where best Є (1, 2,…., np)
9:  while t < Max_iterations do
10:     for j = 1 to np do
11:         Perform the three motion calculation using Eq. (1), (8) and (10)
12:         xj (t + δt) = xj (t) + δt dxj/dt {Update each krill}
13:         Fine-tune xj+1 by using krill operators: Crossover and mutation
14:         Evaluate each krill by xj+1
15:     end for
16: Replace the worst krill with the best krill.
17: Sort the krill and find xbest, where best Є(1, 2, . . ., n p)
18: t=t+1
19: end while
20: Return xbest
```

3 A Case Study of KH

Consider the following minimization function; f(x), where $f(x) = X_1{}^2 + X_2{}^2$; for integer X_1 and X_2, $0 \leq X_1 \leq 12$ and $0 \leq X_2 \leq 12$.

3.1 Calculating First Iteration

Step 1: Initialize the parameters of KH
Let's suppose we have created a population randomly with four agents. Also, initial data structures are shown Table 1.

Step 2: Generate the first population randomly and evaluate the fitness values of random solutions
In the first iteration, calculate the fitness function to find the best krill, worst krill, and best position as shown in Table 2.

Sort the result of the fitness function from the lowest to the highest value. The Best Krill = 13, Worst Krill = 244, and Best position = 2, 3. For all four agents in the first iteration, we assume that F(x) = K and Kgb = Best position = 2, 3. The Lagrangian model is extended to an n-dimensional decision space.

3.2 Calculating Second Iteration

During this stage, we do the following:

$$\frac{dXi}{dt} = N_i + F_i + D_i \tag{1}$$

Table 1. Initial data structures

Notation	Value	Description
NR	4	Number of Runs
NK	4	Number of Krill's
MI	2	Maximum Iteration
C_flag	1	Crossover flag [Yes = 1]
LB	0	Lower boundary
UB	12	Upper boundary
NP	length(LB)	Number of Parameter(s)
Dt	mean(abs(UB-LB))/2	Scale Factor
Vf	0.02	Foraging speed
D^{max}	0.005	Maximum diffusion speed
N^{max}	0.01	Maximum induced speed

Table 2. Create an agent population at random and examine the fitness of each individual.

Agent	X_1	X_2	X_1^2	X_2^2	$F(x) = X_1^2 + X_2^2$
1	**2**	**3**	**4**	**9**	**13**
2	4	6	16	36	52
3	5	8	25	64	89
4	10	12	100	144	244

Step 1: In this step, we evaluate movement induced to find local and target effects. also, assume that ε = 0.4.

$$N_i^{new} = N^{max}\alpha_i + \omega_n N_i^{old} \tag{2}$$

where,

$$\alpha_i = \alpha_i^{local} + \alpha_i^{target} \tag{3}$$

$$\alpha_i^{local} = \sum\nolimits_{j=1}^{NN} \hat{K}_{ij}\hat{X}_{ij} \tag{4}$$

$$\hat{X}_{ij}\frac{X_j - X_i}{//X_j - X_i// + \varepsilon} = \tag{5}$$

$$X_j - X_i = \begin{matrix} -1 & 0 \\ -2 & 0 \\ -3 & 0 \\ -4 & 0 \end{matrix}$$

$$//X_j - X_i// = \begin{matrix} 1 & 0 \\ 2 & 0 \\ 3 & 0 \end{matrix}$$

$$//X_j - X_i// + \varepsilon = \begin{matrix} 1.4 & 0.4 \\ 2.4 & 0.4 \\ 3.4 & 0.4 \\ 2.4 & 0.4 \end{matrix}$$

$$\hat{X}_{ij} = \begin{matrix} 0.7143 & 0 \\ 0.8333 & 0 \\ 0.8824 & 0 \\ 0.8333 & 0 \end{matrix}$$

$$\hat{K}_{ij} = \frac{K_i - K_j}{K^{worst} - K^{best}} \tag{6}$$

$$K_i - K_j = \begin{matrix} 0 \\ 0 \\ 0 \\ 0 \end{matrix}$$

$$K^{worst} - K^{best} = 244 - 13 = 231$$

$$\hat{K}_{ij} = \begin{matrix} 0 \\ 0 \\ 0 \end{matrix}$$

$$\alpha_i^{local} = \begin{matrix} -0.7143 & 0 \\ 0.8333 & 0 \\ 0.8824 & 0 \\ 0.8333 & 0 \end{matrix}$$

where K^{worst} and K^{best} are the krill individuals' best and worst fitness values thus far; Ki denotes the fitness or objective function value of the *i*th krill individual; Kj is the fitness of the *j*th (j = 1, 2,… NN) neighbor; X denotes the associated locations, and NN denotes the number of neighbors. A tiny positive integer, e, is added to the denominator to avoid singularities.

$$\alpha_i^{target} = C^{best} \hat{K}_{I,best} \hat{X}_{I,best} = \tag{7}$$

$$C^{best} = 2(rand + \frac{I}{I_{max}}) \tag{8}$$

Assume that rand [0, 1] = 0.6 and = ω_n 0.3.

$$C^{best} = 2(0.6 + 1/2) = 2(1.1) = 2.2$$

$$\alpha_i^{target} = 491.8914$$

$$\alpha_i = \begin{matrix} 492.0559 \\ 492.2204 \\ 492.3849 \\ 492.2204 \end{matrix}$$

After that, we find all variables using Eq. (2).

$$= \begin{matrix} 4.9206 & 4.9206 \\ 4.9222 & 4.9222 \\ 4.924 & 4.9238 \\ 4.9222 & 4.9222 \end{matrix}$$

Step 2: In this step, we evaluate foraging motion to calculate food attraction.

$$F_i = V_f \beta_i + \omega_f F_i^{old} \tag{9}$$

where,

$$\beta_i = \beta_i^{food} + \beta_i^{best} \tag{10}$$

$$\beta_i^{food} = C^{food} \hat{K}_{i,food} \hat{X}_{i,food} \tag{11}$$

$$C^{food} = 2\left(1 + \frac{I}{I_{max}}\right) \tag{12}$$

$$C^{food} = 2(1 + 1/2) = 2(1.5) = 3$$

$$\text{SF} = (\text{sum}(\text{X/K})) \quad \text{sf = source food,} \quad \text{K = F(x) = fitness function} \tag{13}$$

$$SF = \begin{matrix} 1.6154 & 2.2308 \\ 0.4038 & 0.5577 \\ 0.2360 & 0.3258 \\ 0.0861 & 0.1189 \end{matrix}$$

$$Xf = Sf./(sum(1./K)) \tag{14}$$

The variable Xf is Food Location.

$$
{}^{1}\!/_{K} = \begin{matrix} 0.0769 \\ 0.0192 \\ 0.0112 \\ 0.0041 \end{matrix}
$$

$$
Sum\left({}^{1}\!/_{K}\right) = 0.1114
$$

$$
Xf = \begin{matrix} 14.5009 & 20.02513 \\ 3.624776 & 5.006284 \\ 2.118492 & 2.924596 \\ 0.77289 & 1.067325 \end{matrix}
$$

$$
Kf = cost(Xf) = X_1 + X_2
$$

$$
Kf = \begin{matrix} 34.52603 \\ 8.63106 \\ 5.043088 \\ 1.840215 \end{matrix}
$$

Calculation of distances

$$
\begin{aligned} Rf &= Xf - X \\ &= \begin{matrix} 12.5009 & 17.02513 \\ -0.375224 & -0.993716 \\ -2.881508 & -5.075404 \\ -9.22711 & -10.932675 \end{matrix} \end{aligned}
$$

$$
\begin{aligned} \beta_i^{food} &= C^{food} * (Kf - K)/K^{worst} - K^{best}/sqrt(sum(Rf. * Rf)) * Rf \\ &= 3 * \begin{matrix} 21.52603\ /\ 231/ & 226.6438081 & 308.6689996 \\ -43.36894 & -6.802885893 & -18.0162691 \\ -83.956912 & -52.24231425 & -92.01808592 \\ -242.159785 & -167.289343 & -198.2115764 \end{matrix} \end{aligned}
$$

$$
\beta_i^{food} = \begin{matrix} 0.001233472 & 0.000905691 \\ 0.082793243 & 0.031262465 \\ 0.020871003 & 0.011849296 \\ 0.018799358 & 0.015866542 \end{matrix}
$$

Calculation of BEST position attraction

$$\beta_i^{best} = \hat{K}_{i,best}\hat{X}_{i,best} \tag{15}$$

$$Rib = X - X$$

$$Rib = \begin{matrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{matrix}$$

$$\beta_i^{best} = (Kf - K)/K^{worst} - K^{best}/sqrt(sum(Rib. * Rib)) * Rib$$
$$= \begin{matrix} -21.52603 & / & 231 & / & 0 \\ 43.36894 & & 0 & & \\ 83.956912 & & 0 & & \\ 242.159785 & & 0 & & \end{matrix}$$

$$\beta_i^{best} = \begin{matrix} 0 \\ 0 \\ 0 \\ 0 \end{matrix}$$

$$\omega_f = (0.1 + 0.8 * (1 - I/MI)) = (0.1 + 0.8 * (1 - 1/2)) = 0.5$$

$$F_i = V_f \beta_i + \omega_f F_i^{old}$$

$$F_i = 0.02 * \begin{matrix} 0.001233472 & 0.000905691 \\ 0.082793243 & 0.031262465 \\ 0.020871003 & 0.011849296 \\ 0.018799358 & 0.015866542 \end{matrix} + 0.5 * \begin{matrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{matrix}$$

$$F_i = \begin{matrix} 0.500024669 & 0.500018114 \\ 0.501655865 & 0.500625249 \\ 0.50041742 & 0.500236986 \\ 0.500375987 & 0.500317331 \end{matrix}$$

Step 3: In this step, we evaluate physical diffusion.

$$D_i = D^{max}\delta \tag{16}$$

$$D_i = D^{max}\left(1 - \frac{I}{I_{max}}\right)\delta \tag{17}$$

Assume that rand $\delta = [-1, 1] = 0.4\delta = [-1, 1] = 0.4$

$$D_i = 0.005 * (1 - 1/2) * 0.4$$
$$D_i = 0.001$$

Step 4: In this step, we evaluate genetic operators.

3.3 Genetic Operation

3.3.1 Crossover

$$C\_rate = 0.8 + 0.2 * \left(K - K^{best}\right)/K^{worst} - K^{best} \tag{18}$$

$$K - K^{best} = 0$$
$$39$$
$$76$$
$$231$$
$$C_{rate} = 0.8 + 0.2 * 0/\,231$$
$$39$$
$$76$$
$$231$$
$$C_{rate} = 0.003463203$$
$$0.037229437$$
$$0.069264069$$
$$0.203463203$$

$$Cr = rand(NP, 1) < C\_rate$$

$$Cr = 0$$
$$0$$
$$1$$

$$NK4Cr = round(NK * rand + .5)$$

$$NK4Cr = 3$$

3.3.2 Mutation

$$X = X(NK4Cr). * (1 - Cr) + X * Cr$$

4 FOX

FOX replicates fox foraging behavior when hunting prey. The method is based on ways of measuring the distance between the fox and its prey to conduct an effective leap. The first step in this algorithm, defining a search space and choosing a group of individuals is randomly from the population. In addition, the fitness function was examined to determine the best score and best position. Figure 2 is a flowchart of the FOX algorithm. The depiction of the FOX algorithm's search process is shown in Algorithm 2.

Table 3. A new X is generated

Agent	Xnew1	Xnew2
1	**4.1696**	**6.616**
2	5.2768	8.2768
3	5	8.8304
4	6.384	9.1072

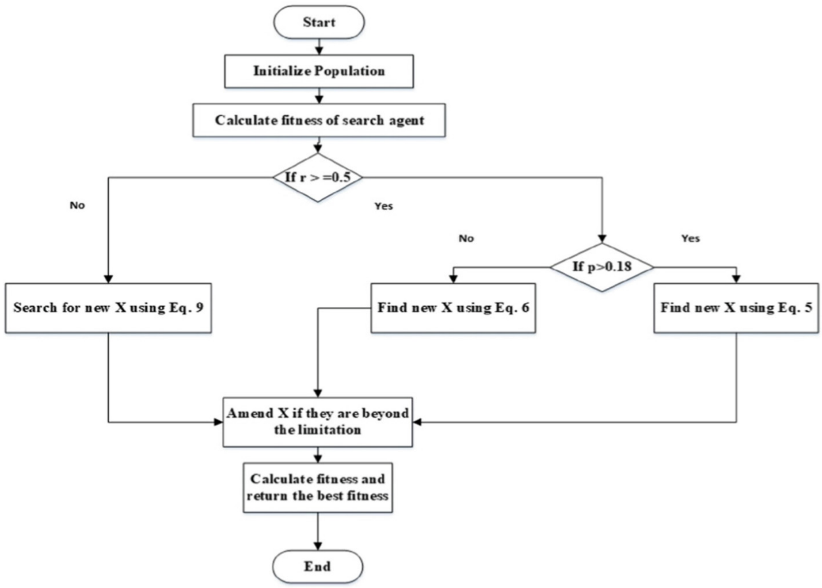

Fig. 2. FOX flowchart

Algorithm 2 FOX Optimization Algorithm

```
Initialize the red fox population Xi (i=1,2,…….,n)
While it<Maxit
Initialize Dist_S_T, Sp_S, Time_S_T, BestX, Dist_Fox_Prey, Jump, MinT,
a, BestFitness
Calculate the fitness of each search agent
Select BestX and BestFitness among the fox population (X) in each iteration
    If1 fitness_i>fitness_i+1
BestFitness= fitness_i+1
BestX=X(I,:)
Endif1
If2 r>=0.5
   If3 p>0.18
              Initialize time randomly;
              Calculate distance _Sound_Travel using Eq,(1)
              Calculate Sp_S from Eq. (2)
              Calculate distance fox from to prey using Eq. (3)
              Tt=average time;
              T=Tt/2;
              Calculate jump using Eq. (4)
              Find X(it+1) using Eq. (5)
   Elseif p<0.18
              Initialize time randomly;
              Calculate distance _Sound_Travel using Eq,(1)
              Calculate Sp_S from Eq. (2)
              Calculate distance fox from to prey using Eq. (3)
              Tt=average time;
              T=Tt/2;
              Calculate jump using Eq. (4)
              Find X(it+1) using Eq. (6)
    EndIf3
else
    Find MinT using Eq.(7)
    Explore X(it+1) using Eq. (9)
EndIf2
Check and amend the position of it goes beyond the limits
Evaluate search agents by their fitness
Update BestX
It=it+1
End while
Return BestX & BestFitness
```

5 A Case Study of the FOX Algorithm

Consider the following minimization function; f(x), where $f(x) = X_1{}^2 + X_2{}^2$; for integer X_1 and X_2, $0 \leq X_1 \leq 12$ and $0 \leq X_2 \leq 12$.

5.1 Calculating First Iteration

Step 1: Let's suppose we have created a population randomly with four agents. In the first iteration, calculate the fitness function to find the best score and best position as shown in Table 4.

Table 4. Create an agent population at random and examine the fitness of each individual.

Agent	X_1	X_2	$X_1{}^2$	$X_2{}^2$	$F(x) = X_1{}^2 + X_2{}^2$
1	2	3	4	9	**13**
2	4	6	16	36	**52**
3	5	8	25	64	**89**
4	10	12	100	144	**244**

Table 5. Initial data structures

Notation	Value	Description
A	2	According to BestX, this setting is utilized to reduce search performance.
Jump	0	Jump height
c1	0.18	These numbers are based on a red fox's leap movement, which is either to the northeast or the opposite.
c2	0.82	
MintT	inf	minimum time average
R	0.6	
P	0.20	

Sort the result of the fitness function from the lowest to the highest value. The Best score = 13, Best position = 2 3. Also, other variables are defined such as shown in Table 5.

5.2 Calculating Second Iteration

During this stage, we do the following:

These are positions for the first agent: x = 2 and x = 3.

Step 1: To find a new position, we must evaluate the distance sound travels by using the below equation.

$$Dist\_S\_T_{it} = Sp\_S * Time\_S\_T_{it} \tag{19}$$

Time_S_ = T_{it} random number between [0,1] = 0.5

Step 2: Compute the speed of the sound in this step by using Eq. (20).

$$Sp\_S = \frac{BestPosition_{it}}{Time\_S\_T_{it}} \tag{20}$$

$$\mathrm{Sp\_S} = [2, 3]/0.5$$

$$= 4, 6$$

$$Dist\_S\_T_{it} = [4, 6] * 0.5$$
$$= 2, 3$$

Step 3: In this step, evaluate the distance of the fox from the prey; it can be calculated by using Eq. 21.

$$Dis\_Fox\_Prey_{it} = Dist\_S\_T_{it} * 0.5 \quad (21)$$

$$= [2 \quad 3] * 0.5$$
$$= 1 \quad 1.5$$

Step 4: In this step, the fox needs to calculate the jump height; it can be calculated by the following equation.

$$Jump_{it} = 0.5 * 9.81 * t^2 \quad (22)$$

The value of the time transition *tt* is computed by dividing the total of the $\mathrm{Time\_S\_T_{it}}$ to dimensions, the equation below shows the tt computation.

$$tt = Time\_S\_T_{it}/dim$$

$$= 0.5/2$$
$$= 0.25$$

Divide *tt* by 2 to get the average time *t*.

$$t = tt/2 = 0.25/2 = 0.125$$

$$Jump_{it} = 0.5 * 9.81 * t^2 = 0.5 * 9.81 * 0.125^2$$
$$= 0.0156$$

Step 5: In this step, the computation of the fox's new position is shown in the equation below.

$$X_{(it+1)} = Dis\_Fox\_Prey_{it} * Jump_{it} * C_1 \quad (23)$$

$$= [11.5] * 0.0156 * 0.18$$
$$= 0.0028 \quad 0.0042$$

Mint = tt = 0.25.

In this section, we calculate Eqs. 19, 20, 21, 22, and 23 these are positions for the second agents x = 4 and x = 6.

Step 1: To find a new position, we must evaluate the distance sound travels.

$Time\_S\_\ T_{it}$= random number between [0, 1] = 0.4

$$Dist\_S\_T_{it} = Sp\_S * Time\_S\_T_{it} \tag{24}$$

Step 2: In this step, we calculate the speed of the sound.

$$Sp\_S = \frac{BestPosition_{it}}{Time\_S\_T_{it}} \tag{25}$$

$$Sp\_S = [2 \quad 3]/0.4 = 5 \quad 7.5$$

$$\begin{aligned} Dist\_S\_T_{it} &= Sp\_S * Time\_S\_T_{it} \\ &= [57.5] * 0.4 \\ &= 2 \quad 3 \end{aligned}$$

Step 3: In this step, evaluate the distance of the fox from the prey; it can be calculated by using Eq. (26).

$$Dis\_Fox\_Prey_{it} = Dist\_S\_T_{it} = *0.5 \tag{26}$$

$$\begin{aligned} &= [2 \quad 3] * 0.5 \\ &= 1 \quad 1.5 \end{aligned}$$

Step 4: The fox needs to calculate the jump height; it can be calculated by the following equation.

$$Jump_{it} = 0.5 * 9.81 * t^2 \tag{27}$$

$$tt = Time\_S\_T_{it}/dim$$

$$\begin{aligned} &= 0.4/2 \\ &= 0.2 \end{aligned}$$

$$t = tt/2 = 0.2/2 = 0.1$$

$$\begin{aligned} Jump_{it} &= 0.5 * 9.81 * t^2 = 0.5 * 9.81 * 0.1^2 \\ &= \ 0.0491 \end{aligned}$$

Step 5: In this step, the computation of the fox's new position is shown in the equation below.

$$X_{(it+1)} = Dis\_Fox\_Prey_{it} * Jump_{it} C_1 \tag{28}$$

$$= 11.5 * 0.0491 * 0.18$$
$$= 0.0088 \quad 0.0133$$

Mint = tt = 0.2

These are positions for the third agent: x = 5 and x = 8.

Step 1: To find a new position, we must evaluate the distance sound travels.

$$Time\_S\_T_{it} = random\ number\ between\ [0, 1] = 0.3$$

$$Dist\_S\_T_{it} = Sp\_S * Time\_S\_T_{it} \tag{29}$$

Step 2: In this step, we calculate the speed of the sound.

$$Sp\_S = \frac{BestPosition_{it}}{Time\_S\_T_{it}} \tag{30}$$

$$Sp\_S = [23]/0.3$$
$$= 6.7 \quad 10$$

$$Dist\_S\_T_{it} = Sp\_S * Time\_S\_T_{it}$$
$$= [6.7 \quad 10] * 0.5$$
$$= 3.4 \quad 5$$

Step 3: In this step, evaluate the distance of the fox from the prey; it can be calculated by using Eq. (30).

$Dis\_Fox\_Prey_{it} = Dist\_S\_T_{it} * 0.5(30)$

$$= [3.4 \quad 5] * 0.5$$
$$= 1.7 \quad 2.5$$

Step 4: In this step, the fox needs to calculate the jump height; it can be calculated by the following equation:

$$Jump_{it} = 0.5 * 9.81 * t^2 \tag{31}$$

$$tt = Time\_S\_T_{it}/dim$$

$$= 0.3/2$$
$$= 0.15$$

$$t = tt/2 = 0.15/2 = 0.075$$

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

$$= 0.5 * 9.81 * 0.075^2$$
$$= 0.0276$$

Step 5: In this step, the computation of the fox's new position is shown in the equation below.

$$X_{(it+1)} = Dis\_Fox\_Prey_{it} * Jump_{it} * C_1 \tag{32}$$

$$\begin{aligned} &= [1.7 \quad 2.5] * 0.0276 * 0.18 \\ &= 0.0084 \quad 0.0124 \end{aligned}$$

Mint = tt = 0.3
These are positions for the fourth agent: x = 10 and x = 12.
We assume that r < 0.5, then Eqs. 7, 8, and 9 are activated.
Mint = 0.2, a = 1.9
$X_{(it+1)} = BestX_{it} * rand(1, dimension) * MinT * a$ (33)
We assume that Rand (1, dimension) = rand (1,2) = 0.2

$$\begin{aligned} X_{(it+1)} &= [2 \quad 3] * 0.2 * 0.2 * 1.9 \\ &= 0.1520 \quad 0.2280 \end{aligned}$$

Note: if p < = 0.18, Eqs. 1, 2, 3, 4, and 6 are used to find a new position. After that second iteration, we have this new x as shown in Table 6.

Then, for the next iteration, the same previous steps are repeated.

6 Result and Discussion

The results of this study show that the results have been improved and that individual productivity has increased as a result of its adoption. According to the obtained results explained in Tables 3 and 6, it appears that the Fox algorithm works more effectively to find the best result.

Table 6. A new X is generated

Agent	Xnew1	Xnew2
1	0.0028	0.0042
2	0.0088	0.0133
3	0.0084	0.0124
4	0.1520	0.2280

7 Conclusion

This study presents the KH algorithm and the FOX algorithm. A case study is intended to describe the steps of the KH and FOX algorithms that may be confusing to readers of these algorithms. In the experimental findings, the KH and FOX algorithms demonstrated their ability to improve and develop attributes, as well as locate the ideal solution. FOX iteratively improves and achieves better solutions.

In future work, researchers should work on hybridizing, modifying, or improving these two algorithms. This suggestion is to improve the ability and efficiency of these algorithms.

References

Abdullah, J. M., & Ahmed, T. (2019). Fitness Dependent Optimizer: Inspired by the Bee Swarming Reproductive Process. *IEEE Access*, *7*, 43473–43486. https://doi.org/10.1109/ACCESS.2019.2907012

Bacanin, N., Antonijevic, M., Bezdan, T., Zivkovic, M., Venkatachalam, K., & Malebary, S. (2022). Energy efficient offloading mechanism using particle swarm optimization in 5G enabled edge nodes. *Cluster Computing 2022*, 1–12. https://doi.org/10.1007/S10586-022-03609-Z

Bacanin, N., Arnaut, U., Zivkovic, M., Bezdan, T., & Rashid, T. A. (2022). Energy Efficient Clustering in Wireless Sensor Networks by Opposition-Based Initialization Bat Algorithm. *Lecture Notes on Data Engineering and Communications Technologies*, *75*, 1–16. https://doi.org/10.1007/978-981-16-3728-5_1/COVER

Bacanin, N., Zivkovic, M., Al-Turjman, F., Venkatachalam, K., Trojovský, P., Strumberger, I., & Bezdan, T. (2022). Hybridized sine cosine algorithm with convolutional neural networks dropout regularization application. *Scientific Reports 2022 12:1*, *12*(1), 1–20. https://doi.org/10.1038/s41598-022-09744-2

Bacanin, N., Zivkovic, M., Bezdan, T., Venkatachalam, K., & Abouhawwash, M. (2022). Modified firefly algorithm for workflow scheduling in cloud-edge environment. *Neural Computing and Applications*, *34*(11), 9043–9068. https://doi.org/10.1007/S00521-022-06925-Y/FIGURES/9

Bacanin, N., Zivkovic, M., Jovanovic, L., Ivanovic, M., & Rashid, T. A. (2022). *Training a Multilayer Perception for Modeling Stock Price Index Predictions Using Modified Whale Optimization Algorithm*. 415–430. https://doi.org/10.1007/978-981-16-9573-5_31

Bacanin, N., Zivkovic, M., Sarac, M., Petrovic, A., Strumberger, I., Antonijevic, M., Petrovic, A., & Venkatachalam, K. (2022). A Novel Multiswarm Firefly Algorithm: An Application for Plant Classification. *Lecture Notes in Networks and Systems*, *504 LNNS*, 1007–1016. https://doi.org/10.1007/978-3-031-09173-5_115/COVER

Bertsimas, D., & Tsitsiklis, J. (1993). Simulated Annealing. *8*(1), 10–15. https://doi.org/10.1214/SS/1177011077

Gandomi, A. H., & Alavi, A. H. (2012). Krill herd: A new bio-inspired optimization algorithm. *Communications in Nonlinear Science and Numerical Simulation*, *17*(12), 4831–4845. https://doi.org/10.1016/j.cnsns.2012.05.010

Gandomi, A. H., Yang, X. S., & Alavi, A. H. (2013). Erratum: Cuckoo search algorithm: A metaheuristic approach to solve structural optimization problems (Engineering with Computers DOI: https://doi.org/10.1007/s00366-011-0241-y). *Engineering with Computers*, *29*(2), 245. https://doi.org/10.1007/s00366-012-0308-4

Glover, F. (1989). Tabu Search—Part I. *1*(3), 190–206. https://doi.org/10.1287/IJOC.1.3.190

Jovanovic, D., Antonijevic, M., Stankovic, M., Zivkovic, M., Tanaskovic, M., & Bacanin, N. (2022). Tuning Machine Learning Models Using a Group Search Firefly Algorithm for Credit Card Fraud Detection. *Mathematics 2022, Vol. 10, Page 2272*, *10*(13), 2272. https://doi.org/10.3390/MATH10132272

M., D. (1992). Optimization, Learning and Natural Algorithms. *Ph.D. Thesis, Politecnico Di Milano*. https://cir.nii.ac.jp/crid/1573668926038702080

Mohammed, H. (2022). *FOX : a FOX-inspired optimization algorithm.*

Passino, K. M. (2002). Biomimicry of Bacterial Foraging for Distributed Optimization and Control. *IEEE Control Systems*, *22*(3), 52–67. https://doi.org/10.1109/MCS.2002.1004010

Rashedi, E., Nezamabadi-pour, H., & Saryazdi, S. (2009). GSA: A Gravitational Search Algorithm. *Information Sciences*, *179*(13), 2232–2248. https://doi.org/10.1016/j.ins.2009.03.004

Salb, M., Jovanovic, L., Zivkovic, M., Tuba, E., Elsadai, A., & Bacanin, N. (2023). Training Logistic Regression Model by Enhanced Moth Flame Optimizer for Spam Email Classification. *Lecture Notes on Data Engineering and Communications Technologies*, *141*, 753–768. https://doi.org/10.1007/978-981-19-3035-5_56/COVER

Sumida, B.H., et al. (1990). Genetic algorithms and evolution. *Journal of Theoretical Biology*, *147*(1), 59–84. https://www.sciencedirect.com/science/article/pii/S0022519305802528

Yang, X.-S. (2010). *Nature-inspired metaheuristic algorithms*. Luniver Press.

Zivkovic, M., Bacanin, N., Arandjelovic, J., Rakic, A., Strumberger, I., Venkatachalam, K., & Joseph, P. M. (2022). *Novel Harris Hawks Optimization and Deep Neural Network Approach for Intrusion Detection*. 239–250. https://doi.org/10.1007/978-981-19-0332-8_17

Zivkovic, M., Bacanin, N., Venkatachalam, K., Nayyar, A., Djordjevic, A., Strumberger, I., & Al-Turjman, F. (2021). COVID-19 cases prediction by using hybrid machine learning and beetle antennae search approach. *Sustainable Cities and Society*, *66*, 102669. https://doi.org/10.1016/J.SCS.2020.102669

Zivkovic, M., Petrovic, A., Venkatachalam, K., Strumberger, I., Jassim, H. S., & Bacanin, N. (2023). Novel Chaotic Best Firefly Algorithm: COVID-19 Fake News Detection Application. *Studies in Computational Intelligence*, *1054*, 285–305. https://doi.org/10.1007/978-3-031-09835-2_16/COVER