18

Krill Herd Algorithm

Ali R. Kashani
Department of Civil Engineering
University of Memphis, Memphis, Tennessee, United States

Charles V. Camp
Department of Civil Engineering
University of Memphis, Memphis, Tennessee, United States

Hamed Tohidi
Department of Civil Engineering
University of Memphis, Memphis, Tennessee, United States

Adam Slowik
Department of Electronics and Computer Science
Koszalin University of Technology, Koszalin, Poland

CONTENTS

18.1 Introduction

The krill herd (KH) algorithm is developed by mimicking the social behavior of krill in a herd [1]. This algorithm belongs to the swarm-intelligence algorithms category. Krill individuals tend to search for food in the form of large schools based on two major motivations: (1) increasing krill density, (2) reaching food. Therefore, the global optimum solution can be defined as a position with a shorter distance from the source of food. Obviously, such a point has a higher density of krill swarm. The KH algorithm considers krill individuals as potential solutions for a specific problem. Therefore, the krill individuals explore the solution space by changing their position during the time. Fundamentally, KH modifies the position of the krill based on two kinds of modifications: Lagrangian movements and evolutionary operations. Lagrangian movements are governed by three basic rules: (1) movements induced by other krill; (2) foraging activity; and (3) random diffusion. Evolutionary operators are crossover and mutation which modifies the solutions before moving to the new positions. As of now, the KH algorithm has received much attentions which has resulted in several variations and improvements in this algorithm such as chaotic KH [2], stud KH [3], fuzzy KH [4], oppositional KH [5], binary KH [6], improved KH [7-9]. The remainder of this chapter is organized accordingly. In Section 18.2 a step-by-step pseudo-code for the original KH algorithm is provided, in Sections 18.3 and 18.4 source-codes of the KH algorithm in both Matlab and C++ are presented, respectively, in Section 18.5 a detailed numerical example is solved by KH, and finally a conclusion is presented in Section 18.6.

18.2 Original KH algorithm

18.2.1 Pseudo-code of the original version of KH algorithm

The pseudo-code for the KH algorithm is depicted in Algorithm 17.

Algorithm 17 Pseudo-code of the original KH.

1: determine the $D-th$ dimensional objective function $OF(.)$
2: determine the range of variability for each $j-th$ dimension $\left[K_{i,j}^{min}, K_{i,j}^{max}\right]$
3: determine the KH algorithm parameter values such as NK – number of krill, MI – maximum iteration, V_f – foraging speed, D_{max} – maximum diffusion, N_{max} – maximum induced speed
4: randomly create swarm P which consists of NK krill individuals (each krill individual is a D-dimensional vector)
5: finding the best krill and its relevant vector
6: $Iter = 0$
7: **while** termination condition not met (here is reaching MI) **do**
8: evaluate $X^{food} = \frac{\sum_{i=1}^{NK} \frac{1}{K_i} \cdot X_i}{\sum_{i=1}^{NK} \frac{1}{K_i}}$
9: **for** each i-th krill in swarm P **do**
10: $\alpha_i^{target} = 2 \cdot \left(rand + \frac{Iter}{MI}\right) \cdot \widehat{K}_{i,best} \cdot \widehat{X}_{i,best}$
11: $R_{z,i} = \sum_{j=1}^{N} \|X_i - X_j\|$ and $d_{z,i} = \frac{1}{5 \cdot N} \cdot R_{z,i}$
12: **if** $R_{z,i} < d_{z,i}$ and $K(i) \neq K(n)$ **then**
13: $\alpha_i^{local} = \sum_{j=1}^{NN} \frac{K_i - K_j}{K^{worst} - K^{best}} \times \frac{X_j - X_i}{\|X_j - X_i\| + \epsilon}$

14: **end if**
15: $\omega = 0.1 + 0.8 \times \left(1 - \frac{1}{MI}\right)$
16: $N_i^{new} = N^{max} \cdot \left(\alpha_i^{target} + \alpha_i^{local}\right) + \omega \cdot N_i^{old}$
17: $\beta_i^{food} = 2 \cdot \left(1 - \frac{Iter}{MI}\right) \cdot \widehat{K}_{i,food} \cdot \widehat{X}_{i,food}$
18: $\beta_i^{best} = \widehat{K}_{i,best} \cdot \widehat{X}_{i,best}$
19: $F_i^{new} = V_f \cdot \left(\beta_i^{food} + \beta_i^{best}\right) + \omega \cdot F_i^{old}$
20: $D_i = D^{max} \cdot \left(1 - \frac{Iter}{MI}\right) \cdot \delta$
21: $\frac{dX_i}{dt} = N_i^{new} + F_i^{new} + D_i$
22: $X_i = crossover\,(X_i, X_c)$
23: $X_i = mutation\,(X_i, X_{best}, M_u)$
24: $X_i^{new} = X_i + \frac{dX_i}{dt}$
25: **end for**
26: update best-found solution
27: $Iter = Iter + 1$
28: **end while**
29: post-processing the results

18.2.2 Description of the original version of KH algorithm

Referring to Algorithm 17, KH algorithm handles a D-dimensional objective function $OF(.)$. Therefore, after defining the objective function, the valid boundary domains for the j-th variable of the i-th krill is defined as $\left[K_{i,j}^{min}, K_{i,j}^{max}\right]$ in the second step. The third step is devoted to the parameter setting of the KH algorithm such as a number of krill (NK), the maximum number of iteration (MI), the foraging speed (V_f), the maximum diffusion speed (D^{max}), the maximum induced speed (N_{max}). The original paper's recommendation for D^{max} is $[0.002, 0.010]$ (ms^{-1}), and for V_f and N_{max} are 0.02 and 0.01 (ms^{-1}), respectively. KH starts with a randomly produced initial population of NK number of krill. Each krill is represented by a D-dimensional vector as follows:

$$K_i = \{k_{i,1}, k_{i,2}, ..., k_{i,D}\} \tag{18.1}$$

where $k_{i,1}$ to $k_{i,D}$ are decision variables varying between $K_{i,j}^{min}$ and $K_{i,j}^{max}$.

In Step 5, the best-found solution is detected. The main loop of the KH algorithm is started in the next step. Within this loop, the location of the virtual food is calculated. In the original paper [1], a method based on the distribution of the krill individuals' fitness proposed to find the center of the food. As mentioned earlier, KH uses the following time-dependent Lagrangian model to simulate krill movements in a D-dimensional search space:

$$\frac{dX_i}{dt} = F_i + N_i + D_i \tag{18.2}$$

where F_i is the foraging action, N_i is movement induced by the other krill individuals, and D_i is random diffusion.

Step 9 in Algorithm 17 is utilized to address Equation (18.2). In step 11, sensing distance for every krill individual is calculated. In Equation (18.2), N_i depends on two different factors α_i^{target} (for the best neighbor effect) and α_i^{local} (for local neighbor effect). Those factors are computed in step 10 and 13, respectively. In step 15, inertia weights for induced motion by other krill individuals (ω_n) and inertia weight of the foraging motion (ω_f) are evaluated. The movement caused by other krill individuals (N_i^{new}) is computed in step 16. In steps 17 and 18, β_i^{food} and β_i^{best}, two necessary factors for evaluating the foraging motion (F_i) are calculated, respectively. After estimating F_i in step 19, physical diffusion of the krill individuals is computed in step 20. In steps 22 and 23, two genetic operators of crossover and mutation are applied to the current population. Finally, the new positions of the krill herd are calculated by adding the term of Δt, resulting from step 21 to their current positions. In step 26 the best-found solution is updated. Finally, the best-found solution is proposed when the termination criteria are satisfied.

18.3 Source-code of the KH algorithm in Matlab

In Listing 18.1 the source-code for the objective function handled by the KH algorithm is shown. In the objective function $OF(.)$, the input parameters are all the krill in a herd. The result of $OF(.)$ function is an N-dimensional column vector with the objective function values for each krill K_i from the whole population of K. The here tackled minimization objective function is given by Equation (18.3). The source-code for the KH algorithm in Matlab is presented in Listing 18.2.

$$OF\left(SA_i\right) = \sum_{j=1}^{D} SA_{i,j}^2; where - 5.12 < SA_{i,j} < 5.12 \tag{18.3}$$

```
function [output]=OF(SA)
[x,y]=size(SA);
output=zeros(x,1);
for i=1:x
    for j=1:y
    output(i,1)=output(i,1)+SA(i,j)^2;
    end
end
```

Listing 18.1
Definition of objective function $OF(.)$ in Matlab.

```
function [Best,Bestsolution,Kgb]= KH (NK,MI,UB,LB)
%% Initial Parameter Setting
C_flag = 1;                                  % Crossover flag [Yes=1]
NP = length(LB); % Number of Parameter(s)
Dt = mean(abs(UB-LB))/2; % Scale Factor
F = zeros(NP,NK);D=zeros(1,NK);N=zeros(NP,NK); %R = zeros(NP,NK);
```

```
Vf = 0.02; Dmax = 0.005; Nmax = 0.01; Sr = 0;
%% Optimization & Simulation %Initial Krills positions
X=rand(NK,NP).*(UB-LB)+LB;
for z1 = 1:NP
    X(z1,:) = LB(z1) + (UB(z1) - LB(z1)).*rand(1,NK);
end
K=zeros(1,NK);
for z2 = 1:NK
    K(z2)=OF(X(:,z2));
end
Kib=K;
Xib=X;
[Kgb,A]=min(K);
Xgb=zeros(NP,MI);
Xgb(:,1)=X(:,A);
Xf=zeros(NP,MI);
Kf=zeros(1,MI);
for j = 1:MI
    %% Virtual Food
    Sf=zeros(1,NP);
    for ll = 1:NP
        Sf(ll) = (sum(X(ll,:)./K));
    end
    Xf(:,j) = Sf./(sum(1./K)); %Food Location
    Xf(:,j) =findlimits(Xf(:,j)',LB,UB,Xgb(:,j)');% Bounds Checking
    Kf(j) = OF(Xf(:,j));
    if 2<=j
        if Kf(j-1)<Kf(j)
            Xf(:,j) = Xf(:,j-1);
            Kf(j) = Kf(j-1);
        end
    end
    Kw_Kgb = max(K)-Kgb(j);
    w=(0.1+0.8*(1-j/MI));
    RR=zeros(NP,NK);
    for i = 1:NK
        % Calculation of distances
        Rf = Xf(:,j)-X(:,i);
        Rgb = Xgb(:,j)-X(:,i);
        for ii = 1:NK
            RR(:,ii) = X(:,ii)-X(:,i);
        end
        R = sqrt(sum(RR.*RR));
        % Movement Induced % Calculation of BEST KRILL effect
        if Kgb(j) < K(i)
            alpha_b = -2*(1+rand*(j/MI))*(Kgb(j) - K(i)) /Kw_Kgb/ sqrt(sum(Rgb.*Rgb)) * Rgb;
        else
            alpha_b=0;
        end
        % Calculation of NEIGHBORS KRILL effect
        nn=0;
        ds = mean(R)/5;
        alpha_n = 0;
        for n=1:NK
            condition=R<ds;
            condition=sum(condition);
            if (condition==NK && n~=i)
                nn=nn+1;
                if and(nn<=4,K(i)~=K(n))
                    alpha_n=zeros(NP,1);
                    alpha_n = alpha_n-(K(n) - K(i)) /Kw_Kgb/ R(n) * RR(:,n);
                end
            else
                alpha_n = 0;
            end
```

```
        end
        % Movement Induced
        N(:,i) = w*N(:,i)+Nmax*(alpha_b+alpha_n);
        % Foraging Motion % Calculation of FOOD attraction
        if Kf(j) < K(i)
            Beta_f=-2*(1-j/MI)*(Kf(j) - K(i)) /Kw_Kgb/ sqrt(sum(Rf.*Rf
    )) * Rf;
        else
            Beta_f=0;
        end
        % Calculation of BEST position attraction
        Rib = Xib(:,i)-X(:,i);
        if Kib(i) < K(i)
            Beta_b=-(Kib(i) - K(i)) /Kw_Kgb/ sqrt(sum(Rib.*Rib)) *Rib;
        else
            Beta_b=0;
        end
        % Foraging Motion
        F(:,i) = w*F(:,i)+Vf*(Beta_b+Beta_f);
        % Physical Diffusion %
        D = Dmax*(1-j/MI)*floor(rand+(K(i)-Kgb(j))/Kw_Kgb)*(2*rand(NP
    ,1)-ones(NP,1));
        % Motion Process %
        DX = Dt*(N(:,i)+F(:,i));
        % Crossover %
        if C_flag ==1
            C_rate = 0.8 + 0.2*(K(i)-Kgb(j))/Kw_Kgb;
            Cr = rand(NP,1) < C_rate ;
            % Random selection of Krill No. for Crossover
            NK4Cr = round(NK*rand+.5);
            % Crossover scheme
            add=X(:,NK4Cr).*(1-Cr)+X(:,i).*Cr;
            for dim=1:NP
                X(dim,i)=add(dim);
            end
        end
        % Update the position
        add=X(:,i)+DX;
        for dim=1:NP
            X(dim,i)=add(dim);
        end
        X(:,i)=findlimits(X(:,i)',LB,UB,Xgb(:,j)'); % Bounds Checking
        K(i)=OF(X(:,i));
        if K(i)<Kib(i)
            Kib(i)=K(i);
            Xib(:,i)=X(:,i);
        end
    end
    % Update the current best
    [Kgb(j+1), A] = min(K);
    if Kgb(j+1)<Kgb(j)
        Xgb(:,j+1) = X(:,A);
    else
        Kgb(j+1) = Kgb(j);
        Xgb(:,j+1) = Xgb(:,j);
    end
end
%% Post-Processing
Best = min(Kgb);
Bestsolution=Xgb(:,end);

function [ns]=findlimits(ns,Lb,Ub,best)
% Evolutionary Boundary Constraint Handling Scheme
n=size(ns,1);
for i=1:n
    ns_tmp=ns(i,:);
    I=ns_tmp<Lb;
```

```
    J=ns_tmp>Ub;
    A=rand;
    ns_tmp(I)=A*Lb(I)+(1-A)*best(I);
    B=rand;
    ns_tmp(J)=B*Ub(J)+(1-B)*best(J);
  ns(i,:)=ns_tmp;
end
```

Listing 18.2
Source-code of KH algorithm in Matlab.

18.4 Source-code of the KH algorithm in C++

The C++ source code for the objective function and the KH algorithm are presented in Listing 18.3 and Listing 18.4, respectively.

```
#include <iostream>
using namespace std;
/* Function Definitions */
double OF(double SA[], int size_array)
{
  double output;
  int j;
  output = 0.0;
  for (j = 0; j < size_array; j++) {
    output += SA[j] * SA[j];
  }
  return output;
}
```

Listing 18.3
Definition of objective function $OF(.)$ in C++.

```
/* Function Definitions */
/* Include files */
#include <iostream>
#include <stdlib.h>
#include <math.h>
using namespace std;
KH(int NK, int MI, int dim, double lb[], double ub[])
{
  /* Initial Krills positions */
  double X[NK][dim]; double K[1][NK];
for (int i = 0; i < NK; i++) {
  srand(time(0));
  for (int j = 0; j < dim; j++) {
    double r = (rand() % 10000) / 10000;
    X[i][j] = (ub[j] - lb[j])*r + lb[j];
  }
  K[i] = OF(X[i], dim);
}
double Kib = K; double Xib = X; double Kgb; double Xgb[1][dim]; double
      RR[dim][NK];
Kgb = K[1];
for (int i = 0; i < NK; i++) {
  if (K[i + 1] < Kgb) {
    Kgb = K[i + 1]; A = i + 1;
    for (int j = 0; j < dim; j++) {
```

```
            Xgb[1][j] = X[i + 1][j];
        }
    }
}
double Xf[dim][MI]; double Kf[1][MI]; double Sf[1][dim]; double Kw_Kgb
    ; double Rf; double RR[dim][NK]; double R[1][dim]; double
    alpha_b_numerator; double R_ave;
double condition[1][NK]; double sum_condition; double alpha_n; double
    Food_multiplier_trans[dim][1]; double Food_multiplier; double Rib[
    dim][1]; double Sum_attraction_multipliers;
double D_multiplier[dim][1];
for (int t = 0; t < MI; t++) {
    /* Virtual Food */
    for (int int i = 0; i < dim; i++) {
        double Null = 0;
        for (int j = 0; j < NK; j++) {
            Sf[i] += X[i][j] / K[j];
            Null += 1 / K[j];
            Xf[i][j] = Sf[i] / Null; /* Food Location */
        }

    }
    for (int i = 0; i < NK; i++) {
        for (int j = 0; j < dim; j++) {
            /* Bounds Checking */
            if (Xf[i][j] > ub[j]) Xf[i][j] = ub[j];
            if (Xf[i][j] < lb[j]) Xf[i][j] = lb[j];
        }
        Kf[i] = OF(Xf[i], dim);
    }
    if (t >= 2) {
        if (Kf[t-1] < Kf[t]) {
            for (int i = 0; i < dim; i++) {
                Xf[i][t] = Xf[i][t-1];
            }
        }
    }
        Kw_Kgb = max(K) - Kgb[t];
    double w = (0.1 + 0.8*(1 - t / MI));
    for (int i = 0; i < NK; i++) {
        /* Calculation of distance */
        for (int j = 0; j < dim; j++) {
            Rf[j][1] = Xf[j][t] - X[j][i];
            Rgb[1][j] = Xgb[1][j] - X[j][i];
            alpha_b_numerator += (Rgb[1][j] * Rgb[1][j]);
            Food_multiplier_trans[j][1] = Rf[j][1] * Rf[j][1];
            Food_multiplier += Food_multiplier_trans[j][1];
        }
        for (int j = 0; j < NK; j++) {
            for (int k = 0; k < dim; k++) {
                RR[k][j] = X[k][j] - X[k][i];
                R[1][j] += (RR[k][j] * RR[k][j]);
            }
            R[1][j] = sqrt(R[1][j]);
            R_ave += R[1][j];
        }
        /* Movement Induced */
        double alpha;
        if (Kgb[t] < K[i]) {
            srand(time(0));
            r = (rand() % 10000) / 10000;
            alpha_b = -2 * (1 + r*(t / MI))*(Kgb[t] - K[i]) / Kw_Kgb / sqrt(
        alpha_b_numerator) * Rgb;
        }
        else {
            alpha_b = 0;
        }
```

```
    /* Calculation of neighbors krill effect */
    int nn = 0;
    double ds = R_ave / (5 * NK);
    for (int n = 0; n < NK; n++) {
      for (int j = 0; j < NK; j++) {
        condition[1][j] = R[1][j] < ds[1][j];
        sum_condition += condition[1][j];
      }
      if (sum_condition == NK && n != i) {
        nn += 1;
        if (nn <= 4 && K(i) != K(n)) {
          for (int j = 0; j < dim; j++) {
            RR_multiplier[j][1] = RR[j][n];
          }
          alpha_n = alpha_n - (K[n] - K[i]) / Kw_Kgb / R[n] * RR_multiplier;
        }
      }
    }
    /* Movement Induced */
    for (j = 0; j < dim; j++) {
      N[dim][i] = w*N[dim][i] + Nmax*(alpha_b + alpha_n);
    }
    /* Calculation of food attraction */
    Food_multiplier = sqrt(Food_multiplier);
    if (Kf[j] < K[i]) {
      double Beta_f = -2 * (1 - j / MI)*(Kf[j] - K[i]) / Kw_Kgb / Food_multiplier * Rf;
    }
    else {
      double Beta_f = 0;
    }
    /* Calculation of best psition attraction */
    for (int j = 0; j < dim; j++) {
      Rib[j][1] = Xib[j][i] - X[j][i];
      best_attraction_multipliers[j][1] = Rib[j][1] * Rib[j][1];
      Sum_attraction_multipliers += best_attraction_multipliers[j][1];
    }
    if (Kib[i] < K[i]) {
      double Beta_b = -(Kib[i] - K[i]) / Kw_Kgb / sqrt(Sum_attraction_multipliers) *Rib;
    }
    else {
      double Beta_b = 0;
    }
    /* Foraging Motion */
    for (int j = 0; j < dim; j++) {
      F[j][i] = w*F[j][i] + Vf*(Beta_b + Beta_f);
    }
    /* Physical Diffusion */
    for (int j = 0; j < dim; j++) {
      srand(time(0)); double r = (rand() % 10000) / 10000;
      D_multiplier[j][1] = 2 * r - 1;
    }
    srand(time(0));
    double r = (rand() % 10000) / 10000;
    double D = Dmax*(1 - t / MI)*floor(r + (K[i] - Kgb[t]) / Kw_Kgb)*D_multiplier;
    double C_rate = 0.8 + 0.2*(K[i] - Kgb[t]) / Kw_Kgb;
    /* Motion Process */
    for (int j = 0; j < dim; j++) {
      double Dx[j][i] = Dt*(N[j][i] + F[j][i] + D[j]);
    }
    if (C_flag == 1) {
      for (int j = 0; j < dim; j++) {
        srand(time(0));
        R_vec[j][1] = rand() % 10000) / 10000;
```

```
        double Cr = R_vec[j][1] < C_rate;
        int NK4Cr = nearbyint(KN*((rand() % 10000) / 10000) + 0.5);
        X[j][i] = X[j][NK4Cr] * (1 - Cr) + X[j][i] * Cr;
      }
    }
    for (int j = 0; j < dim; j++) {
      Delta[j][1] = X[j][i];
      X[j][i] = Delta[j][1] + DX[j][i];
      /* Bounds Checking */
      if (X[i][j] > ub[j]) X[i][j] = ub[j];
      if (X[i][j] < lb[j]) X[i][j] = lb[j];
    }
    K[i] = OF(X[i], dim);
    if (K[i] < Kib[i]) {
      Kib[i] = K[i];
      for (int j = 0; j < dim; j++) {
        Xib[j][i] = X[j][i];
      }
    }
  }
```

Listing 18.4
Source-code of KH algorithm in C++.

18.5 Step-by-step numerical example of KH algorithm

In this section, a detailed computational process of the objective function defined by Equation (18.3) has been provided. In this study, the number of design variables and the number of krill in a herd are 5 and 6, respectively. The results are reported based on running the algorithm only for one iteration. Notably, the presented results may be slightly different from the real calculation. It has occurred because we used four decimal approximation here although the original algorithm uses the long format for variables. In the first step, foraging speed (V_f), maximum diffusion (D_{max}), and maximum induced speed (N_{max}) are initialized as 0.02, 0.005, and 0.01, respectively. In the second step, a herd of six krill is produced randomly within the acceptable boundaries domain.

$K_1 = \{3.2228, -2.2682, 4.6814, 2.9922, 1.8302\}$
$K_2 = \{4.1553, 0.4801, -0.1497, 4.7052, 2.6392\}$
$K_3 = \{-3.8196, 4.6848, 3.0749, 1.5948, 2.4897\}$
$K_4 = \{4.2330, 4.7604, -3.6671, -4.7543, -1.1036\}$
$K_5 = \{1.3553, -3.5060, -0.8012, 3.5751, 1.5921\}$
$K_6 = \{-4.1212, 4.8189, 4.2571, 4.4441, -3.3670\}$

In the third step, the relevant objective values for every krill is computed.

$OF(K_1) = 10.3862$
$OF(K_2) = 17.2666$

$OF(K_3) = 14.5898$
$OF(K_4) = 17.9180$
$OF(K_5) = 1.8370$
$OF(K_6) = 16.9842$

For calculating the best krill effect (α_b), in the fourth step, the objective values of the krill and their relevant solution vectors will be saved in Kib and Xib, respectively. Next, the best-found solution and its relevant vector are stored in Kgb and Xgb, respectively. In the next step, the main loop of the algorithm will be run iteratively until satisfying the termination criteria. To update the position of the krill school, movement induced by other krill (N_i), foraging activities (F_i), and random diffusion (D_i) are needed to be estimated. This procedure is simulated in the KH algorithm using a for loop for every i-th krill individual in step 14 (line 43 in Listing 18.2). However, before going through this step, we need a virtual food location and the distance from this food resource to compute N_i. In this way, S_f for every d-th dimension of the problem is defined as the accumulation of the d-th design variables for all the krill divided by their relevant objective values as follows (step 7):

$S_f = \{1.0205, -1.2286, 0.2627, 2.6124, 1.1066\}$

In order to calculate food location in step 8, each component of S_f is divided by the summation of the inverse of each objective value.

$X_f = \{1.1573, -1.3933, 0.2979, 2.9626, 1.2549\}$

The necessary step after estimating the food location is to check boundary constraints and bring the violated particle back to the valid domain. As there is no boundary constraint violation, the objective value for this food location is computed as follows (step 10):

$K_f = OF(X_f) = 1.3395$

It should be noted that since the second iteration, if there is no improvement in the food location KH will not update its position (step 11). In the next step, we evaluate the difference between the best and worst individuals, $K_{w}\_K_{gb}$, which is required for calculating α and β:

$K_{w}\_K_{gb} = 17.9180 - 1.8370 = 16.0810$

In step 13, the inertia weight is defined as follows:

$\omega = \left(0.1 + 0.8 \times \left(1 - \frac{Iter}{MI}\right)\right) = (0.1 + 0.8 \times (1 - 1/1)) = 0.1$

As mentioned previously step 14 evaluates N_i, F_i and D_i. Two necessary factors for computing N_i are α_b (effect of the best krill) and α_n (effect of neighbor). α_b needs R_{gb} that is specified by the difference between the best particle and i-th solution (i.e., $R_{gb} = X_{gb} - X_i$) in step 15 as follows:

$$R_{gb1} = \{-1.8674, -1.2379, -5.4825, 0.5829, -0.2381\}$$
$$R_{gb2} = \{-2.7999, -3.9861, -0.6514, -1.1301, -1.0472\}$$
$$R_{gb3} = \{5.1750, -8.1909, -3.8760, 1.9803, -0.8976\}$$
$$R_{gb4} = \{-2.8776, -8.2665, 2.8659, 8.3294, 2.6957\}$$
$$R_{gb5} = \{0, 0, 0, 0, 0\}$$
$$R_{gb6} = \{5.4765, -8.3249, -5.0583, -0.8690, 4.9591\}$$

Using R_{gb} and $K_{w}\_K_{gb}$ we can calculate α_b based on the following formula:

$$\alpha_b = -2 \times \left(1 + r \times \left(\frac{Iter}{MI}\right)\right) \times \frac{K_{gb} - K_i}{K_{w}\_K_{gb} \times \sqrt{R_{gb} \otimes R_{gb}}} \times R_{gb} \tag{18.4}$$

where $Iter$ represents the current iteration, and r is a uniform random number. Therefore, α_b values based on step 16 in this study are depicted as follows:

$$\alpha_{b1} = \{-0.4256, -0.2822, -1.2498, 0.1329, -0.0543\}$$
$$\alpha_{b2} = \{-1.2767, -1.8176, -0.2970, -0.5153, -0.4775\}$$
$$\alpha_{b3} = \{1.3531, -2.1416, -1.0134, 0.5178, -0.2347\}$$
$$\alpha_{b4} = \{-0.8666, -2.4894, 0.8630, 2.5083, 0.8118\}$$
$$\alpha_{b5} = 0$$
$$\alpha_{b6} = \{1.6347, -2.4849, -1.5098, -0.2594, 1.4802\}$$

Next, R is defined as the second square root of the summation of elements of the piece-wise multiplication of the RR vector. Therefore, in step 17, the RR vector is determined as the difference between the i-th and all the other krill.

$$RR_1 = \begin{bmatrix} 0 & 0.9325 & -7.0424 & 1.0102 & 1.0102 & -1.8674 \\ 0 & 2.7482 & 6.9530 & 7.0286 & 7.0286 & -1.2379 \\ 0 & -4.8311 & -1.6065 & -8.3485 & -8.3485 & -5.4825 \\ 0 & 1.7130 & -1.3974 & -7.7465 & -7.7465 & 0.5829 \\ 0 & 0.8090 & 0.6594 & 0.6594 & -2.9338 & -0.2382 \end{bmatrix}$$

$$RR_2 = \begin{bmatrix} -2.6710 & 0 & -7.9749 & 0.0776 & -2.8000 & -8.2765 \\ -2.7627 & 0 & 4.2048 & 4.2804 & -3.9861 & 4.3388 \\ 4.7671 & 0 & 3.2246 & -3.5173 & -0.6514 & 4.4069 \\ -2.5480 & 0 & -3.1104 & -9.4595 & -1.1301 & -0.2611 \\ 1.8784 & 0 & -0.1496 & -3.7428 & -1.0472 & -6.0063 \end{bmatrix}$$

$$RR_3 = \begin{bmatrix} 5.3040 & 5.1096 & 0 & 8.0526 & 5.1750 & -0.3015 \\ -6.9675 & -4.2979 & 0 & 0.0756 & -8.1909 & 0.1340 \\ 1.5425 & -3.2398 & 0 & -6.7419 & -3.8760 & 1.1822 \\ 0.5624 & 3.0840 & 0 & -6.3491 & 1.9803 & 2.8493 \\ 2.0280 & 0.1251 & 0 & -3.5933 & -0.8976 & -5.8567 \end{bmatrix}$$

$$RR_4 = \begin{bmatrix} -2.7486 & -2.9430 & -0.2208 & 0 & -2.8776 & -8.3541 \\ -7.0431 & -4.3734 & -0.1852 & 0 & -8.2665 & 0.0584 \\ 8.2845 & 3.5021 & 0.8760 & 0 & 2.8659 & 7.9242 \\ 6.9115 & 9.4331 & 6.3756 & 0 & 8.3294 & 9.1984 \\ 5.6213 & 3.7184 & 3.5813 & 0 & 2.6957 & -2.2634 \end{bmatrix}$$

$$RR_5 = \begin{bmatrix} 0.1290 & -0.0654 & 2.6568 & 2.8332 & 0 & -5.4765 \\ 1.2234 & 3.8930 & 8.0812 & -0.0982 & 0 & 8.3249 \\ 5.4186 & 0.6362 & -1.9899 & 4.0063 & 0 & 5.0583 \\ -1.4178 & 1.1037 & -1.9538 & 0.2239 & 0 & 0.8690 \\ 2.9256 & 1.0227 & 0.8855 & -1.0902 & 0 & -4.9591 \end{bmatrix}$$

$$RR_6 = \begin{bmatrix} 5.6055 & 5.4112 & 8.1333 & 8.3098 & 5.4765 & 0 \\ -7.1015 & -4.4319 & -0.2437 & -8.4232 & -0.2437 & 0 \\ 0.3603 & -4.4221 & -7.0482 & -1.0520 & -3.7452 & 0 \\ -2.2868 & 0.2347 & -2.8228 & -0.6451 & -0.8690 & 0 \\ 7.8847 & 5.9819 & 5.8447 & 3.8690 & 4.9591 & 0 \end{bmatrix}$$

Then, using the RR vectors, we can evaluate R as follows:

$R_1 = \{0, 5.9457, 10.1444, 13.7381, 5.9560, 11.5525\}$
$R_2 = \{6.8930, 0, 10.0685, 11.5841, 5.1504, 11.9537\}$
$R_3 = \{9.1371, 8.0376, 0, 12.7878, 10.6594, 6.6277\}$
$R_4 = \{14.3234, 11.9526, 7.3705, 0, 12.7073, 14.9105\}$
$R_5 = \{6.4377, 4.2224, 8.9959, 5.0325, 0, 12.2569\}$
$R_6 = \{12.2222, 10.2135, 12.5704, 12.5098, 8.3323, 0\}$

KH uses the average of the R vector's elements (d_s) as a threshold for the sensing distance. In this way, for every krill which meets this criterion α_n will be calculated as follows:

$$\alpha_n = \alpha_n - 2 \times \left(1 + r \times \left(\frac{Iter}{MI}\right)\right) \times \frac{K_n - K_i}{K_{w}\_K_{gb} \times R_n} \times \overrightarrow{RR_n} \tag{18.5}$$

where $\overrightarrow{RR_n} = X_i - X_j$.

Therefore, in step 19 d_s was calculated as follows:

$d_s = \{1.5779, 1.4886, 1.6751, 2.1884, 1.5416, 1.9061\}$

In step 20 and 21, a counter called nn as well as the parameter α_n are initialized as zero. In the next step, the threshold for updating α_n has been checked and updated. In this study, the mentioned condition is not satisfied for any of the krill individuals. As a result, N_i for i-th krill individual is depicted accordingly.

$$N_i = \begin{bmatrix} -0.0042 & -0.0128 & 0.0135 & -0.0087 & 0 & 0.0163 \\ -0.0028 & -0.0182 & -0.0214 & -0.0249 & 0 & -0.0248 \\ -0.0125 & -0.0030 & -0.0101 & 0.0086 & 0 & -0.0151 \\ 0.0013 & -0.0052 & 0.0052 & 0.0251 & 0 & -0.0026 \\ -0.0005 & -0.0048 & -0.0023 & 0.0082 & 0 & 0.0148 \end{bmatrix}$$

The next movement is caused by foraging motion. Two effective factors in this motion are food attraction (β_f) and best position attraction (β_b). For the individuals with objective values more than K_f and K_{ib} the factors of β_f and β_b resulted from the Equations (18.4) and (18.5), respectively.

$$\beta_f = -2 \times \left(1 - \frac{Iter}{MI}\right) \times \frac{K_f - K_i}{K_{w}\_K_{gb} \times \sqrt{R_f \otimes R_f}} \times R_f \tag{18.6}$$

$$\beta_b = -\frac{K_{ib} - K_i}{K_{w}\_K_{gb} \times \sqrt{R_{ib} \otimes R_{ib}}} \times R_{ib} \tag{18.7}$$

In these relationships, R_f is defined as the difference between X_f and i-th krill and R_{ib} as the difference between X_{ib} and i-th krill. In step 23, R_f is computed as follows:

$$\begin{aligned}
R_{f1} &= \{-2.0654, 0.8748, -4.3835, -0.0296, -0.5753\} \\
R_{f2} &= \{-2.9979, -1.8734, 0.4476, -1.7426, -1.3843\} \\
R_{f3} &= \{4.9770, -6.0782, -2.7770, 1.3678, -1.2348\} \\
R_{f4} &= \{-3.0756, -6.1538, 3.9650, 7.7169, 2.3585\} \\
R_{f5} &= \{-0.1980, 2.1127, 1.0990, -0.6125, -0.3372\} \\
R_{f6} &= \{5.2785, -6.2122, -3.9592, -1.4815, 4.6219\}
\end{aligned}$$

β_f values were calculated in step 24 using the R_f values. In this case of study β_f values for all the krill have been equal to zero. After that, the effect of β_b for evaluating F_i needed to be considered. Therefore, R_{ib} values as the difference between X_{ib} and the current location of each krill, are required in this step. During the first iteration of the algorithm as the values of X_{ib} are determined to be equal to X_i, the values for R_{ib} would be a zero vector. As a result, in this case, the values for β_b would be zero. Now, we can examine F_i using ω, V_f, β_f, and β_b. In step 27, F_i proved to be a zero vector in the first iteration.

In step 28, D_i is calculated using the following equation:

$$\beta_f = D_{max} \times \left(1 - \frac{t}{MI}\right) \times \left(rand + \frac{K_i - K_{gb}^t}{K_{w}\_K_{gb}}\right) \times (2 \times V_r - V_u) \tag{18.8}$$

where t represents the current iteration, $rand$ is a random number, and K_{gb}^t is the t-th global best solution. D_{max} is equal to 0.005 as discussed earlier. Let dim be the number of design variables, so V_r is a vector of a random number, and V_u is a vector of unit elements both with the size of dim by one. Here, β_f is a zero vector with the size of dim by one.

In step 29, the movement interval (D_x) is computed by summing N_i, F_i, and D_i.

$$D_x = \begin{bmatrix} -0.02179 & -0.0654 & 0.0693 & -0.0444 & 0 & 0.0837 \\ -0.0144 & -0.0931 & -0.1096 & -0.1275 & 0 & -0.1272 \\ -0.0640 & -0.0152 & -0.0519 & 0.0442 & 0 & -0.0772 \\ 0.0068 & -0.0264 & 0.0265 & 0.1284 & 0 & -0.0133 \\ -0.0028 & -0.0244 & -0.0120 & 0.0416 & 0 & 0.0758 \end{bmatrix}$$

Before adding the term of D_x to the current positions of the herd of krill, a crossover operator is applied to the current solution (X^i) and one of the randomly selected krill (X^{random}). Therefore, in step 30, the rate of crossover is defined by the following expression:

$$C_{rate} = 0.8 + 0.2 \times \left(\frac{K_i - K_{gb}^t}{K_{w\_}K_{gb}} \right) \tag{18.9}$$

Then, a uniform random number called C_r will be produced for every variable of the problem. For every dimension with the random number greater than C_{rate} the design variable will be replaced by the following term:

$$X_{dim}^i = X_{dim}^{random} \times (1 - C_r) + X_{dim}^i \times C_r \tag{18.10}$$

In this study, the values for C_{rate} are summarized as follows:

$C_{rate} = \{0.9063, 0.9919, 0.9586, 1, 0.8000, 0.9884\}$

The values of C_r for every dimension of each krill are gathered in the following:

$C_{r1} = \{0.8687, 0.0046, 0.8173, 0.9619, 0.7749, 0.0496\}$
$C_{r2} = \{0.0844, 0.3998, 0.2599, 0.8001, 0.4314, 0.9027\}$
$C_{r3} = \{0.1818, 0.9106, 0.2638, 0.1455, 0.1361, 0.9448\}$
$C_{r4} = \{0.8693, 0.5797, 0.5499, 0.8530, 0.1449, 0.4909\}$
$C_{r5} = \{0.6220, 0.3509, 0.5132, 0.4018, 0.0760, 0.4892\}$
$C_{r6} = \{0.2399, 0.1233, 0.1839, 0.2399, 0.4173, 0.9991\}$

Randomly selected krill for the crossover are 5, 2, 4, 3, 3, and 2. The current population after applying the crossover operator changed into the following positions:

$K_1 = \{3.2228, -2.2682, 4.6814, 2.9922, 1.8302\}$
$K_2 = \{4.1553, 0.4801, -0.1497, 4.7052, 2.6392\}$
$K_3 = \{-3.8196, 4.6848, 3.0749, 1.5948, 2.4897\}$
$K_4 = \{4.2330, 4.7604, -3.6671, -4.7543, -1.1036\}$
$K_5 = \{1.3553, 4.5752, -2.7911, 3.5751, 1.5921\}$
$K_6 = \{-4.1212, 4.8189, 4.2571, 4.4441, -3.3670\}$

A mutation operator is applied to the output of the crossover function at the next step. To this end, the following equation is utilized:

$$X_{dim}^{i} = \begin{cases} K_{gb,dim}^{t} + \mu \left(X_{dim}^{p} - X_{dim}^{q} \right) & \text{if } rand_{dim}^{i} < Mu \\ X_{dim}^{i} & \text{otherwise} \end{cases} \tag{18.11}$$

where p and q are two randomly selected krill. The results in our study are collected as below:

$rand^1 = \{0.16, 0.37, 0.35, 0.02, 0.29\}, p = 3, q = 6$
$K_1 = \{1.5061, -2.2682, 4.6814, 2.1504, 4.5204\}$
$rand^2 = \{0.24, 0.76, 0.76, 0.74, 0.74\}, p = 4, q = 4$
$K_2 = \{1.3554, 0.4801, -0.1497, 4.7052, 2.6392\}$
$rand^3 = \{0.10, 0.82, 0.17, 0.46, 0.66\}, p = 5, q = 3$
$K_3 = \{3.9429, 4.6849, -2.7392, 1.5948, 2.4897\}$
$rand^4 = \{0.95, 0.14, 0.28, 0.04, 0.11\}, p = 6, q = 4$
$K_4 = \{4.2330, -3.4768, 3.1609, 8.1743, 0.4603\}$
$rand^5 = \{0.95, 0.54, 0.04, 0.68, 0.81\}, p = 2, q = 3$
$K_5 = \{1.3554, 4.5752, 0.5119, 3.5751, 1.5921\}$
$rand^6 = \{0.95, 0.04, 0.54, 0.68, 0.81\}, p = 1, q = 6$
$K_6 = \{-4.1212, -7.0568, 4.2571, 4.4441, -3.3670\}$

Now, the term of D_x is added to the current solutions to update the positions of the krill herd as follows:

$K_1 = \{1.4843, -2.2826, 4.6174, 2.1572, 4.5177\}$
$K_2 = \{1.2900, 0.3870, -0.1650, 4.6788, 2.6148\}$
$K_3 = \{4.0121, 4.5752, -2.7911, 1.6213, 2.4777\}$
$K_4 = \{4.1886, -3.6043, 3.2051, 8.3027, 0.5019\}$
$K_5 = \{1.3553, 4.5752, 0.5119, 3.5751, 1.5921\}$
$K_6 = \{-4.0375, -7.1840, 4.1798, 4.4308, -3.2913\}$

After checking the boundary limitations, the objective values for every krill will be evaluated. In this example there is no boundary violation, therefore, the objective function values are calculated as follows:

$OF(K_1) = 2.2032$
$OF(K_2) = 1.6641$
$OF(K_3) = 16.0973$
$OF(K_4) = 17.5444$
$OF(K_5) = 1.8370$
$OF(K_6) = 16.3013$

Finally, the global best individual will be updated accordingly:

$X_{gb} = \{1.2900, 0.3870, -0.1650, 4.6788, 2.6148\}$
$K_{gb} = 1.6641$

After determination of the main loop of the algorithm, the global best solution found in the previous step will be proposed as the final result of the algorithm.

18.6 Conclusion

In this chapter, the strategy behind the KH algorithm to handle a given objective function has been explained. Therefore, all the steps for handling the objective function through a pseudo-code were provided. To better conveyance of the concept, we presented the source-codes for the original KH algorithm in both Matlab and C++ programming language were presented. In addition, a simple numerical example is tackled for detailed computation with the aim of better understanding the mechanism of the KH algorithm. The whole computational loads during the KH process were provided for this example. This chapter can be helpful to better understanding of the fundamentals of KH or desires to rewrite this algorithm in any other programming language.

References

1. A.H. Gandomi, A.H. Alavi. "Krill herd: a new bio-inspired optimization algorithm". *Communications in Nonlinear Science and Numerical Simulation*, vol. 17(12), pp. 4831-4845, 2012.
2. G.G Wang, L. Guo, A.H. Gandomi, G.S. Hao, H. Wang. "Chaotic krill herd algorithm". *Information Sciences*, vol. 274, pp. 17-34, 2014.
3. G.G Wang, A.H. Gandomi, A.H. Alavi. "Stud krill herd algorithm". *Neurocomputing*, vol. 128, pp. 363-370, 2014.
4. E. Fattahi, M. Bidar, H.R. Kanan. "Fuzzy krill herd optimization algorithm" in *IEEE First International Conference on Networks & Soft Computing (ICNSC)*, pp. 423-426, August 2014.
5. S. Sultana, P.K. Roy. "Oppositional krill herd algorithm for optimal location of capacitor with reconfiguration in radial distribution system". *International Journal of Electrical Power & Energy Systems*, vol. 74, pp. 78-90, 2016.

6. D. Rodrigues, L.A. Pereira, J.P. Papa, S.A. Weber. "A binary krill herd approach for feature selection" in *22nd IEEE International Conference on Pattern Recognition (ICPR)*, pp. 1407-1412, August 2014.
7. G.G. Wang, A.H. Gandomi, A.H. Alavi. "An effective krill herd algorithm with migration operator in biogeography-based optimization". *Applied Mathematical Modelling*, vol. 38(9-10), pp. 2454-2462, 2014.
8. J. Li, Y. Tang, C. Hua, X. Guan. "An improved krill herd algorithm: Krill herd with linear decreasing step". *Applied Mathematics and Computation*, vol. 234, pp. 356-367, 2014.
9. R.R. Bulatovic, G. Miodragovic, M.S. Boskovic. "Modified Krill Herd (MKH) algorithm and its application in dimensional synthesis of a four-bar linkage". *Mechanism and Machine Theory*, vol. 95, pp. 1-21, 2016.