10

Dynamic Virtual Bats Algorithm

Ali Osman Topal

Department of Computer Engineering
Epoka University, Tirana, Albania

CONTENTS

10.1 Introduction

Dynamic Virtual Bats Algorithm (DVBA) is proposed as another bat-inspired algorithm in 2014 by Topal and Altun [1, 2]. It is inspired by the bat's ability to manipulate frequency and wavelength of emitted sound waves. Although the algorithm is fundamentally inspired from Bat Algorithm (BA) [3], it is conceptually very different. In DVBA, a role based search is developed by using two bats: explorer and exploiter bat. Each bat has its own role in the algorithm and during the search they exchange these roles dynamically according to their positions. The bat which is in a better position becomes the exploiter; meanwhile the other one becomes the explorer. While the exploiter bat increases the intensification of the search around a preferable position, the explorer bat will keep looking for a better position. Until the explorer bat finds a better position, the exploiter bat will increase intensification of the search after each iteration to attain the optimal solution.

DVBA has been tested on well-known test functions (from CEC-2014 [4]) and a supply chain cost problem. The results show that DVBA, similar to other evolutionary algorithms, has some challenging problems, like the convergence

speed and avoiding local optima traps. Recently, these problems in DVBA have been eradicated by introducing probabilistic selection restart techniques in an improved version of DVBA (IDVBA) [5]. The rest of this chapter is organized as follows. In Section 10.2 an overview of DVBA is given with its pseudo code, in Sections 10.3 and 10.4 source code of the algorithm are presented in Matlab and C++ programming language respectively, in Section 10.5 a numerical example of DVBA is shown in detail, and the conclusion is drawn in Section 10.6.

10.2 Dynamic virtual bats algorithm

In DVBA, there are two bats with two different roles. In Figure 10.1, these roles are simulated. Here, black triangles represent the bats and plus signs represent the prey. The black dots are the positions on the waves which are going to be checked for a better solution. As shown in Figure 10.1a during exploration the search points which are created by the explorer bat are distributed widely in the search space. However in Figure 10.1b the exploiter bat creates a very small search scope where search points have become closer to each other.

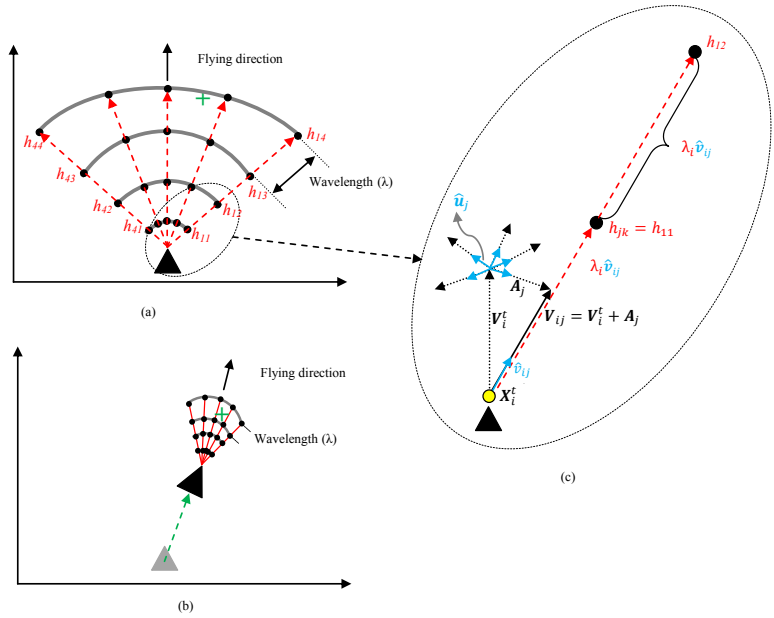

FIGURE 10.1
(a) Exploration: explorer bat searching a prey, (b) Exploitation: exploiter bat chasing a prey, (c) Search positions in the search scope of a bat.

10.2.1 Pseudo-code of DVBA

The pseudo-code of DVBA is represented in Algorithm 9. Step by step the algorithm is explained in the next section.

Algorithm 9 Description of DVBA with pseudo-code.

```
1: Objective function f(x), x = (x_1, ..., x_d)^T
2: Initialize the bat population x_i and v_i, (i = 1, 2)
3: Initialize wavelength λ_i and frequency freq_i
4: Find the increment rate ρ
5: Define the increment divisor β, the scope width variable b
6: Initialize the number of the waves(j) and search points(k)
7: Find f_gbest based on the bats' starting position.
8: while (t < Max number of FEs) do
9:     for each bat do
10:        Create a sound waves scope
11:        Evaluate the solutions on the waves
12:        Choose the best solution on the waves, h_*
13:        if f(h_*) is better than f(x_i) then
14:            Move to h_*, update x_i
15:            Change v_i towards to the better position
16:            Decrease λ_i and increase f_i
17:        end if
18:        if f(x_i) is not better than the best solution f_gbest then
19:            Change the direction randomly
20:            Increase λ_i and decrease freq_i
21:        end if
22:        if f(x_i) is the best found solution then
23:            Minimize λ_i and maximize freq_i
24:            Change the direction randomly
25:        end if
26:    end for
27:    Rank the bats and find the current best x_gbest
28: end while
```

where f_{gbest} is the global best solution and d is the number of dimensions.

10.2.2 Description of DVBA

In this section, we will explain how bats' hunting strategies became an optimization algorithm by presenting DVBA's pseudo-code in Algorithm 9. DVBA starts with defining a multidimensional objective function $f(x)$ in step 1. Similar to real bats, virtual bats are looking for superior solutions (prey) in their search scope $f(x)$. The solution (prey) in DVBA is the minimum value of the objective function $f(x)$.

Both bats start flying from random positions x_i with random velocities v_i using random wavelength and frequency (step 2 and 3). In step 4, 5, and 6 the parameters such as number of wave vectors, search points on the wave vectors, increment rate divisor β, and the scope width variable b are defined. Increment rate ρ is calculated by using the search space boundaries (10.8), so the size of the bat's search scope can be adapted with the size of the search space. In steps 16 and 20, ρ is used as the increment rate for frequency and wavelength (10.6-10.7). During the search, the range of the search scope changes dynamically: it expands when exploring (Figure 10.1a) or it shrinks if prey gets closer (Figure 10.1b). The length and the width of the search scope are controlled by wavelength λ_i and the frequency $freq_i$, respectively.

After defining all the parameters and initializing the bats' position and velocities, the main loop starts in step 8. Function of evaluation (FEs) is used as stopping criteria for the main loop. Generally, maximum number of FEs is set to $100,000$ for 10 dimensional functions and $300,000$ for 30 dimensional functions in optimization algorithms [4].

For each bat, the search scope $H_{j,k}$ (10.5) is generated in step 10. The search scope is simulated by using wave direction vectors V_j^i and search points h_{jk} on the vectors as shown in Figure 10.1a and Figure 10.1c. To distribute the wave direction vectors in the search scope the unit vectors $\hat{u}_j$ are generated which give direction to the scope width vectors $\mathbf{A}_j$ (Figure 10.1c). The unit vectors $\hat{u}_j$ are generated randomly; consequently, the wave vectors are distributed randomly in the search scope as well. The magnitude of A_j changes the width of the search scope, and is inversely proportional with the frequency of the waves $freq_i$ (10.1). When the frequency $freq$ increases reflects what is shown in Figure 10.1b. The search points h_{jk} are distributed by scaling the unit vector $\hat{v}_{ij}$ (10.3) with the same length as wavelength λ_i. The scope width vector $\mathbf{A}_j$, the wave vector $\mathbf{V}_{ij}$, and the positions of the search points h_{ij} in the bat's search scope at time step t are given by (10.1-10.5),

$$\mathbf{A}_j = \frac{\hat{u}_j b}{freq_i}, (j = 1, ..., w), (i = 1, ..., n) \tag{10.1}$$

$$\mathbf{V}_{ij} = \mathbf{V}_i^t + \mathbf{A}_j \tag{10.2}$$

$$\hat{\mathbf{v}}_{ij} = \frac{\mathbf{V}_{ij}}{\|\mathbf{V}_{ij}\|} \tag{10.3}$$

$$h_{j,k} = \mathbf{X}_i^t + \hat{\mathbf{v}}_{ij} k \lambda_i, (k = 1, ..., m) \tag{10.4}$$

$$\mathbf{H}_{j,k} = \begin{bmatrix} h_{1,1} & h_{1,2} & \cdots & h_{1,k} \\ h_{2,1} & h_{2,2} & \cdots & h_{2,k} \\ \vdots & \vdots & \ddots & \vdots \\ h_{j,1} & h_{j,2} & \cdots & h_{j,k} \end{bmatrix} \tag{10.5}$$

where w is the number of wave vectors, i is the index of bat (maximum 2), $b \in (20, 40)$ is the scope width variable, and m is the number of search positions on the wave vectors.

If the frequency increases, vectors will get closer and the wavelength (distance between search points) will get shorter (10.7). This is inspired by a bat's dynamic search ability for hunting. The changes of frequency $freq_i$, and wavelength λ_i at time step $t+1$ are given by (10.6 - 10.8),

$$freq_i^{t+1} = freq_i^t \pm \rho \tag{10.6}$$

$$\lambda_i^{t+1} = \lambda_i^t \pm \rho \tag{10.7}$$

$$\rho = mean(\frac{\mathbf{U} - \mathbf{L}}{\beta}), \qquad \{\beta \in \Re : \beta > 0\} \tag{10.8}$$

where ρ and β are positive real constants, U is the upper bounds and L is the lower bounds of the search space, and ρ is used as the increment rate for frequency and wavelength. β is used as the increment rate divisor. Choosing the range of the wavelength is very important. If the wavelength is too short, in large search spaces, convergence could be very slow. Conversely, if the wavelength is too long, it might bypass the global best and fail to locate the optimum position. To overcome this problem the length of the wavelength is linked to the range of the problem. For simplicity the following approximations were used. The wavelength range λ, in a range $[\lambda_{min}, \lambda_{max}] = [0.1\rho, 10\rho]$ corresponds to a range of the frequency $[freq_{min}, freq_{max}] = [\rho, 10\rho]$.

If the best position of search scope h_* is better than the bat's current position x_i^t, the bat will fly to this position (10.9)(step 13 and 14). And also in step 15, its direction will be changed towards the next position (10.10). In step 16, the bat's wavelength will be shortened (10.7), and its frequency will be increased (10.6) so it will become the exploiter bat. As long as it has a better position in its sound waves scope it will increase intensification of the search. To avoid very small or big $\mathbf{V}_i^{t+1}$ it is normalized in equation 10.11.

$$x_i^{t+1} = h_* \tag{10.9}$$

$$\mathbf{V}_i^{t+1} = |\mathbf{x}_i^{t+1} - \mathbf{x}_i^t| \tag{10.10}$$

$$\hat{\mathbf{v}}_i = \frac{\mathbf{V}_i^{t+1}}{\|\mathbf{V}_i^{t+1}\|} \tag{10.11}$$

If h_* is worse than the bat's current position, the algorithm checks whether the current position x_i^t is the best one ever found (step 22) x_{gbest} or not (step 18).

If the current solution is not the best solution ever found then the bat becomes the explorer bat, changes its direction randomly, increases the wavelength, and decreases the frequency expanding the search scope (step 18, 19, and 20). These actions help the bat keep exploring the search space without getting trapped in a local optima and provides a random walk.

If the bat is already on the best found position (x_{gbest}) or a better one, the bat will become the exploiter bat. The wavelength will be minimized $(\lambda_i^{t+1}) = (\lambda_{min})$, the frequency will be maximized $(freq_i^{t+1}) = (freq_{max})$, and the search direction will be changed randomly (step 22, 23, and 24). As a result, the bat can increase the intensification of the search around the best position.

After repeating same steps for the second bat, the first iteration of the while loop will be over. Until the number of the max function of evaluation is reached, the bats will dynamically change their roles and keep searching.

10.3 Source-code of DVBA in Matlab

The source code of an objective function is given in Listing 10.1. The input of the objective function "x" is the positions of the bats and the $sphere(x)$ function returns the function values of the bats as a column vector. The aim of DVBA is to find the minimum value of this objective function. The objective function is given in (10.12). In Listing 10.2, the dimension of the function D is set to 10 and DVBA minimizes this function within 100.000 FEs.

$$f(x) = \sum_{i=1}^{D} x_i^2 \qquad \text{where } -5.12 \leqslant x_i \leqslant 5.12 \tag{10.12}$$

```
% Objective function f
function f = sphere(x)
    x = x.^2;
    f = sum(x,2);
end
```

Listing 10.1
Definition of objective function $f(x)$ in Matlab.

```
f = @sphere;
ub = 5.12; lb = -ub; % [ub,lb]:upper and lower bounds
%  declaration of the parameters of DVBA
j=5; k=6; b=20; beta=100; D=10; P=2; maxAssesment= 100000;
% ro: increment rate of frequency and wavelength
ro = mean((ub-lb)/beta);
Lmax = ro * 10; Lmin = ro * 0.1; % wavelength range
freqMax = ro * 10.0; freqMin = ro * 1.0; % frequency range
freq = freqMin + (freqMax - freqMin) * rand(1,P);  % frequency
lambda = Lmin + (Lmax - Lmin) * rand(1,P);  % wavelength
% initialize the bat population and directions.
x = lb + (ub-lb)*rand(P,D);
```

```
v = -1 + 2 * rand(P,D);
% find the best fitness value based on the bat's initial position.
fgbest = min(f(x));
% main program loop starts
while maxAssesment >= 1
% for each bat
    for i=1:P
% wave vectors directions are generated
        u = -1 + 2*rand(j,D);
        A = u*b/freq(i);
        V = bsxfun(@plus,v(i,:), A) ;
        V = V./norm(V);
% search scope vector is generated for the bat
        H = []; % search scope column vector
        for r = 1:j     %  wave vectors
            for z = 1:k   % search points on the wave vector
                xtrans = x(i,:) + V(r,:)*z*lambda(i);
% keep the search points in the search scope [ub, lb]
                xtrans(xtrans>ub)=ub;
                xtrans(xtrans<lb)=lb;
% add each search points to the search scope matrix of the bat
                H = [H ; xtrans];
            end
        end
% compare the current position with the search scope points first
        fnew = min(f(H));
        maxAssesment = maxAssesment-(k*j + 5);
% in each loop, (k*j + 5) times FEs are done
        idx = find(f(H)==fnew);
        xnew = H(idx(1),:);
        xtemp = x(i,:); % assign xtemp to use for the next steps
% the bat becomes the EXPLOITER bat,
        if fnew < f(x(i,:))
% its direction will be changed towards the next position
            v(i,:) = (xnew - x(i,:))./norm(xnew - x(i,:));
            x(i,:) = xnew;
% shrink the search scope of the bat within the ranges
            if lambda(i) > Lmin+ro
                lambda(i) = lambda(i) - ro;
            else
                lambda(i) = Lmin;
            end
            if freq(i) < freqMax - ro
               freq(i) = freq(i) + ro;
            else
                freq(i) = freqMax;
            end
        end
% the bat becomes the EXPLORER bat
        if  f(xtemp) > fgbest
% change its direction randomly
            v(i,:) = -1 + 2 * rand(1,D);
% expand the search scope of the bat
            if lambda(i) < Lmax - ro
                lambda(i) = lambda(i) + ro;
            end
            if freq(i) > freqMin + ro
                freq(i) = freq(i) - ro;
            end
        end
% the bat is already on the best found position
        if f(xtemp) <= fgbest && f(xtemp) < fnew
% change its direction randomly and shrink the search scope
            v(i,:) = -1 + 2 * rand(1,D);
            lambda(i) = Lmin; freq(i) = freqMax;
        end
% update best values: fgbest, xgbest
```

```
        if f(x(i,:)) <= fgbest
            fgbest = f(x(i,:)); xgbest = x(i,:);
        end
    end
end
% the result of DVBA is returned
disp(fgbest)
```

Listing 10.2
Source-code of DVBA in Matlab.

10.4 Source-code of DVBA in C++

```
#include <iostream>
#include <cstdlib>
#include <ctime>
using namespace std;
// definition of the objective function f(x)
float ub=5.12; float lb=-5.12; //ub, lb: upper and lower bounds.
float f(float x[], int size_array){
   float t = 0;
   for(int i=0; i<size_array; i++){
      t=t+x[i]*x[i];}
   return t;}
// generate pseudo random values from the range [0, 1)
float r(){
   return float(rand()%1000)/1000;}
// "checkBoundary" function to check the constraints:
float checkBoundary(float x, float min, float max){
   if (x<min) x = min;
   if (x>max) x = max;
   return x;}
// main program function
int main(){
srand(time(0));
// initialization of the parameters
int P=2; int maxAssessment=100000; int assessment=0; int D=10;
int beta=100; int j=5; int k=6; int b=20;
int theBest=0;
float xmin[D]; float xmax[D]; float rho[D];
float freqMax[D],freqMin[D],freq[P][D];
float Lmax[D], Lmin[D], lambda[P][D];
// initialization of the constraints
 for (int z=0; z<D; z++){
    xmin[z] = -5.12; xmax[z] = 5.12;
    rho[z] = (xmax[z] - xmin[z])/beta;
    freqMax[z] = 10*rho[z]; freqMin[z] = rho[z];
    Lmax[z] = 10*rho[z]; Lmin[z] = 0.01*rho[z];}
// initialization of all data structures which are needed by DVBA
   float x[P][D]; float xtemp[D]; float xnew[D];
   float v[P][D];float xgbest[D];
   float u[j][D], A[D][j], V[j][D], H[j*k][D];
// randomly create bat's positions, velocities, frequency(freq) and
    wavelength(lambda)
    for(int i=0; i<P; i++){
       for(int z=0; z<D; z++){
        x[i][z]=(xmax[z]-xmin[z]) * r() + xmin[z];
        v[i][z] = (-1 + 2 * r());
        freq[i][z] = (freqMax[z]-freqMin[z]) * r() + freqMin[z];
        lambda[i][z] = (Lmax[z]-Lmin[z]) * r() + Lmin[z];}
// evaluate the bats, find the best bat
       if (f(x[i],D)<f(x[theBest],D)) theBest=i;}
```

```
// assign the best bat to xgbest
    for (int i=0; i<D; i++) xgbest[i]=x[theBest][i];
// main program loop:
    while(assessment<maxAssessment){
// for each bat
        for(int i=0; i<P; i++){
            assessment = assessment + (k*j);
            int n=0, theHbest=0;
// generate the search scope matrix for the bat:
            for(int z=0; z<j; z++){  // wave vectors
             for(int m=0; m<k; m++){ //search points
                 for(int s=0; s<D; s++){ // for each dimension
                  u[z][s]=-1+2*r();
                  A[z][s]=b*u[z][s]/freq[i][s];
                  V[z][s]=v[i][s]+A[z][s];
                  V[z][s]=(V[z][s]-xmin[s])/(xmax[s]-xmin[s]);
                  H[n][s]=x[i][s]+(z+1)*V[z][s]*lambda[i][s];
                  H[n][s]=checkBoundary(H[n][s],xmin[s],xmax[s]);}
// find the best position on the bat's search scope xnew:
                 if (f(H[n],D)<f(H[theHbest],D)) theHbest = n;
                 n = n + 1;}}
            for (int u=0; u<D; u++) xnew[u] = H[theHbest][u];
            for (int t=0;t<D;t++) xtemp[t] = x[i][t];
// the bat becomes the EXPLOITER bat,
             if (f(H[theHbest],D) < f(x[i],D)){
               for(int t=0; t<D; t++){
// move towards the next position, shrink the search scope
                 v[i][t]=((xnew[t]-x[i][t]))/(xmax[t]-xmin[t]);
                 x[i][t]=xnew[t];
                 lambda[i][t]=checkBoundary(lambda[i][t]-rho[t],Lmin[t],
    Lmax[t]);
                 freq[i][t]=checkBoundary(freq[i][t]+rho[t],freqMin[t],
    freqMax[t]);}}
// the bat becomes the EXPLORER bat
            if (f(xtemp,D) > f(x[theBest],D)){
// change its direction randomly, expand the search scope
             for(int t=0; t<D; t++){
               v[i][t] = (-1 + 2 * r());
               lambda[i][t]=checkBoundary(lambda[i][t]+rho[t],Lmin[t],
    Lmax[t]);
                 freq[i][t]=checkBoundary(freq[i][t]-rho[t],freqMin[t],
    freqMax[t]);}}
// the bat is already on the best found position
           if (f(xtemp,D)<=f(x[theBest],D)&&f(xtemp,D)<f(xnew,D)){
// change its direction randomly, expand the search scope
             for(int t=0; t<D; t++){
                v[i][t] = (-1 + 2 * r());
                lambda[i][t] = Lmin[t];
                freq[i][t] = freqMax[t];}}
// update best values: fbest, xbest
            if (f(x[i],D) <= f(x[theBest],D)) theBest = i;
            for (int t=0; t<D; t++) xgbest[t] = x[theBest][t];}
         cout<<"fbest = "<<f(x[theBest],D)<<endl;}
    cout<<"xgbest = ";
    for (int t=0;t<D;t++) cout<<xgbest[t]<<" ";}
```

Listing 10.3
Source-code of the DVBA in C++.

10.5 Step-by-step numerical example of DVBA

We will use DVBA to minimize the function given by equation 10.12. For simplification purposes, the dimension will be set to 3, $j = 2$, and $k = 3$. So there will be only two wave vectors, three search points on each of them, and

in total $j \times k = 6$ points in the search scope of the bats. Search scope wideness variable b is set to 20, and increment rate divisor is set to 100. In the first step all these parameters will be set.

In the second step, increment rate ρ will be calculated by using equation 10.8, then the wavelength and the frequency will be initialized randomly for each bat within the ranges mentioned previously: $\lambda \in [0.1\rho, 10\rho]$ and the $freq \in [\rho, 10\rho]$.

$\rho = 0.1024, \quad stepsize$

$[freqMin, freqMax] = [0.1024 \quad 1.024], \quad frequency(freq) \quad range$

$[Lmin, Lmax] = [0.01024 \quad 1.0240], \quad wavelength(lambda) \quad range$

$freq1 = 0.5527$

$freq2 = 1.0065$

$lambda1 = 0.9682$

$lambda2 = 0.1649$

In the third step, two bats will start the search from random positions within the range of the search space $[-5.12 \quad 5.12]$ with random velocities(directions) in the range of (-1,1). And the best function value($fgbest$) will be updated based on the bats' current positions.

$x1 = [5.0652, \quad -4.2795, \quad -1.9859]$

$x2 = [1.3005, \quad -0.5550, \quad -0.5254]$

$v1 = [-0.4553, \quad 0.9136, \quad 0.7642]$

$v2 = [-0.5224, \quad 0.6805, \quad 0.8534]$

$$f(x1) = \sum_{i=1}^{3} x_i^2 = [5.0652^2 + (-4.2795)^2 + (-1.9859)^2] = 47.9140$$

$$f(x2) = \sum_{i=1}^{3} x_i^2 = [1.3005^2 + (-0.5550)^2 + (-0.5254)^2] = 2.2755$$

$fgbest = min(f) = 2.2755$

In the fourth step the main loop of the algorithm starts. If the algorithm termination condition is fulfilled we jump to the fourteenth step, if not we continue to step five.

In the fifth step, the first bat's search scope column vector(H) will be generated by using the equations from 10.1 to 10.4. Since there are just two wave vectors there will be only $A11$, $A12$, $V11$, and $V12$ vectors for the search scope.

$u = -1 + 2 * rand(j, d)$

$u1 = [0.0309, \quad 0.0278, \quad -0.0334]$

$u2 = [-0.0299, \quad 0.0252, \quad -0.0177]$

$A11 = u1 * b/freq1 = [1.1198, \quad 1.0074, \quad -1.2079]$
$A12 = u2 * b/freq1 = [-1.0808, \quad 0.9105, \quad -0.6400]$
$V11 = norm(v1 + A11) = [0.2420, \quad 0.6996, \quad -0.1616]$
$V12 = norm(v1 + A12) = [-0.5594, \quad 0.6644, \quad 0.0453]$

$H1 = x1 + V11 * 1 * lambda1 = [5.1200, \quad -3.6021, \quad -2.1424]$
$H2 = x1 + V11 * 2 * lambda1 = [5.1200, \quad -2.9247, \quad -2.2989]$
$H3 = x1 + V11 * 3 * lambda1 = [5.1200, \quad -2.2473, \quad -2.4553]$
$H4 = x1 + V12 * 1 * lambda1 = [4.5235, \quad -3.6362, \quad -1.9421]$
$H5 = x1 + V12 * 2 * lambda1 = [3.9819, \quad -2.9930, \quad -1.8983]$
$H6 = x1 + V12 * 3 * lambda1 = [3.4402, \quad -2.3497, \quad -1.8545]$

$H1$, $H2$, and $H3$ are the search points on the first wave vector, and the last three are the search points on the second wave vector.

In the sixth step, all the positions in the first bat's search scope (H) will be evaluated by using the objective function, the best value and the position will be assigned as $fnew$ and $xnew$, respectively.

$$f(H1) = \sum_{i=1}^{3} H_i^2 = [5.1200^2 + (-3.6021)^2 + (-2.1424)^2] = 43.7792$$

if we repeat the same competition for all the search points in the scope, we obtain f(H) values:

$f(H1) = 43.7792$
$f(H2) = 40.0529$
$f(H3) = 37.2932$
$f(H4) = 37.4563$
$f(H5) = 28.4168$
$f(H6) = 20.7954$
$fnew = min(f(H)) = f(H6) = 20.7954$
$xnew = H6 = [3.4402, \quad -2.3497, \quad -1.8545]$
$xtemp = x1$

$x1$ is assigned to $xtemp$, so we can use $xtemp$ for the next steps as $x1$'s not-updated value.

In the seventh step, $fnew$ and the bat's position $f(x1)$ will be compared and the bat's role will be decided. In this case, $fnew$ is better than the first bat's current position ($20.7954 < 47.9140$), so the first bat will become the exploiter bat. It will move to $xnew$, its direction will be changed towards the

next position, and its search scope will be shrunk by decreasing its wavelength ($lambda1$) and increasing its frequency ($freq1$).

$v1 = (xnew - x1)/norm(xnew - x1) = [-0.6432, \quad 0.7639, \quad 0.0520]$
$x1 = xnew = [3.4402, \quad -2.3497, \quad -1.8545]$
$lambda1 = lambda1 - \rho = 0.8658$
$freq1 = freq1 + \rho = 0.6551$

Since $fnew$ and $f(xtemp)$ are not better than $fgbest$, $fgbest$ is not updated, only $x1$ is updated, and the first bat's evaluation is over here.

In the eighth step, we return to the fourth step and repeat the same competition for the second bat. Firstly, its search scope column vector (H) will be generated as follows:

$u = -1 + 2 * rand(j, d)$
$u1 = [-0.0292, \quad -0.0081, \quad -0.0337]$
$u2 = [0.0002, \quad -0.0103, \quad 0.0184]$
$A21 = u1 * b/freq2 = [-0.5812, \quad -0.1617, \quad -0.6693]$
$A22 = u1 * b/freq2 = [0.0039, \quad -0.2046, \quad 0.3659]$
$V21 = norm(v2 + A21) = [-0.6572, \quad 0.3089, \quad 0.1096]$
$V22 = norm(v2 + A22) = [-0.3087, \quad 0.2834, \quad 0.7261]$

$H1 = x2 + V21 * 1 * lambda2 = [1.1922, \quad -0.5041, \quad -0.5073]$
$H2 = x2 + V21 * 2 * lambda2 = [1.0839, \quad -0.4531, \quad -0.4892]$
$H3 = x2 + V21 * 3 * lambda2 = [0.9755, \quad -0.4022, \quad -0.4712]$
$H4 = x2 + V22 * 1 * lambda2 = [1.2497, \quad -0.5083, \quad -0.4057]$
$H5 = x2 + V22 * 2 * lambda2 = [1.1988, \quad -0.4616, \quad -0.2860]$
$H6 = x2 + V22 * 3 * lambda2 = [1.1479, \quad -0.4149, \quad -0.1663]$

In the ninth step we repeat the sixth step for the second bat's search scope H:

$$f(H1) = \sum_{i=1}^{3} H_i^2 = [1.1922^2 + (-0.5041)^2 + (-0.5073)^2] = 1.9328$$

if we repeat the same competition for all the search points in the scope we obtain f(H) values:

$f(H1) = 1.9328$
$f(H2) = 1.6195$
$f(H3) = 1.3354$
$f(H4) = 1.9846$

$f(H5) = 1.7318$
$f(H6) = 1.5173$
$fnew = min(f(H)) = f(H3) = 1.3354$
$xnew = H3 = [0.9755, \quad -0.4022, \quad -0.4712]$
$xtemp = x2$

In the tenth step, $fnew$ and the bat's position $f(x2)$ will be compared and the bat's role will be decided. In this case, $fnew(1.3354)$ is better than $f(x2)(2.2755)$. So the first bat will become the exploiter bat as well. It will move to $xnew$, its direction will be changed towards the next position, and its search scope will be shrunk by decreasing its wavelength ($lambda2$) and increasing its frequency ($freq2$).

$v2 = (xnew - x2)/norm(xnew - x2) = [-0.8949, \quad 0.4206, \quad 0.1493]$
$x2 = xnew = [0.9755, \quad -0.4022, \quad -0.4712]$
$lambda2 = lambda2 - \rho = 0.0625$
$freq2 = freq2 + \rho = 1.1089 = 1.0240$

$freq2$ was bigger than $freqMax$, so it is constrained by $freqMax = 1.0240$.

In the eleventh step, we compare $f(xtemp)$ with $fgbest$ whether the bat is already on the best found position or not. The bat's current position $f(x2)$ is better than $fgbest$, so steps 23 and 24 will be executed in Algorithm 9: The wavelength will be minimized, the frequency will be maximized, and the bat's search direction will be changed randomly. So its search scope will be at its minimum size.

$lambda2 = Lmin = 0.0102$
$freq2 = freqMax = 1.0240$
$v2 = -1 + 2 * rand(1, D) = [0.9415, \quad -0.2689, \quad -0.1303]$

In the twelfth step, since $fnew$ is better than $fgbest$, $fgbest$ and $xgbest$ will be updated as follows:

$fgbest = fnew = 1.3354$
$xgbest = xnew = [0.9755, \quad -0.4022, \quad -0.4712]$

At the end of the twelfth step, the both bats are evaluated and the first bat $x1$ moved from 47.9140 to 20.7954, and the second bat $x2$ moved from 2.2755 to 1.3354. Until now, we spent $2 * (j \times k) + 10 = 32$ function of evaluations(FEs). And we have still around $9,700$ FEs to stop the algorithm.

In the thirteenth step, we return to the fourth step.

In the fourteenth step, after the function of evaluations (FEs) is exhausted we return the fgbest and xgbest as a result of the algorithm.

10.6 Conclusions

In this chapter we have aimed to show how a bat's behaviour in nature can be modeled and become an optimization algorithm. And we have shown step-by-step how the algorithm works. We have also provided source-codes in MATLAB and in C++ programming language which could help researchers to implement the algorithm on optimization problems and work on improving the algorithm. Finally, the step-by-step numerical example of DVBA has been presented in detail, so we can see how virtual bats improve the solution during the competition process.

References

1. A.O. Topal, O. Altun. "Dynamic virtual bats algorithm (DVBA) for global numerical optimization," in *Proc. of the IEEE International Conference on Congress on Intelligent Networking and Collaborative Systems (INCoS)*, 2014, pp. 320-327 .
2. A.O. Topal, O. Altun. "A novel meta-heuristic algorithm: Dynamic Virtual Bats Algorithm." *Information Sciences*, vol. 354 (2016): 222-235.
3. X.-S. Yang, "A new metaheuristic bat-inspired algorithm", in: *Nature Inspired Cooperative Strategies for Optimization (NICSO 2010)*, Springer, 2010, pp. 65-74.
4. C. Yu, L. Kelley, S. Zheng, Y. Tan, "Fireworks algorithm with differential mutation for solving the CEC 2014 competition problems", in *Proc. of the IEEE Congress on Evolutionary Computation (CEC)*, IEEE, 2014, pp. 3238-3245.
5. A.O. Topal, Y.E. Yildiz, M. Ozkul. "Improved dynamic virtual bats algorithm for global numerical optimization." *Proc. of the World Congress on Engineering and Computer Science.* Vol. 1, pp. 462-467, 2017.