19

Monarch Butterfly Optimization

Pushpendra Singh
Department of Electrical Engineering
Govt. Women Engineering College, Ajmer, India

Nand K. Meena
School of Engineering and Applied Science
Aston University, Birmingham, United Kingdom

Jin Yang
School of Engineering and Applied Science
Aston University, Birmingham, United Kingdom

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

CONTENTS

19.1 Introduction 249
19.2 Monarch butterfly optimization 250
19.2.1 Migration operator 251
19.2.2 Butterfly adjusting operator 252
19.3 Algorithm of monarch butterfly optimization 253
19.4 Source-code of MBO algorithm in Matlab 254
19.5 Source-code of MBO algorithm in C++ 256
19.6 Step-by-step numerical example of MBO algorithm 258
19.7 Conclusion 262
Acknowledgement 262
References 262

19.1 Introduction

The natural inspired algorithms (NIAs) influenced by nature; these are found to be very effective and popular in the area of mathematics, computer science and decision making variables. Generally, the natural inspired algorithm

can be divided into two categories, swarm based algorithm (SBAs) and evolutionary algorithm (EAs). The SBAs are a well-known model of NIA and are widely used techniques [1] in diversified areas of science and engineering. Many of such optimization techniques are inspired by the behaviour of cuckoos, chicken, honey bees, bats, etc. These methods are utilizing the swarm-intelligence for collective and interacting agents which are based on some set of defined rules [2]. The popularly known SBAs may include particle swarm optimization (PSO) [3], ant colony optimization (ACO) [4] etc. Similarly, the EAs are based on the prototype of biology and natural evolution; the popularly known EAs may include evolutionary programming (EP), genetic algorithm (GA), differential evolution (DE), and evolutionary strategy (ES) [5].

In the year 2015, Wang et al. [7] developed a new swarm-intelligence based optimization technique. The monarch butterfly, considered to be the most beautiful of butterflies, is known as the king of butterflies and is therefore named "monarch". The monarch butterfly is a species of insects generally found in the North American region. This creature has black and orange patterned wings and belongs to the milkweed butterfly part of the family of Nymphalidae [6, 8]. However, some white morphs of monarch are also found in the Hawaiian Islands. The male and female butterflies have different wing creation and therefore can be differentiated easily. These eastern North American butteries are well known for their migration from the USA and the southern part of Canada, to California and then Mexico, for winter. They fly thousands of miles over the Rocky Mountains, California to Mexico. When days begin to turn longer, they start their journey back to the North. They ley eggs somewhere during the journey and this process keeps repeating until they reach to north, e.g. for 4 to 5 generations.

A recent investigation [9], on preserved male and female monarch butterflies, carried out in 2015 revealed significant differences in wing sizes and body structure. The female usually has thicker wings to provide greater tensile strength which reduces the tendencies to get hurt during migration. On average, the male has wider wings and is heavier than a female.

During the journey to Mexico, females lay their eggs to produce offspring. In their life cycle, these butterflies pass through four major stages and a four-generation cycle in one year. The four stages may include egg, caterpillar, chrysalis and adult respectively. The monarch butterfly optimisation (MBO) is inspired from the migration behaviour of monarch butterflies from one continent to other, i.e., monarch butterfly optimization (MBO). Figure 19.1 shows the movement of the monarch butterfly from one region to another. They keep their reproduction cycle active during this cross county movement [10]. The migration behaviour is modelled into some set of mathematical formulation, presented in the following section.

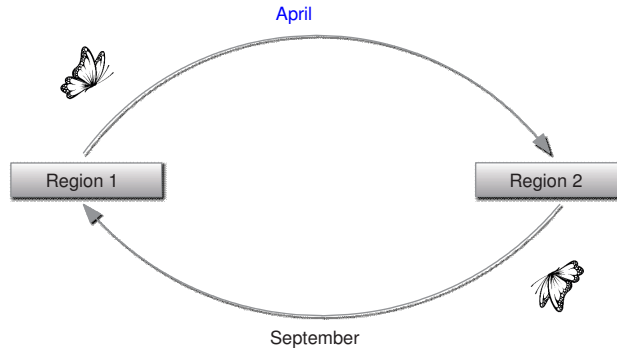

FIGURE 19.1
Migration behaviour of monarch butterflies.

19.2 Monarch butterfly optimization

In order to model the realistic migration behaviour of monarch butterflies, some set of pre-defined rules is formulated. These pre-defined rules are listed below [7].

- All the members of the monarch fleet should be located either in region 1 or 2. The complete flutter of that region is considered to be the total population in MBO.
- The new offspring of monarch butterflies in region 1 or 2 are produced by the migration operator which can be controlled by the migration ratio.
- In MBO, the total population of the monarch flutter remains unaltered. The offspring replace their respective parents, if they exhibit better fitness than their parent.
- The fittest butterfly of the flutter remains in the fleet and not affected by the migration operator.

19.2.1 Migration operator

As presented in Fig. 19.1, the monarch butterflies migrate from region 1 to 2 in the month of April every year and return in September of the same year. It is examined that the monarch butterflies flutter stays in region 1 from September to March and April to August in region 2. Suppose, the number of monarch butterflies staying in region 1 is considered as 'subpopulation 1' and represented by P_{n1}. Here, P_{n1} is determined as the nearest integer greater than or equal to $R \cdot (P_n \times P_{n1})$, where, R and P_n are the ratio of the butterfly

flutter staying in region 1 and total number of monarch butterfly population respectively. Similarly, the butterflies found in region 2 are considered to be 'subpopulation 2', $P_{n2} = P_n - P_{n1}$.

This migration process of butterflies is mathematically expressed as

$$s_{ij}(t+1) = s_{r_1,j}(t) \tag{19.1}$$

where, $s_{ij}(t+1)$, represents the jth position element/variable of the ith butterfly in $t+1$ generation, whereas, $s_{r_1,j}(t)$, represents the jth element of s_{r_1} which is the new position of monarch butterfly r_1 of current generation t. The butterfly r_1 is randomly selected from subpopulation 1 or region 1.

For decision making, a random number r is generated as follows

$$r = \psi \times rand \tag{19.2}$$

where, $rand$ is a random number obtained from the uniform distribution; whereas, ψ represents the migration period considered to 1.2 for 12 months. If $r \leq R$, then the new element j of butterfly is produced by (19.1); otherwise, the new born monarch butterfly is produced as

$$s_{ij}(t+1) = s_{r_2,j}(t) \tag{19.3}$$

where, r_2 is the randomly selected monarch butterfly from subpopulation 2 or region 2. It has been noted that by adjusting the value of R, the direction of migration can be balanced. For example, if the value of R is high then more members of monarch flutter P_{n1} will be expected. On the other hand, a low value of R increases the number of monarch butterflies P_{n2} in subpopulation 2 or region 2. The selection of R is very important in order to produce new monarch butterflies. In this chapter, the value of R is considered to be $5/12 = 0.4166$.

The important steps involved in the migration operator are listed below.

Migration operator

1. Calculate the value of subpopulation 1, i.e., P_{n1}, for region 1;
2. calculate r as suggested in (19.2). If $r \leq R$ then use (19.1) otherwise (19.3) for updating monarch butterfly of region 1;
3. a correction method may be applied to correct the infeasible positions of butterflies, if any;
4. repeat steps 1 to 3 until the positions of all individuals are updated.

19.2.2 Butterfly adjusting operator

Recently, it has been researched that monarch butterflies follow the Lévy flight movement pattern when moving from one place to the other [10]. Besides the

migration operator, the positions of monarch butterflies are also updated by a 'butterfly adjusting operator'. Mathematically it is expressed as

$$s_{ij}(t+1) = s_{best,j}(t) \tag{19.4}$$

where, $s_{ij}(t+1)$, represents the jth variable or element of the ith monarch butterfly for generation $t+1$. Besides, $s_{best,j}(t)$ represents the jth element in the best monarch butterfly of region 1 and 2 for present generation t. Similar to the migration operator, a random number *rand* is generated for decision making and then compared with R. If $rand \leq R$ then the jth position of butterfly i is updated by (19.4); otherwise it is modified as follows

$$s_{ij}(t+1) = s_{r_3,j}(t) \tag{19.5}$$

where, $s_{r_3,j}(t)$ represents the jth element of $s_{r_3,j}$, selected randomly from region 2, $r_3 \in \{1, 2, 3, \ldots, P_{n2}\}$. If *rand*>BAR (Butterfly adjustment rate) then s is further updated under this condition as suggested below

$$s_{ij}(t+1) = s_{ij}(t+1) + \alpha \cdot (ds_j - 0.5) \tag{19.6}$$

where, ds_j is a small walk step of the ith butterfly which is calculated by Lévy flight movement as

$$ds_j = L[s_j(t)] \tag{19.7}$$

In (19.6), α is representing the weighting factor, determined as

$$\alpha = S_{max}/t^2 \tag{19.8}$$

where, S_{max} is the maximum value of the walk step taken by the monarch butterfly in a single move. It means that the larger the value of α, the higher will be the influence of ds on $s_{ij}(t+1)$. So, it encourages the exploration process of the method. Furthermore, the small size of α decreases the influence of ds on $s_{ij}(t+1)$, enhancing the exploitation process of MBO.

The algorithm for the butterfly adjusting operator is presented as

Butterfly adjusting operator

1. Calculate the value of subpopulation, P_{n2};
2. calculate the value of ds_j and weighing factor α, as suggested in (19.7) and (19.8) respectively for region 2;
3. generate a random number *rand* following uniform distribution. If $rand \leq R$ then update the jth element of the ith butterfly of region 2 by (19.4), otherwise by (19.5);
4. If $rand > R$ and *rand*>BAR then further update by (19.6);
5. apply a correction technique to correct infeasible individuals;
6. repeat steps 1 to 4 until all elements of the population are updated.

19.3 Algorithm of monarch butterfly optimization

In this section, the algorithm of MBO is presented for basic understanding. The essential steps of the algorithm involved in MBO are given below.

1. Set the value of required parameters and maximum number of iterations, e.g. $S_{Max.}$, p, BAR, ψ, t etc.;
2. initialize the random but feasible population P_n of the monarch flutter;
3. calculate the fitness of each butterfly i, based on suggested position elements;
4. sort all monarch butterfly individuals based on their fitness values and then divide into two populations P_{n1} and P_{n2} which correspond to region 1 and 2 respectively;
5. for P_{n1} of region 1, generate new subpopulation by migration operator and P_{n2} of region 2 by butterfly adjusting operator, as discussed in Sections 19.2.1 and 19.2.2 respectively;
6. repeat steps 3 to 5 until convergence criteria or maximum value of iteration is reached;
7. print the best solution along with its fitness value.

The flowchart of standard MBO, discussed in the previous section, is presented in Fig. 19.2.

19.4 Source-code of MBO algorithm in Matlab

In Listing 19.1, the source-code for the MBO algorithm is presented. Here, the $OF(.)$ is representing the address of the objective function.

```
clc
clear
LB=[0 0 0]; %set upper limits of each dimension
UB=[5 5 5]; %set upper limits of each dimension
%==========MBO parameters=========================
Max_iter=50;          %maximum number of iterations.
Pn=30;                %population size.
D=length(UB);         %number of butterfly variables.
Stepmax=1.0;          %Maximum walk size
R=5/12;               %ratio of butterfly flutter...
Keep=2;               %populations keep in elitism...
phi=1.2;              %migration period
BAR=5/12;             %butterfly adjusting rate
Pn1= ceil(Pn*5/12); Pn2= Pn - Pn1;%butterflies in regions 1&2
Lnd1=zeros(Pn1,D); Lnd2=zeros(Pn2,D);%flutter of regions 1&2
s=[Lnd1;Lnd2];          %total flutter in both regions
fitness=zeros(Pn,1); %fitness values
```

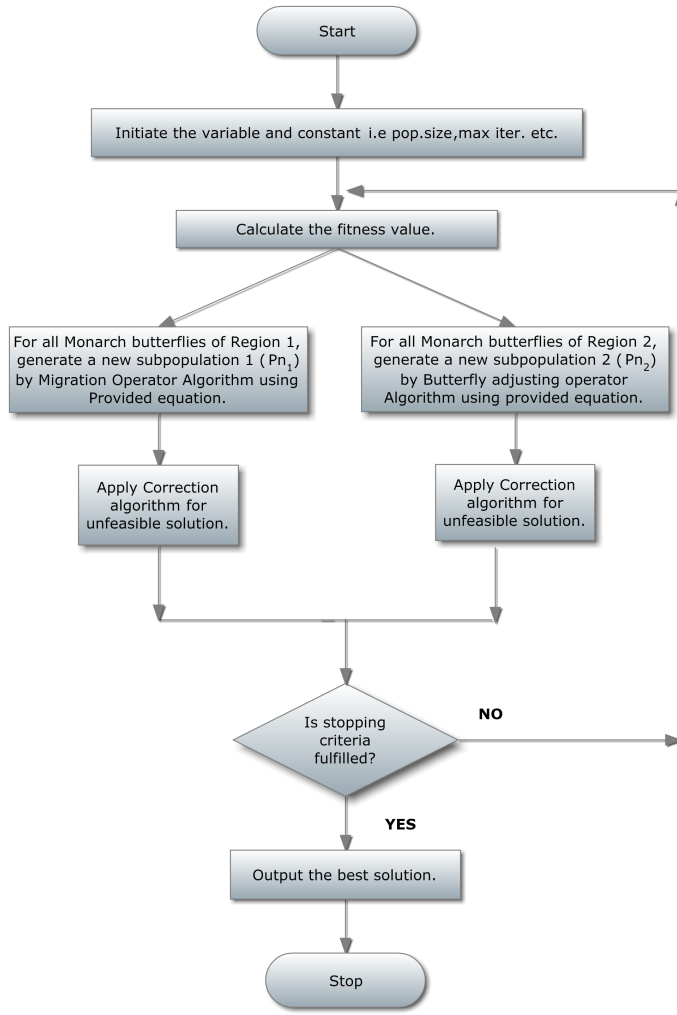

FIGURE 19.2
Flow chart of monarch butterfly optimization.

```
%=====================initialization=================
for i = 1:Pn
for j = 1:D
s(i,j) = LB(j) + rand*(UB(j)-LB(j)); %random populations
end
fitness(i) = OF(s(i,:));        %fitness calculations...
end
best_fit = min(fitness);      %best butterfly fitness
nn_best = find(best_fit==fitness); %position of best butterfly
s_best = s(nn_best(1),:);     %best butterfly
iter = 0;                     %initialize the iterations.
while iter < Max_iter         %iterations start here...
```

```
iter = iter + 1;              %update iteration
COMB_POP=[s, fitness];
s_srt=sortrows(COMB_POP, D+1);%sort the population
s_keep = s_srt(1:Keep,:);     %elitism...
s_srt((Pn-Keep+1):Pn,:)=s_keep;
s = s_srt(:,1:D);             %replace by sorted population
fitness = (s_srt(:,D+1));     %replace by sorted fitness
%=========Split population into two parts=============
pop1=s(1:Pn1,:);            %butterflies pop in region 1
pop2=s(Pn1+1:Pn,:);         %butterflies pop in region 2
%================Migration operator==================
for i =1:Pn1
for j = 1:D
r=phi*rand;      %decision making....
if r<=R
r1=randi([1 Pn1],1,1);
Lnd1(i,j)=pop1(r1,j);
else
r2=randi([1 Pn2],1,1);
Lnd1(i,j)=pop2(r2,j);
end
end
end
%============Butterfly adjustment operator===========
for i=1:Pn2
alpha=Stepmax/(iter)^2;   %weighting factor
stepsize=ceil(exprnd(2*Max_iter,1,1)); %step size
ds=LevyFlight(stepsize,D);   %walk step
for j=1:D
if rand<=R
Lnd2(i,j)=s_best(1,j);
else
r3=randi([1 Pn2],1,1);
Lnd2(i,j)=pop2(r3,j);
if rand> BAR
Lnd2(i,j)=Lnd2(i,j)+ alpha*(ds(j)-0.5);
end
end
end
end
%combined updated butterflies population
s_new=[Lnd1;Lnd2];     %combined butterfly population
for i = 1:Pn
[fitness(i)]=OF(s_new(i,:)); %Fitness calculations
end
%============Preserve the best solution=========
best_fit1 = min(fitness);%find best butterfly in new pop
nn_best = find(best_fit1==fitness); %location of best butterfly
s_best1 = s_new(nn_best(1),:);       %best butterfly of new pop
if best_fit1 < best_fit %replace if found better than this
best_fit = best_fit1;
s_best = s_best1;
end
s = s_new;           %replace the pop with new pop
end
disp(best_fit)
disp(s_best)
```

Listing 19.1
Source-code of MBO in Matlab.

19.5 Source-code of MBO algorithm in C++

```
#include <iostream>
#include <math.h>
#include <time.h>
#include <algorithm>
using namespace std;
//--- function for swapping two values in the table
void swape(float *xp, float *yp)
{float temp=*xp; *xp=*yp; *yp=temp;}
//--- function for random generation of number in exp distribution
double ran_expo(float lambda)
{double u; u = rand()/(RAND_MAX + 1.0);
return -log(1-u)/lambda;}
//--- definition of objective function
float OF(float x[])
{float Fit=pow((x[0]-0.2),2)+pow((x[1]-1.7),2)+pow((x[2]-5.1),2);
return Fit;}
//--- function for generation of random values from the range [0, 1)
float ra(){return (float)(rand()%1000)/1000;}
//--- function for generation of Levy-Flight values
float LF(int StepSize)
{float suma=0;
for(int j=0; j<StepSize; j++){
float fx=tan(M_PI * ra());
suma=suma+fx;}
return suma;}
//--- main function
int main()
{   srand(time(NULL));
int Max_iter=50, Pn=30, D=3, Keep=2, Pn1=ceil(Pn*5/12);
int Pn2=Pn-Pn1, nr_fit[Pn];
float LB[3]={0, 0, 0}, UB[3]={5, 5, 5}, Stepmax=1;
float R=5/12, phi=1.2, BAR=5/12, s_best[D];
float Lnd1[Pn1][D], Lnd2[Pn2][D], s[Pn1+Pn2][D], fitness[Pn];
float s_srt[Pn1+Pn2][D];
//--- initialization
for(int i=0; i<Pn; i++){
for(int j=0; j<D; j++){s[i][j]=LB[j]+ra()*(UB[j]-LB[j]);}
fitness[i]=OF(s[i]); nr_fit[i]=i;}
float best_fit=*min_element(fitness,fitness+Pn);
int nn_best=min_element(fitness, fitness+Pn)-fitness;
for(int i=0; i<D; i++){s_best[i]=s[nn_best][i];}
int iter=0;
while(iter<Max_iter)
{   iter++;
for(int i=0; i<Pn; i++){
for(int j=0; j<Pn-i; j++){
if (fitness[j]>fitness[j+1]){
swape(&fitness[j],&fitness[j+1]);
int tmp=nr_fit[j];
nr_fit[j]=nr_fit[j+1];
nr_fit[j+1]=tmp;}}}
for(int i=0; i<Pn; i++){
for(int j=0; j<D; j++){
s_srt[i][j]=s[nr_fit[i]][j];}}
for(int i=0; i<Keep; i++){
for(int j=0; j<D; j++){
s_srt[Pn-1-i][j]=s_srt[i][j];
fitness[Pn-1-i]=fitness[i];}}
for(int i=0; i<Pn; i++){
for(int j=0; j<D; j++){
s[i][j]=s_srt[i][j];}}
//--- divide the whole population into two parts
```

```
float pop1[Pn1][D], pop2[Pn2][D];
for(int i=0; i<Pn1;i++){
for(int j=0; j<D; j++){pop1[i][j]=s[i][j];}}
for(int i=0; i<Pn2;i++){
for(int j=0; j<D; j++){pop2[i][j]=s[Pn1+i][j];}}
//--- migration operator
for(int i=0; i<Pn1; i++){
for(int j=0; j<D; j++){
float r=phi*ra();
if (r<=R){int r1=rand()%Pn1+1;
Lnd1[i][j]=pop1[r1][j];}
else {int r2=rand()%Pn2+1;
Lnd1[i][j]=pop2[r2][j];}}}
//--- butterfly adjustment operator
for(int i=0; i<Pn2; i++){
float alpha=Stepmax/(iter*iter);
float stepsize=ceil(ran_expo(2*Max_iter));
for (int j=0; j<D; j++){
if(ra()<=R){
Lnd2[i][j]=s_best[j];}
else {int r3=rand()%Pn2+1;
Lnd2[i][j]=pop2[r3][j];
if (ra()>BAR){
Lnd2[i][j]=Lnd2[i][j]+alpha*(LF(stepsize)-0.5);}}}}
//--- combined updated butterflies population
float s_new[Pn1+Pn2][D];
for(int i=0; i<Pn1; i++){
for(int j=0; j<D; j++){s_new[i][j]=Lnd1[i][j];}}
for(int i=0; i<Pn2; i++){
for(int j=0; j<D; j++){s_new[Pn1+i][j]=Lnd2[i][j];}}
for(int i=0; i<Pn; i++){
fitness[i]=OF(s_new[i]);}
//--- preserve the best solution
float best_fit1=*min_element(fitness,fitness+Pn);
int nn_best=min_element(fitness, fitness+Pn)-fitness;
if (best_fit1<best_fit){
best_fit=best_fit1;
for(int i=0; i<D; i++){s_best[i]=s_new[nn_best][i];}}
for(int i=0;i<Pn;i++){
for(int j=0; j<D; j++){s[i][j]=s_new[i][j];}}
//--- print the best result in each iteration
cout<<"Iteration: "<<iter<<" - The Best: "<<best_fit<<endl;}
getchar();
return 0;
}
```

Listing 19.2
Source-code of MBO algorithm in C++.

19.6 Step-by-step numerical example of MBO algorithm

Example 4 *Determine the global minima of function* $f(x)$*, where* $x_i \epsilon [0,5]\ \forall\ i = 1,2,3$

$$f(x) = (x_1 - 0.2)^2 + (x_2 - 1.7)^2 + (x_3 - 5.1)^2 \tag{19.9}$$

Solution: In this problem, the optimal values of three variables x_1, x_2, x_3 have to be determined for the minima of $f(x)$. Listing 19.3 presents the Matlab source-code for function, $F(x)$, used in optimization.

```
function [Fit] = OF(x)
Fit=(x(1)-0.2)^2 + (x(2)-1.7)^2 +(x(3)-5.1)^2;
```

Listing 19.3
Definition of function $OF(.)$ in Matlab.

In this problem, x is a 3-dimensional column vector. In the first step, we set upper and lower limits of x for all dimensions as $UB = [5, 5, 5]$ and $LB = [0, 0, 0]$.

In the second step, we set the values of the algorithm parameters. In this example, we assume that maximum step size, $S_{max} = 1$; population size, $P_n = 6$; partition ratio $R = 5/12 = 0.4167$; subpopulation 1, $P_{n1} = ceil(Pn * R) = 3$; subpopulation 2, $P_{n2} = P_n - P_{n1} = 3$; butterfly adjusting rate, $BAR = 5/12 = 0.4167$; migration period, $\psi = 1.2$.

For the third step, we initialise a random but feasible population of P_n monarch butterflies as shown below.
$x_1 = \{0.1383\ 2.8725\ 3.0034\}$
$x_2 = \{0.3189\ 2.2329\ 3.0163\}$
$x_3 = \{2.5637\ 1.4946\ 1.3914\}$
$x_4 = \{1.8562\ 1.1483\ 3.0960\}$
$x_5 = \{3.5239\ 4.0837\ 4.3452\}$
$x_6 = \{3.4696\ 3.3391\ 4.7787\}$

In the fourth step, the fitness values of each butterfly is calculated by using $OF(.)$, as illustrated below.
$f_1 = OF(x_1) = 5.774$
$f_2 = OF(x_2) = 4.639$
$f_3 = OF(x_3) = 19.383$
$f_4 = OF(x_4) = 7.063$
$f_5 = OF(x_5) = 17.300$
$f_6 = OF(x_6) = 13.480$

In the fifth step, the best butterfly is identified based on best fitness value. We observed that butterfly x_2 is the best butterfly, as our goal is to minimise the function. Therefore,
$f_{best} = 4.639$
$x_{best} = \{0.3189, 2.2329, 3.0163\}$

In the sixth step, the main loop of the algorithm starts with iteration. We check whether the algorithm termination condition is fulfilled. If yes, we jump to step seventeen. If no, we move to step seven.

In step seven, we sort the population of butterflies according to their fitness values (best to worst fitnesses), as is done below.
$x_1 = \{0.3189, 2.2329, 3.016\}$ $f_1 = 4.639$
$x_2 = \{0.1383, 2.8725, 3.003\}$ $f_2 = 5.774$

$x_3 = \{1.8562, 1.1483, 3.096\}$ $f_3 = 7.063$
$x_4 = \{3.4696, 3.3391, 4.778\}$ $f_4 = 13.480$
$x_5 = \{3.5239, 4.0837, 4.345\}$ $f_5 = 17.300$
$x_6 = \{2.5637, 1.4946, 1.391\}$ $f_6 = 19.383$

In the eighth step, split the sorted butterflies population into two parts. The upper half will be known as subpopulation 1(P_{n1}):
$Region1_1 = x_1 = \{0.3189, 2.2329, 3.0163\}$; $fRegion1_1 = f_1 = 4.639$
$Region1_2 = x_2 = \{0.1383, 2.8725, 3.0034\}$; $fRegion1_2 = f_2 = 5.774$
$Region1_3 = x_3 = \{1.8562, 1.1483, 3.0960\}$; $fRegion1_3 = f_3 = 7.063$
The second half will be known as subpopulation 2(P_{n2}) :
$Region2_1 = x_4 = \{3.4696, 3.3391, 4.7787\}$; $fRegion2_1 = f_4 = 13.480$
$Region2_2 = x_5 = \{3.5239, 4.0837, 4.3452\}$; $fRegion2_2 = f_5 = 17.300$
$Region2_3 = x_6 = \{2.5637, 1.4946, 1.3914\}$; $fRegion2_3 = f_6 = 19.383$

In the ninth step, we apply the migration operator to update positions of butterflies in subpopulation 1 (P_{n1}).
To do this, we generate a random number, e.g. $rand = 0.7750$, and then determine r as, $r = \psi \times rand = 1.2 \times 0.7750 = 0.9300$. Here, $r > R$ so $Region1_{1,1}$ is updated by (19.3), as illustrated below.
Now, choose a butterfly randomly from subpopulation 2 i.e., r_2. Let's assume $r_2 = 2$; then $Region1_{1,1}^{new}$ is updated as, $Region1_{1,1}^{new} = Region2(r_2, 1) = 3.5239$

For the second dimension, again we generate a random number, e.g. $rand = 0.1262$; then $r = 1.2 \times 0.1262 = 0.1514$. This time $r < R$ therefore $Region1_{1,2}$ will be updated as suggested in (19.1). Now, we choose a butterfly from subpopulation 1 randomly, i.e., r_1. We assume that $r_1 = 1$; then $Region1_{1,2}$ is updated as $Region1_{1,2}^{new} = Region1(r_1, 2) = 2.2329$.
For the third dimension, we repeat the process and assume that $rand = 0.2172$ and then $r = 1.2 \times 0.2172 = 0.2607$. Again, $r < R$ therefore, a butterfly is selected randomly from subpopulation 1. Let's assume that $r_1 = 3$; then $Region1_{1,3}$ is updated as, $Region1_{1,3}^{new} = Region1(r_1, 3) = 3.0960$.
Finally, $Region1_1^{new}$ can be expressed as $Region1_1^{new} = \{3.5239\ 2.2329\ 3.0960\}$

Similarly, other butterflies are updated and given below:
$Region1_2^{new} = \{3.4696\ 3.3391\ 3.0960\}$
$Region1_3^{new} = \{0.1383\ 2.8725\ 3.0034\}$

In the tenth step, we apply the butterfly adjustment operator to update butterflies in subpopulation 2(P_{n2}).
For this, we determine $\alpha = Stepmax/t^2 = 1/(1^2) = 1$ and ds by using the Levy distribution. Let's assume that $ds = \{0.2857\ -7.2522\ -3.4262\}$.
Now, generate a random number, e.g., $rand = 0.3753$, as $rand < R$; then $Region2_{1,1}$ is updated by using (19.4). Therefore,
$Region2_{1,1}^{new} = x_{best}(1) = 0.3189$

For the second dimension $j = 2$, we again generate a random number, e.g. $rand = 0.6950$. This time, $rand > R$ thus $Region2_{1,2}$ will be updated by (19.5), as illustrated here.
We randomly select a butterfly r_3 from subpopulation 2. Let's assume $r_3 = 3$; then $Region2_{1,2}$ is updated as
$Region2^{new}_{1,2} = Region2_{r_3,2} = Region2_{3,2} = 1.4946$.
Now, we generate a random number, e.g. $rand = 0.9289$. If $rand > BAR$ then $Region2^{new}_{1,2}$ is further updated by (19.6), as illustrated here.
$Region2^{new}_{1,2} = Region2^{new}_{1,2} + \alpha(ds(2) - 0.5) = 1.4946 + 1.(-7.2522 - 0.5) = -6.2576$.
It is observed that this variable crosses the boundary limits $[0,\ 5]$ therefore, correction is applied as
$Region2^{new}_{1,2} = LB(2) + rand(UB(2) - LB(2)) = 0 + 0.7613(5 - 0) = 3.8065$.

For the third dimension $j = 3$, we repeat the process by generating a random number, e.g. $rand = 0.1903$ which is less than R therefore, $Region2_{1,3}$ is updated by (19.4) as: $Region2^{new}_{1,3} = x_{best}(3) = 3.0163$. Therefore
$Region2^{new}_{1} = \{0.3189,\ 3.8065,\ 3.0163\}$.

Similarly, $Region2_2$ and $Region2_3$ are updated as
$Region2^{new}_{2} = \{0.3189\ 2.2329\ 3.0163\}$
$Region2^{new}_{3} = \{0.3117\ 0.1184\ 3.0163\}$

In the eleventh step, we combine both subpopulation 1 and 2, and then store in x^{new} as illustrated below.
$x^{new}_1 = Region1^{new}_1 = \{3.5239\ 2.2329\ 3.0960\}$
$x^{new}_2 = Region1^{new}_2 = \{3.4696\ 3.3391\ 3.0960\}$
$x^{new}_3 = Region1^{new}_3 = \{0.1383\ 2.8725\ 3.0034\}$
$x^{new}_4 = Region2^{new}_1 = \{0.3189\ 3.8065\ 3.0163\}$
$x^{new}_5 = Region2^{new}_2 = \{0.3189\ 2.2329\ 3.0163\}$
$x^{new}_6 = Region2^{new}_3 = \{0.3117\ 0.1184\ 3.0163\}$

In step twelve, we determine the fitness values of all butterflies at updated positions $x^{new}_{i,j}\ \forall i, j$ by using $OF(,)$, as
$f_1 = OF(x^{new}_1) = 15.348$
$f_2 = OF(x^{new}_2) = 17.393$
$f_3 = OF(x^{new}_3) = 5.774$
$f_4 = OF(x^{new}_4) = 8.793$
$f_5 = OF(x^{new}_5) = 4.640$
$f_6 = OF(x^{new}_6) = 6.856$

In the thirteenth step, the best butterfly is identified from the updated population, based on best fitness value. It is observed that x^{new}_5 is the best butterfly. Therefore,
$f^{new}_{best} = 4.640$
$x^{new}_{best} = \{0.3189\ 2.2329\ 3.0163\}$

In the fourteenth step, the best fitness of the new population f_{best}^{new} is compared with f_{best}. If $f_{best}^{new} < f_{best}$ then we replace the x_{best} with x_{best}^{new} and f_{best} with f_{best}^{new}, otherwise proceed to step fifteen.
In this iteration $f_{best}^{new} = 4.640 > f_{best} = 4.639$; therefore, we move to step fifteen.

In the fifteenth step, we replace the $x_{i,j}$ with $x_{i,j}^{new} \ \ \forall\ i, j$.
$x_{i,j} = x_{i,j}^{new} \ \ \forall\ i, j$

In the sixteenth step, we return to step six.

In the seventeenth step, we print the optimal solution x_{best} with f_{best} and stop the algorithm.

19.7 Conclusion

A swarm based meta-heuristic optimization technique is presented in this chapter which is inspired by the migration behaviour of monarch butterflies and called MBO. The migration and butterfly adjusting operators are discussed and explained individually. To understand the application of this method, a simple step-by-step numerical example is demonstrated for one iteration of the algorithm. To help the new users, we have also presented the algorithm source-codes in Matlab and C++.

Acknowledgement

This work was supported by the Engineering and Physical Sciences Research Council (EPSRC) of United Kingdom (Reference Nos.: EP/R001456/1 and EP/S001778/1).

References

1. Cui Z, Gao X "Theory and applications of swarm intelligence" *Neural Comput Appl.*, 2012, pp. 205-206.
2. Fister Jr I, Yang XS, Fister I, Brest J, Fister D. "A brief review of nature-inspired algorithms for optimization" *arXiv preprint arXiv:1307.4186*, 2013.

3. Kennedy J, Eberhart R, "Particle swarm optimization" In paper presented at the Proceedings of *The IEEE International Conference on Neural Networks*, 2013.
4. Dorigo M, Maniezzo V, Colorni "A Ant system: optimization by a colony of cooperating agents" *IEEE Trans Syst Man Cybern B Cybern*, 1996, doi:10.1109/3477.484436
5. Back T, "*Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*" Oxford University Press, 1996.
6. Wang G-G, Gandomi AH, Alavi AH, Hao G-S, "Hybrid krill herd algorithm with differential evolution for global numerical optimization", *Neural Comput Appl*, 2014, pp. 297-308, doi:10.1007/s00521-013-1485-9.
7. Wang G-G, Deb S, Cui Z "Monarch butterfly optimization", *Neural Comput Appl.*, 2015. doi:10.1007/s00521-015-1923-y.
8. Garber SD, "*The Urban Naturalist*", Dover Publications, Mineola, 1998.
9. A. K. Davis and M. T. Holden. "Measuring intraspecific variation in flight-related morphology of monarch butterflies (Danaus plexippus): which sex has the best flying gear?" *Journal of Insects*, vol. 2015, 6 pages, 2015. https://doi.org/10.1155/2015/591705. A. K. Davis and M. T. Holden. "Measuring intraspecific variation in flight-related morphology of monarch butterflies (Danaus plexippus): which sex has the best flying gear?". *Journal of Insects*, vol. 2015, 6 pages, 2015. https://doi.org/10.1155/2015/591705.
10. Breed GA, Severns PM, Edwards AM, "Apparent powerlaw distributions in animal movements can arise from intraspecific interactions" *J R Soc Interface*, 2015, doi:10.1098/rsif.2014.0927.