5

Cat Swarm Optimization

Dorin Moldovan
Department of Computer Science
Technical University of Cluj-Napoca, Romania

Viorica Chifu
Department of Computer Science
Technical University of Cluj-Napoca, Romania

Ioan Salomie
Department of Computer Science
Technical University of Cluj-Napoca, Romania

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

CONTENTS

5.1 Introduction

CSO is introduced in [1] by Chu et al. where the first version of the CSO algorithm is proposed. The main inspiration of the CSO algorithm is represented by the behavior of the cats and the algorithm optimizes the searching of a solution in an M-dimensional space according to a fitness function. The solution sets are portrayed by cats and each cat is characterized by a number of dimensions, velocities for each dimension, a flag which describes whether the cat is in seeking mode or in tracing mode and a fitness value. The best solution is represented by the cat that has the best fitness value. The seeking mode models the situation in which the cat in a resting position looks around and seeks the next position to move to while the tracing mode models the situation in which the cat traces some targets. In literature there are many modifications of the CSO algorithm: CSO for clustering [2], parallel CSO [3], a parallel version of CSO that is based on the Taguchi Method [4], a modified version of the CSO algorithm in which the concept of craziness is introduced which is called Crazy-CSO [5], Multi-Objective Binary Cat Swarm Optimization (MOBCSO) [6], a gray image segmentation algorithm based on CSO [7], Harmonious Cat Swarm Optimization (HCSO) [8], Discrete Binary Cat Swarm Optimization (DBCSO) [9] and Quantum Cat Swarm Optimization Clustering (QCSOC) algorithm [10]. The CSO algorithm is generally applied for solving various engineering problems and we applied this algorithm successfully in [11] for generating diets for elders. The rest of the paper is organized as follows: Section 5.2 presents and discusses the global version and the local version of the CSO algorithm, Section 5.3 presents the source-code of the global version of the CSO algorithm in Matlab, Section 5.4 presents the source-code of the global version of the CSO algorithm in C++, in Section 5.5 is presented a detailed numerical example of the global version of the CSO algorithm and in Section 5.6 are presented the main conclusions.

5.2 Original CSO algorithm

5.2.1 Pseudo-code of global version of CSO algorithm

The pseudo-code for the global version of the CSO algorithm is presented in Algorithm 4 and is adapted after the one that is described in [12]. The algorithm takes as input the following parameters: M - the number of dimensions of the search space, $\left[P_{i,j}^{min}, P_{i,j}^{max}\right]$ - the range of variability for the positions of the cats, MR - the mixture ratio which is a percent that describes how many cats are in seeking mode and how many cats are in tracing mode, SMP - the seeking memory pool, SRD - the seeking range of the selected dimension, CDC - the count of dimension to change, SPC - the self-position consideration, N - the total number of cats, max - the maximum number of iterations and c_1 - a constant which is used for updating the velocity of the cats in tracing mode. The output is represented by the global best, which is the best position achieved by a cat. In this chapter we present a simplified version of the global version of the CSO algorithm in which we suppose that the flag SPC is equal to 0 and the point to move to in the seeking mode is chosen randomly from the SMP copies, each copy having the same probability of being chosen. In step 1 is created the initial population of N cats randomly and each cat is represented using an M-dimensional vector of values. Then the *Gbest*, the global best, is updated considering the best cat (step 3). While the stopping criterion is not satisfied or the number of iterations is less than the maximum number of iterations, a set of steps is repeated. According to MR some cats will be in tracing mode, while the rest will be in seeking mode (step 5). The global best cat is updated after the fitness values of all cats are computed (step 6) and then for each cat the values of the new positions are determined considering whether the cat is in seeking mode or in tracing mode. If the cat is in seeking mode then SMP copies are created, the positions of the copies are updated considering the value of SRD and the new position to move is selected randomly. In the tracing mode, the velocity is updated first (step 15) and the new position is determined using the updated velocity (step 17). Finally, in step 21 the global best *Gbest* is returned as the final result of the algorithm.

Algorithm 4 Pseudo-code of the global version of CSO.

```
Input $M$ - the number of dimensions
$[P_{i,j}^{min}, P_{i,j}^{max}]$ - range of variability for $i$-th cat and $j$-th dimension
$MR$ - mixture ratio
$SMP$ - seeking memory pool
$SRD$ - seeking range of the selected dimension
$CDC$ - count of dimensions to change
$SPC$ - self-position consideration
$N$ - the total number of cats
$max$ - maximum number of iterations
$c_1$ - a constant for updating the velocity of the cats in tracing mode
Output $Gbest$ - the best position achieved by a cat
 1: randomly create the population of cats $X_i (i = 1, 2, ..., N)$ (each cat is a
    $M$-dimensional vector)
 2: create the $M$-dimensional $Gbest$ vector
 3: assign the best cat $X_i$ to the $Gbest$
 4: while stopping criterion not satisfied or $I < I_{max}$ do
 5:     Set the cats in tracing mode or seeking mode according to $MR$
 6:     Calculate the fitness values of all cats and update $Gbest$
 7:     for $i = 1 : N$ do
 8:         if $X_i$ is in seeking mode then
 9:             create $SMP$ copies
10:             update the position of each copy using the formula
11:             $X_{cn} = X_c \times (1 \pm SRD \times R)$
12:             pick randomly a position to move to from the set of $SMP$ copies
13:         else
```

```
14:             update the velocity of the cat using the formula
15:             $v_{i,d} = v_{i,d} + R \times c_1 \times (X_{best,d} - X_{i,d})$
16:             update the position of the cat using the formula
17:             $X_{i,d,new} = X_{i,d,old} + v_{i,d}$
18:         end if
19:     end for
20: end while
21: return the $Gbest$ as a result
```

5.2.2 Description of global version of CSO algorithm

In this section is described the global version of the CSO algorithm. The algorithm starts by defining the algorithm's parameters and then the initial positions and velocities of the cats are generated randomly. The cats are distributed into tracing mode or seeking mode according to the value of a parameter that is called MR (Mutation Ratio). The fitness values of the cats are computed using an objective function and the best solution so far is the position of the best cat (X_{best}). Next are applied the seeking mode or the tracing mode process steps based on the values of the flags. If the termination criteria are met then the process is terminated otherwise the steps which follow after initialization are repeated. Some common termination criteria for the CSO are the number of the iterations, the running time and the amount of improvement. The seeking mode and the tracing mode are described in more detail next.

5.2.2.1 Seeking mode (resting)

The seeking mode describes the mode when the cat is resting. If the cat senses a danger then it moves cautiously and slowly. In the seeking mode the cat observes the M-dimensional space of solutions in order to decide what is the next move. The cat is aware of (a) its environment, (b) its own situation and (c) the choices it can make. In the CSO algorithm these facts are represented by the following parameters: (a) seeking memory pool (SMP), (b) seeking range of selected dimension (SRD), (c) count of dimensions to change (CDC) - the number of dimensions that will be mutated and (d) self-position consideration (SPC). The seeking mode process is described by the next steps: (1) for each cat X_i, SMP copies are created, and if the flag SPC is true then the cat's current position is considered as one of those SMP copies; (2) according to CDC the new position for each cat is computed using the following equation:

$$X_{cn} = X_c \times (1 \pm SRD \times R) \tag{5.1}$$

where X_{cn} is the new position, X_c is the current position and R is a random number from the interval $[0, 1]$; (3) for each position compute the fitness values of the new positions and if all the fitness values are equal then set the probability for all of the candidates' points to 1, otherwise use the equation described in the next step; (4) pick the point to move to randomly from the set of candidate points and replace the position of the cat X_i using the following equation:

$$P_i = \frac{FS_i - FS_b}{FS_{max} - FS_{min}} \tag{5.2}$$

where $0 < i < j$, P_i is the probability of the candidate cat X_i, FS_i is the fitness value of the cat X_i, FS_{max} is the maximum value of the fitness function, FS_{min} is the minimum value of the fitness function, $FS_{max} = FS_b$ for maximization problems and $FS_b = FS_{min}$ for minimization problems.

5.2.2.2 Tracing mode (movement)

The tracing mode simulates the chasing of a prey by a cat. The equations for updating the velocities and the positions of the cats are:

$$v_{k,d} = v_{k,d} + R \times c_1 \times (X_{best,d} - X_{k,d}) \tag{5.3}$$

and

$$X_{k,d,new} = X_{k,d,old} + v_{k,d} \tag{5.4}$$

where c_1 is a constant whose optimal value is determined considering the particularities of the optimization problem that will be solved and it does not have a specific range of variability, its value being determined in most of the cases using a trial and error procedure, R is a random value from the interval $[0, 1]$, $v_{k,d}$ is the velocity of the cat k for the d dimension, $X_{k,d}$ is the position of the cat k for the d dimension, $X_{best,d}$ is the position of the cat that has the best solution for the d dimension, $X_{k,d,new}$ is the new value of the position of cat k for the d dimension and $X_{k,d,old}$ is the current position of the cat k for the d dimension.

5.2.3 Description of local version of CSO algorithm

In the global version of the CSO algorithm each cat goes towards the best cat from the whole swarm. On the other hand in the local version of the CSO algorithm each cat goes towards the best cat from its neighborhood. Thus the formula:

$$v_{k,d} = v_{k,d} + R \times c_1 \times (X_{best,d} - X_{k,d}) \tag{5.5}$$

is replaced with the formula:

$$v_{k,d} = v_{k,d} + R \times c_1 \times (X_{Lbest,d} - X_{k,d}) \tag{5.6}$$

where X_{Lbest} describes the local best and X_{best} describes the global best. Another variant is the one that includes ω which is the inertia factor.

$$v_{k,d} = \omega \times v_{k,d} + R \times c_1 \times (X_{Lbest,d} - X_{k,d}) \tag{5.7}$$

The inertia factor ω is a numerical value which is used for controlling the rate at which the value of the velocity decreases. In literature its initial value is a constant from the interval $[0, 1]$, usually greater than 0.9, and its value decreases at each iteration of the algorithm with a constant numerical value such as 0.01, 0.001 or 0.0001. The neighborhood can be geometrical or social. Geometrical neighborhoods can be computed using several distances such as the Euclidean distance, the Manhattan distance or the Chebyshev distance. The social neighborhood depends on the index of the cat. Some common topologies are the ring topology in which the left neighbor and the right neighbor are considered or the star topology in which there is a central element connected to all other nodes. In the ring topology the neighbors of the i-th cat are determined using the following formulas:

$$Left_i = \begin{cases} i+1 & \text{if } i < N \\ 1 & \text{if } i = N \end{cases} \tag{5.8}$$

$$Right_i = \begin{cases} i-1 & \text{if } i > 1 \\ N & \text{if } i = 1 \end{cases} \tag{5.9}$$

The local best for cat i is updated considering the best value among its value, $Left_i$ and $Right_i$.

The star topology considers the relation between the cat mother and its kittens and the initial swarm of cats is divided into several smaller swarms that consist of one mother cat and its offspring. In this case the local best for each cat from the swarm is updated considering the best value obtained by a kitten or the mother cat. Figure 5.1 presents illustrative examples for these two topologies.

Some advantages are the fact that the algorithm might explore different areas and it can detect different local optima. However a disadvantage might be the fact that the local groups will have a smaller number of individuals and thus this modified version of the algorithm might not detect the global optimum.

5.3 Source-code of global version of CSO algorithm in Matlab

In Listing 5.1 is presented the source-code for the objective function which will be optimized by the CSO algorithm. In the function $OF(P)$, the input parameter is the position of a cat C. The objective function is given by formula 5.10, where $P_{i,j}$, $1 \leq j \leq M$, is the position of cat C_i for dimension j.

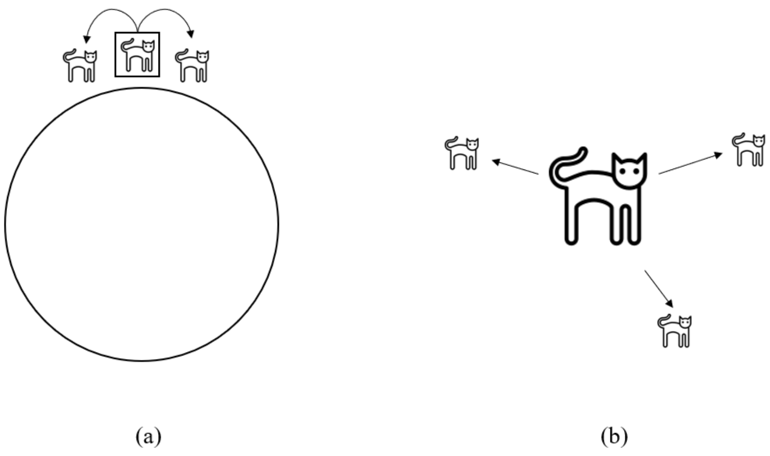

FIGURE 5.1
(a) Ring topology that considers the left and the right neighbor cats, (b) Star topology in which the cat mother is in the center.

$$OF(P) = \sum_{j=1}^{M} P_{i,j}^2 \qquad \text{where } -5.12 \leqslant P_{i,j} \leqslant 5.12 \tag{5.10}$$

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

Listing 5.1
Definition of objective function $OF(P)$ in Matlab.

In Listing 5.2 is presented the definition of the shuffle function in Matlab, a function that is used to permute randomly the initial values from the array given as input.

```
function [out]=shuffle(x)
out=x(randperm(length(x)));
end
```

Listing 5.2
Definition of shuffle function $shuffle(.)$ in Matlab.

```
%--- definition of the parameters of CSO algorithm
M=5; Pmin=zeros(1,M); Pmax=zeros(1,M); N=6; MAX=100; MR=0.2;
SMP=5; SRD=0.2; CDC=3; c1=0.5;
```

```
%--- definition of the constraints value
Pmin(1,:)=-5.12; Pmax(1,:)=5.12;
VMAX=10; VMIN=-10;
%--- definition of arrays required by CSO algorithm
copies=zeros(SMP,M); numberOfCopies=SMP;
P=zeros(N,M); V=zeros(N,M);
EvalPbest=zeros(N,1); Gbest=zeros(1,M);
flag=zeros(N,1); dimensionsToChange=zeros(1,M); Iter=0;
%--- population of cats is randomly created
P=P+(Pmax(1,:)-Pmin(1,:)).*rand(N,M)+Pmin(1,:);
Pbest=P;
%--- population of cats is evaluated
Eval=OF(P);
EvalPbest(:,1)=Eval(:,1);
%--- seeking the best cat
[Y,I]=min(Eval(:,1));
TheBest=I;
EvalGbest=Y;
%--- the main loop of the CSO algorithm
while(Iter<MAX)
  Iter=Iter+1;
  catsInSeekingMode=(1-MR)*N;
  for i=1:N
    if i<catsInSeekingMode
      flag(i,1)=0;
    else
      flag(i,1)=1;
    end
  end
  flag=shuffle(flag);
  for i=1:N
    if flag(i,1)==0
      dimensionsToChange(1,:)=0;
      dimensionsToChange(1,1:CDC)=1;
      for c=1:numberOfCopies
        copies(c,:)=P(i,:);
      end
      for c=1:numberOfCopies
        dimensionsToChange=shuffle(dimensionsToChange);
        for j=1:M
          if dimensionsToChange(1,j)==1
            if rand()>0.5
              copies(c,j)=copies(c,j)+SRD*copies(c,j)*rand();
            else
              copies(c,j)=copies(c,j)-SRD*copies(c,j)*rand();
            end
          end
        end
      end
      selectedCopy=randi(SMP);
      for j=1:M
        P(i,j)=copies(selectedCopy,j);
        if P(i,j)>Pmax(j) P(i,j)=Pmax(j); end
        if P(i,j)<Pmin(j) P(i,j)=Pmin(j); end
      end
    else
      for j=1:M
        V(i,j)=V(i,j)+rand()*c1*(P(TheBest,j)-P(i,j));
        if V(i,j)>VMAX V(i,j)=VMAX; end
        if V(i,j)<VMAX V(i,j)=VMIN; end
        P(i,j)=P(i,j)+V(i,j);
        if P(i,j)>Pmax(j) P(i,j)=Pmax(j); end
        if P(i,j)<Pmin(j) P(i,j)=Pmin(j); end
      end
    end
  end
  TheBest=1;
  for i=1:N
    Eval(i,1)=OF(P(i,:));
    if Eval(i,1)<EvalPbest(i,1)
      Pbest(i,:)=P(i,:);
      EvalPbest(i,1)=Eval(i,1);
    end
    if Eval(i,1)<Eval(TheBest,1)
      TheBest=i;
    end
  end
  if Eval(TheBest,1)<EvalGbest
    Gbest(1,:)=P(TheBest,:);
    EvalGbest=Eval(TheBest,1);
  end
end
disp(EvalGbest);
```

Listing 5.3
Source-code of the global version of the CSO in Matlab.

5.4 Source-code of global version of CSO algorithm in C++

```
#include <iostream>
#include <cstdlib>
#include <ctime>
#include <string>
using namespace std;
float OF(float x[], int size_array) {
  float t = 0;
  for(int i = 0; i < size_array; i++) {
    t = t + x[i] * x[i];
  }
  return t;
}
float r() {
  return (float)(rand()%1000)/1000;
}
void shuffle(int x[], int size_array) {
  int temporaryValue = 0; int randomIndex = 0;
  for(int i = 0; i < size_array; i++) {
    randomIndex = rand() % size_array; temporaryValue = x[i];
    x[i] = x[randomIndex]; x[randomIndex] = temporaryValue;
  }
}
int main() {
  srand(time(NULL));
  int M = 5; float Pmin[M]; float Pmax[M]; int N = 6;
  int TheBest = 0; int MAX = 100; int Iter = 0; float MR = 0.2;
  for(int i = 0; i < M; i++) {
    Pmin[i] = -5.12; Pmax[i] = 5.12;
  }
  int SMP = 5; float SRD = 0.2; int CDC = 3; float c1 = 0.5;
  float VMAX = 10.00; float VMIN = -10.00;
  float P[N][M]; float Pbest[N][M]; float V[N][M];
  float Eval[N]; float EvalPbest[N]; float Gbest[M]; float EvalGbest;
  int flag[N]; int dimensionsToChange[M];
  for(int i = 0; i < N; i++) {
    flag[i] = 0;
    for(int j = 0; j < M; j++) {
      P[i][j] = (Pmax[j] - Pmin[j]) * r() + Pmin[j];
      Pbest[i][j] = P[i][j]; V[i][j] = 0;
    }
    Eval[i] = OF(P[i], M); EvalPbest[i] = Eval[i];
    if(Eval[i] < Eval[TheBest]) TheBest = i;
  }
  EvalGbest = Eval[TheBest];
  while(Iter < MAX) {
    Iter++; int catsInSeekingMode = (1 - MR) * N;
    for(int i = 0; i < N; i++) {
      flag[i] = (i < catsInSeekingMode) ? 0 : 1;
    }
    shuffle(flag, N);
    for(int i = 0; i < N; i++) {
      if(flag[i] == 0) {
      int numberOfCopies = SMP; float copies[numberOfCopies][M];
      for(int j = 0; j < M; j++) dimensionsToChange[j] = 0;
      for(int j = 0; j < CDC; j++) {
        dimensionsToChange[j] = (j < CDC) ? 1 : 0;
      }
      for(int c = 0; c < numberOfCopies; c++) {
        for(int j = 0; j < M; j++) {
          copies[c][j] = P[i][j];
        }
      }
      for(int c = 0; c < numberOfCopies; c++) {
        shuffle(dimensionsToChange, M);
        for(int j = 0; j < M; j++) {
          if(dimensionsToChange[j] == 1) {
            copies[c][j] = (r() > 0.5)
              ? copies[c][j] + SRD * copies[c][j] * r()
              : copies[c][j] - SRD * copies[c][j] * r();
            }
          }
        }
        int selectedCopy = std::rand() % SMP;
        for(int j = 0; j < M; j++) {
          P[i][j] = copies[selectedCopy][j];
          P[i][j] = P[i][j] > Pmax[j] ? Pmax[j] :
            (P[i][j] < Pmin[j] ? Pmin[j] : P[i][j]);
        }
      } else {
        for(int j = 0; j < M; j++) {
          V[i][j] += r() * c1 * (P[TheBest][j] - P[i][j]);
          if(V[i][j] < VMIN || V[i][j] > VMAX) {
            V[i][j] = (V[i][j] < VMIN) ? VMIN : VMAX;
          }
          P[i][j] = P[i][j] + V[i][j];
          P[i][j] = P[i][j] > Pmax[j] ? Pmax[j] :
            (P[i][j] < Pmin[j] ? Pmin[j] : P[i][j]);
        }
      }
    }
    TheBest = 0;
    for(int i = 0; i < N; i++) {
      Eval[i] = OF(P[i], M);
      if(Eval[i] < EvalPbest[i]) {
        for(int j = 0; j < M; j++) {
          Pbest[i][j] = P[i][j];
        }
        EvalPbest[i] = Eval[i];
      }
      TheBest = (Eval[i] < Eval[TheBest]) ? i : TheBest;
    }
    if(Eval[TheBest] < EvalGbest) {
      for(int j = 0; j < M; j++) {
          Gbest[j] = P[TheBest][j];
        }
        EvalGbest = Eval[TheBest];
      }
  }
  cout << "EvalGBest = " << EvalGbest << endl;
  return 0;
}
```

Listing 5.4
Source-code of the global version of the CSO in C++.

5.5 Step-by-step numerical example of global version of CSO algorithm

In the first step let us assume that we want to minimize an objective function $OF(.)$ given by equation 5.10, where M, the number of dimensions, is equal to 5.

In the second step we determine the parameter values of the CSO algorithm such as $N = 6$, $MR = 0.2$, $SMP = 5$, $SRD = 0.2$, $CDC = 3$, $MAX = 100$ and $c_1 = 0.5$.

In the third step the swarm which consists of 6 cats is created. Each cat is a 5-dimensional vector.
$Cat_1 = \{0.849, -0.501, 4.761, 1.597, 3.983\}$
$Cat_2 = \{0.460, -3.553, 0.061, -3.338, -1.095\}$
$Cat_3 = \{-1.802, -2.856, -2.662, 3.061, -1.464\}$
$Cat_4 = \{2.385, 2.467, 0.215, -3.061, 0.849\}$
$Cat_5 = \{-1.720, 2.682, -2.314, 4.300, -1.249\}$
$Cat_6 = \{-0.399, 4.874, -2.457, 2.785, -0.296\}$

In the fourth step, the 5-dimensional *Gbest* vector is created.
$Gbest = \{0, 0, 0, 0, 0\}$

In the fifth step, the $Catbest_i$ cat is created for each i-th cat Cat_i. In the beginning the cats $Catbest_i$ have the same values as the cats Cat_i.
$Catbest_1 = \{0.849, -0.501, 4.761, 1.597, 3.983\}$
$Catbest_2 = \{0.460, -3.553, 0.061, -3.338, -1.095\}$
$Catbest_3 = \{-1.802, -2.856, -2.662, 3.061, -1.464\}$
$Catbest_4 = \{2.385, 2.467, 0.215, -3.061, 0.849\}$
$Catbest_5 = \{-1.720, 2.682, -2.314, 4.300, -1.249\}$
$Catbest_6 = \{-0.399, 4.874, -2.457, 2.785, -0.296\}$

In the sixth step, for each cat Cat_i the 5-dimensional velocity vector V_i is created. At the start each vector V_i consists of zeros.
$V_1 = \{0, 0, 0, 0, 0\}$, $V_2 = \{0, 0, 0, 0, 0\}$, $V_3 = \{0, 0, 0, 0, 0\}$
$V_4 = \{0, 0, 0, 0, 0\}$, $V_5 = \{0, 0, 0, 0, 0\}$, $V_6 = \{0, 0, 0, 0, 0\}$

In the seventh step we evaluate each cat Cat_i using the objective function $OF(.)$ (please see equation 5.10).
$Eval\_Catbest_1 = OF(Cat_1) = 42.065$, $Eval\_Catbest_2 = OF(Cat_2) = 25.186$
$Eval\_Catbest_3 = OF(Cat_3) = 30.017$, $Eval\_Catbest_4 = OF(Cat_4) = 21.925$
$Eval\_Catbest_5 = OF(Cat_5) = 35.570$, $Eval\_Catbest_6 = OF(Cat_6) = 37.803$

In the eighth step we take the best cat Cat_i that has the smallest value for the objective function as our goal is to minimize the value of $OF(.)$ and assign that value to $Gbest$. In our case the best cat is Cat_4 and therefore $Gbest = Cat_4 = \{2.385, 2.467, 0.215, -3.061, 0.849\}$ and $OF(Gbest) = 21.925$.

In the ninth step, the main loop of the algorithm starts. We check whether the algorithm termination condition is fulfilled and in the example presented here we check whether the value of the current iteration is less than $MAX = 100$. If the termination condition is met then we jump to the final step; otherwise we continue with the tenth step.

In the tenth step we determine the number of cats in seeking mode and the number of cats in tracing mode. The number of cats in seeking mode is given by the formula $(1 - MR) \times N = 0.8 \times 6 = 4.8$ in which we consider only the integer part; thus the actual number is 4. The number of cats in tracing mode is equal to $6 - 4 = 2$. The cats are selected randomly for the seeking mode and for the tracing mode and after this selection the cats in seeking mode are $\{cat_1, cat_2, cat_3, cat_6\}$ and the cats is tracing mode are $\{cat_4, cat_5\}$.

Seeking mode

In the first substep of seeking mode we start the creation of $SMP = 5$ copies for each cat.

In the second substep of seeking mode we consider $CDC = 3$ dimensions to change for each copy. In other words for each copy we change 3 out of the 5 dimensions that characterize the current position of the copy and the 3 dimensions to change are selected randomly from those 5 dimensions. The dimensions to change for each copy of each cat are the following:
$D_{cat_1} = \{(d_1, d_2, d_4), (d_1, d_3, d_4), (d_1, d_2, d_5), (d_1, d_2, d_3), (d_2, d_3, d_4)\}$
$D_{cat_2} = \{(d_1, d_3, d_4), (d_1, d_2, d_5), (d_1, d_3, d_4), (d_1, d_3, d_5), (d_3, d_4, d_5)\}$

$D_{cat_3} = \{(d_1, d_3, d_4), (d_1, d_4, d_5), (d_1, d_3, d_5), (d_2, d_4, d_5), (d_2, d_3, d_5)\}$
$D_{cat_6} = \{(d_2, d_4, d_5), (d_1, d_2, d_4), (d_1, d_2, d_4), (d_2, d_3, d_4), (d_1, d_2, d_3)\}$
The dimensions to change will take new values using the formula from the next substep while the other dimensions will have the same values as the original. For example, in the case of the first copy of cat_1, the dimensions d_1, d_2 and d_4 will be changed. That means that this copy will have the value $(x_1, x_2, x_3 = 4.761, x_4, x_5 = 3.983)$ where x_1, x_2 and x_4 are new values that will computed in the next substep.

In the third substep of seeking mode we create the 5 copies for each cat, $5 \times 4 = 20$ total copies, using the following formula:
$X_{cn} = X_c \times (1 \pm SRD \times R)$
where X_{cn} is the new position, X_c is the current position, R is a random number from $[0, 1]$ and SRD has the value 0.2 and describes the percentage of how much the current dimension is modified. We consider that in the formula the signs plus and minus are chosen randomly with a probability of 50%.

In the fourth substep of seeking mode for each cat we select randomly one copy from the 5 copies and we check whether the values are in the acceptable range of variability. In the original version of the algorithm for each copy the probability of being selected is proportional with the value returned by $OF(.)$. In the version presented in this article we consider that each cat has the same probability of being chosen and that is why we do not necessarily choose the cat with the lowest value for $OF(.)$. The positions of the cats after this step are:
$P_1 = Copy(Cat_1) = \{0.766, -0.461, 4.761, 1.569, 3.983\}$
$P_2 = Copy(Cat_2) = \{0.465, -3.553, 0.056, -3.338, -1.013\}$
$P_3 = Copy(Cat_3) = \{-1.802, -2.890, -2.190, 3.061, -1.390\}$
$P_6 = Copy(Cat_6) = \{-0.448, 5.12, -2.422, -2.785, -0.296\}$

Tracing mode

In the first substep of tracing mode we update the velocities of the cats using the formula:
$v_{k,d} = v_{k,d} + R \times c_1 \times (X_{best,d} - X_{k,d})$
where $v_{k,d}$ is the velocity value for cat k and dimension d, R is a random number from the interval $[0, 1]$, c_1 is a constant that has value 0.5, $X_{best,d}$ is the value of the position of the best cat for the dimension d and $X_{k,d}$ is the current position of the cat k.

In the second substep of tracing mode we check whether the values of velocity are in the accepted range of variability and if they are not we update those values that are outside the range of variability. After this step the velocities of the two cats in tracing mode are:
$V_4 = \{0, 0, 0, 0, 0\}$
$V_5 = \{1.554, -0.030, 0.838, -3.055, 0.044\}$

The velocity of cat_4 is zero because cat_4 was the best cat in the first step of the algorithm and the equation that is used for updating the velocities of the cats considers the position of the best cat.

In the third substep of tracing mode we update the current position of the cat using the formula:
$X_{k,d,new} = X_{k,d,old} + v_{k,d}$
where $X_{k,d,new}$ is the new value for the k-th cat for dimension d, $X_{k,d,old}$ is the old value for the k-th cat for dimension d and $v_{k,d}$ is the value of velocity of the k-th cat for dimension d.

In the fourth substep of tracing mode we update the values of the positions in order to be in the accepted range of variability. The values of the positions for the two cats in tracing mode are:
$P_4 = \{2.385, 2.467, 0.215, -3.061, 0.849\}$
$P_5 = \{-0.166, 2.652, -1.475, 1.245, -1.205\}$

In the eleventh step we compute the fitness values of the cats, we update the values of their local bests and we update the value of *Gbest* if a better value is found. Then we return to the tenth step.
$OF(Cat_1) = 41.803$, $OF(Cat_2) = 25.016$, $OF(Cat_3) = 27.705$
$OF(Cat_4) = 21.925$, $OF(Cat_5) = 12.243$, $OF(Cat_6) = 40.128$
$Eval\_Catbest_1 = 41.803$, $Eval\_Catbest_2 = 25.016$, $Eval\_Catbest_3 = 27.705$
$Eval\_Catbest_4 = 21.925$, $Eval\_Catbest_5 = 12.243$, $Eval\_Catbest_6 = 37.803$
$Gbest = \{-0.166, 2.652, -1.475, 1.245, -1.205\}$
$OF(Gbest) = 12.243$

In the twelfth step we return *Gbest* as the result of the algorithm operation and we stop the algorithm. After 100 iterations the value of *Gbest* is $\{0.371, -0.054, -1.085, -0.024, -0.136\}$ and $OF(Gbest)$ is equal with 1.337

5.6 Conclusions

In this chapter we showed the main principles of the CSO algorithm. We showed how the algorithm works in the global version and we provided the source codes both in Matlab and in C++. These source codes could help others for a better understanding of the CSO algorithm and we believe that this chapter will make the implementations of others of their own versions of the CSO algorithm in any programming language easier to achieve.

References

1. S.-C. Chu, P.-w. Tsai, J.-S. Pan. “Cat Swarm Optimization” in *Lecture Notes in Artificial Intelligence*, vol. 4099, 2006, pp. 854-858.
2. B. Santosa, M. K. Ningrum. “Cat Swarm Optimization for Clustering” in *Proc. of IEEE 2009 International Conference of Soft Computing and Pattern Recognition*, 2009, pp. 54-59.
3. P.-W. Tsai, J.-S. Pan, S.-M. Chen, B.-Y. Liao. “Parallel Cat Swarm Optimization” in *Proc. of the 7th International Conference on Machine Learning and Cybernetics*, 2008, pp. 3328-3333.
4. P.-W. Tsai, J.-S. Pan, S.-M. Chen, B.-Y. Liao. “Enhanced Parallel Cat Swarm Optimization Based on the Taguchi Method” in *Expert Systems with Applications*, vol. 39, 2012, pp. 6309-6319.
5. A. Sarangi, S. K. Sarangi, M. Mukherjee, S. P. Panigrahi. “System Identification by Crazy-cat Swarm Optimization” in *Proc. of the 2015 International Conference on Microwave, Optical and Communication Engineering (ICMOCE)*, 2015, pp. 439-442.
6. L. Pappula, D. Ghosh. “Planar Thinned Antenna Array Synthesis using Multi-objective Binary Cat Swarm Optimization” in *Proc. of the 2015 IEEE International Symposium on Antennas and Propagation & USNC/URSI National Radio Science Meeting*, 2015, pp. 2463-2464.
7. W. Ansar, T. Bhattacharya. “A New Gray Image Segmentation Algorithm Using Cat Swarm Optimization” in *Proc. of the 2016 International Conference on Communication and Signal Processing (ICCSP)*, 2016, pp. 1004-1008.
8. K. C. Lin, K. Y. Zhang, J. C. Hung. “Feature Selection of Support Vector Machine Based on Harmonious Cat Swarm Optimization” in *Proc. of the 2014 7th International Conference on Ubi-Media Computing and Workshops*, 2014, pp. 205-208.
9. Y. Sharafi, M. A. Khanesar, M. Teshnehlab. “Discrete Binary Cat Swarm Optimization Algorithm” in *Proc. of the 2013 3rd IEEE International Conference on Computer, Control and Communication (IC4)*, 2013, pp. 1-6.
10. D. Yan, Y. Wang, P. Zhang, H. Cao, X. Yu. “Working Conditions Classification of Ball Mill Pulverizing System Based on Quantum Cat Swarm Optimization Clustering Algorithm” in *Proc. of the 2016 31st Youth Academic Annual Conference on Chinese Association of Automation (YAC)*, 2016, pp. 348-352.
11. D. Moldovan, P. Stefan, C. Vuscan, V. R. Chifu, I. Anghel, T. Cioara, I. Salomie. “Diet Generator for Elders using Cat Swarm

Optimization and Wolf Search" in *Proc. of the International Conference on Advancements of Medicine and Health Care through Technology*, 2016, pp. 238-243.

12. M. Bahrami, O. Bozorg-Haddad, X. Chu. "Cat Swarm Optimization (CSO) Algorithm" in *O. Bozorg-Haddad (ed.), Advanced Optimization by Nature-Inspired Algorithms, Studies in Computational Intelligence*, vol. 720, 2018, pp. 9-18.