7

Cockroach Swarm Optimization

Joanna Kwiecien
AGH University of Science and Technology
Department of Automatics and Robotics, Krakow, Poland

CONTENTS

7.1 Introduction 85
7.2 Original cockroach swarm optimization algorithm 86
7.2.1 Pseudo-code of CSO algorithm 86
7.2.2 Description of the CSO algorithm 87
7.3 Source-code of CSO algorithm in Matlab 88
7.4 Source-code of CSO algorithm in C++ 90
7.5 Step-by-step numerical example of CSO algorithm 92
7.6 Conclusions 95
References 95

7.1 Introduction

The cockroach swarm optimization algorithm is one of the algorithms belonging to the group of swarm intelligence algorithms. It should be mentioned that swarm intelligence has focused on how social insects solve various problems by their mutual cooperation. CSO draws inspiration from the social behavior of cockroaches looking for food. The most common behaviors of cockroaches such as chasing, swarming, escaping from light and being ruthless are the basis of this approach. In 2010, Chen and Tang in [1] proposed the basic version of the CSO algorithm. Typically, each iteration of the CSO algorithm consists of three procedures, namely, chase-swarming, dispersion, and ruthless. In CSO, each individual represents a solution vector, and initialization is randomly over the entire search space. Changes to the position of cockroaches within the search space are based on using varying combinations of movements.

It should be mentioned that a number of basic modifications to CSO have been developed to improve the quality of solutions obtained with this algorithm. One of the first proposals was a modified CSO algorithm, where an inertia weight was introduced [2]. Furthermore, Ogagbuwa and Adewumi follow a somewhat different approach by introducing the hunger behavior after chase-swarming operation [3]. Some papers illustrated the ability of CSO to solve optimization problems. With some additional assumptions, the cockroach swarm optimization algorithm was used to solve discrete optimization problems, including route planning [4-6]. Moreover, cockroach-inspired algorithms for robot path planning were used in [7].

The rest of this paper is organized as follows: in Section 7.2 the basic principles of CSO, and its pseudo-code are presented and described in detail in order to allow for easy implementation by future users, and, consequently, practical applications of this algorithm. In Sections 7.3 and 7.4 the source-codes of the CSO algorithm are shown in Matlab and C++ programming language, respectively. In Section 7.5, we illustrate in detail the numerical example and simulation results of the CSO algorithm, and some conclusions are revealed in Section 7.6.

7.2 Original cockroach swarm optimization algorithm

In general, the CSO algorithm can search through possible movement choices to find a movement sequence that will guide the cockroach individuals from initial solutions to the global optimum. During each cycle, the CSO algorithm looks around possible solutions in order to find an even better one. The solutions are modified through some procedures, including chase-swarming, dispersion, and ruthlessness.

7.2.1 Pseudo-code of CSO algorithm

The pseudo-code for the basic version of the CSO algorithm is presented in Algorithm 6.

Algorithm 6 Pseudo-code of CSO.

determine the D-dimensional objective function $OF(.)$
initialize the CSO algorithm parameter values such as N – number of cockroaches in the swarm, $visual$ – the visibility parameter, $Stop$ – termination condition
randomly create swarm, the $i-th$ individual represents a vector $X_i = (x_{i1}, x_{i2}, ..., x_{iD})$
for each $i-th$ cockroach from swarm **do**
    evaluate quality of the cockroach X_i using $OF(.)$ function
end for
select the best cockroach P_g in initial swarm
while termination condition not met **do**

```
9:      for i = 1 to N do
10:         for j = 1 to N do
11:             if OF(X_i) is local optimum then
12:                 move cockroach i towards P_g using formula
13:                 X_i = X_i + step · rand · (P_g − X_i)
14:             else
15:                 move cockroach i towards P_i (within visual scope) using formula
16:                 X_i = X_i + step · rand · (P_i − X_i)
17:             end if
18:         end for j
19:     end for i
20:     if OF(X_i) is better than OF(P_g) then
21:         P_g = X_i
22:     end if
23:     for i = 1 to N do
24:         move cockroach randomly using formula
25:         X_i = X_i + rand(1, D)
26:         if OF(X_i) is better than OF(P_g) then
27:             P_g = X_i
28:         end if
29:     end for
30:     select cockroach h randomly
31:     X_h=P_g
32: end while
33: return the best one as a result
```

7.2.2 Description of the CSO algorithm

In Algorithm 6 the pseudo-code of the first version of the CSO algorithm was presented. In order to better understand CSO, it is necessary to conduct its detailed description. At the start, we should have defined an objective function (step 1) and the CSO algorithm parameters (step 2) such as number of cockroaches (N), visual range ($visual$), and the stopping criterion ($Stop$). The visibility parameter denotes the visual distance of cockroaches. In the third step, we have to initialize the swarm with random solutions. A D-dimensional vector $X_i = (x_{i1}, x_{i2}, ..., x_{iD})$ represents the ith cockroach, $i = 1, 2, ..., N$. The position of each individual is a potential solution, and the objective function value is calculated for all entities (step 5). Given such evaluation, the global best position is kept and denoted as global optimum P_g (step 7). At the core of CSO lies a loop starting from the $8th$ step, until a stopping condition is satisfied. The simplest criterion is a maximum number of iterations that the CSO algorithm executes, or a limited number of fitness function evaluations.

In the chase-swarming procedure (steps 9 to 19), in the new cycle, the strongest cockroaches carry the local best solutions P_i, form small swarms, and move forward to the global optimum P_g according to the formula (step 13): $X_i = X_i + step \cdot rand \cdot (P_g - X_i)$. Within this procedure, each individual X_i moves to its local optimum P_i in the range of its visibility (step 16) using the formula: $X_i = X_i + step \cdot rand \cdot (P_i - X_i)$.

There can occur a situation when a cockroach moving in a small group becomes the strongest by finding a better solution, because individuals follow in other ways than their local optimum. When the global best position of cockroaches found so far is improved (with respect to the objective function), this new cockroach position will become the new P_g (step 21).

In addition, a dispersion procedure is incorporated into the running process (steps 23 to 29). In order to improve the ability of local searching and to avoid getting stuck in local minima, each cockroach is randomly dispersed (step 25) using the formula: $X_i = X_i + rand\,(1, D)$, where $rand\,(1, D)$ is a D-dimensional random vector (D is the space dimension).

In step 27, the position of the best cockroach is updated. In step 31, the phenomenon of replacing a randomly chosen individual by the current best individual (step 31) is given. This is a behavior that corresponds the situation, when the stronger cockroach eats the weaker. All aforementioned procedures are applied repeatedly until the stopping criterion is satisfied. When the stopping criterion is met, the global best position of individuals (a local or global optimum) is returned.

7.3 Source-code of CSO algorithm in Matlab

In Listing 7.1 the source-code for the objective function which will be optimized by the CSO algorithm is shown. The result of $OF(.)$ function is an D-dimensional column vector with the objective function values for each cockroach from the swarm. We used the well-known Sum Squares function as the objective function. It is convex and unimodal, and has the global optimum (0) at x* = (0, 0, ..., 0). Therefore, we assume that the CSO algorithm minimizes the objective function given by formula 7.1 and evaluated on the hypercube $x_i \in [-5.12, 5.12]$. To simplify, the cockroach form is augmented to include the solution quality.

$$OF(X_i) = \sum_{j=1}^{D} j X_{i,j}^2 \qquad \text{where } -5.12 \leqslant X_{i,j} \leqslant 5.12 \tag{7.1}$$

```
function [X]=OF(X, i, D)
X(i,D+1)=0
for j=1:D
    X(i,D+1)=X(i,D+1)+j*X(i,j)^2;;
end
```

Listing 7.1
Definition of objective function $OF(.)$ in Matlab.

```
% declaration of the parameters of the CSO algorithm
N=5; visual=10; step=2; iter=50; D=3;
% constraints definition for all decision variables
Xmin=-5.12; Xmax=5.12;
% the initial swarm is created randomly
X=zeros(N,D+1);
for i=1:N;
 for j=1:(D);
    X(i,j)=Xmin+(Xmax-Xmin)*rand;
 end
 [X]=OF(X,i,D+1);
end
% find Pg
 [M Nr]=min(X(:,D+1));
Pg=X(Nr,:); % vector of Pg
vPg=M; % value of the objective function for Pg
%main program loop starts
for it=1:iter
% chase-swarming procedure
for i=1:N
  if X(i,D+1)~=vPg
  sl=X(i,:); % sl is the vector of the i-th cockroach
    vsl=X(i,D+1); % the position of the i-th cockroach
    Pi=sl; % assign Pi
    vPi=vsl; % assign value of the objective function for Pi
    flag=0; % if flag = 0 the i-th cockroach does not see better individual
    for l=1:N
      if (i ~= l)
      vl=X(l,D+1); % value of the objective function for the l-th cockroach
      dist=abs(vl-vsl); % distance between two cockroaches
       if (dist <= visual) & (vl < vPi)  % the l-th individual is better than Pi within visual scope
        Pi=X(l,:);
        vPi=vl;
        flag=1; % the i-th cockroach sees better individual
         end
      end
    end
  % if the i-th cockroach sees better individual, it moves toward local optimum Pi
   if flag==1
      for j=1:D
       X(i,j)=X(i,j)+step*rand*(Pi(1,j)-X(i,j));
       if X(i,j)<Xmin
        X(i,j)=Xmin;
       end
       if X(i,j)>Xmax
        X(i,j)=Xmax;
       end
    end
    [X]=OF(X,i,D);
  end
  if flag==0;
    for j=1:D
      X(i,j)=X(i,j)+step*rand*(Pg(1,j)-X(i,j));
```

```
        if X(i,j)<Xmin
          X(i,j)=Xmin;
        end
        if X(i,j)>Xmax
          X(i,j)=Xmax;
        end
      end
% evaluation of all solutions
      [X]=OF(X,i,D);
   end
   end
end
%update Pg
[M Nr]=min(X(:,D+1));
newPg=X(Nr,:);
newvPg=M;
if newvPg<vPg
   Pg=newPg;
   vPg=newvPg;
end
 % dispersion procedure
 for i=1:N
   for j=1:D
     X(i,j)=X(i,j)+rand;
     if X(i,j)<Xmin
         X(i,j)=Xmin;
     end
     if X(i,j)>Xmax
         X(i,j)=Xmax;
     end
   end
% evaluation of all solutions
   [X]=OF(X,i,D);
 end
 % update Pg
 [M Nr]=min(X(:,D+1));
 newPg=X(Nr,:);
 newvPg=M;
 if newvPg<vPg
   Pg=newPg;
   vPg=newvPg;
 end
 % ruthless behavior
 r=randi(N);
 X(r,:)=Pg;
 end
  % the result of the CSO algorithm is returned
 disp(Pg);

```

Listing 7.2
Source-code of CSO in Matlab.

7.4 Source-code of CSO algorithm in C++

```
#include <iostream>
#include <cmath>
using namespace std;
// definition of the objective function OF
double OF(double x[], int size_array)
{
    double f=0;
```

```
    for(int j=0; j<size_array; j++){
        f=f+j*x[j]*x[j];
    }
    return f;
}
// main program function
int main(){
// initialization of the parameters
    int N=5; int iter=50; int D=3;
    double visual = 10;
    double step = 2;
    int i,j,it;
    double r;
    bool flag;
    double X[N][D]; //cockroach position
    double Pg[D]; //the best solution found so far
    double Pi[N][D]; //the local optimum
    double v[N]; //the fitness value
    double vPg; //value of the objective function for Pg
    int Best=0;
    double Xmin[D]; double Xmax[D];
// initialization of the constraints
  for (int j=0; j<D; j++){
    Xmin[j]=-5.12; Xmax[j]=5.12;
  }
//initialize the cockroach swarm
    for(int i=0; i<N; i++) {
      for(int j=0; j<D; j++) {
       r=((double)rand()/((double)(RAND_MAX)+(double(1)));
       X[i][j]=(Xmax[j]-Xmin[j])*r()+Xmin[j];
       Pi[i][j]=X[i][j];
      }
// evaluate all the cockroaches in the swarm
       v[i]=OF(X[i],D);
       vPi[i]=v[i];
// find the best individual in the swarm
       if (v[i]<v[Best]) Best=i;
     }
// assign the best cockroach to Pg
   for(j=0; j<D; j++) {
    Pg[j]=X[Best][j];
    vPg=v[Best];
   }
//main program loop
 while(it<iter)
 {
  for (i=0; i<N, i++){
    for (k=0; k<N, k++){
      if(fabs(v[i]-v[k])<=visual && (v[k])<v[i]))
      {
      Pi[i][j]=X[k][j];
      flag=true;
      }
         if(flag==false){
         Pi[i][j]=X[i][j];
         }
         flag=false;
         if(Pi[i][j]==X[i][j]){
         r=((double)rand()/((double)(RAND_MAX)+(double(1)));
        X[i][j]=X[i][j]+((step*r)*(Pg[j]-X[i][j]));
      }
      else {
       r=((double)rand()/((double)(RAND_MAX)+(double(1)));
          X[i][j]=X[i][j]+(step*r*(Pi[i][j]-X[i][j]));
      }
    }
    v[i]=OF(X[i],D);
```

```
   }
//update Pg
   if (v[i]<vPg) {Best = i;
   for(int j=0; j<D; j++) Pg[j]=X[Best][j];
   vPg=v[Best];
   }
//dispersion
   for (i=0; i<N; i++){
      for (j=0; j<D;j++){
           X[i][j]=X[i][j]+rand();
            if (X[i][j]<Xmin[j]) X[i][j]=Xmin[j];
            if (X[i][j]>Xmax[j]) X[i][j]=Xmax[j];
      }
   v[i]=OF(X[i],D);
      if (v[i]<vPg) {Best = i;
         for(int j=0; j<D; j++) Pg[j]=X[Best][j];
         vPg=v[Best];
      }
   }
// ruthless behavior
   int RANDOM = rand() %N;
   for (j=0; j<D; j++){
   if(X[RANDOM][j]!=Pg[j]{
      X[RANDOM][j]=Pg[j];}
   }
  }
  cout<<"Value of the global best solution =" << vPg << endl;
}
```

Listing 7.3
Source-code of CSO in C++.

7.5 Step-by-step numerical example of CSO algorithm

In this section, we show step-by-step the CSO algorithm based on one mathematical function. First, we assume that the aim is to minimize the objective function given by equation 7.1, where D is equal to 3. In the second step, we initialize the following fixed parameters: the population size (N) equalling 5 individuals, $visual = 10$, $step = 2$. The stopping criterion is taken to be a predefined maximum number of iterations ($iter$) which equals 50. X_{min} and X_{max} are used to represent the minimum and maximum limits of variables. Next, CSO creates a swarm of 5 cockroaches randomly. To simplify numerical illustrations, consider the case that the cockroach i is expressed in the D-dimensional vector. Therefore, the swarm consists of 5 cockroaches:

X_1=$\{-0.0635, 0.9296, -4.6603\}$
X_2=$\{0.6601, 1.3675, -0.0195\}$
X_3=$\{-4.5095, 0.1553, -4.5753\}$
X_4=$\{3.5188, 0.9694, 1.9560\}$
X_5=$\{1.5076, 1.2640, 4.6475\}$.

As mentioned in the previous section, each cockroach has a fitness value computed by optimization function $OF(.)$ and represents a complete solution. Hence, for all individuals we have the following values:

$OF(X_1) = \sum_{j=1}^{D} jX_{1,j}^2$=66.888
$OF(X_2) = \sum_{j=1}^{D} jX_{2,j}^2$=4.1769
$OF(X_3) = \sum_{j=1}^{D} jX_{3,j}^2$=83.1831
$OF(X_4) = \sum_{j=1}^{D} jX_{4,j}^2$=25.7397
$OF(X_5) = \sum_{j=1}^{D} jX_{5,j}^2$=70.2664.

In the next step, the best cockroach from the swarm is selected. In our case, the best cockroach is X_2 with the value vP_g (see the $16-th$ line in Listing 7.2) equalling 4.1769. Then, the main loop of the CSO algorithm starts, until the stopping criterion has been satisfied, usually a maximum number of iterations.

For all cockroaches different from P_g we calculate distance ($dist$) between their values of objective function. Therefore, the chase-swarming procedure starts on the $1-st$ cockroach. The following distances are obtained:

$dist(OF(X_1), OF(X_2)) = 62.7111$
$dist(OF(X_1), OF(X_3)) = 16.2951$
$dist(OF(X_1), OF(X_4)) = 41.1483$
$dist(OF(X_1), OF(X_5)) = 3.3784$.

Next, during the run of CSO, the flag of the cockroaches is setting to make them move towards P_i or P_g according to the equations of the chase-swarming procedure. Once a cockroach goes into the chase-swarming mode, it moves according to the better one for every dimension.

Only $dist(OF(X_1), OF(X_5))$ is within the range of visibility ($visual$ = 10), but $OF(X_5)$ is worse. In that way, the first individual cannot see a better one ($flag = 0$), and it moves towards the swarm leader P_g. Here, we assume that after movement of the first cockroach, other individuals see only its new position and new value of $OF(X_1)$.

Taking account $rand = 0.2197$, the $1-st$ cockroach has the following form: X_1=$\{0.2544, 1.1220, -2.6213\}$ with its OF equalling 23.1962. The new solution should satisfy the constraint of the range, and in this case it is done. The same steps are done for all cockroaches (apart from P_g). In other words, the distance between individuals is calculated and checked, and after setting the flag their specific movement toward P_g or P_i is applied.

Therefore, for the $3-rd$ cockroach, we have the following distances:

$dist(OF(X_3), OF(X_1)) = 59.9869$
$dist(OF(X_3), OF(X_2)) = 79.0062$
$dist(OF(X_3), OF(X_4)) = 57.4434$
$dist(OF(X_3), OF(X_5)) = 12.9167$.

This individual cannot see a better one ($flag = 0$), hence it moves towards P_g. With $rand = 0.4596$, we obtain new solution X_3=$\{0.2428, 1.2697, -0.3872\}$ with its OF equalling 3.7329.

For the $4-th$ cockroach, we have the following distances:

$dist(OF(X_4), OF(X_1)) = 2.5435$
$dist(OF(X_4), OF(X_2)) = 21.5628$

$dist(OF(X_4), OF(X_3)) = 22.0068$
$dist(OF(X_4), OF(X_5)) = 44.5267.$

Note that X_1 is the locally best individual for X_4 (its $flag = 1$), so the $4-th$ cockroach can move toward X_1.Assume $rand = 0.9585$, the new cockroach X_4 has the following vector: $\{-2.7392, 1.2619, -5.1200, OF = 89.3316\}$.

For the last $(5 - th)$ cockroach, we have the following distances:

$dist(OF(X_5), OF(X_1)) = 47.0702$
$dist(OF(X_5), OF(X_2)) = 66.0895$
$dist(OF(X_5), OF(X_3)) = 66.5335$
$dist(OF(X_5), OF(X_4)) = 19.0652.$

If $rand$= 0.79, then the positions of all cockroaches are as follows:

X_1=$\{0.25441.1220 - 2.6213\}$
X_2=$\{0.6601, 1.3675, -0.0195\}$
X_3=$\{0.24281.2697 - 0.3872\}$
X_4=$\{-2.73921.2619 - 5.1200\}$
X_5=$\{0.16851.4275 - 2.7268\}$

Their objective functions have the following values:

$OF(X_1)$=23.1962, $OF(X_2)$=4.1769, $OF(X_3)$=3.7329, $OF(X_4)$=89.3316, $OF(X_5)$=26.4102.

After calculating the objective function one by one, the update of the position and the value of the cockroach are considered, to determine which owns the best fitness value we find so far. It should be noted, that the third cockroach becomes new P_g.

For calculations in the dispersion procedure, assume that:

$rand(1, D) = [0.4519, 0.3334, 0.0591]$ for the $1 - st$ cockroach,
$rand(1, D) = [0.7409, 0.5068, 0.1999]$ for the $2 - nd$ cockroach,
$rand(1, D) = [0.4272, 0.1687, 0.7517]$ for the $3 - th$ cockroach,
$rand(1, D) = [0.3684, 0.9418, 0.0172]$ for the $4 - th$ cockroach,
$rand(1, D) = [0.8291, 0.6266, 0.5387]$ for the $5 - th$ cockroach.

For such values, it is easy to determine values for the cockroaches:

X_1=$\{0.7063, 1.4554, -2.5622\}$
X_2=$\{1.4010, 1.8743, 0.1804\}$
X_3=$\{0.6700, 1.4383, 0.3645\}$
X_4=$\{-2.3709, 2.2038, -5.1028\}$
X_5=$\{0.9975, 2.0541, -2.1880\}$

The fitness of individuals has the following values:

$OF(X_1)$=24.4302, $OF(X_2)$=9.0864, $OF(X_3)$=4.9851, $OF(X_4)$=93.4507, $OF(X_5)$=23.7965.

It may, of course, happen that a new best position is found, but in this procedure P_g is not updated.

To explain the ruthless procedure, assume that the $4-th$ individual is replaced by updated P_g. Then:

X_1={0.7063, 1.4554, −2.5622}
X_2={1.4010, 1.8743, 0.1804}
X_3={0.6700, 1.4383, 0.3645}
X_4={0.2428, 1.2697, −0.3872}
X_5={0.9975, 2.0541, −2.1880},

and their objective functions have the following values: $OF(X_1)$=24.4302, $OF(X_2)$=9.0864, $OF(X_3)$=4.9851, $OF(X_4)$=3.7329, $OF(X_5)$=23.7965.

It should be noted at this point that after the first iteration the better global solution is obtained with $OF = 3.7329$ ($X_4 = [0.2428, 1.2697, -0.3872]$).

7.6 Conclusions

In this chapter, we have described the cockroach swarm optimization algorithm that has a flexible architecture and can be used for various optimization problems. We presented the basic principles to boost the interest of readers by providing a comprehensive tutorial to entry into this method. We show how CSO can be implemented using Matlab and C++ languages. One example of function optimization was given to illustrate the application of CSO. Although the procedures of the CSO algorithm were presented for a certain task, one can design CSO for other similar problems. To apply the CSO algorithm to solve the discrete problem, we therefore need defining movement in the search space and to employ it in the structure of the considered algorithm. We believe that there still is a great potential for solving problems in various domains.

References

1. Z.Chen, H. Tang. "Cockroach swarm optimization" in *Proc. of 2nd International Conference on Computer Engineering and Technology (ICCET)*, 2010, pp. 652-655.
2. Z. Chen. "A modified cockroach swarm optimization" in *Energy Procedia*, vol. 11, pp. 4-9, 2011.
3. I.C. Obagbuwa, A.O. Adewumi. "An Improved Cockroach Swarm Optimization" in *The Scientific World Journal*, Article ID 375358, 2014.

4. J. Kwiecien. "Use of different movement mechanisms in cockroach swarm optimization algorithm for traveling salesman problem" in *Artificial Intelligence and Soft Computing*, vol. 9693, pp. 484-493, 2016.

5. J. Kwiecien, M. Pasieka. "Cockroach swarm optimization algorithm for travel planning" in *Entropy*, 19, 213, 2017.

6. L. Cheng, Z. Wang, S. Yanhong, A. Guo. "Cockroach swarm optimization algorithm for TSP" in *Adv. Eng. Forum*, vol. 1, pp. 226-229, 2011.

7. C. Le, H. Lixin, Z. Xiaoqin, B. Yuetang, Y. Hong. "Adaptive cockroach colony optimization for rod-like robot navigation" in *Journal of Bionic Engineering*, vol. 12, pp. 324-337, 2015.