8

Crow Search Algorithm

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

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

CONTENTS

8.1 Introduction

Crow Search Algorithm CSA is a novel metaheuristic that was introduced in [1] and its main application is the solving of the constrained engineering optimization problems. Its main source of inspiration is the behavior of crows, birds which are considered to be among the most intelligent animals in the whole world, and the main principles of the algorithm are the organization of the crows in the form of flocks, the memorization of the hiding places that are used for storing excess food, the following of each other when they do a theft and the protection of their caches from being stolen. These principles lead to the development of a unique algorithm that is much different from other algorithms that have as main inspiration the behavior of the birds in nature such as: Chicken Swarm Optimization (CSO) [2], Cuckoo Search (CS) [3], Bird Swarm Algorithm (BSA) [4], Bird Mating Optimizer (BMO) [5] and Peacock Algorithm (PA) [6]. The algorithms that might be considered as the main source of inspiration for CSA are Particle Swarm Optimization (PSO) [7], Genetic Algorithms (GA) [8] and Harmony Search (HS) [9], but compared to those algorithms CSA has fewer configurable parameters and thus it reduces the effort of parameter setting which is time-consuming work. CSA can be applied for various engineering optimization problems and in the original article that introduces the algorithm [1], several applications are presented such as the three-bar truss, the welded beam and the gear train design problems. The solutions of CSA are represented by crows, and each crow has a position in the D-dimensional space, a memory and a fitness value. Even though CSA is a novel bio-inspired algorithm, in literature there are already a lot of variations of it such as: Multi-Objective Crow Search Algorithm (MOCSA) [10], Binary Crow Search Algorithm (BCSA) [11] and Chaotic Crow Search Algorithm (CCSA) [12]. Moreover CSA was also used in combination with other algorithms and several examples are Hybrid Cat Swarm Optimization - Crow Search Algorithm (HCSO-CSA) [13] and hybrid Grey Wolf Optimizer (GWO) with CSA (GWOCSA) [14]. CSA has already been used in literature for solving a diversity of optimization problems such as data clustering [15], electromagnetic optimization [16], parameter estimation of Software Reliability Growth Models (SRGMs) [17], photovoltaic model parameters identification [18], economic environmental dispatch [19], performance improvement for inverter-based distributed generation systems [20] and enhancement of the performance of medium-voltage distribution systems [21]. The chapter has the following structure: Section 8.2 presents the original version of CSA, Section 8.3 presents the source-code of CSA in Matlab, Section 8.4 presents the source-code of CSA in C++, Section 8.5 presents a numerical example of CSA and Section 8.6 presents conclusions.

8.2 Original CSA

The pseudo-code of CSA is presented in Algorithm 7 and it is adapted after the original version that is presented in [1].

Algorithm 7 Pseudo-code of CSA.

for $i = 1 : N$ **do**
    initialize crow C_i in the D-dimensional search space
    initialize memory M_i to C_i
    evaluate the fitness value of C_i
end for
$t = 0$
while $t < Iter_{max}$ **do**
    for $i = 1 : N$ **do**
        select a random value k from $\{1, ..., N\}$
        if $r \geq AP$ **then**
            for $j = 1 : D$ **do**

12: $C_{i,j} = C_{i,j} + r_i \times fl \times (M_{k,j} - C_{i,j})$
13: **end for**
14: **else**
15: **for** $j = 1 : D$ **do**
16: $C_{i,j} = r_j \times (C_{max} - C_{min}) + C_{min}$
17: **end for**
18: **end if**
19: **end for**
20: **for** $i = 1 : N$ **do**
21: **for** $j = 1 : D$ **do**
22: update the value of $C_{i,j}$ to be in the interval $[C_{min}, C_{max}]$
23: **end for**
24: **end for**
25: **for** $i = 1 : N$ **do**
26: evaluate the new position of C_i
27: evaluate the new value of memory M_i
28: **if** $OF(C_i) < OF(M_i)$ **then**
29: M_i is updated to C_i
30: **end if**
31: **end for**
32: $t = t + 1$
33: **end while**
34: return M_i from memory for which $OF(M_i)$ is minimal

The **inputs** of the algorithm are: N - the number of crows, D - the number of dimensions of the search space, $Iter_{max}$ - the maximum number of iterations, $[C_{min}, C_{max}]$ - the range of variability of the positions of the crows, AP - the awareness probability and fl - the flight length. The **output** of CSA is represented by that i-th value from memory M for which the value of $OF(M_i)$ is minimal in the minimization case or maximal in the maximization case.

In steps 1-5 of the algorithm the initial population of N crows is initialized as follows: for each crow the D-dimensional vector C_i that describes the position is initialized with random numbers from the interval $[C_{min}, C_{max}]$ and the initial value of the memory M_i is initialized with the value of C_i. Initially $Memory(M) = Flock(C)$.

$$Flock = \begin{bmatrix} C_{1,1} & ... & C_{1,D} \\ ... & ... & ... \\ C_{N,1} & ... & C_{N,D} \end{bmatrix} \tag{8.1}$$

$$Memory = \begin{bmatrix} M_{1,1} & ... & M_{1,D} \\ ... & ... & ... \\ M_{N,1} & ... & M_{N,D} \end{bmatrix} \tag{8.2}$$

The fitness of C_i is evaluated using the objective function that is described by formula 8.5. In *step 6* the value of the current iteration is initialized to 0.

The next steps, from *step 8* to *step 32*, are repeated for a number of iterations equal to $Iter_{max}$. In steps 8-19 the initial population of N crows is initialized using formula 8.3 if the value of a random variable r from $[0, 1]$ is greater than or equal to AP or using the formula 8.4 otherwise. The first case corresponds to the situation in which the crow C_i follows another crow C_j from the flock having as main objective the discovery of the memory M_j of that crow, and the second case corresponds to the situation in which the new position is initialized randomly in the D-dimensional search space.

$$C_{i,j} = C_{i,j} + r_i \times fl \times (M_{k,j} - C_{i,j}) \tag{8.3}$$

$$C_{i,j} = r_j \times (C_{max} - C_{min}) + C_{min} \tag{8.4}$$

In formula 8.3 the value of r_i is a random value from $[0, 1]$ and k is a number from $\{1, ..., N\}$ selected randomly prior to the updating of the crow position. In formula 8.4 the value of r_j is a random number from $[0, 1]$ for each dimension j such that $j \in \{1, ..., D\}$.

The feasibility of C_i is checked for each crow. In this chapter a position C_i is considered feasible if all values of the D-dimensional vector C_i are in the interval $[C_{min}, C_{max}]$. In steps 20-24 the positions of the crows are updated to take values from the interval $[C_{min}, C_{max}]$ as follows: if $C_{i,j} < C_{min}$ then $C_{i,j} = C_{min}$ and if $C_{i,j} > C_{max}$ then $C_{i,j} = C_{max}$.

In steps 25-31 the memory of each crow C_i is updated as follows: in *step 26* the value of the position C_i is evaluated using the formula 8.5, in *step 27* the value of the memory M_i is evaluated using the same formula 8.5 and if the value of $OF(C_i)$ is less than the value of $OF(M_i)$ (or greater than the value of $OF(M_i)$ for maximization problems) then M_i is updated to the value of C_i (*step 29*). In *step 32* the value of the current iteration t is increased with 1.

Finally, in *step 34* the algorithm returns the memory M_i from the entire set of memory values for which the value of $OF(M_i)$ is minimal in the minimization case or maximal in the maximization case.

8.3 Source-code of CSA in Matlab

The objective function that is optimized by CSA is given by formula 8.5. In the function $OF(x, D)$ the input parameter x is the vector of decision variables, and D represents the number of dimensions of the search space.

$$OF(x, D) = \sum_{i=1}^{D} x_i^2 \qquad \text{where } -5.12 \leqslant x_i \leqslant 5.12 \tag{8.5}$$

Listing 8.1 presents the source-code of the objective function.

```
function [y]=OF(x,D)
  y=0;
  for i=1:D
    y=y+x(1,i)*x(1,i);
  end
end
```

Listing 8.1
Definition of objective function $OF(x,D)$ in Matlab.

```
N=10; d=5; Pmax=5.12; Pmin=-5.12;
IterMax=30;
fl=0.9;
AP=0.5;
Crows=zeros(N,d);
Memory=zeros(N,d);
EvalCrows=zeros(N,1);
EvalMemory=zeros(N,1);
for i=1:N
  for j=1:d
    Crows(i,j)=rand()*(Pmax-Pmin)+Pmin;
  end
end
Memory=Crows;
EvalCrows=OF(Crows);
Iter=0;
while Iter<IterMax
  for i=1:N
    ri=rand();
    k=randi(N);
    if rand()>=AP
      for j=1:d
        Crows(i,j)=Crows(i,j)+ri*fl*(Memory(k,j)-Crows(i,j));
      end
    else
      for j=1:d
        Crows(i,j)=rand()*(Pmax-Pmin)+Pmin;
      end
    end
  end
  for i=1:N
    for j=1:d
      if Crows(i,j)<Pmin
        Crows(i,j)=Pmin;
      end
      if Crows(i,j)>Pmax
        Crows(i,j)=Pmax;
      end
    end
  end
  EvalCrows=OF(Crows);
  EvalMemory=OF(Memory);
  for i=1:N
    if EvalCrows(i,1)<EvalMemory(i,1)
      Memory(i,:)=Crows(i,:);
      EvalMemory(i,1)=EvalCrows(i,1);
    end
  end
  Iter=Iter+1;
end
[x,y]=min(EvalMemory);
disp('FINAL RESULT:');
disp(Memory(y,:));
disp(x);
```

Listing 8.2
Source-code of CSA in Matlab.

8.4 Source-code of CSA in C++

```
#include <iostream>
#include <cstdlib>
#include <ctime>

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;}

int main()
{
  srand(time(NULL));
  int N=10; int d=5; int IterMax=30; int Iter=0;
  float Crowmax=5.12; float Crowmin=-5.12;
  float fl=0.9; float AP=0.5;
  float Crows[N][d]; float Memory [N][d];
  float EvalCrows[N]; float EvalMemory[N];
  for(int i=0; i<N; i++) {
    for(int j=0; j<d; j++) {
      Crows[i][j]=r()*(Crowmax-Crowmin)+Crowmin;
      Memory[i][j]=Crows[i][j];
    }
    EvalCrows[i]=OF(Crows[i],d);
  }
  while(Iter<IterMax) {
    for(int i=0; i<N; i++) {
      float ri=r();
      int k=rand()%N;
      if (r()>=AP) {
        for(int j=0; j<d; j++) {
          Crows[i][j]+=ri*fl*(Memory[k][j]-Crows[i][j]);
        }
      } else {
        for(int j=0; j<d; j++) {
          Crows[i][j]=r()*(Crowmax-Crowmin)+Crowmin;
        }
      }
    }
    for(int i=0; i<N; i++) {
      for(int j=0; j<d; j++) {
        if (Crows[i][j]<Crowmin) {
          Crows[i][j]=Crowmin;
        }
        if (Crows[i][j]>Crowmax) {
          Crows[i][j]=Crowmax;
        }
      }
    }
    for(int i=0; i<N; i++) {
      EvalCrows[i]=OF(Crows[i], d);
      EvalMemory[i]=OF(Memory[i], d);
      if(EvalCrows[i]<EvalMemory[i]) {
        for(int j=0; j<d; j++) {
          Memory[i][j]=Crows[i][j];
        }
        EvalMemory[i]=EvalCrows[i];
      }
    }
    Iter=Iter+1;
```

```
  }
  float minimum=EvalMemory[0]; int minimumIndex=0;
  for(int i=1; i<N; i++) {
    if(EvalMemory[i]<minimum) {
      minimum=EvalMemory[i];
      minimumIndex=i;
    }
  }
  cout<<"FINAL RESULT: ["<<Memory[minimumIndex][0];
  for(int j=1; j<d; j++) {
    cout<<", "<<Memory[minimumIndex][j];
  }
  cout<<"]"<<endl;
  cout<<"OF="<<EvalMemory[minimumIndex]<<endl;
  getchar();
  return 0;
}
```

Listing 8.3
Source-code of CSA in C++.

8.5 Step-by-step numerical example of CSA

First step

The objective of the algorithm is to minimize the value of the function $OF(x, D)$ which is given by equation 8.5 where the number of dimensions D is equal to 5.

Second step

The values of the parameters of CSA are initialized as follows:
N - the number of crows is equal to 10
D - the number of dimensions is equal to 5
$Iter_{max}$ - the maximum number of iterations is equal to 30
$[C_{min}, C_{max}]$ - the positions of the crows are in the interval $[-5.12, 5.12]$
AP - the awareness probability is 0.5
fl - the flight length is equal to 0.9

Third step

The initial population of crows that consists of 10 crows is created randomly and each crow is represented by one 5-dimensional vector that describes the position and one 5-dimensional vector that describes the memory. Initially for each crow the value of the position is equal to the value of the memory.
$C_1 = M_1 = \{1.300, -5.027, 1.904, -0.696, -2.314\}$
$C_2 = M_2 = \{1.525, -5.017, -1.894, -1.290, 3.307\}$
$C_3 = M_3 = \{-2.969, -3.962, -1.914, -4.515, -0.993\}$
$C_4 = M_4 = \{0.389, 1.730, 1.464, 2.836, -0.778\}$
$C_5 = M_5 = \{-2.222, 0.460, 2.703, 1.884, 0.716\}$
$C_6 = M_6 = \{0.378, 0.491, -0.870, 0.153, 0.143\}$
$C_7 = M_7 = \{-1.484, -2.232, -0.993, 2.928, 1.689\}$
$C_8 = M_8 = \{2.519, 0.286, 3.307, -4.075, 0.757\}$

$C_9 = M_9 = \{-1.280, 4.751, 1.710, 3.440, 0.092\}$
$C_{10} = M_{10} = \{-4.546, -0.184, -0.983, -4.423, 2.058\}$

Fourth step
The position of each crow is evaluated using the objective function $OF(x, D)$.
$EvalCrow_1 = OF(C_1) = 36.438, EvalCrow_2 = OF(C_2) = 43.697,$
$EvalCrow_3 = OF(C_3) = 49.569, EvalCrow_4 = OF(C_4) = 13.941,$
$EvalCrow_5 = OF(C_5) = 16.522, EvalCrow_6 = OF(C_6) = 1.186,$
$EvalCrow_7 = OF(C_7) = 19.606, EvalCrow_8 = OF(C_8) = 34.551,$
$EvalCrow_9 = OF(C_9) = 38.984, EvalCrow_{10} = OF(C_{10}) = 45.476$

For a number of iterations equal to $Iter_{max} = 30$ repeat the **Fifth step**, the **Sixth step** and the **Seventh step**.

Fifth step
The position of each crow is updated considering the following two possible cases: (case 1) if the value of a random numerical value is greater than the value of $AP = 0.5$ then the new position considers both the current value of the position and the current value of the memory of the crow, (case 2) otherwise the new position is initialized randomly. The new positions of the crows are:
(case 2) $C_1 = \{-4.792, 3.143, -1.505, -4.997, -4.751\}$
(case 1) $C_2 = \{1.363, -4.780, -1.695, -1.153, 3.195\}$
(case 1) $C_3 = \{-1.679, -2.247, -1.512, -2.717, -0.555\}$
(case 1) $C_4 = \{0.389, 1.730, 1.464, 2.836, -0.778\}$
(case 1) $C_5 = \{-2.222, 0.460, 2.703, 1.884, 0.716\}$
(case 2) $C_6 = \{4.710, -4.741, 3.502, 1.392, 2.682\}$
(case 1) $C_7 = \{-1.400, 0.646, 0.121, 3.139, 1.031\}$
(case 1) $C_8 = \{2.519, 0.286, 3.307, -4.075, 0.757\}$
(case 1) $C_9 = \{-2.117, 3.485, 1.019, 1.423, 0.596\}$
(case 2) $C_{10} = \{-4.116, -2.283, -3.604, 5.027, -0.798\}$

Sixth step
The values of the positions of the crows are updated so that they are in the interval $[C_{min}, C_{max}] = [-5.12, 5.12]$. After this step the positions have the same values because all values are already in the interval $[-5.12, 5.12]$.

Seventh step
For each crow the fitness value of the position and the fitness value of the memory are calculated. In addition if the fitness value of the position of the crow is less than the fitness value of the memory of the crow then the value of the memory is initialized with the value of the crow position and the fitness value of the memory is also updated with the fitness value of the crow position in order to reflect the change.

$EvalCrow_1 = OF(C_1) = 82.661, EvalMemory_1 = OF(M_1) = 36.438$
$EvalCrow_2 = OF(C_2) = 39.134, EvalMemory_2 = OF(M_2) = 43.697$
$OF(C_2) < OF(M_2) \Rightarrow M_2 = C_2,\ EvalMemory_2 = OF(C_2) = 39.134$
$EvalCrow_3 = OF(C_3) = 17.850, EvalMemory_3 = OF(M_3) = 49.569$
$OF(C_3) < OF(M_3) \Rightarrow M_3 = C_3,\ EvalMemory_3 = OF(C_3) = 17.850$
$EvalCrow_4 = OF(C_4) = 13.941, EvalMemory_4 = OF(M_4) = 13.941$
$EvalCrow_5 = OF(C_5) = 16.522, EvalMemory_5 = OF(M_5) = 16.522$
$EvalCrow_6 = OF(C_6) = 66.067, EvalMemory_6 = OF(M_6) = 1.186$
$EvalCrow_7 = OF(C_7) = 13.314, EvalMemory_7 = OF(M_7) = 19.606$
$OF(C_7) < OF(M_7) \Rightarrow M_7 = C_7,\ EvalMemory_7 = OF(C_7) = 13.314$
$EvalCrow_8 = OF(C_8) = 34.551, EvalMemory_8 = OF(M_8) = 34.551$
$EvalCrow_9 = OF(C_9) = 20.054, EvalMemory_9 = OF(M_9) = 38.984$
$OF(C_9) < OF(M_9) \Rightarrow M_9 = C_9,\ EvalMemory_9 = OF(C_9) = 20.054$
$EvalCrow_{10} = OF(C_{10}) = 61.069, EvalMemory_{10} = OF(M_{10}) = 45.476$
$OF(C_{10}) < OF(M_{10}) \Rightarrow M_{10} = C_{10},\ EvalMemory_{10} = OF(C_{10}) = 45.476$

Finally after $Iter_{max} = 30$ iterations, the best solution from memory is returned. As the best solution we mean the solution for which the value of the objective function is the lowest (in the minimization case). The content of the memory after 30 iterations is:
$M_1 = \{-0.323, 0.516, -0.406, -0.576, -0.676\}$
$EvalMemory_1 = OF(M_1) = 1.327$
$M_2 = \{-0.060, -0.166, 0.284, -0.199, -0.361\}$
$EvalMemory_2 = OF(M_2) = 0.282$
$M_3 = \{1.150, 0.437, -0.162, 0.052, 0.188\}$
$EvalMemory_3 = OF(M_3) = 1.578$
$M_4 = \{-0.242, -0.003, -0.025, -0.203, 0.249\}$
$EvalMemory_4 = OF(M_4) = 0.162$
$M_5 = \{0.041, -0.071, -0.011, -0.231, -0.368\}$
$EvalMemory_5 = OF(M_5) = 0.196$
$M_6 = \{0.378, 0.491, -0.870, 0.153, 0.143\}$
$EvalMemory_6 = OF(M_6) = 1.186$
$M_7 = \{0.068, 0.247, -0.618, -0.464, -0.362\}$
$EvalMemory_7 = OF(M_7) = 0.795$
$M_8 = \{-0.050, -0.365, 0.067, -0.299, -0.071\}$
$EvalMemory_8 = OF(M_8) = 0.235$
$M_9 = \{0.131, 0.096, 0.144, -0.142, 0.184\}$
$EvalMemory_9 = OF(M_9) = 0.102$
$M_{10} = \{0.230, 0.022, -0.035, -0.166, -0.240\}$
$EvalMemory_{10} = OF(M_{10}) = 0.140$

Eighth step
The final result corresponds to the memory of the ninth crow.
$Result = M_9 = \{0.131, 0.096, 0.144, -0.142, 0.184\}$
$EvalResult = EvalMemory_9 = OF(M_9) = 0.102$

8.6 Conclusions

This chapter presented the main principles of CSA. It presented the pseudo-code of CSA and the corresponding source-code both in Matlab and in C++. In addition it showed how this algorithm works providing a step-by-step numerical example. This chapter will facilitate the development of other versions of the algorithm in other programming languages and it might be the source of inspiration for new modifications that can be applied in complex engineering optimization problem solving.

References

1. A. Askarzadeh. "A novel metaheuristic method for solving constrained engineering optimization problems: Crow search algorithm" in *Computers and Structures*, vol. 169, 2016, pp. 1-12.
2. X. Meng, Y. Liu, X. Gao, H. Zhang. "A New Bio-inspired Algorithm: Chicken Swarm Optimization" in *Lecture Notes in Computer Science*, vol. 8794, Springer, 2014, pp. 86-94.
3. C. Zefan, Y. Xiaodong. "Cuckoo search algorithm with deep search" in *Proc. of the 2017 3rd IEEE International Conference on Computer and Communications (ICCC)*, 2017, pp. 2241-2246.
4. X.-B. Meng, X. Z. Gao, L. Lu, Y. Liu, H. Zhang. "A new bio-inspired optimisation algorithm: Bird Swarm Algorithm" in *Journal of Experimental & Theoretical Artificial Intelligence*, vol. 28, no. 4, 2016, pp. 673-687.
5. A. Arram, M. Z. A. Nazri, M. Ayob, A. Abunadi. "Bird mating optimizer for discrete berth allocation problem" in *Proc. of the 2015 International Conference on Electrical Engineering and Informatics (ICEEI)*, 2015, pp. 450-455.
6. R. Chaudhary, H. Banati. "Peacock Algorithm" in *Proc. of the 2019 IEEE Congress on Evolutionary Computation (CEC)*, 2019, pp. 2331-2338.
7. R. Eberhart, J. Kennedy. "A new optimizer using particle swarm theory" in *MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, 1995, pp. 39-43.
8. J. Stender. "Introduction to genetic algorithms" in *IEE Colloquium on Applications of Genetic Algorithms*, 1994.

9. J. Zhang, P. Zhang. "A study on harmony search algorithm and applications" in *Proc. of 2018 Chinese Control And Decision Conference (CCDC)*, 2018, pp. 736-739.
10. H. Nobahari, A. Bighashdel. "MOCSA: A multi-objective crow search algorithm for multi-objective optimization" in *Proc. of the 2017 2nd Conference on Swarm Intelligence and Evolutionary Computation (CSIEC)*, 2017, pp. 60-65.
11. R. C. T. De Souza, L. d. S. Coelho, C. A. De Macedo, J. Pierezan. "A V-shaped binary crow search algorithm for feature selection" in *Proc. of the 2018 IEEE Congress on Evolutionary Computation (CEC)*, 2018, pp. 1-8.
12. G. I. Sayed, A. Darwish, A. E. Hassanien. "Chaotic crow search algorithm for engineering and constrained problems" in *Proc. of the 2017 12th International Conference on Computer Engineering and Systems (ICCES)*, 2017, pp. 676-681.
13. A. B. Pratiwi. "A hybrid cat swarm optimization - crow search algorithm for vehicle routing problem with time windows" in *Proc. of the 2017 2nd International conferences on Information Technology, Information Systems and Electrical Engineering (ICITISEE)*, 2017, pp. 364-368.
14. S. Arora, H. Singh, M. Sharma, S. Sharma, P. Anand. "A new hybrid algorithm based on grey wolf optimization and crow search algorithm for unconstrained function optimization and feature selection" in *IEEE Access*, vol. 7, 2019, pp. 26343-26361.
15. Z.-X. Wu, K.-W. Huang, A. S. Girsang. "A whole crow search algorithm for solving data clustering" in *Proc. of the 2018 Conference on Technologies and Applications of Artificial Intelligence (TAAI)*, 2018, pp. 152-155.
16. L. dos Santos Coelho, C. Richter, V. C. Mariani, A. Askarzadeh. "Modified crow search approach applied to electromagnetic optimization" in *Proc. of the 2016 IEEE Conference on Electromagnetic Field Computation (CEFC)*, 2016, pp. 1-1.
17. A. Chaudhary, A. P. Agarwal, A. Rana, V. Kumar. "Crow search optimization based approach for parameter estimation of SRGMs" in *Proc. of the 2019 Amity International Conference on Artificial Intelligence (AICAI)*, 2019, pp. 583-587.
18. A. Omar, H. M. Hasanien, M. A. Elgendy, M. A. L. Badr. "Identification of the photovoltaic model parameters using the crow search algorithm" in *The Journal of Engineering*, vol. 2017, 2017, pp. 1570-1575.
19. A. A. A. E. Ela, R. A. El-Sehiemy, A. M. Shaheen, A. S. Shalaby. "Application of the crow search algorithm for economic environmen-

tal dispatch" in *Proc. of the 2017 Nineteenth International Middle East Power Systems Conference (MEPCON)*, 2017, pp. 78-83.

20. D. A. Zaki, H. M. Hasanien, N. H. El-Amary and A. Y. Abdelaziz. "Crow search algorithm for improving the performance of an inverter-based distributed generation system" in *Proc. of the 2017 Nineteenth International Middle East Power Systems Conference (MEPCON)*, 2017, pp. 656-663.
21. A. M. Shaheen, R. A. El-Sehiemy. "Optimal allocation of capacitor devices on MV distribution networks using crow search algorithm" in *CIRED - Open Access Proceedings Journal*, vol. 2017, 2017, pp. 2453-2457.