8

Crow Search Algorithm - Modifications and Application

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) [2] is a novel bio-inspired algorithm introduced in 2016. The main source of inspiration of CSA is the behavior of intelligent crows in nature when they search for food. CSA presents similarities with the following three algorithms: Genetic Algorithms (GA) [3], Particle Swarm Optimization (PSO) [4] and Harmony Search (HS) [5]. Like in the case of GA and PSO, CSA is a non-greedy algorithm and like in the case of PSO and HS, CSA keeps the good solutions in memory.

Other representative algorithms from the family of the bio-inspired algorithms [6] are Ant Colony Optimization (ACO) [7], Elephant Search Algorithm (ESA) [8], Chicken Swarm Optimization (CSO) [9], Artificial Algae Algorithm (AAA) [10], Kangaroo Mob Optimization (KMO) [11] and Moth Flame Optimization (MFO) [12]. In the research literature CSA was applied for solving various types of optimization problems or optimization tasks such as the optimal power flow problem [13], electromagnetic optimization [14] and continuous optimization tasks [15].

Even though the algorithm is a relatively new bio-inspired algorithm, in literature there are already modifications of CSA which were applied in solving various types of optimization problems. A part of these modifications namely, Chaotic Crow Search Algorithm (CCSA) [16], Modified Crow Search Algorithm (MCSA) [17] and Binary Crow Search Algorithm (BCSA) [18] are presented in more detail in the section of the chapter which illustrates modifications of CSA.

The chapter has the following structure: Section 8.2 presents the original CSA in brief, Section 8.3 presents illustrative modifications of CSA, Section 8.4 presents the application of CSA for the tuning of the number of nodes of each layer of a deep neural network applied for jobs status prediction using as experimental support the CIEMAT Euler log and Section 8.5 presents the conclusions and future research directions.

8.2 Original CSA in brief

In this section is presented briefly the original version of CSA [2].

Algorithm 11 Pseudo-code of CSA.

```
1: initialize the population of crows $C_i$ $(i = 1, ..., N)$
2: for each crow $C_i$ calculate the fitness value using objective function $OF(.)$
3: initialize the memory of the crows $M_i$ $(i = 1, ..., N)$
4: define the awareness probability $AP$ and the flight length $fl$
5: $t = 0$
6: while $t < Iter_{max}$ do
7:     for $i = 1 : N$ do
8:         select a random value $k$ from $\{1, ..., N\}$
9:         generate a random number $r$ in $[0, 1]$
10:        if $r \geq AP$ then
11:            generate the new position of the crow using formula 8.1
12:        else
13:            generate the new position of the crow randomly
14:        end if
15:    end for
16:    for each crow $C_i$ check the feasibility of the solution represented by $C_i$
```

```
17:     compute the fitness value (OF) for each crow
18:     update the memory for each crow
19:     $t = t + 1$
20: end while
21: return $M_i$ from memory for which $OF(M_i)$ is minimal/maximal
```

In *step 1* the population of N crows C_i $(i = 1, ..., N)$ is initialized randomly in the D - dimensional space such that the values for each dimension are in the interval $[Min_j, Max_j]$ $(j \in [1, D])$. For each crow C_i the fitness value is computed using an objective function in *step 2* and in *step 3* the memory of the crows M_i $(i = 1, ..., N)$ is initialized. In this step $M_i = C_i$ for all $i \in [1, N]$. In *step 4* the awareness probability AP which is a number from $[0, 1]$ and the flight length fl are defined. These parameters are considered in the equations that are used for the generation of the new positions of the crows.

The current iteration t is initialized to 0 in *step 5* and then for a number of iterations equal to $Iter_{max}$ the steps 7-19 are executed. In steps 7-15 the positions of the crows are updated as follows: in *step 8* the value of k is selected randomly from $\{1, ..., N\}$ and in *step 9* a random number r is generated from $[0, 1]$. If the value of r is greater than or equal to AP (step 10) then the new position of C_i is generated using the formula:

$$C_{i,j} = C_{i,j} + r \times fl \times (M_{k,j} - C_{i,j}) \tag{8.1}$$

where $j \in [1, D]$. Otherwise the new position of C_i is generated randomly in the D - dimensional search space (step 13). The feasibility of each crow C_i is checked in *step 16*, and in this chapter a position is considered feasible if the values for each dimension are in the interval $[Min_j, Max_j]$ $(j \in [1, D])$. If this condition is not respected then the new values are updated accordingly to Min_j or Max_j $(j \in [1, D])$. The fitness value for each crow C_i is computed in *step 17* and in *step 18* the memory is updated to the value of the position if the fitness value of the new position is better. The current iteration t is incremented in *step 19*. Finally, in *step 21*, the algorithm returns that value M_i from memory for which the corresponding fitness value is minimal or maximal depending on the type of the optimization problem.

8.3 Modifications of CSA

This section presents a selection of representative modifications of CSA [2].

8.3.1 Chaotic Crow Search Algorithm (CCSA)

In the case of the CCSA modification of CSA [16], the searching mechanism of CSA introduces various chaotic maps [20] and the formulas for some of the most representative chaotic maps applied in that article are presented next. In all three cases the initial value of x_0 is a random number from $[0, 1]$.

(a) Chebyshev chaotic map

$$x_{t+1} = cos\left(t \times cos^{-1}(x_t)\right) \tag{8.2}$$

where $x_t \in [0, 1]$.

(b) Logistic chaotic map

$$x_{t+1} = c \times x_t \times (1 - x_t) \tag{8.3}$$

where c is a number from $[0, 4]$ and $x_t \in [0, 1]$.

(c) Piecewise chaotic map

$$x_{t+1} = \begin{cases} \frac{x_t}{p} & \text{if } 0 \leq x_t < p \\ \frac{x_t - p}{\frac{1}{2} - p} & \text{if } p \leq x_t < \frac{1}{2} \\ \frac{1 - p - x_t}{\frac{1}{2} - p} & \text{if } \frac{1}{2} \leq x_t < 1 - p \\ \frac{1 - x_t}{p} & \text{if } 1 - p \leq x_t < 1 \end{cases} \tag{8.4}$$

where $p = 0.2$ and $x_t \in [0, 1]$.

The chaotic maps are applied in the equation that is used for updating the position of the crow as follows:

$$C_{i,j} = \begin{cases} C_{i,j} + m_i \times fl \times (M_{k,j} - C_{i,j}) & \text{if } m_z \geq AP \\ r_j \times (Max_j - Min_j) + Min_j & \text{otherwise} \end{cases} \tag{8.5}$$

where r_i and r from the original equations of CSA are substituted with m_i and m_z, values that are obtained using the chaotic map.

8.3.2 Modified Crow Search Algorithm (MCSA)

MCSA [17] addresses the premature convergence and the stagnation of the classical CSA. The AP and fl parameters are tuned using an approach based on information about population diversity and Gaussian distribution as follows:

$$AP = 0.02 \times |R| \tag{8.6}$$

$$fl = 0.4 \times |R| \times (1 - div) \tag{8.7}$$

where R is generated by $\mathcal{N}(0, 1)$, the Gaussian distribution with mean 0 and standard deviation 1, and div represents the mean of the population diversity for the current iteration after it is truncated in the interval $[0, 1]$ according to the approach described in more detail in [21].

8.3.3 Binary Crow Search Algorithm (BCSA)

This version is applied in the case of optimization problems in which the search space is discrete and the solutions of BCSA [18] can be represented as arrays of zeros and ones. The formula applied in order to transform the positions of the crows in binary vectors is:

$$B_{i,j} = \begin{cases} 1 & \text{if } F(C_{i,j}) \geq 0.5 \\ 0 & \text{otherwise} \end{cases} \tag{8.8}$$

where F is defined by the formula:

$$F(C_{i,j}) = \left| \frac{C_{i,j}}{\sqrt{1 + C_{i,j}^2}} \right| \tag{8.9}$$

such that $C_{i,j}$ is the position of the crow C_i for the j-th dimension and $B_{i,j}$ is the binary variant.

A possible alternative of BCSA would be one in which another function is applied in order to convert the positions of the crows in arrays of zeros and ones like in the case of Binary Chicken Swarm Optimization (BCSO) [19] where a sigmoid function that has the formula presented below is applied:

$$F(C_{i,j}) = \frac{1}{1 + e^{-C_{i,j}}} \tag{8.10}$$

Formula 8.9 describes a V-shape transfer function while formula 8.10 describes an S-shape transfer function. According to the experiments performed in [22] on another bio-inspired algorithm called Ant Lion Optimization (ALO), the approach based on the V-shape transfer function improves significantly the performance of the optimization algorithm.

8.4 Application of CSA for jobs status prediction

8.4.1 Problem description

The problem that is approached in this chapter as an illustrative application of CSA for the solving of engineering problems in the tuning of the number of nodes of each layer of a deep neural network that is applied for the prediction of the jobs status. The problem is complex because the deep neural network has 10 hidden layers and each layer has a number of nodes from the set $\{10, ..., 100\}$. A brute force approach would consider $(100 - 10 + 1)^{10} = 91^{10}$ different combinations of nodes, where 91 is the size of the set $\{10, ..., 100\}$ and 10 is the number of hidden layers, and consequently a brute force approach is not feasible because the deep neural network would require 91^{10} training operations, one training for each possible combination of nodes.

The problem described in this chapter is a multiobjective optimization problem. The first objective is the minimization of the number of nodes of the deep neural network and the second objective is the maximization of the accuracy of the deep neural network. The multiobjective function used by CSA has the formula:

$$OF(C) = w_1 \times \frac{no_{nodes}(dnn(C)) - min_{nodes}}{max_{nodes} - min_{nodes}} + w_2 \times (1 - accuracy(dnn(C))) \tag{8.11}$$

The parameters used in formula 8.11 have the following meaning: min_{nodes} is the minimum possible number of nodes of the hidden layers of the deep neural network and has the value $min_{nodes} = 10 \times 10 = 100$, max_{nodes} is the maximum possible number of nodes of the hidden layers of the deep neural network and has the value $max_{nodes} = 10 \times 100 = 1000$, $dnn(C)$ is the deep neural network trained and tested using a configuration of nodes for the hidden layers indicated by the position of the crow C, $no_{nodes}(dnn(C))$ is the number of nodes of $dnn(C)$ and $accuracy(dnn(C))$ is the accuracy of $dnn(C)$. Moreover, the weights w_1 and w_2 respect the relation:

$$w_1 + w_2 = 1 \tag{8.12}$$

Due to the fact that in the second term of formula 8.11 the value of the accuracy is subtracted from 1, the optimization problem presented in this chapter minimizes both criteria and consequently the ideal solution has the fitness value equal to 0 and the worst solution has the fitness value equal to 1. The optimization problem uses as experimental support the CIEMAT Euler log [1]. The main characteristics of that dataset are summarized in Table 8.1.

TABLE 8.1
Main Characteristics of the CIEMAT Euler log [1].

Characteristic	**Value**
monitoring duration	from November 2008 to December 2017
number of jobs	9263012
number of features	18

The significance of each feature is presented briefly in Table 8.2 and more details about each feature can be found using the reference [1].

The meaning of the status of each job [1] is presented in Table 8.3.

In this chapter only the last 100000 records are used in experiments due to the time constraints associated with the training of the deep learning model. Those records are selected as the last records because they describe the newest monitored data and they are also consecutive records in order to respect the timeseries nature of the monitored data. The training data consists of 80000 records and the testing data consists of 20000 records which are selected randomly from the set of 100000 records. Moreover, from the initial set of 18

TABLE 8.2
CIEMAT Euler Log Features Description.

Feature No.	Description	Value
1	the number of the job	integer
2	the submit time	seconds
3	the wait time	seconds
4	the run time	seconds
5	the number of processors used by the job	integer
6	the average CPU time used	seconds
7	the used memory	kilobytes
8	the requested number of processors	integer
9	the requested time	seconds
10	the requested memory	kilobytes
11	the status of the job	integer
12	the ID of the user	positive integer
13	the ID of the group	positive integer
14	the application number	positive integer
15	the queue number	positive integer
16	the partition number	positive integer
17	the number of the preceding job	positive integer
18	the think time from the preceding job	seconds

TABLE 8.3
The Meaning of the Status of Each Job.

Status	Meaning
0	the job failed
1	the job was completed successfully
2	a partial execution that will be continued
3	the job was completed (the last partial execution)
4	the job failed (the last partial execution)
5	the job was canceled

features the features $\{10, 16, 17, 18\}$ are removed because they are constant for all records. The influence of the other features on the final classification results was validated using the Boruta [23] algorithm for features selection from R. In Figure 8.1 are presented the features selection results after the Boruta algorithm was applied for all 100000 records.

The Boruta algorithm compares iteratively the importances of the attributes with the importances of the shadow attributes (permuted copies). The attributes which are significantly better than the shadowMax are admitted as confirmed and the attributes left without any decision are considered tentative. The remaining attributes are rejected. As shown in Figure 8.1, from

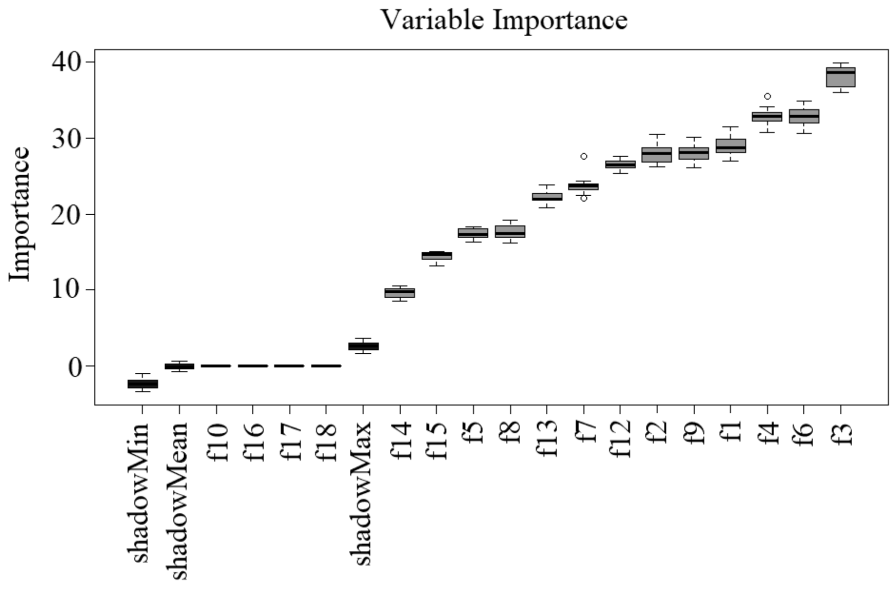

FIGURE 8.1
Features selection results after the application of the boruta algorithm.

the total number of 17 features, 13 features are confirmed and 4 features are rejected. The feature number 11 is the class label and consequently both for the training data and for the testing data the records have 13 features and 1 label. The labels of the records have only one status from the set $\{0, 1, 2, 3, 4, 5\}$ therefore the deep learning model has six possible outputs that correspond to each type of status.

8.4.2 How can CSA be used for this problem?

This subsection describes how CSA is used for the problem of tuning a deep neural network which is applied for job status prediction. The architecture of the neural network applied in this chapter is presented in Figure 8.2. The number of nodes of the first layer and the number of nodes of the last layer are static while the hidden layers have variable numbers of neurons given by the position of the crow associated with that deep neural network.

The deep learning model applied for the classification of the job status was developed in Python using Keras and TensorFlow and it is based on a sequential model with 1 epoch and a batch size equal to 64. An epoch describes the passing of the entire dataset forward and backward through the deep neural network only once and the batch size describes the total number of training samples that are present in a single batch or part of the training dataset.

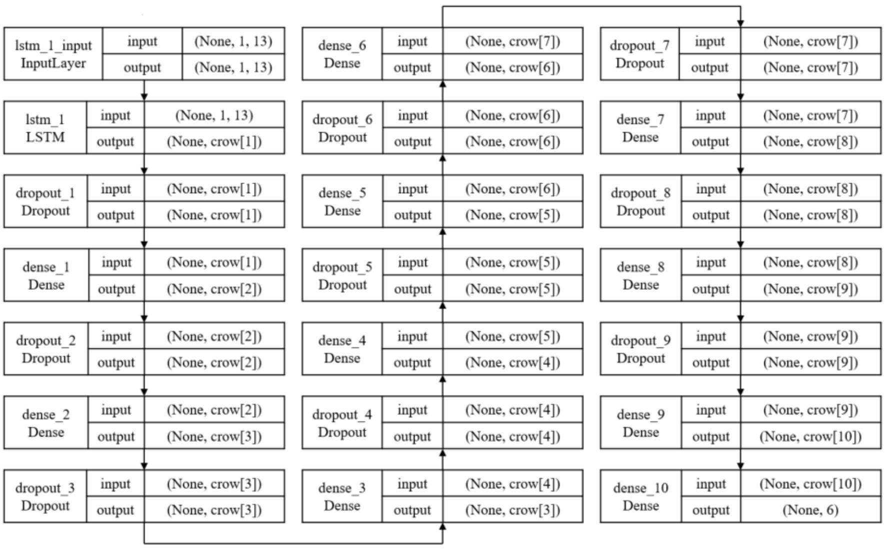

FIGURE 8.2
Architecture of the deep neural network tuned using CSA applied for the prediction of the jobs status.

The values of the features are normalized prior to the application of the deep neural network in order to obtain better accuracy results. The first hidden layer is a LSTM (Long-Short Term Memory) layer and the other 9 layers are dense layers with *relu* activation. Between each 2 consecutive hidden layers a dropout equal to 0.5 is considered. The last layer has *softmax* activation and 6 outputs. The model is compiled using the *categorical_ crossentropy* loss and the *adam* optimizer.

The application of CSA for this problem leads to an adaptation of the classical version of the algorithm. The position of each crow is a vector with 10 values such that each value describes the number of nodes of each hidden layer and is a value from $[10, 100]$. These values are chosen in order to create a complex search space for which a brute force approach is not feasible. Moreover, these values are also chosen in order to prevent overfitting. The positions of the crows are discretized converting the real values from the interval $[10, 100]$ to integer values immediately after (step 16) of the original Algorithm 11 where the feasibility of the solutions is checked. This transformation is mandatory because each layer of the deep neural network must be characterized by an integer number of nodes. Therefore, for a crow C_i from the search space the discretized position is given by the vector:

$$([C_{i,1}], ..., [C_{i,D}]) \tag{8.13}$$

where [] is the mathematical formula for the integer part and $(C_{i,1}, ..., C_{i,D})$ is the position of the crow C_i before the discretization phase.

8.4.3 Experiments description

Three types of experiments were considered depending on the criteria used in the seeking of the solutions in the search space. These criteria are summarized in Table 8.4.

TABLE 8.4
Description of the Criteria Applied in the CSA Experiments.

Criterion	**Description**
$(w_1, w_2) = (1.0, 0.0)$	the solutions are sought using only the first criterion
$(w_1, w_2) = (0.5, 0.5)$	the solutions are sought with regards to both criteria simultaneously
$(w_1, w_2) = (0.0, 1.0)$	the solutions are sought using only the second criterion

For each experiment 10-fold repetitions were performed and the values of the average (r_{avg}) and of the standard deviation (SD) were computed. The formulas of the two evaluation metrics are:

$$r_{avg} = \frac{\sum_{i=1}^{10} r_i}{10} \tag{8.14}$$

and

$$SD = \sqrt{\frac{\sum_{i=1}^{10} (r_i - r_{avg})^2}{9}} \tag{8.15}$$

where r_i with $i \in \{1, ..., 10\}$ is the result obtained in the i-th fold of the experiment.

For each criterion the search space is explored as follows:

- criterion 1 $(w_1 = 1.0, w_2 = 0.0)$: the number of nodes in each layer is optimized without considering the value of the accuracy;
- criterion 2 $(w_1 = 0.5, w_2 = 0.5)$: the number of nodes in each layer is optimized in order to obtain a higher accuracy;
- criterion 3 $(w_1 = 0.0, w_2 = 1.0)$: the accuracy is optimized without considering the number of nodes in each layer;

The performance of CSA is compared with the performance of PSO and of CSO and consequently both for PSO and for CSO three experiments which consist of 10-fold repetitions are performed. In Tables 8.5, 8.6 and 8.7 are presented the values of the configuration parameters used in experiments for CSA, PSO and CSO, respectively.

TABLE 8.5
Values of the CSA Configuration Parameters.

Configuration Parameter	**Value**
number of crows (N)	10
number of dimensions (D)	10
maximum number of iterations ($Iter_{max}$)	50
minimum possible value for crow positions (Min)	10
maximum possible value for crow positions (Max)	100
flight length (fl)	5
awareness probability (AP)	0.5

TABLE 8.6
Values of the PSO Configuration Parameters.

Configuration Parameter	**Value**
number of particles (N)	10
number of dimensions (D)	10
maximum number of iterations ($Iter_{max}$)	50
minimum possible value for particles positions (Min)	10
maximum possible value for particles positions (Max)	100
minimum possible value for particles velocity (V_{min})	-1
maximum possible value for particles velocity (V_{max})	1
cognitive coefficient value (c_1)	2
social coefficient value (c_2)	2

TABLE 8.7
Values of the CSO Configuration Parameters.

Configuration Parameter	**Value**
number of chickens (N)	10
number of dimensions (D)	10
maximum number of iterations ($Iter_{max}$)	50
hierarchical order updating period (G)	5
minimum value of the flight length (FL_{min})	0.5
maximum value of the flight length (FL_{max})	0.9
a number used for avoiding division by zero (e)	0.000000001
roosters percent (RP)	10%
hens percent (HP)	30%
minimum possible value for chickens positions (Min)	10
maximum possible value for chickens positions (Max)	100

8.4.4 Results

In Table 8.8 are presented the values of the average (r_{avg}), of the standard deviation (SD) and of the best result ($best$) returned after the execution of the 10-folds of the experiments in the case for each combination of optimization algorithm and criterion.

TABLE 8.8
Summary of the Experimental Results for Each Optimization Algorithm and for Each Criterion.

Metric	Algorithm	Criterion 1	Criterion 2	Criterion 3
r_{avg}	CSA	0.02977	0.07541	0.05319
	PSO	0.27910	0.19207	0.05311
	CSO	0.09799	0.11799	0.05456
SD	CSA	0.04794	0.03521	0.00016
	PSO	0.05807	0.03067	0.00007
	CSO	0.08110	0.04796	0.00235
$best$	CSA	0.00000	0.02910	0.05295
	PSO	0.19444	0.13007	0.05299
	CSO	0.02333	0.06533	0.05349

In Tables 8.9, 8.10 and 8.11 are presented the configurations of the deep neural network which correspond to the best results returned by CSA, PSO and CSO, respectively.

TABLE 8.9
Summary of the Best Results Returned by CSA.

Criterion	Result	Crow Position
$(w_1, w_2) = (1.0, 0.0)$	0.00000	$(10, 10, 10, 10, 10, 10, 10, 10, 10, 10)$
$(w_1, w_2) = (0.5, 0.5)$	0.02910	$(10, 10, 10, 10, 10, 10, 10, 10, 10, 10)$
$(w_1, w_2) = (0.0, 1.0)$	0.05295	$(48, 100, 10, 98, 83, 90, 100, 90, 45, 77)$

TABLE 8.10
Summary of the Best Results Returned by PSO.

Criterion	Result	Particle Position
$(w_1, w_2) = (1.0, 0.0)$	0.19444	$(25, 50, 10, 18, 10, 31, 66, 13, 13, 39)$
$(w_1, w_2) = (0.5, 0.5)$	0.13007	$(21, 21, 10, 10, 21, 63, 10, 63, 29, 10)$
$(w_1, w_2) = (0.0, 1.0)$	0.05299	$(65, 66, 55, 66, 56, 35, 69, 53, 43, 68)$

In Figure 8.3 is presented the evolution of the OF(.) value for each criterion for CSA, PSO and CSO for the folds of the experiments that led to the best results which are summarized in Tables 8.9, 8.10 and 8.11.

TABLE 8.11
Summary of the Best Results Returned by CSO.

Criterion	Result	Chicken Position
$(w_1, w_2) = (1.0, 0.0)$	0.02333	$(11, 10, 10, 12, 10, 10, 28, 10, 10, 10)$
$(w_1, w_2) = (0.5, 0.5)$	0.06533	$(18, 10, 16, 10, 10, 22, 10, 10, 26, 10)$
$(w_1, w_2) = (0.0, 1.0)$	0.05349	$(83, 98, 57, 39, 44, 92, 12, 96, 46, 63)$

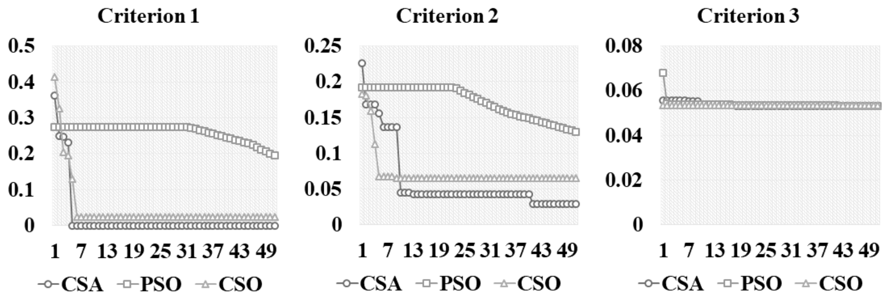

FIGURE 8.3
Evolution of the OF(.) value for CSA, PSO and CSO for each criterion for the folds of the experiments that returned the best results.

The best results were obtained by CSA for all three criteria and these results might be justified by the big value of the flight length (fl) which is equal to 5 and by the value of the awareness probability (AP) which is equal to 0.5. These values of the two configuration parameters lead to a premature convergence of the CSA algorithm. A similar behavior is presented by CSO which returns the second best results for the first criterion and for the second criterion. In the case of PSO the search space is exploited more due to the fact that the velocity of the particles is limited to values from the interval $[-1, 1]$, but that limitation also leads to a slow exploration rate of the particles. However, even with this limitation PSO returns a value almost identical to the value returned by CSA in the case of the third criterion.

8.5 Conclusions

This chapter described briefly the original version of CSA, presented modifications of CSA such as CCSA, MOCSA and BCSA, and finally showed how CSA can be applied for the tuning of the number of nodes of each layer of a deep neural network used for the prediction of jobs status. As future research work the following research directions are proposed: (1) the development of novel modifications of CSA having as a main source of inspiration the modifications of other classical bio-inspired algorithms, (2) the research of the hybridizations of CSA with other bio-inspired algorithms and (3) the application of CSA in the solving of other engineering problems.

References

1. The CIEMAT Euler log, https://www.cse.huji.ac.il/labs/parallel/workload/l_ciemat_euler/index.html
2. A. Askarzadeh. "A novel metaheuristic method for solving constrained engineering optimization problems: Crow search algorithm" in *Computers & Structures*, vol. 169, 2016, pp. 1-12.
3. J. Stender. "Introduction to genetic algorithms" in *IEE Colloquium on Applications of Genetic Algorithms*, 1994, pp. 1/1-1/4.
4. Eberhart, Y. Shi. "Particle swarm optimization: developments, applications and resources" in *Proc. of the 2001 Congress on Evolutionary Computation (IEEE Cat. No.01TH8546)*, 2001, pp. 81-86.
5. J. Zhang, P. Zhang. "A study on harmony search algorithm and applications" in *Proc. of the 2018 Chinese Control And Decision Conference (CCDC)*, 2018, pp. 736-739.
6. A. Darwish. "Bio-inspired computing: Algorithms review, deep analysis, and the scope of applications" in *Future Computing and Informatics Journal*, 2018, pp. 231-246.
7. Y. Pei, W. Wang, S. Zhang. "Basic Ant Colony Optimization" in *Proc. of the 2012 International Conference on Computer Science and Electronics Engineering*, 2012, pp. 665-667.
8. S. Deb, S. Fong, Z. Tian. "Elephant Search Algorithm for optimization problems" in *Proc. of the 2015 Tenth International Conference on Digital Information Management (ICDIM)*, 2015, pp. 249-255.
9. D. Moldovan, V. Chifu, C. Pop, T. Cioara, I. Anghel, I. Salomie. "Chicken Swarm Optimization and deep learning for manufacturing processes" in *Proc. of the 2018 17th RoEduNet Conference: Networking in Education and Research (RoEduNet)*, 2018, pp. 1-6.
10. M. Kumar, J. S. Dhillon. "Hybrid Artificial Algae Algorithm for global optimization" in *Proc. of the 2017 3rd International Conference on Advances in Computing,Communication & Automation (ICACCA) (Fall)*, 2017, pp. 1-6.
11. D. Moldovan, I. Anghel, T. Cioara, I. Salomie, V. Chifu, C. Pop. "Kangaroo Mob heuristic for optimizing features selection in learn-

ing the daily living activities of people with Alzheimer's" in *Proc. of the 2019 22nd International Conference on Control Systems and Computer Science (CSCS)*, 2019, pp. 236-243.

12. N. Jangir, M. H. Pandya, I. N. Trivedi, R. H. Bhesdadiya, P. Jangir, A. Kumar. "Moth-Flame optimization algorithm for solving real challenging constrained engineering optimization problems" in *Proc. of the 2016 IEEE Students' Conference on Electrical, Electronics and Computer Science (SCEECS)*, 2016, pp. 1-5.
13. A. Saha, A. Bhattacharya, P. Das, A. K. Chakraborty. "Crow search algorithm for solving optimal power flow problem" in *Proc. of the 2017 Second International Conference on Electrical, Computer and Communication Technologies (ICECCT)*, 2017, pp. 1-8.
14. 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.
15. P. A. Kowalski, K. Franus, S. Lukasik. "Crow Search Algorithm for continuous optimization tasks" in *Proc. of the 2019 6th International Conference on Control, Decision and Information Technologies (CoDIT)*, 2019, pp. 7-12.
16. 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.
17. L. dos Santos Coelho, C. E. Klein, V. C. Mariani, C. A. R. do Nascimento, A. Askarzadeh. "Electromagnetic optimization based on gaussian crow search approach" in *Proc. of the 2018 International Symposium on Power Electronics, Electrical Drives, Automation and Motion (SPEEDAM)*, 2018, pp. 1107-1112.
18. 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.
19. M. Han, S. Liu. "An improved Binary Chicken Swarm Optimization algorithm for solving 0-1 knapsack problem" in *Proc. of the 2017 13th International Conference on Computational Intelligence and Security*, 2017, pp. 207-210.
20. A. H. Abdullah, R. Enayatifar, M. Lee. "A hybrid genetic algorithm and chaotic function model for image encryption" in *AEU - International Journal of Electronics and Communications*, vol. 66, 2012, pp. 806-816.

21. L. S. Coelho, T. C. Bora, V. C. Mariani. "Differential evolution based on truncated Levy-type flights and population diversity measure to solve economic load dispatch problems" in *International Journal of Electrical Power & Energy Systems*, vol. 57, 2014, pp. 178-188.

22. M. Mafarja, D. Eleyan, S. Abdullah, S. Mirjalili. "S-shaped vs. V-shaped transfer functions for ant lion optimization algorithm in feature selection problem" in *CoRR*, vol. abs/1712.03223, 2017, pp. 1-7.

23. Package 'Boruta', https://cran.r-project.org/web/packages/Boruta/Boruta.pdf