6

Chicken Swarm Optimization - Modifications and Application

Dorin Moldovan
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

6.1 Introduction

Chicken Swarm Optimization (CSO) was introduced in 2014 in [1] and it is a part of the family of algorithms that are generically called nature inspired algorithms. Some illustrative algorithms from this family of algorithms are Particle Swarm Optimization (PSO) [2], Ant Colony Optimization (ACO) [3], Cuckoo Search (CS) [4], Lion Optimization Algorithm (LOA) [5], Kangaroo Mob Optimization (KMO) [6] and Crab Mating Optimization (CMO) [7]. Some of these algorithms are well known and they represent the source of inspiration for other bio-inspired algorithms, while other algorithms are relatively new and they are adaptations of the original PSO algorithm for different types of animal behaviors. The CSO algorithm is inspired by the behavior of the chickens when they search for food and it is a bio-inspired algorithm that can be applied for solving various types of engineering problems that are characterized by a search space with many dimensions. Each solution is represented by a chicken that has a position and there are three types of chickens namely, roosters, hens and chicks. The algorithm has as a main objective the identification of the best chicken according to an objective function which depends on the optimization problem that is solved. Some hybrid algorithms from literature that are based on the CSO algorithm are: Bat-Chicken Swarm Optimization (B-CSO) [8], Cuckoo Search-Chicken Swarm Optimization (CS-CSO) [9] and Chicken Swarm Optimization-Teaching Learning Based Optimization (CSO-TLBO) [10]. The chapter is organized as follows: Section 6.2 presents a short description of the global version of the CSO algorithm, Section 6.3 illustrates modifications of the CSO algorithm, Section 6.4 presents the application of the CSO algorithm for falls detection in daily living activities [11], [12] and Section 6.5 presents the main conclusions.

6.2 Original CSO algorithm in brief

The original CSO algorithm in global version can be represented using the pseudo-code from Algorithm 7.

Algorithm 7 Pseudo-code of the original CSO.

1: create the initial population of chickens C_i $(i = 1, 2, ..., N)$ randomly such that each chicken is represented by a D-dimensional vector
2: evaluate the fitness values of all N chickens
3: create the D-dimensional vector $Gbest$
4: assign the best chicken C_i to $Gbest$
5: $t = 0$
6: **while** $t < I_{max}$ **do**

if t modulo $G == 0$ **then**
    rank the fitness values of the chickens and establish the hierarchical order of the swarm
    divide the swarm of chickens into several groups and determine the relations between the chicks and the associated mother hens in each group
end if
for $i = 1 : N$ **do**
    if $C_i ==$ rooster **then**
        $C_{i,j}^{t+1} = C_{i,j}^{t} \times \left(1 + \mathcal{N}(0, \sigma^2)\right)$
    end if
    if $C_i ==$ hen **then**
        $C_{i,j}^{t+1} = C_{i,j}^{t} + S_1 \times R_1 \times \left(C_{r_1,j}^{t} - C_{i,j}^{t}\right) + S_2 \times R_2 \times \left(C_{r_2,j}^{t} - C_{i,j}^{t}\right)$
    end if
    if $C_i ==$ chick **then**
        $C_{i,j}^{t+1} = C_{i,j}^{t} + FL \times (C_{m,j}^{t} - C_{i,j}^{t})$
    end if
    update the value $C_{i,j}^{t+1}$ if it is not in the interval $[C_{min}, C_{max}]$
    evaluate the fitness value of the new solution
    if the new solution is better than *Gbest* then update *Gbest*
end for
$t = t + 1$
end while
return the *Gbest* as a result

6.2.1 Description of the original CSO algorithm

The pseudo code of the original CSO algorithm is presented in Algorithm 7 and in this subsection the algorithm is described in more detail. The inputs of the algorithm are represented by: D - the number of dimensions, N - the number of chickens, RN - the number of roosters, HN - the number of hens, CN - the number of chicks, MN - the number of mother hens, FL - a random number from the interval $[0.5, 0.9]$ which is used when the positions of the chicks are updated, G - a number that specifies how often the hierarchy of the chicken swarm is updated, I_{max} - the maximum number of iterations and $[C_{min}, C_{max}]$ - an interval that indicates the minimum and the maximum possible values of the positions of the chickens. The output of the algorithm is represented by the best position achieved by a chicken. In the global version of the algorithm this position is represented by the *Gbest* vector.

In *step 1* of the algorithm, the initial population of chickens C_i where $i = 1, 2, ..., N$ is created randomly such that each chicken is described by a D-dimensional vector. In *step 2* of the algorithm, each chicken is evaluated using an objective function $OF(.)$. In *step 3* the D-dimensional vector *Gbest* is created and the best chicken C_i is assigned to *Gbest* in the global version of the algorithm in *step 4*. In *step 5* of the algorithm, the current iteration t is initialized to 0 and the next steps are repeated for a number of iterations that is equal to I_{max}. If the current iteration is divisible without rest by G then

the chickens are ranked considering their fitness values, the novel hierarchical order of the swarm is established, the swarm of chickens is divided into several groups, each group having a dominant rooster, and the relations between the chicks and the corresponding mother hens are determined for each group of chickens. The position of each chicken is updated considering the type of the chicken. The roosters update their positions using the formula:

$$C_{i,j}^{t+1} = C_{i,j}^{t} \times \left(1 + \mathcal{N}(0, \sigma^2)\right) \tag{6.1}$$

where $t+1$ is the next iteration, t is the current iteration, i is the index of the chicken, j is the index of the dimension and has values from the set $\{1, ..., D\}$ and $\mathcal{N}(0, \sigma^2)$ is a Gaussian with mean 0 and standard deviation σ^2. The formula for σ^2 is:

$$\sigma^2 = \begin{cases} 1 & \text{if } OF(C_i) \leq OF(C_k) \\ e^{\frac{OF(C_k) - OF(C_i)}{|OF(C_i)| + \epsilon}} & \text{otherwise} \end{cases} \tag{6.2}$$

where ϵ is a small positive constant which is used in order to avoid division by 0 and k is the index of a rooster that is selected randomly from the group of roosters. In the global version of the algorithm the hens update their positions using the formula:

$$C_{i,j}^{t+1} = C_{i,j}^{t} + S_1 \times R_1 \times \left(C_{r_1,j}^{t} - C_{i,j}^{t}\right) + S_2 \times R_2 \times \left(C_{r_2,j}^{t} - C_{i,j}^{t}\right) \tag{6.3}$$

where R_1 and R_2 are random numbers from $[0, 1]$, r_1 is the index of the hen's rooster mate according to the hierarchical order that is reestablished every G iterations, r_2 is the index of another rooster or of a hen selected randomly from the swarm such that $r_1 \neq r_2$ and S_1 and S_2 have the formulas:

$$S_1 = e^{\frac{OF(C_i) - OF(C_{r_1})}{|OF(C_i)| + \epsilon}} \tag{6.4}$$

$$S_2 = e^{(OF(C_{r_2}) - OF(C_i))} \tag{6.5}$$

In the above formulas ϵ is a small positive constant that is used in order to avoid division by 0, like in the case of the formula that is used for updating the positions of the roosters. In the local version of the algorithm the formula that is used in order to update the positions of the hens is:

$$C_{i,j}^{t+1} = C_{i,j}^{t} + S_1 \times R_1 \times \left(C_{r,j}^{t} - C_{i,j}^{t}\right) + S_2 \times R_2 \times \left(C_{h,j}^{t} - C_{i,j}^{t}\right) \tag{6.6}$$

where $r = r_1$ and h is the index of a hen that is selected randomly from the group of hens in which C_i belongs. If there is no such hen then the formula becomes:

$$C_{i,j}^{t+1} = C_{i,j}^{t} + S_1 \times R_1 \times \left(C_{r,j}^{t} - C_{i,j}^{t}\right) \tag{6.7}$$

The chicks update their positions using the formula:

$$C_{i,j}^{t+1} = C_{i,j}^{t} + FL \times (C_{m,j}^{t} - C_{i,j}^{t}) \tag{6.8}$$

where FL is a random number from the interval $[0.5, 0.9]$ and m is the mother hen of the chicken with the index i. The rest of the parameters have the same signification as in the case of the formulas (6.1) and (6.3). If the value of $C_{i,j}^{t+1}$ is not in the interval $[C_{min}, C_{max}]$ (step 21) then that value is updated. In *step 22* the fitness value of the new solution is computed using the objective function $OF(.)$ and in *step 23* if the new solution is better than the previous solution then the value of *Gbest* is updated. Finally in the last step of the algorithm the best solution is returned. In the case of the global version of CSO algorithm the best solution is *Gbest*.

6.3 Modifications of the CSO algorithm

In this section are presented illustrative modifications of CSO [13].

6.3.1 Improved Chicken Swarm Optimization (ICSO)

In this version [14] the chicks learn both from their mother and from the rooster of the group. The situation in which the chicks get trapped in local optima is avoided and the new formula that is used in order to update the positions of the chicks is:

$$C_{i,j}^{t+1} = w \times C_{i,j}^{t} + FL \times (C_{m,j}^{t} - C_{i,j}^{t}) + c \times (C_{r,j}^{t} - C_{i,j}^{t}) \tag{6.9}$$

where w is the self-learning coefficient of the chick C_i, c is the learning factor and r is the index of the rooster of the group to which the chick C_i belongs. The value of w decreases from 0.9 to 0.4 in each iteration according to the formula [15]:

$$w = w_{min} \times \left(\frac{w_{max}}{w_{min}}\right)^{\frac{1}{1+10\times\frac{t}{I_{max}}}} \tag{6.10}$$

In the formula (6.10) w_{min} is the minimum value of w, w_{max} is the maximum value of w, t is the current iteration and I_{max} is the maximum number of iterations. FL is a random number from the interval $[0.5, 0.9]$ and the value of c is 0.4.

6.3.2 Mutation Chicken Swarm Optimization (MCSO)

The version presented in [16] introduces the following formula in order to update the positions of the chicks:

$$C_{i,j}^{t+1} = C_{i,j}^{t+1} \times \left(1 + \frac{1}{2} \times \eta\right) \tag{6.11}$$

where η is a random value from the Gaussian distribution $\mathcal{N}(0,1)$ that has the mean 0 and the standard deviation 1. The objective of the introduction of the mutation operator is to avoid the situations in which the chicks get trapped in local optima.

6.3.3 Quantum Chicken Swarm Optimization (QCSO)

In this version of the algorithm [17] the formula that is used for updating the positions of the chicks is:

$$C_{i,j}^{t+1} = C_{i,j}^{t} + FL \times (C_{m,j}^{t} - C_{i,j}^{t}) \times \log\left(\frac{1}{rand}\right) \tag{6.12}$$

such that:

$$FL = FL_{max} - (FL_{max} - FL_{min}) \times \frac{t}{I_{max}} \tag{6.13}$$

where FL_{max} is the maximum possible value of FL, FL_{min} is the minimum possible value of FL, t is the current iteration, I_{max} is the maximum number of iterations and $rand$ is a random number from $(0,1)$.

6.3.4 Binary Chicken Swarm Optimization (BCSO)

This version of the algorithm [18] is used for solving discrete optimization problems in which the search space is discrete and the solution can be represented using a binary vector. The formula that is used for transforming the positions of the chickens in arrays of zeros and ones is:

$$B_{i,j} = \begin{cases} 1 & \text{if } S(C_{i,j}) \geq 0.5 \\ 0 & \text{if } S(C_{i,j}) < 0.5 \end{cases} \tag{6.14}$$

where S is a sigmoid function that has the formula:

$$S(x) = \frac{1}{1 + e^{-x}} \tag{6.15}$$

and $C_{i,j}$ is the position of the chicken C_i for the j-th dimension while $B_{i,j}$ is the binary version of the position of the chicken C_i for the j-th dimension.

6.3.5 Chaotic Chicken Swarm Optimization (CCSO)

The CCSO algorithm is introduced in [19] where the authors apply knowledge from chaos theory [20]. The application of chaos theory has as the main objective the improvement of the performance of the CSO algorithm. The objective of the chaotic maps is to solve the problem of generation of values for the random variables which are used when the positions of the chickens are updated and two illustrative maps are the tent map and the logistic map.

The equation for the tent map is [21]:

$$x^{t+1} = \begin{cases} 2 \times \lambda \times x^t & \text{if } 0 \leq x^t \leq \frac{1}{2} \\ 2 \times \lambda \times (1 - x^t) & \text{if } \frac{1}{2} \leq x^t \leq 1 \end{cases} \tag{6.16}$$

where λ is a number from the interval $[0, 1]$, x^{t+1} is the new value and x^t is the current value. The equation for the logistic map is:

$$y^{t+1} = \mu \times y^t \times (1 - y^t) \tag{6.17}$$

where μ is a value from the interval $[0, 4]$. The values of x^t and y^t are in the interval $[0, 1]$. When $\lambda = 1$ and $\mu = 4$, the maps are in the chaos regions. Each map leads to a different version of the CCSO algorithm as follows: the tent maps are used in the CCSO Tent Map version and the logistic maps are used in the CCSO Logistic Map version. These maps help in a better exploration of the search space and they also solve the problem of getting trapped in local optima.

6.3.6 Improved Chicken Swarm Optimization - Rooster Hen Chick (ICSO-RHC)

The abbreviation ICSO-RHC [22] is given by the fact that the improved CSO algorithm includes position update modes for roosters, hens and chicks, and in addition it also includes a strategy for population update. In the case of the roosters the equations that are used for updating the positions consider information about the positions of the hens. The equations which are used for updating the positions of the hens are improved considering the guidance of the hens towards elite individuals from the chicken swarm. The chicks update their positions considering not only the position of the mother hen but also the position of the rooster which is the head of the swarm they belong to. Finally, in order to ensure the diversity of the chicken swarm, the following update strategy is applied:

$$C_i^{t+1} = \begin{cases} C_i^t & \text{if } R_i \geq P_e \\ C_{min} + rand \times (C_{max} - C_{min}) & \text{if } R_i < P_e \end{cases} \tag{6.18}$$

where $rand$ and R_i are random numbers from $[0, 1]$ selected uniformly and P_e is the elimination probability which is usually equal to 0.1.

6.4 Application of CSO for detection of falls in daily living activities

6.4.1 Problem description

In this subsection is described briefly the problem that is approached in this chapter using a modified version of the CSO algorithm. The problem is pre-

sented in more detail in [11] and in the approach presented in this article the problem is simplified. The input of the problem is represented by data generated from monitoring sensors placed on different parts of the body and the output can be of two types, 1 if the monitored subject performs a fall or 0 if the monitored subject performs a regular daily living activity. The problem is a classification problem and the classifications are performed using the classical Random Forest (RF) algorithm [23] from Apache Spark [24]. The dataset that is used as experimental support is taken from [11] and it contains information from different subjects that perform 16 types of daily living activities and 20 types of falls. Data is collected using 6 sensors and for each fall or activity 5 repetitions are performed.

For simplicity in this chapter in the experiments are considered only two subjects, a male subject and a female subject, and for each subject only one repetition out of five repetitions is considered for each fall or activity. This simplicity was introduced because the repetitions for each fall or activity are similar when they are performed by the same monitored subject and because the algorithm uses the same configuration of parameters for each subject. In a version of the algorithm that uses different parameter values for each subject it would be recommended to run experiments for various subjects, but in this chapter due to the fact that the algorithm uses the same configuration of parameters for both subjects, the application of data generated from the monitoring of two representative subjects is justified. The first subject is a male that weighs 81 kilograms, is 174 centimeters and is 21 years old. The second subject is a female that weighs 60 kilograms, is 165 centimeters in height and is 20 years old.

In Table 6.1 are presented the main characteristics of the data used in experiments after normalization for the male subject and for the female Psubject.

TABLE 6.1
Characteristics of the Data Used in Experiments.

Characteristic	Male	Female
number of features	126	126
number of labels	2	2
number of falls samples	5964	6739
number of daily living activities samples	7525	8249

6.4.2 How can the CSO algorithm be used for this problem?

The two major modifications that are performed in order to apply the CSO algorithm for falls detection using data generated from the monitoring of daily living activities using sensors placed on different parts of the body are:

- the modification of the global version of the CSO algorithm such that the positions of the chickens are transformed into arrays of zeros and ones;

- the evaluation of the fitness values of the chickens using a RF classifier in which the features of the data that is used in training are indicated by the positions of the chickens;

In this approach the fitness value of each chicken C_i is evaluated using the formula:

$$OF(C_i) = RF(train, test, B_i) \tag{6.19}$$

where C_i is the i-th chicken from the set of N chickens, RF is a Random Forest classifier, *train* is the training data, *test* is the testing data and B_i is the binary version of the position of C_i (see the formula 6.14). The validation data is data which is not used when the classification model is created and it is used for evaluating the performance of the classification model.

In the BCSO algorithm for classification of falls in daily living activities the line 22 from Algorithm 7 is replaced by the lines:

1: $B_{i,j}^{t+1} = 0$
2: **if** $S(C_{i,j}^{t+1}) \geq 0.5$ **then**
3: $\quad B_{i,j}^{t+1} = 1$
4: **end if**
5: $OF(C_i) = RF(train, test, B_i)$

The pseudo-code of the RF algorithm that is used for computing the fitness values of the chickens is presented in Algorithm 8. The input data is represented by *train* - the training data, *test* - the testing data, B - the binary version of the position of the chicken and T - the size of the forest. The output of the algorithm is represented by the accuracy of the classification model. Each step is described briefly in the pseudo-code of the algorithm. F represents the selected features according to B which is the binary position of the chicken and K is a number that should be much less than the number of features from the set F.

Algorithm 8 Pseudo-code of RF for Chicken Fitness Values Computation.

1: select randomly K features from the set of features F
2: compute the best node d using the best split which is calculated using the *train* data and the set of features K
3: compute the daughter nodes using the best split
4: repeat steps 1, 2, 3 until a certain number of nodes are computed
5: repeat steps 1, 2, 3, 4 until the size of the forest is T
6: $accuracy = RFClassificationModel(test)$
7: return *accuracy* as a result

6.4.3 Description of experiments

In the experiments are considered the following values for the configurable parameters: the population of chickens is equal to 10, the number of iterations is 30, the hierarchy of the swarm is updated every 5 iterations, FL_{min} is

equal to 0.5, FL_{max} is equal to 0.9, the value of ϵ is 10^{-9}, the percentage of roosters is 20%, the percentage of hens is 60%, the percentage of chicks is 20% and $[C_{min}, C_{max}] = [-5.12, 5.12]$. The values -5.12 and 5.12 are the values which are used in the global version of the algorithm and using exactly these numerical values is not very important. Any interval $[-L, L]$ where L is a number greater than 0 can be considered as an alternative. In each iteration of the algorithm the fitness values of the chickens are computed considering the training data and the testing data and finally the chicken with the best fitness value is returned as the final result of the optimization problem. The objective of the optimization problem is to determine a subset of features from the entire set of features such that the performance of the classification algorithm is high. The original datasets have 126 features and that means that there are 2^{126} possible combinations of features. The checking of all those combinations is too expensive in terms of computational resources and thus the application of an algorithm that determines a near-optimal solution fast is justified. The fitness value of each chicken is described by the accuracy of a RF classification model that is trained using the training data and tested using the testing data. The value of the accuracy in terms of TP (True Positives), TN (True Negatives), FP (False Positives) and FN (False Negatives) is:

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{6.20}$$

The global best chicken is used further in order to train a RF classification model that is applied on the validation data. In addition to accuracy, the metrics that are used for the evaluation of the performance of the classification model are:

$$precision = \frac{TP}{TP + FP} \tag{6.21}$$

$$F1Score = \frac{2 \times TP}{2 \times TP + FP + FN} \tag{6.22}$$

$$recall = \frac{TP}{TP + FN} \tag{6.23}$$

The accuracy is a metric that describes the closeness of a value that is measured to a known value while the precision describes the closeness of two measurements to each other. The recall or the sensitivity is equal to the ratio of positive observations that are correctly predicted to all observations and the F1Score is the weighted average between the values of the precision and of the recall.

6.4.4 Results obtained

In Tables 6.2 and 6.3 are presented the values of the accuracy and of the number of selected features in the initial iteration of the algorithm and in the final iteration of the algorithm. In the case of the male subject the value of the

accuracy increases with 0.004 after 30 iterations and in the case of the female subject the value of the accuracy increases with 0.008 after 30 iterations. In contrast the number of selected features increases by 3 in the case of the male subject while the number of selected features decreases by 3 in the case of the female subject. Even though in the case of the male subject the number of features increases, the number of features by which that number increases is a small number that represents approximately 2% of the total number of features.

TABLE 6.2
Accuracy and Number of Features in the Initial Iteration and in the Final Iteration of the Optimization Algorithm for the Male Subject.

Criteria	First Iteration	Last Iteration
accuracy	0.974	0.978
number of features	66	69

TABLE 6.3
Accuracy and Number of Features in the Initial Iteration and in the Final Iteration of the Optimization Algorithm for the Female Subject.

Criteria	First Iteration	Last Iteration
accuracy	0.975	0.983
number of features	59	56

Table 6.4 presents the values for accuracy, precision, F1Score and recall obtained in the validation data after the RF classification model is trained using the configurations indicated by the chickens that represent the values returned by the optimization algorithm for the male and for the female subjects. In the case of the male subject the value of the accuracy 0.980 is better than the values returned by the BCSO algorithm in each of the 30 iterations and which are in the interval $[0.974, 0.978]$. On the other hand in the case of the female subject the value of the accuracy 0.976 is in the interval $[0.975, 0.983]$. In both cases the value of the accuracy is better than the value of the accuracy from the first iteration of the BCSO algorithm.

TABLE 6.4
Evaluation Metrics Results That Correspond to the Best Chicken Returned by the Optimization Algorithm.

Metric	Male	Female
accuracy	0.980	0.976
precision	0.971	0.959
F1Score	0.985	0.973
recall	1.000	0.987

6.4.5 Comparison with other classification approaches

In Table 6.5 are presented the results obtained using other approaches in the case of the male subject and in Table 6.6 are presented the results obtained using other approaches in the case of the female subject. The tables also contain the values obtained after the application of the BCSO approach. The experiments were performed in Konstanz Information Miner (KNIME) [25] and like in the case of the BCSO approach, the training data is 60% and the testing data is 20%. The accuracy column contains the accuracy results of the classification models using the standard configurations from KNIME in the case of the other approaches and in the case of the BCSO approach the accuracy column contains the value of the accuracy from the last iteration of the BCSO algorithm.

The two features selection approaches that are compared are an approach in which the features are selected using Principal Component Analysis (PCA) with a value for the minimum information to preserve equal to 95% and an approach in which the features are selected using Correlation Filter (CF) with a threshold equal to 55%. The values for the configuration parameters of PCA and CF were selected after a series of experiments such that the number of selected features would be approximately equal to the number of features returned by the BCSO algorithm.

As can be seen in Tables 6.5 and 6.6 the approaches in which the applied classification algorithm is RF and the features are selected using PCA and CF are better than the approach in which the classification algorithm is RF and the features are selected using BCSO. Even though the methods that are based on Gradient Boosted Trees (GBT) are better than the approach described in this chapter, when the data is classified using Decision Trees (DT) the method based on RF and BCSO is better and this proves once again the fact that in some cases the appropriate use of BCSO in combination with a classical classification algorithm is better than the use of some combinations of classical classification and features selection algorithms.

TABLE 6.5
Comparison of the Results Obtained Using the BCSO Approach with Other Approaches for the Male Subject.

Classification Algorithm	Features Selection Approach	Accuracy	Number of Features
GBT	CF	0.995	55
GBT	PCA	0.988	51
DT	CF	0.976	55
DT	PCA	0.965	51
RF	BCSO	0.978	69
RF	CF	0.997	55
RF	PCA	0.993	51

TABLE 6.6
Comparison of the Results Obtained Using the BCSO Approach with Other Approaches for the Female Subject.

Classification Algorithm	Features Selection Approach	Accuracy	Number of Features
GBT	CF	0.995	60
GBT	PCA	0.989	50
DT	CF	0.980	60
DT	PCA	0.975	50
RF	BCSO	0.983	56
RF	CF	0.996	60
RF	PCA	0.996	50

Even though the results that are presented in this chapter cannot be compared perfectly with the results from other articles due to the fact that for each subject only one test data was used for each monitoring sensor and the feature that describes the temperature was eliminated due to the fact that it has constant values, compared to the results from [11] in which 10-fold cross validation is used, the values of the accuracy returned in the last iteration of the BCSO algorithm, namely 0.978 for the male subject and 0.983 for the female subject, are better or comparable with the values of the accuracy obtained in the case of Dynamic Time Warping (DTW) and of Artificial Neural Network (ANN). In the case of DTT the value of the accuracy is 0.978 and in the case of the ANN the value of the accuracy is 0.956.

However the comparison cannot be considered highly accurate because in the approach presented in this chapter the ratio between the number of samples from the training data and the number of samples from the testing data is $60 : 20 = 3 : 1$ and in the case of the 10-fold cross validation the ratio is $9 : 1$. Moreover in this chapter only one trial out of five trials is considered for each fall or activity for each monitored subject. These major differences might be enough to justify why the results obtained in the approach based on BCSO are not as good as the ones obtained in [11] using Support Vector Machines (SVM), K-Nearest Neighbors (K-NN), Bayesian Decision Making (BDM) and Least Squares Method (LSM) which return accuracy results better than 0.991.

6.5 Conclusions

In this chapter we described the original version of the CSO algorithm in brief, we presented illustrative modifications of the algorithm such as ICSO, MCSO, QCSO, BCSO, CCSO and ICSO-RHC, and we discussed how this algorithm can be applied for features selection in classification problems in which the data is characterized by a big number of features and has two

labels. The CSO algorithm in its binary version was applied further for the problem of falls classification in daily living activities. The CSO algorithm results are good classification results and the performance can be improved by varying the configuration parameters or by applying a modified version of the algorithm in order to avoid the situations in which the chicks are trapped in local optima. As future research work we want to: (1) propose different modifications of the CSO algorithm that are not approached in literature yet but which were approached in the case of other bio-inspired algorithms, (2) apply the CSO algorithm in other engineering problems and (3) conduct a more complex research in order to determine for each modification of the CSO algorithm the most representative classes of engineering problems that can be solved using that modification.

References

1. X. Meng, Y. Liu, X. Gao, H. Zhang. "A New Bio-inspired Algorithm: Chicken Swarm Optimization" in *Proc. of International Conference in Swarm Intelligence ICSI 2014: Advances in Swarm Intelligence*, Lecture Notes in Computer Science, vol. 8794, Springer, 2014, pp 86-94
2. C. Xiang, X. Tan, Y. Yang. "Improved Particle Swarm Optimization algorithm in dynamic environment" in *Proc. of the 26th Chinese Control and Decision Conference (2014 CCDC)*, 2014, pp. 3098-3102.
3. G. Ping, X. Chunbo, L. Jing, L. Yanqing. "Adaptive ant colony optimization algorithm" in *Proc. of the 2014 International Conference on Mechatronics and Control (ICMC)*, 2014, pp. 95-98.
4. S. Dhabal, S. Tagore, D. Mukherjee. "An improved Cuckoo Search Algorithm for numerical optimization" in *Proc. of the 2016 International Conference on Computer, Electrical & Communication Engineering (ICCECE)*, 2016, pp. 1-7.
5. R. Babers, A. E. Hassanien, N. I. Ghali. "A nature-inspired metaheuristic Lion Optimization Algorithm for community detection" in *Proc. of the 2015 11th International Computer Engineering Conference (ICENCO)*, 2015, pp. 217-222.
6. D. Moldovan, I. Anghel, T. Cioara, I. Salomie, V. Chifu, C. Pop. "Kangaroo mob heuristic for optimizing features selection in learning 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.

7. V. R. Chifu, I. Salomie, E. S. Chifu, A. Negrean, M. Antal. "Crab mating optimization algorithm" in *Proc. of the 2014 18th International Conference on System Theory, Control and Computing (ICSTCC)*, 2014, pp. 353-358.
8. S. Liang, T. Feng, G. Sun, J. Zhang, H. Zhang. "Transmission power optimization for reducing sidelobe via bat-chicken swarm optimization in distributed collaborative beamforming" in *Proc. of the 2016 2nd IEEE International Conference on Computer and Communications (ICCC)*, 2016, pp. 2164-2168.
9. S. Liang, T. Feng, G. Sun. "Sidelobe-level suppression for linear and circular antenna arrays via the cuckoo search–chicken swarm optimisation algorithm" in *IET Microwaves, Antennas & Propagation*, vol. 11, 2017, pp. 209-218.
10. S. Deb, K. Kalita, X.-Z. Gao, K. Tammi, P. Mahanta. "Optimal placement of charging stations using CSO-TLBO algorithm" in *Proc. of the 2017 Third International Conference on Research in Computational Intelligence and Communication Networks (ICRCICN)*, 2017, pp. 84-89.
11. A. T. Ozdemir, B. Barshan. "Detecting falls with wearable sensors using machine learning techniques" in *Sensors*, vol. 14, no. 6, 2014, pp. 10691-10708.
12. A. T. Ozdemir. "An analysis on sensor locations of the human body for wearable fall detection devices: principles and practice" in *Sensors*, vol. 16, no. 1161, 2016, pp. 1-25.
13. S. Deb, X.-Z. Gao, K. Tammi, K. Kalita, P. Mahanta. "Recent studies on Chicken Swarm Optimization algorithm: a review (2014–2018)" in *Artificial Intelligence Review*, 2019, pp. 1-29.
14. D. Wu, F. Kong, W. Gao, Y. Shen, Z. Ji. "Improved Chicken Swarm Optimization" in *Proc. of 2015 IEEE International Conference on Cyber Technology in Automation, Control, and Intelligent Systems (CYBER)*, 2015, pp. 681-686.
15. G. Chen, J. Jia, Q. Han. "Study on the strategy of decreasing inertia weight in particle swarm optimization algorithm" in *Journal of Xi'an Jiao Tong University*, vol. 40, no. 1, 2006, pp. 53-61.
16. K. Wang, Z. Li, H. Cheng, K. Zhang. "Mutation chicken swarm optimization based on nonlinear inertia weight" in *Proc. of the 2017 3rd IEEE International Conference on Computer and Communications (ICCC)*, 2017, pp. 2206-2211.
17. X. B. Meng, H. X. Li. "Dempster-Shafer based probabilistic fuzzy logic system for wind speed prediction" in *Proc. of the 2017 International Conference on Fuzzy Theory and its Applications (iFUZZY)*, 2017, pp. 1-5.

18. 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.

19. K. Ahmed, A. E. Hassanien, S. Bhattacharyya. "A novel chaotic chicken swarm optimization algorithm for feature selection" in *2017 Third International Conference on Research in Computational Intelligence and Communication Networks (ICRCICN)*, 2017, pp. 259-264.

20. X. Wang, L. Liu, Y. Zhang. "A novel chaotic block image encryption algorithm based on dynamic random growth technique" in *Optics and Lasers in Engineering*, vol. 66, 2015, pp. 10-18.

21. B. Wang, W. Li, X. Chen, H. Chen. "Improved chicken swarm algorithms based on chaos theory and its application in wind power interval prediction" in *Mathematical Problems in Engineering*, no. 1240717, 2019, pp. 1-10.

22. J. Wang, Z. Cheng, O. K. Ersoy, M. Zhang, K. Sun, Y. Bi. "Improvement and application of chicken swarm optimization for constrained optimization" in *IEEE Access*, vol. 7, 2019, pp. 58053-58072.

23. Y. Xu. "Research and implementation of improved random forest algorithm based on Spark" in *Proc. of the 2017 IEEE 2nd International Conference on Big Data Analysis (ICBDA)*, 2017, pp. 499-503.

24. G. Gousios. "Big data software analytics with Apache Spark" in *Proc. of the 2018 IEEE/ACM 40th International Conference on Software Engineering: Companion (ICSE-Companion)*, 2018, pp. 542-543.

25. L. Feltrin. "KNIME an open source solution for predictive analytics in the geosciences [software and data sets]" in *IEEE Geoscience and Remote Sensing Magazine*, vol. 3, no. 4, 2015, pp. 28-38.