Article

An Asymmetric Chaotic Competitive Swarm Optimization Algorithm for Feature Selection in High-Dimensional Data

Supailin Pichai , Khamron Sunat * and Sirapat Chiewchanwattana

Department of Computer Science, Faculty of Science, Khon Kaen University, Khon Kaen 40002, Thailand; supailin@kkumail.com (S.P.); sunkra@kku.ac.th (S.C.)

* Correspondence: skhamron@kku.ac.th

Received: 18 September 2020; Accepted: 22 October 2020; Published: 27 October 2020

Abstract: This paper presents a method for feature selection in a high-dimensional classification context. The proposed method finds a candidate solution based on quality criteria using subset searching. In this study, the competitive swarm optimization (CSO) algorithm was implemented to solve feature selection problems in high-dimensional data. A new asymmetric chaotic function was proposed and used to generate the population and search for a CSO solution. Its histogram is right-skewed. The proposed method is named an asymmetric chaotic competitive swarm optimization algorithm (ACCSO). According to the asymmetrical property of the proposed chaotic map, ACCSO prefers zero than one. Therefore, the solution is very compact and can achieve high classification accuracy with a minimal feature subset for high-dimensional datasets. The proposed method was evaluated on 12 datasets, with dimensions ranging from 4 to 10,304. ACCSO was compared to the original CSO algorithm and other metaheuristic algorithms. Experimental results show that the proposed method can increase accuracy and it reduces the number of selected features. Compared to different optimization algorithms with other wrappers, the proposed method exhibits excellent performance.

Keywords: asymmetry; chaos; skewed distribution; competitive swarm optimizer; metaheuristic algorithm; high-dimensional; feature selection

1. Introduction

The rapid development of inter-networking technology can gather data from many sources, such as, the Internet of Things, social networks, websites, health-related systems, and mobile devices, to name a few. The increment of the usage requirements and types of the device causes the number of attributes to rise. The increase in data volume is usually due to storage in real-time. The historical data will be used to assist in decision-making processes [1,2], in which a popular tool is machine learning (ML). However, the increased data attributes may lead to redundancy and irrelevance, resulting in the efficiency of an algorithm [3].

Generally, the feature extraction/representation modules and the classification are two main processing steps of ML [3]. Since the first step may produce high redundant or irrelevant data, selecting the data with essential features and deleting irrelevant features before feeding them to the classification step can increase efficiency and accuracy [2]. The decrease in the number of high-dimensional data features makes it challenging to design an efficient algorithm because the computational complexity is very high [2]. However, this challenge is worthy. A metaheuristic algorithm is usually an exploration-oriented population-based algorithm and development-oriented search algorithm [4]. For the past decade, metaheuristic algorithms, which are superior to precise

search and random search, have been widely used for feature selection. Though it might not be the best solution to the problem, the algorithm allows the user to produce an acceptable solution within a limited time [5]. Many search options have been applied to feature selection. Several articles reviewed metaheuristic algorithms for feature selection, such as References [2,3,6].

Particle swarm optimization (PSO) is a highly cited and generally widely used metaheuristic algorithm [6]. PSO is in the group of swarm algorithms. Recently, competitive swarm optimization (CSO), which is a significant variant of PSO, was proposed by Chen and Jin [7]. Based on the numerical benchmarks function in Reference [7], CSO performs well at large-scale numerical optimization problems. The critical steps of CSO are competition and update steps. The competition divides the population into winners and losers: the winners continue in further iterations, while the losers are updated. CSO has a lower computational cost than PSO because only half of its population is evaluated and updated. CSO has been improved to increase its efficiency [8–10]. Furthermore, CSO has been used to solve other problems, such as feature selection [11,12], to increase the efficiency of extreme learning machines [13], and for applications in cyber-physical systems [14]. These are shreds of evidence showing the excellent ability of the CSO algorithm in the literature. Based on PSO and CSO's comparison, conducted on six datasets, in the feature selection of high-dimensional classification [12], PSO decreased efficiency when the data dimension is high, while the CSO did not.

A nonlinear phenomenon called chaos has been used to enhance several metaheuristic algorithms. For instance, the CPSO algorithm is PSO with embedded chaos [15]. It outperformed the original PSO. Chaotic sequences help PSO efficiently balance the exploration and exploitation abilities [15], and the search capability of the algorithm increases when optimizing complex high-dimensional functions [16]. Furthermore, chaos can improve other swarm algorithms [17–21].

The hybridization of metaheuristic algorithms could also improve their performance compared to the canonical algorithms [22]. An exciting algorithm for hybridizing with a metaheuristic algorithm is the simulated annealing (SA) algorithm. SA enables local search and dramatically enhances local optimal hybrid algorithms [4].

In this study, we suppose that a dataset has n attributes; in other words, the dataset has a length of n . The critical point of feature selection is identifying the solution in $\{0, 1\}^n$. The complexity of this problem is very high [2]. However, a metaheuristic optimization algorithm can optimize for an acceptable solution. The optimization algorithms work on real values. Therefore, each particle's value must be mapped to the range of $(0, 1)$ to produce a binary solution. A "0" means the feature is not selected, and a "1" means the corresponding attribute is chosen. For example, an explanation of a dataset of length 6 with a value of "001101" is that attributes 3, 4, and 6 are selected, and the rest are discarded. For any two different solutions producing the same accuracy, fewer 1s, aka a more compact solution, is preferable. Therefore, we propose a new asymmetric chaotic map to use for generating new particles in CSO. This chaotic map distribution is asymmetric and right-skewed, producing zero value with a higher probability than one value.

Other than applying the original CSO algorithm to solve binary optimization problems with high dimensions, the main contributions of this paper are as follows:

1. We propose a novel feature selection method based on an embedding of the proposed asymmetric chaotic map in CSO. The proposed method can deal with high-dimensional problems effectively. This paper is the first work combining asymmetric chaotic CSO.
2. The proposed method is compared with wrapper feature selection methods based on other swarm optimization algorithms in terms of classification accuracy and the selected feature subset's compactness.
3. A graphical competitive magic quadrant is used to depict the ability of the proposed method.

This paper is organized as follows. Section 1 introduces the overall background of the study and presents other related studies. The materials and methods used in this study are described in Section 2.

The proposed method is elaborated in Section 3. Section 4 presents and discusses the experiments and results. Finally, Section 5 presents the conclusions.

2. Materials and Methods

Several metaheuristic or evolutionary algorithms have achieved increased search abilities by incorporating many methods to improve the algorithms. One example of such an improved swarm algorithm is CSO. It can be combined with chaos and SA. The explanations of them are as follows.

2.1. Competitive Swarm Optimization (CSO)

The recently proposed CSO is an efficient algorithm for large-scale optimization [7]. The algorithm has been utilized with test functions of 2000 and 5000 dimensions. That testing is the highest challenge that has ever been reported in the evolutionary optimization literature [7].

Compared with a PSO algorithm, CSO requires less computational cost. A particle learns from a randomly selected competitor in CSO instead of the global or personal best position, as in PSO. In each iteration, the swarm is randomly divided into two groups, and pairwise competitions are carried out between the particles from each group. After each pair of competitions, the winner particle is directly passed to the next iteration, while the loser particle updates its position and velocity by learning from the winner particle as follows:

$$v_l^{t+1} = R_1^t \odot v_l^t + R_2^t \odot (x_w^t - x_l^t) + \varnothing R_3^t \odot (x^{-t} - x_l^t), \quad (1)$$

$$x_l^{t+1} = x_l^t + v_l^{t+1}, \quad (2)$$

where t is the iteration counter. The three vectors, R_1^t , R_2^t and R_3^t are randomly generated vectors within $[0, 1]^n$. x_w^t and x_l^t denote the winner particle and the loser particle, respectively. x^{-t} indicates the mean position of the current swarm in iteration t , and $\varnothing$ is a parameter that controls the influence of x^{-t} ; in our experiment, $\varnothing = 0.2$, $\odot$ is element-wise multiplication.

2.2. Chaos

Chaos is a nonlinear phenomenon and has complicated and semi-random behaviors. Its qualities include ergodicity, randomness, and sensitivity. A chaotic map is usually generated by a simple deterministic function and can pass through all states in a range without duplicates. It is very unstable to the initial value, meaning that small changes in the initial value may lead to differences in the output [23].

Chaos has been used to improve the efficiency and variety of algorithms widely used in various science fields by combining chaotic maps. For example, the appropriate chaos function allows the control parameters of the gray wolf optimization (GWO) algorithm to find the optimal solution more quickly and adjust the algorithmic convergence rate [19]. Either they generated chaotic sequences to replace the random sequences of gravitational search algorithm (GSA) parameters or used the chaos to perform local searches and had a better result than the original GSA did [24]. For combining PSO with the chaos algorithm (CPSO), the built-in chaos outperforms the original PSO and balances the exploration and exploitation capability reasonably and efficiently [15].

2.3. Simulated Annealing (SA)

The SA algorithm is derived from the annealing process and used to reduce the energy to a stable state. This idea was proposed by Kirkpatrick et al. in 1983 to solve the problem of stagnating in local optima [25]. SA has a probability of accepting worse solutions, and the algorithm starts with a randomly generated solution. The algorithm tries to produce the best neighborhood solution so far according to the predefined neighborhood structure and evaluates it with the fitness function in each round. Improvement is acceptable, while a worse neighbor is accepted with a certain probability

determined by the Boltzmann probability, $p = e^{-\theta/T}$, where θ is the difference between the fitness of the best solution (*BestSol*) and that of the generated neighbor (*TrialSol*). However, the T value, a parameter also known as temperature, gradually decreases periodically during the cooling process. In this study, the initial temperature is specified as $T_0 = 2 * |N|$, where $|N|$ is the number of attributes in each dataset, and the cooling time is scheduled by $T = 0.93 * T$, as in Reference [4]. The pseudocode of SA is shown in Algorithm 1.

The main difference between a swarm-based algorithm and SA is that the former generates multiple trial points at a time, while the latter produces one possible solution. According to the literature [26], SA could improve the solution of the swarm-based algorithm. A hybridization between CSO and SA produces CSO followed by the SA algorithm (CSO-SA). Therefore, we also studied another combination, ACCSO followed by SA (ACCSO-SA). The detailed procedure of CSO-SA is summarized in Algorithm 2. Note that if the algorithm is ACCSO, then line numbers 20 and 21 of Algorithm 2 will be removed.

Algorithm 1: The pseudocode of the SA algorithm

```

1:   $T_0 = 2 * |N|$ , where  $|N|$  is the number of attributes for each dataset
2:   $BestSol = S_i$ 
3:   $\delta(BestSol) = \delta(S_i)$  //  $\delta$  indicates the quality of the solution
4:  while  $T > T_0$ 
5:    generate at random a new solution TrailSol in a neighbor of  $S'_i$ 
6:    calculate  $\delta(TrailSol)$ 
7:    if ( $\delta(TrailSol) > \delta(BestSol)$ )
8:       $S'_i = TrailSol$ 
9:       $BestSol = TrailSol$ 
10:    $\delta(S_i) = \delta(TrailSol)$ 
11:    $\delta(BestSol) = \delta(TrailSol)$ 
12:   else if ( $\delta(TrailSol) = \delta(BestSol)$ )
13:     Calculate  $|TrailSol|$  and  $|BestSol|$ 
14:     if ( $|TrailSol| < |BestSol|$ )
15:        $S'_i = TrailSol$ 
16:        $BestSol = TrailSol$ 
17:        $\delta(S_i) = \delta(TrailSol)$ 
18:        $\delta(BestSol) = \delta(TrailSol)$ 
19:     end if
20:   else // accepting a worse solution
21:     Calculate  $\theta = \delta(TrailSol) - \delta(BestSol)$ 
22:     Generate a uniform random number,  $P = [0,1]$ 
23:     if ( $P \leq e^{-\theta/T}$ )
24:        $S'_i = TrailSol$ 
25:        $\delta(S_i) = \delta(TrailSol)$ 
26:     end if
27:   end if
28:    $T = 0.93 * T$  //update temperature
29: end while
30: Output BestSol

```

Algorithm 2: The pseudocode of competitive swarm optimization (CSO) followed by SA (CSO-SA)

```

1:  t = 0;
2:  randomly initialize particles P(0);                                     //(*)
3:  while the termination condition is not satisfied do
4:    Calculate the fitness of each particle in P(t);
5:    U = P(t), P(t + 1) = {};
6:    while U is not empty do
7:      uniformly random choose two particles  $X_1(t)$ ,  $X_2(t)$  from U;
8:      if  $f(X_1(t)) \leq f(X_2(t))$  then
9:         $X_w(t) = X_1(t)$ ,  $X_1(t) = X_2(t)$ ;
10:     else
11:        $X_w(t) = X_2(t)$ ,  $X_1(t) = X_1(t)$ ;
12:     end if
13:     add  $X_w(t)$  into P(t + 1)
14:     randomly generate  $R_1^t, R_2^t, R_3^t$                                      //(*)
15:     update  $X_1(t)$  using (1) and (2);
16:     add the updated  $X_1(t + 1)$  to P(t + 1)
17:     Remove  $X_1(t)$ ,  $X_2(t)$  from U;
18:   end while
19:   t = t + 1;
20:
21:   Use the SA algorithm (Algorithm 1) to find a better gbest
22:   t = t + 1;
23: end while

```

Note: (1) The symbol “//” is a comment, (2) if the algorithm is without SA, then line numbers 20 and 21 are removed.

3. The Proposed Asymmetric Chaotic Competitive Swarm Optimization

This section describes the development of the proposed asymmetric chaotic map. The proposed chaotic map is used to develop an asymmetric chaotic CSO (ACCSO) algorithm and an ACCSO followed by SA algorithm (ACCSO-SA).

3.1. The Proposed Asymmetric (Right-Skewed) Chaotic Map

Existing PSO variants attempt to modify the global best solution, resulting in only limited performance [7]. CSO has also been done on developing PSO. However, CSO has shown performance for the high-dimensional data. In CSO, the new particles are usually updated from the loser's velocity vectors using three uniform random vectors: R_1^t , R_2^t , and R_3^t . This paper proposes an asymmetric chaotic map based on a combination of Kent map and the neuron map for producing those three vectors. The existing Kent map, which is computed by Equation (3) and produces values in (0, 1), has been used in many applications [24]. Similarly, neuron map chaos, which is computed by Equation (4) and generates values in (−1.5, 0.5), has achieved excellent performance on several vital benchmarks [23]. The proposed chaotic map generates chaotic sequences in (0, 1), and a new map is formed as Equation (5). Switching between the Kent map and the neuron map is controlled by the parameter m , defined as 0.72 in this paper.

$$x_{k+1} = \begin{cases} \frac{x_k}{m}, & 0 < x_k \leq m \\ \frac{1-x_k}{1-m}, & m < x_k \leq 1, \quad 0 < m < 1 \end{cases} \quad (3)$$

Kent map, Equation (3), generates sequences in (0, 1) if $0 < m < 1$.

$$x_{k+1} = \eta - 2 \tanh(\gamma) \exp(-3x_n^2) \quad (4)$$

Neuron map, Equation (4), generates sequences in $(-1.5, 0.5)$ if $\eta = 0.5$, $\gamma = 5$, where η and γ denote the attenuation factor and proportionality factor, respectively.

$$x_{k+1} = \begin{cases} \frac{x_k}{m}, & m < x_k \leq 1 \\ \eta - 2\tanh(\gamma) \exp(-3x_\eta^2) & \end{cases} \quad (5)$$

The proposed asymmetric chaotic map, Equation (5), generates sequences in $(0, 1)$ if $0 < m < 1$.

Figure 1a shows a histogram of the Kent map, and Figure 1b shows a histogram of the neuron map. The histogram and examples of chaotic sequences produced by the proposed chaotic map are shown in Figure 2a,b, respectively.

Figure 1. (a) Histogram of the Kent map, and (b) histogram of the neuron map.

Figure 2. (a) Histogram of asymmetric chaotic sequences, and (b) chaotic sequences generated by the proposed chaotic map, Equation (5).

The incorporation of the two chaotic maps results in a new hybrid chaotic map that is different from the neuron map and Kent map. The Kent map histogram illustrates a nearly uniform structure ranging from 0 to 1. The neuron map histogram structure roughly shows three clusters ranging from -1.5 to 1. The histogram of the proposed chaotic map shown in Figure 2a depicts an asymmetric chaos histogram structure and is in a step shape. Figure 2b suggests that the proposed chaotic map sequences range from 0 to 1. Though the asymmetric chaos map generates values in the range of $(0, 1)$, it has the highest possibility of generating values in the range of $(0.3, 0.4)$, whereas values in the range of $(0, 0.077)$ are scarce.

Feature selection is a binary and high-dimensional problem. The values generated from each chaotic function are mapped to the range of $(0, 1)$. Let U be the set of generated values greater than

or equal to 0.5, and L be the set of generated values that are less than 0.5. A value of L will be interpreted as a 0 or not selected, whereas a value of U will be interpreted as a 1 or selected. From our experiments, the ratios of $\frac{|U|}{|L|}$ produced from the asymmetric chaos map, neuron map, and Kent map are 0.7496, 0.8223, and 0.9977, respectively. These results mean that the asymmetric chaos map has more asymmetry than the neuron map and Kent map do. CSO can use a chaotic map to produce its particle. Therefore, an algorithm that uses the proposed asymmetric chaotic function will prefer “0” (not selected) to “1” (selected). Thus, the solution produced by the proposed method should be more compact than those produced from different algorithms.

3.2. The Process Approach

A multiobjective optimization with two different objectives can regard feature selection. The first objective is to obtain the least number of selected features, and the second objective is to produce the highest accuracy. A solution can be regarded as a better solution when it involves fewer selected features with higher accuracy. Two goals are combined as one fitness function and shown in Equation (6). The dataset with the selected features is classified by the K-nearest neighbor (KNN) classifier, as in Reference [4]. The search algorithms use Equation (6) as their fitness function.

$$fitness = \alpha \gamma_R(D) + \beta \frac{|R|}{|N|}, \quad (6)$$

where $\gamma_R(D)$ defines the given classification error rate, $|R|$ is the cardinality of the selected subset, $|N|$ is the total number of features in the dataset, and α and β are parameters corresponding to the importance of the classification of the number of selected features. $\alpha = [0, 1]$ and $\beta = (1 - \alpha)$, and $\beta = 0.01$ in current experiments [27].

3.3. An Asymmetric Chaotic CSO

This study proposes a way to increase CSO's effectiveness by using an asymmetric chaos map instead of a uniform random generator. Therefore, we name the new method as an asymmetric chaotic CSO (ACCSO). For $m = 0.72$, the chaotic sequences in the interval (0, 1) can be obtained. The pseudocode of ACCSO is shown in Algorithm 3.

Algorithm 3: An asymmetric chaotic competitive swarm optimization (ACCSO)

- 1: The pseudocode of Algorithm 3 is as same as Algorithm 2.
 - 2: There are three modifications in Algorithm 2.
 - 3: The first modification is at line number 2: a population of N particles is initialized using the proposed asymmetric chaos map.
 - 4: The second change is at line number 14: the three vectors R_1^t , R_2^t , and R_3^t are randomly generated vectors within $[0, 1]^n$ using the proposed asymmetric chaos function.
 - 5: Line numbers 20 and 21 of Algorithm 2 will be removed.
The rest remains the same.
-

3.4. An Asymmetric Chaos CSO Followed by SA (ACCSO-SA)

If the SA algorithm is used to hybrid with ACCSO, then the method is called an ACCSO-SA. The pseudocode is shown in Algorithm 4.

Algorithm 4: An asymmetric chaotic competitive swarm optimization followed by SA (ACCSO-SA)

- 1: The pseudocode of Algorithm 4 is as the same as Algorithm 3, but line numbers 20 and 21 are uncommented.
 - 2: The rest remains the same.
-

4. Experimental Results

4.1. Dataset

The proposed method was applied to datasets taken from University of California Irvine Machine Learning Repository (UCI taken from <https://archive.ics.uci.edu/mL/datasets.html>) and Arizona State University (ASU taken from <http://featureselection.asu.edu/datasets.php>). The characteristics of the datasets are shown in Table 1. The data include the number of instances, the number of classes, and the number of attributes as the number of features ranges from low to high dimensions.

Table 1. List of benchmark classification datasets.

No.	Dataset Name	Number of Instances	Number of Attributes	Number of Classes
1	Iris	150	4	3
2	Lirbras	360	90	15
3	ORL	200	1024	40
4	Colon	62	2000	2
5	Madelon	2600	500	2
6	Lymphoma	96	4026	9
7	Glioma	50	4434	4
8	Prostate_GE	102	5966	2
9	Nci9	60	9172	9
10	Carcinom	174	9182	11
11	Pixraw10P	100	10,000	10
12	Orlraw10P	100	10,304	10

4.2. Parameter Settings

In this study, to achieve an effective product for a wrapper approach based on the KNN classifier with Euclidean classification (where $K = 5$, as in Reference [4]), the value was identically set for every dataset [22]. Regarding K-fold cross-validation, K-1 is used for training and validation, and the rest are used for testing.

The population in this study is 10, and the number of iterations is 100. Each algorithm was run 30 times with a random seed via the MATLAB2017b environment, Windows 10, and an Intel Core i7 3.40 GHz processor with 8 GB of RAM. The population in SA is 10, and the run time is 30.

4.3. Comparison of CSO-Kent, CSO-Neuron, and ACCSO

We consider each of the three chaos maps to generate the initial population and update the CSO's search agent positions: Kent map, neuron map, and the proposed asymmetric chaos map. In the first part of the experiment, comparing these three methods' performance was conducted on 12 datasets. Table 2 displays the experimental results in terms of the accuracy and the number of selected features. Note that the best results among each dataset are highlighted with bold text. It can be seen that ACCSO is better than CSO-Kent and CSO-Neuron regarding the summation of the chosen features. Moreover, the solution of ACCSO produces more accuracy than that produced by CSO-Kent and CSO-Neuron. These achievements of ACCSO are caused by the asymmetrical property of the proposed chaos map in the initial population generating and each search agent's position updating.

Table 2. Comparison of the average number of selected features and accuracy among the proposed and related work methods.

Dataset	Attribute			Accuracy		
	CSO-Kent	CSO-Neuron	ACCSO	CSO-Kent	CSO-Neuron	ACCSO
Iris	2	2	2	0.9600	0.9600	0.9467
Lirbras	41.5	43.4	43.1	0.7924	0.7844	0.8196
Orl	503.7	506.8	504.4	0.8572	0.8930	0.7968
Colon	988.4	970	973.6	0.7806	0.7161	0.7796
Madelon	249.9	259	278.51	0.6900	0.6985	0.8247
Lymphoma	1961	1953	1975.4	0.9167	0.9792	0.9000
Glioma	2185.9	2185.8	2179.7	0.6027	0.9280	0.8453
Prostate_ge	2957.1	2981.2	2916.4	0.8595	0.7804	0.9033
Nci9	4816.5	4848.2	4768.4	0.8567	0.8680	0.8333
Carcinom	4560.9	4561.4	4555.7	0.8808	0.9011	0.9042
Pixraw10p	4852.8	4856.2	4876.7	0.9800	1.0000	1.0000
Orlraws10p	5061.8	5074.4	5002.5	0.9400	0.8800	0.9400
Sum all	28,181.5	28,241.4	28,076.41	10.1166	10.3887	10.4935
Ranking	2	3	1	3	2	1

4.4. Comparison of ACCSO and ACCSO-SA

In the second part of the experiment, we want to check if SA should follow ACCSO. Figure 3a depicts the histogram of the accuracy, and Figure 3b displays the histogram of the selected features for 12 datasets, respectively. ACCSO uses the proposed asymmetric chaos map to generate the initial population and to update the position of each search agent, and ACCSO-SA allows SA to search for a better local optimal solution. There are six datasets for which the accuracy of ACCSO is higher than that of CSO, see Figure 3a. Additionally, there are nine datasets for which the accuracy of ACCSO is higher than that of ACCSO-SA. ACCSO achieves the best accuracy among the three algorithms. Similarly, Figure 3b shows that ACCSO-SA recognized the smallest number of the selected features. Therefore, between ACCSO-SA and ACCSO-SA, we cannot immediately conclude which one has the best performance in finding the smallest feature size and yields the highest accuracy on most of the datasets.

(a)

Figure 3. Cont.

Figure 3. Comparison between the proposed asymmetric chaotic CSO algorithm and other algorithms based on the accuracy (a) and the number of features (b).

4.5. Comparison with Other Algorithms

The features are selected by a group of existing algorithms or frequently used swarm optimization algorithms such as PSO, gray wolf optimization (GWO), symbiotic organisms search (SOS), the bat algorithm (BAT), and CSO. Table 3 shows the average accuracy for each dataset when using each algorithm, including ACCSO and ACCSO-SA. Each algorithm's accumulated accuracy values are ranked from the highest to the lowest in the following order: the ACCSO, ACCSO-SA, CSO-Kent, CSO-Neuron, PSO, CSO, GWO, SOS, and BAT. According to Table 3, we can see that ACCSO comes first. This result implies that CSO's performance using chaos to generate the population was superior to that of the original CSO that did not use a chaos map, in terms of accuracy. It can also be seen through observations from the results shown in Table 3 that the ACCSO, CSO-Neuron, CSO-Kent, and ACCSO-SA perform better than the original CSOs.

Table 3. Classification accuracy and ranking comparison between the proposed and related work methods.

Dataset	PSO	GWO	SOS	BAT	CSO	CSO-Kent	CSO-Neuron	ACCSO	ACCSO-SA
Iris	0.9600	0.9600	0.9556	0.9562	0.9733	0.9600	0.9600	0.9467	0.9600
Lirbras	0.7789	0.8133	0.7912	0.7914	0.8278	0.7924	0.7844	0.8196	0.8289
ORL	0.864	0.8050	0.7756	0.7730	0.8720	0.8572	0.893	0.7968	0.9200
Colon	0.8094	0.8645	0.7333	0.7373	0.7097	0.7806	0.7161	0.7796	0.7419
Madelon	0.6935	0.7018	0.7163	0.7067	0.7011	0.6900	0.6985	0.8247	0.6926
Lymphoma	0.9958	0.8958	0.9396	0.8250	0.9792	0.9167	0.9792	0.9000	0.8542
Glioma	0.8400	0.8000	0.6863	0.6943	0.7200	0.6027	0.9280	0.8453	0.7840
Prostate_ge	0.8331	0.7451	0.8357	0.8437	0.8235	0.8595	0.7804	0.9033	0.8863
Nci9	0.6200	0.6800	0.7685	0.6067	0.6133	0.8567	0.8680	0.8333	0.7340
Carcinom	0.8076	0.9103	0.7075	0.6997	0.9356	0.8808	0.9011	0.9042	0.9011
Pixraw10p	0.9400	0.9400	0.9524	0.9523	0.9800	0.9800	1	1	0.9600
Orlraws10p	0.9600	0.9400	0.8984	0.8989	0.9400	0.9400	0.8800	0.9400	0.9240
Average	10.1023	10.0558	9.7604	9.4852	10.0755	10.1166	10.3887	10.4935	10.187
Ranking	5	7	8	9	6	4	2	1	3

Table 4 indicates the average number of the selected features and their ranking in the summation of all selected features. The results show that the ACCSO outperformed the other algorithms on the feature selection of high-dimensional data. Notably, the proposed asymmetric chaotic map helps CSO to surpass CSO-Kent and CSO-Neuron in feature selection.

Table 4. Average number of selected features and ranking comparison between the proposed and related work methods.

Dataset	PSO	GWO	SOS	BAT	CSO	CSO-Kent	CSO-Neuron	ACCSO	ACCSO-SA
Iris	1	2	1	1	1	2	2	2	2
Lirbras	42.8	46.2	43	47	42.4	41.5	43.4	43.1	42.4
Orl	513.4	497.8	532	516	505.6	503.7	506.8	504.4	497
Colon	990.6	1012	1036	967	948	988.4	970	973.6	993.2
Madelon	253.6	245.4	257	239	258.2	249.9	259	278.51	245.8
Lymphoma	2029.8	1993.6	1992	1921	1951.4	1961	1953	1975.4	1940.2
Glioma	2155	2172.8	2204	2246	2162.6	2185.9	2185.8	2179.7	2200.8
Prostate_ge	2942	2943.8	2971	2959	2931.4	2957.1	2981.2	2916.4	2941.2
Nci9	4804.8	4725.8	4942	4674	4798	4816.5	4848.2	4768.4	4834.2
Carcinom	4552.4	4551.2	4593	4591	4577.2	4560.9	4561.4	4555.7	4524.6
Pixraw10p	4854.4	4857	5011	4988	4868	4852.8	4856.2	4876.7	4855.6
Orlraws10p	5048.8	5106.6	5145	5190	5108.4	5061.8	5074.4	5002.5	5043.4
Average	28,188.6	28,154.2	28,727	28,339	28,152.2	28,181.5	28,241.4	28,076.41	28,120.4
Ranking	6	4	9	8	3	5	7	1	2

4.6. Magic Quadrant

The magic quadrant graphically shows the algorithms' competitive positions according to their accuracy and the selected number of dimensions. Figure 4 shows the competitiveness of nine algorithms that present a considerable dissimilarity. This figure was adapted from Reference [28]. The formation depends on the accuracy and the number of dimensions and depicts the magic quadrant to reveal how different each algorithm is. The x-axis shows the ranks based on the number of selected dimensions, taken from Table 4. The y-axis shows the positions based on the classification accuracy, taken from Table 3.

Figure 4. The magic quadrant is produced by 9 competitive algorithms divided by quartile.

The magic quadrant suggests that the positions closer to the top-right corner indicate better performance. The algorithms in Q1 are in the leader quadrant because they are superior to the algorithms in other quadrants in terms of dimension reduction and having high accuracy. The ACCSO, ACCSO-SA,

and CSO-Kent are located in the top-right corner, which is called Q1. However, the ACCSO has higher-ranking performance for dimension reduction than ACCSO-SA and CSO-Kent. CSO-Neuron and PSO are in Q2—the Rank Challengers. Q3, the Visionaries, accommodates CSO and GWO, while SOS and BAT belong to Q4, the Niche Players.

It can now be concluded that for feature selection of high-dimensional data, ACCSO performs well in accuracy and the number of selected features. This achievement is affected by using the proposed asymmetric chaotic map with CSO.

5. Conclusions

In this study, CSO using an asymmetric chaos map was proposed to solve the problem of feature selection and was validated using 12 datasets taken from UCI and ASU. The fitness function depends on the number of selected features and the classification error rate. The proposed asymmetric chaos map makes CSO prefer a “0” than a “1”.

By comparing three algorithms, CSO, ACCSO, and ACCSO-SA, on the 12 datasets, we found that ACCSO-SA is best on more datasets than the other algorithms, 6 out of 12, based on the number of selected features, and ACCSO is best on more datasets than the other algorithms, 9 out of 12, based on the classification accuracy. If the summation of the numbers of selected features is computed from all datasets, then ACCSO has the lowest number of chosen features among the three algorithms. This evidence shows that SA should not follow ACCSO because the algorithm will produce an overfitting solution.

Therefore, the proposed asymmetric chaotic map combined with CSO can boost the algorithm search capability to achieve higher accuracy and more compact selected features than the Kent map and neuron map.

The ACCSO outperformed the other algorithms, ACCSO-SA, PSO, GWO, SOS, BAT, and CSO, for feature selection in high-dimensional data. All the algorithms were then ranked and presented via a magic quadrant. The magic quadrant revealed that the improved CSO outperformed the original CSO algorithm, PSO, GWO, SOS, and BAT in solving the problem of feature selection for high dimensions.

There are still many aspects to be developed for further research, such as increasing ACCSO's effectiveness by hybridizing it with other state-of-the-art metaheuristic algorithms to solve engineering problems or other challenging issues. Furthermore, the proposed asymmetric chaotic map might be effectively used in different metaheuristic algorithms.

Author Contributions: Conceptualization, methodology, writing—original draft, software, writing—review, S.P.; writing—review, editing, supervision, investigation, K.S.; writing—review, editing, investigation, S.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Hossain, E.; Khan, I.; Un-noor, F.; Sikander, S.S.; Sunny, S.H. Application of Big Data and Machine Learning in Smart Grid, and Associated Security Concerns: A Review. *IEEE Access* **2019**, *7*, 13960–13988. [\[CrossRef\]](#)
2. Rong, M.; Gong, D.; Gao, X. Feature Selection and Its Use in Big Data: Challenges, Methods, and Trends. *IEEE Access* **2019**, *7*, 19709–19725. [\[CrossRef\]](#)
3. Al-Tashi, Q.; Abdulkadir, S.J.; Rais, H.M.; Mirjalili, S.; Alhussian, H. Approaches to Multi-Objective Feature Selection: A Systematic Literature Review. *IEEE Access* **2020**, *8*, 125076–125096. [\[CrossRef\]](#)
4. Jia, H.; Li, J.; Song, W.; Peng, X.; Lang, C. Spotted Hyena Optimization Algorithm With Simulated Annealing for Feature Selection. *IEEE Access* **2019**, *7*, 71943–71962. [\[CrossRef\]](#)
5. Dhiman, G.; Kumar, V. Spotted hyena optimizer: A novel bio-inspired based metaheuristic technique for engineering applications. *Adv. Eng. Softw.* **2017**, *114*, 48–70. [\[CrossRef\]](#)
6. Brezočnik, L.; Fister, I.; Podgorelec, V. Swarm intelligence algorithms for feature selection: A review. *Appl. Sci.* **2018**, *8*, 1521. [\[CrossRef\]](#)

7. Cheng, R.; Jin, Y. A Competitive Swarm Optimizer for Large Scale Optimization. *IEEE Trans. Cybern.* **2014**, *45*, 191–204. [\[CrossRef\]](#)
8. Sun, C.; Ding, J.; Zeng, J.; Jin, Y. Regular research paper a fitness approximation assisted competitive swarm optimizer for large scale expensive optimization problems. *Memetic Comput.* **2018**, *10*, 123–134. [\[CrossRef\]](#)
9. Xiong, G.; Shi, D. Orthogonal learning competitive swarm optimizer for economic dispatch problems. *Appl. Soft Comput.* **2018**, *66*, 134–148. [\[CrossRef\]](#)
10. Ling, T.; Zhan, Z.-H.; Wang, Y.; Wang, Z.-J.; Yu, W.-J.; Zhang, J. Competitive Swarm Optimizer with Dynamic Grouping for Large Scale Optimization. In Proceedings of the 2018 IEEE Congress on Evolutionary Computation (CEC), Rio de Janeiro, Brazil, 8–13 July 2018; pp. 2655–2660.
11. Too, J.; Abdullah, A.R.; Saad, N.M. Binary Competitive Swarm Optimizer Approaches for Feature Selection. *Computation* **2019**, *7*, 31. [\[CrossRef\]](#)
12. Gu, S.; Cheng, R.; Jin, Y. Feature selection for high-dimensional classification using a competitive swarm optimizer. *Soft Comput.* **2018**, *22*, 811–822. [\[CrossRef\]](#)
13. Eshtay, M.; Faris, H.; Obeid, N. Improving Extreme Learning Machine by Competitive Swarm Optimization and its application for medical diagnosis problems. *Expert Syst. Appl.* **2018**, *104*, 134–152. [\[CrossRef\]](#)
14. Huang, S.; Tao, M. Competitive swarm optimizer based gateway deployment algorithm in cyber-physical systems. *Sensors* **2017**, *17*, 209. [\[CrossRef\]](#) [\[PubMed\]](#)
15. Liu, B.; Wang, L.; Jin, Y.-H.; Tang, F.; Huang, D.-X. Improved particle swarm optimization combined with chaos. *Chaos Solitons Fractals* **2005**, *25*, 1261–1271. [\[CrossRef\]](#)
16. Dong, S.; Dong, Z.; Ma, J.; Chen, K. Improved PSO algorithm based on chaos theory and its application to design flood hydrograph. *Water Sci. Eng.* **2010**, *3*, 156–165.
17. Gandomi, A.H.; Yun, G.J.; Yang, X.S.; Talatahari, S. Chaos-enhanced accelerated particle swarm optimization. *Commun. Nonlinear Sci. Numer. Simul.* **2013**, *18*, 327–340. [\[CrossRef\]](#)
18. Wang, G.-G.; Deb, S.; Gandomi, A.H.; Zhang, Z.; Alavi, A.H. Chaotic cuckoo search. *Soft Comput.* **2016**, *20*, 3349–3362. [\[CrossRef\]](#)
19. Kohli, M.; Arora, S. Chaotic grey wolf optimization algorithm for constrained optimization problems. *J. Comput. Des. Eng.* **2017**, *5*, 458–472. [\[CrossRef\]](#)
20. Saha, S.; Mukherjee, V. A novel chaos-integrated symbiotic organisms search algorithm for global optimization. *Soft Comput.* **2017**, *22*, 3797–3816. [\[CrossRef\]](#)
21. Wang, Y.; Li, H.; Gao, H.; Kwong, S. A Chaotic Based Artificial Bee Colony Algorithm. In Proceedings of the 2018 Fifth HCT Information Technology Trends (ITT), Dubai, UAE, 28–29 November 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 165–169.
22. Talbi, E.-G. A Taxonomy of Hybrid Metaheuristics. *J. Heuristics* **2002**, *8*, 541–564. [\[CrossRef\]](#)
23. Feng, J.; Zhang, J.; Zhu, X.; Lian, W. A novel chaos optimization algorithm. *Multimed. Tools Appl.* **2017**, *76*, 17405–17436. [\[CrossRef\]](#)
24. Gao, S.; Vairappan, C.; Wang, Y.; Cao, Q.; Tang, Z. Gravitational search algorithm combined with chaos for unconstrained numerical optimization. *Appl. Math. Comput.* **2014**, *231*, 48–62. [\[CrossRef\]](#)
25. Kirkpatrick, S.; Gelatt, C.D.; Vecchi, M.P. Optimization by simulated annealing. *Science* **1983**, *220*, 671–680. [\[CrossRef\]](#) [\[PubMed\]](#)
26. Wang, G.-G.; Guo, L.; Gandomi, A.H.; Alavi, A.H.; Duan, H. Simulated Annealing-Based Krill Herd Algorithm for Global Optimization. *Abstr. Appl. Anal.* **2013**, *2013*, 213853. [\[CrossRef\]](#)
27. Emary, E.; Zawbaa, H.M.; Hassanien, A.E. Binary grey wolf optimization approaches for feature selection. *Neurocomputing* **2016**, *172*, 371–381. [\[CrossRef\]](#)
28. Nasa-ngium, P.; Sunat, K. Impacts of Linnik Flight Usage Patterns on Cuckoo Search for Real-Parameter Global Optimization Problems. *IEEE Access* **2019**, *7*, 83932–83961. [\[CrossRef\]](#)

Publisher’s Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

© 2020 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).