Research Article

A Clustering-Guided Integer Brain Storm Optimizer for Feature Selection in High-Dimensional Data

Jia Yun-Tao ¹, Zhang Wan-Qiu,² and He Chun-Lin²

¹Zhuhai Campus, Beijing Institute of Technology, Zhuhai, China

²School of Information and Control Engineering, China University of Mining and Technology, Xuzhou, China

Correspondence should be addressed to Jia Yun-Tao; jyt-002@126.com

Received 14 July 2021; Revised 19 August 2021; Accepted 4 September 2021; Published 15 September 2021

Academic Editor: Shi Cheng

Copyright © 2021 Jia Yun-Tao et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

For high-dimensional data with a large number of redundant features, existing feature selection algorithms still have the problem of “curse of dimensionality.” In view of this, the paper studies a new two-phase evolutionary feature selection algorithm, called clustering-guided integer brain storm optimization algorithm (IBSO-C). In the first phase, an importance-guided feature clustering method is proposed to group similar features, so that the search space in the second phase can be reduced obviously. The second phase applies oneself to finding optimal feature subset by using an improved integer brain storm optimization. Moreover, a new encoding strategy and a time-varying integer update method for individuals are proposed to improve the search performance of brain storm optimization in the second phase. Since the number of feature clusters is far smaller than the size of original features, IBSO-C can find an optimal feature subset fast. Compared with several existing algorithms on some real-world datasets, experimental results show that IBSO-C can find feature subset with high classification accuracy at less computation cost.

1. Introduction

Feature selection (FS), as an important dimension reduction method, has been applied in various real problems, such as image processing and text classification [1, 2]. In general, a lot of irrelevant/redundant features will slow down the speed of learning algorithms, even reducing their learning accuracy. The purpose of FS is to eliminate those irrelevant and/or redundant features, thus shortening the learning time while improving the learning accuracy [3–6].

Swarm intelligence (SI) is a branch of evolutionary computation [5]. Since it can find optimal or suboptimal solutions by global search strategies, SI has become an effective approach to solve FS problems. Now many SI-based algorithms have been applied in FS problems, such as particle swarm optimization [7–12], differential evolution [13–15], artificial bee colony algorithm [16–18], firefly algorithm [19, 20], salp swarm algorithm [21], ant colony optimization [6], and whale optimization algorithm [22].

Brain storm optimization (BSO) is a new swarm intelligence algorithm simulating collective behavior of human

being [23, 24]. And it has been applied to a lot of real problems including system identification and electromagnetic antenna design [25–33]. Recently BSO-based FS algorithms have received much attention. Zhang et al. applied BSO in FS problems for the first time and proposed a continuous BSO-based FS algorithm (CBSO) [34]. Furthermore, they have also developed an improved discrete BSO [35], where two new idea clustering and new idea updating mechanisms were proposed to improve the performance of BSO. Liang et al. proposed a hybrid FS algorithm by combining ant colony optimization and BSO [36]. Combining the Fuzzy Min-Max neural network and BSO to undertake feature selection and classification problems, Pourpanan et al. developed also a hybrid BSO-based FS method [37]. Furthermore, they also presented a novel improved hybrid BSO-based FS method [38] by combining a fuzzy ARTMAP model with BSO. Papa et al. introduced an improved binary BSO-based FS algorithm, where a real-valued solution is mapped onto a Boolean hyper cube by using different transfer functions [39]. These methods above all enhance the capability of BSO on solving FS problems.

However, as the feature space increases exponentially, the search capability of existing BSO-based methods is inevitably reduced because of the lack of effective space reduction strategies.

For high-dimensional data, this paper develops a new evolutionary feature selection algorithm, called the clustering-guided integer BSO algorithm (IBSO-C). IBSO-C works in the two phases. In the first phase, an importance-guided feature clustering method is developed to group all features into multiple clusters according to their redundancy. Following that, the second phase selects the most representative feature from each feature cluster by employing an improved integer BSO, and all representative features form the final feature subset. Applying IBSO-C in several high-dimensional FS problems, experimental results show its superiority and effectiveness over some state-of-the-art methods, including one filter method and three evolutionary wrapper feature selection methods.

The main contributions of this paper are as follows:

- (1) The paper proposes a new two-phase hybrid evolutionary FS framework, which effectively combines the capability of fast dimensionality reduction of clustering-based method with the global search ability of evolutionary algorithm. Since the number of feature clusters is far smaller than the size of original features, the second phase can find an optimal feature subset fast.
- (2) roposes a new feature clustering method, called the importance-guided feature clustering method. By effectively fusing feature importance and feature correlation, the proposed method can group all features into multiple clusters according to their redundancy at relatively small computational cost.
- (3) The paper proposes an improved integer BSO (IBSO) for feature selection problems. Several new strategies, including the integer encoding strategy and the time-varying integer update strategy, improve the search performance of IBSO.

The remaining contents are organized as follows: Section 2 shows basic conceptions. Next, the proposed BSO-based FS algorithm is introduced in Section 3. Following that, Section 4 provides experimental analyses. Section 5 concludes this paper.

2. Related Work

2.1. Feature Selection Problem. Considering a dataset with D features and H instances, the objective of feature selection is to select d features ($d < D$) from the original feature set, $Fset$, so that the classification accuracy, AC, become better. Using a binary string, X , to represent a feature subset, we have

$$X = (x_1, x_2, \dots, x_D), \quad x_j \in \{0, 1\}, \quad (1)$$

where $x_j = 1$ indicates that the j -th feature is selected into the feature subset X ; otherwise, it is not selected. A feature selection problem is as follows:

$$\begin{aligned} \max & AC(X) \\ \text{s.t. } & X = (x_1, x_2, \dots, x_D). \end{aligned} \quad (2)$$

To deal with FS problems, existing methods can be divided into three categories [8]: filter, wrapper, and embedded. The filter firstly calculates the importance degrees of all features by a specified measure, such as information gain, distance measure, and dependency measure. Next, all features are ordered with their importance degrees. This kind of approach has less computational cost, but its classification accuracy often is worse than the other two kinds of methods. The wrapper utilizes a learning algorithm to evaluate feature subsets and uses some search methods to find good feature subsets. Because of the need of repeatedly evaluating new feature subsets (or solutions) by a classifier, this kind of methods has high computational cost. But, its classification accuracy often is better than the filter. The embedded method automatically carries out feature selection when training classifier. Since those selected features are closely related to such the used learning algorithm, the robustness of embedded method is poor to the change of algorithms. Since the proposed FS algorithm also uses a classifier to evaluate new solutions (i.e., feature subset) and utilizes BSO to search feature subsets, it belongs to the wrapper. The purpose of this paper is to study a new BSO-based feature selection algorithm for high-dimensional data.

2.2. Brain Storm Optimization Algorithm. In BSO, an idea (i.e., an individual) represents a potential solution of optimized problem. It generates continually new ideas by repeatedly implementing the following three phases, i.e., individual clustering, individual update, and elite selection. Note that an idea is an individual in other evolutionary optimization algorithms.

In the phase of individual clustering, individuals are first grouped into multiple clusters. Here, the most commonly used clustering method is the K-means clustering. Next, the phase of individual update uses those cluster centers or ideas from one/two clusters to generate new solutions. Here, two probability values, i.e., P_{gen} and P_{cluster} , are used to determine which cluster center or ideas to be used. After selecting the two solutions, a new individual is generated by a crossover-like strategy as follows:

$$\begin{cases} X_{\text{new}} = X'_{\text{new}} + \xi \times \text{Rand}_2, \\ X'_{\text{new}} = \text{Rand}_1 \times X_i + (1 - \text{Rand}_1) \times X_j, \\ \xi = \text{logsig}\left(\frac{0.5 \times T - t}{c}\right) \times \text{Rand}_2, \end{cases} \quad (3)$$

where X_i and X_j are two ideas/individuals or cluster centers from the two clusters; Rand is a random number within $[0, 1]$. The disturbance factor, ξ , is utilized to enhance the diversity of new ideas; t is the current iteration time and T is the maximal iteration time.

Finally, the elite selection compares those new ideas with corresponding old ones. And the better ones are saved and recorded as new ideas.

2.3. Symmetric Uncertainty. Compared with mutual information, symmetric uncertainty (SU) can correct the FS bias by normalizing mutual information values, so that it can fairly analyze the correlation between features. Now, it has been successfully used in FS problems [7, 40].

Taking two random variables, X and Y , as example, their SU value is calculated by

$$SU(X, Y) = 2 \frac{IG(X|Y)}{H(X) + H(Y)}, \quad (4)$$

where $H(X|Y) = -\sum_{y \in Y} P(y) \sum_{x \in X} P(x|y) \log_2 P(x|y)$ is the conditional entropy, which is used to evaluate the uncertainty of X when Y is given. $H(X) = -\sum_{x \in X} P(x) \log_2 P(x)$ is the entropy of X . $IG(X|Y) = H(X) - H(X|Y)$ is the information gain, which is used to evaluate the decrease degree of the uncertainty of X when Y is known.

3. The Proposed IBSO-C Algorithm

3.1. Framework. To deal with the problem of “curse of dimensionality,” the paper proposes a two-phase evolutionary FS algorithm, called clustering-guided integer BSO algorithm (IBSO-C). Figure 1 shows its framework. The framework includes the following phases: clustering features and selecting representative features. Firstly, the first phase groups all features into multiple clusters according to their similarity, by using the proposed importance-guided feature clustering method. After that, the second phase selects representative features from these feature clusters by employing an improved integer BSO, so generating the final feature subset. The main contributions of this paper have been marked with red dotted lines in Figure 1.

3.2. Importance-Guided Feature Clustering Strategy. A good clustering method should be able to group similar features into the same cluster at a less computational cost. The frequently used K-means can divide data more accurately, but it usually has a high computational cost. For the above reason, we propose a new feature clustering method, i.e., the importance-guided feature clustering (IFC).

Algorithm 1 shows the implementing steps of IFC. Firstly, the SU measure is used to evaluate the importance of each feature in F_{set} (step 1). The greater the SU value between a feature and class labels is, the more important the feature is. Secondly, all the features in F_{set} are sorted in the decreasing order of SU (step 2), denoted the sorting result by F_{sorted} . After that, the following steps (steps 4–12) are executed circularly until all features are classified into corresponding clusters: (1) the first feature in F_{sorted} is set to be a new cluster center, $Center_i$, and the i -th feature cluster is initialized as $Fcluster_i = \{Center_i\}$. (2) All the features in F_{sorted} are checked against the new cluster center. If the correlation between a feature and the cluster center is more than a threshold η , then the feature is put into the feature

FIGURE 1: Framework of the proposed IBSO-C.

cluster, $Fcluster_i$. Repeating the above method, we can get the new feature cluster, $Fcluster_i$. (3) After that, we reset $F_{sorted} = F_{sorted}/Fcluster_i$. If F_{sorted} is not empty, return to step 5; otherwise, stop the clustering method and output all the feature clusters. Note that, we use also US to calculate the correlation between feature and cluster center.

Compared with existing clustering algorithms, IFC has the following advantages: (1) IFC does not need to set the clustering number in advance. Compared with the clustering number, it is easier to set the correlation threshold η . (2) IFC has a lower computation complexity. Considering the worst case, running the 11-th line of Algorithm 1 every time can delete two features from F_{sorted} , and the times of running the SU measure to calculate the correlation between features and cluster centers are $D(D-1)/4$, where D is the size of feature. Moreover, evaluating the importance degrees of all features needs running the SU measure D times. Therefore, the times of running the SU measure by IFC are $(D^2 + 3D)/4$. In most of the existing clustering methods, determining the redundancy between all features usually needs running the SU measure $(D-1)!$ times.

3.3. Selecting Representative Features by an Improved Integer BSO. In this section, an improved integer BSO is proposed to select representative features from those feature clusters, thus generating the final feature subset. Firstly, we give the encoding method of idea/individual and the fitness evaluation strategy of idea.

3.3.1. Encoding and Fitness Evaluation Strategies. By implementing the method in Section 3.2, all features can be divided into Z feature clusters. $Fcluster_i$, $i = 1, 2, \dots, Z$. The

Input: The original feature set, $Fset$;
Output: The feature clustering result, $\{Fcluster_1, \dots, Fcluster_K\}$

- (1) Use the SU measure to evaluate the importance of each feature in $Fset$;
- (2) Sort all features in $Fset$ in the decreasing order of SU values, denoted the sorted result by $Fsorted$;
- (3) Set $i = 1$;
- (4) **While** $|Fsorted| > 0\%|Fsorted|$ is the size of $Fsorted$
- (5) Set the first feature in $Fsorted$ to be the i -th cluster center, $Center_i$;
- (6) Initialize $Fcluster_i = \{Center_i\}$;
- (7) **For** $j = 1: |Fsorted|$
- (8) Calculate the correlation between $Center_i$ and the j -th feature in $Fsorted$;
- (9) If the correlation $> \eta$, then save the j -th feature into $Fcluster_i$;
- (10) **Endfor**
- (11) Reset $Fsorted = Fsorted/Fcluster_i$, and $i = i + 1$.
- (12) **Endwhile**

ALGORITHM 1: The proposed IFC.

goal of this section is to produce a good feature subset, X , by selecting a representative feature from each feature cluster, so that the classification performance gets the largest. Taking an integer vector to construct a solution, the optimization model is as follows:

$$\begin{aligned} \max F(X) \\ \text{s.t. } X = (x_1, x_2, \dots, x_z) \\ x_i \in \{1, \dots, \infty, |Fcluster_i|\}, \quad i = 1, 2, \dots, Z, \end{aligned} \quad (5)$$

where $x_i = s$ ($0 \leq s \leq |Fcluster_i|$) indicates that the s -th feature in the i -th feature cluster is selected into the feature subset, X . Following the above model, we directly use the integer vector, X , to represent an individual in the population.

This paper adopts the leave-one-out cross-validation (LOOCV) of k-NN to calculate the fitness of an idea in BSO. Due to easy implementation, the one nearest neighbor (1-NN) method is used as the classifier in the following experiment. As we know, the k-NN has used by many FS methods [7, 8, 10]. In the LOOCV with 1-NN, a single instance from the original dataset is selected as a testing sample, and the remaining ones are used as training samples. Then the 1-NN predicts which class this instance belongs to. The above process is repeated so that each instance in the original dataset is used once as the testing sample. Based on this, the classification accuracy of an idea X_i is as follows:

$$AC(X_i) = \frac{FP_size}{All_size}, \quad (6)$$

where All_size is the size of all instances in the original dataset and FP_size is the number of correctly predicted samples by K-NN. In the proposed method, the AC value of an idea is its fitness.

3.3.2. Time-Varying Integer Update of Idea. It can be seen by analyzing (3) that since two weights are randomly assigned to the two normal ideas or cluster centers selected, X_i and

X_j , the new generated idea may swing back and forth between the two. If X_i and X_j are always far away in the iteration process of BSO, such ideas corresponding the two clusters will be difficult to converge, reducing the convergence performance of BSO.

Moreover, since traditional BSO is proposed for continuous optimization problems, we must develop a new integer update rule for solving the optimization model described in (5).

To overcome the above problems, this section proposes a time-varying integer update strategy of idea (TVIU). In this strategy, the best one among the two normal ideas or cluster centers will get a large learning weight. And this learning weight will increase as the number of iterations increases. The new update rule is as follows:

$$\left\{ \begin{aligned} X_{new} &= [X_{new}' + \xi \times \text{Rand}_2], \\ X_{new}' &= w_1 \times X_i + w_2 \times X_j, \\ w_1 &= \frac{\text{Rand}_3 + \Delta w_1}{1 + t/T}, \\ w_2 &= \frac{1 - \text{Rand}_3 + \Delta w_2}{1 + t/T}, \\ \Delta w_1 &= \frac{t}{T} \times \frac{AC(X_i)}{AC(X_i) + AC(X_j)}, \\ \Delta w_2 &= \frac{t}{T} \times \frac{AC(X_j)}{AC(X_i) + AC(X_j)}, \end{aligned} \right. \quad (7)$$

where rand_2 and rand_3 are two random numbers within $[0, 1]$, $[\cdot]$ is the roundup function, w_1 and w_2 are two weights, which are used to control the learning proportions of the new solution from two ideas or cluster centers, T is the maximum iteration times of the algorithm, t is the current iteration times, and $AC(X_i)$ and $AC(X_j)$ are the

AC values of the two normal ideas or cluster centers, respectively.

From (7) we can see that (1) the bigger the AC value of a normal idea or cluster center, the higher the weight (w) of this idea or center, so that the new solution X_{new} can learn more knowledge from this idea or center. This can improve the quality of new solution to a certain extent. (2) As the iteration times increase, the influence of the AC value of a normal idea or cluster center on the weight gets higher. This indicates that the learning degree from the best idea is higher and higher with the increase of iteration times. Furthermore, this can speed up the convergence of the population in the later stage of the algorithm.

3.3.3. Disturbance Operator. In the proposed IBSO, a new disturbance operator is unitized to improve the new ideas' diversity. Checking each element in the idea X_i in turn, if the random number $\text{rand}' \in [0, 1]$ is smaller than the probability p_m , the element will be reinitialized within its search space. In this paper, we set $p_m = 1/D$, where D is the number of features.

3.4. Implement Steps of IBSO-C. Like traditional BSO algorithms [41], the proposed IBSO-C includes still three main steps: clustering ideas, updating ideas, and selecting elite ideas. In the IBSO-C, an idea represents a solution of the optimized problem. The feature clustering strategy proposed in Section 3.2 is used to cluster features, and the improved integer BSO proposed in Section 3.3 is used to update the positions of ideas.

In the first step (i.e., clustering ideas), all ideas are grouped into several clusters. In traditional BSO, the K-means clustering is the commonly used method. Due to the need to repeatedly cluster all instances, the K-means has still the disadvantage of high-computational cost. Focusing on it, Cao et al. [42] introduced random grouping to minimize the clustering cost. In this method, after the population is grouped into randomly M clusters, the fittest idea in each cluster is selected as its center. Compared with the K-means, this method significantly reduces the computational cost of population clustering.

In the second step (i.e., updating ideas), new ideas are generated based on two cluster centers or normal ideas in two clusters. Here the time-varying integer update strategy in Section 3.3 is used to generate new solutions. And the proposed disturbance operator is unitized to improve the new ideas' diversity.

In the third step, the elite selection will be implemented. For the i -th idea in the population, if the classification accuracy of the new idea X_i^{new} is better than that of the old one, X_i , then replace X_i by X_i^{new} , i.e., $X_i \leftarrow X_i^{\text{new}}$. If X_i^{new} and X_i have the same classification accuracy, but the feature number in X_i^{new} is fewer than that of X_i , then $X_i \leftarrow X_i^{\text{new}}$.

Moreover, Algorithm 2 shows the pseudocode of the proposed IBSO-C. Note that, there are two clustering results in Algorithm 1, i.e., lines 2 and 6. In line 2, the method proposed in Section 3.2 is used to cluster features. Its input is the data to be processed, and the output is feature clusters.

Line 6 is to cluster the N ideas or individuals, and its input is the population of BCO.

4. Experiments and Analyses

This section verifies the effectiveness of IBSO-C. Firstly, we discuss the effectiveness of the proposed two key operators on the performance of IBSO-C, i.e., the proposed clustering strategy and the time-varying integer update strategy. Secondly, IBSO-C is compared with four existing FS algorithms.

4.1. Experimental Preparation. Eight real datasets are used to verify the performance of IBSO-C. Table 1 shows their basic information. These datasets have been used in many studies [5, 7, 10], which can be downloaded from <http://www.ics.uci.edu/mllearn/MLRepository.html> and <http://gems-system.org/>.

Four representative feature selection algorithms were used, including the ReliefF algorithm (ReliefF) in [43], the binary PSO algorithm (BPSO) in [44], the binary BSO-based algorithm (BBSO) in [35], and the self-adaptive PSO algorithm (SaPSO) in [12]. For fair comparison, all population-based algorithms use the same swarm/population size (50) and the same maximal iterations (1000). For other parameters, their values are set followed by their original literatures, as shown in Table 2.

Three performance indexes are used to evaluate the quality of an algorithm, i.e., the classification accuracy (AC), the number of selected features (FN), and the running time (Time). This paper employs the 10-fold cross-validation method to demonstrate the effectiveness of an algorithm. The 9 parts are taken as training data in turn, the remaining part is taken as test data, and the average value of 10 times is used as the final result. All experiments in this paper are carried out on Intel (R) core (TM) i7-8700 CPU, 3.2 GHz, and 16.00 GB RAM.

4.2. Analysis on the Proposed Clustering Strategy. The proposed clustering strategy plays a key role in improving the performance of IBSO-C. We analyze the effectiveness of the strategy in this section. Here, the conventional BSO without feature clustering [34] (CBSO) is selected as a comparison method. IBSO-C and CBSO use the same parameters shown in Table 2.

Table 3 show the values of AC, FN, and the running time obtained by IBSO-C and CBSO. We can see that, (1) with the help of the feature clustering strategy proposed in Section 3.2, IBSO-C obtained the best average AC values for all the eight datasets, which are significantly higher than those of CBSO. (2) More importantly, compared with CBSO, IBSO-C takes only very few features to get such good AC values, as shown in their average FN values. Taking the dataset CNS as example, IBSO-C only used less than 90 features to get the classification accuracy, 86.04%. (3) Due to only using very few features, the running time of IBSO-C is also significantly less than that of CBSO. Taking the dataset CNS as example, IBSO-C only costs 1.5936 minutes to get a good solution, while the running time of CBSO is more than two hours.

Input: The data set to be solved;
Output: The optimal feature subset;

- (1) Set related parameters, including the population size, N , the maximal iteration times, T , P_{cluster} and $P_{t\text{-cluster}}$, and so on; $t = 0$;
- (2) Cluster all the features into K clusters by using the method in Section 3.2.
- (3) Randomly generate N integer ideas or individuals.
- (4) Evaluate the fitness of each idea by equation (6);
- (5) **While** $t < T$
- (6) Grouping all the N ideas into M clusters by the method in [42];
- (7) Select the best idea from each cluster as the cluster center; % the phase of updating ideas %
- (8) **For** $i = 1: N$ % from the first idea to the last one
- (9) **If** a random number $\text{rand}() < P_{\text{gen}}$, **then**
- (10) Randomly select a cluster and determine its cluster center;
- (11) **If** a random number $\text{rand}() < P_{\text{cluster}}$,
- (12) Select the cluster center;
- (13) Generate a new idea by the equation (7);
- (14) Implement the proposed disturbance operator;
- (15) **Else**
- (16) Randomly select a normal idea from this cluster
- (17) Generate a new idea by the equation (7);
- (18) Implement the proposed disturbance operator;
- (19) **End if**
- (20) **Else**
- (21) Randomly select two clusters;
- (22) **If** a random number $\text{rand}() < P_{\text{cluster}}$, **then**
- (23) Select two cluster centers;
- (24) Generate a new idea by the equation (7);
- (25) Implement the proposed disturbance operator;
- (26) **Else**
- (27) Randomly select two normal ideas from the two clusters respectively;
- (28) Generate a new idea by the equation (7);
- (29) Implement the proposed disturbance operator;
- (30) **End if**
- (31) **End if** % the phase of selecting elite ideas %
- (32) Evaluate the new idea and update correspond old idea by Section 3.3.
- (33) **End for**
- (34) **End while**

ALGORITHM 2: The proposed IBSO-C.

TABLE 1: Basic information for the test datasets.

Datasets	# of features	# of samples	# of classes
SCADI	205	70	7
GFE01	301	1062	2
Yale_64	1024	165	15
Colon	2000	62	2
SRBCT	2308	83	4
WrapAR10P	2400	130	10
Leukemia_small	3571	72	2
DBWorld	4703	64	2
CNS	7129	60	2
Lung	12600	203	2

4.3. Analysis on the Time-Varying Integer Update Strategy.
The time-varying integer update strategy (TVIU) proposed in Section 3.3 also plays a key role in improving the performance of IBSO-C. This subsection will analyze its effectiveness. Here, the integer update version of (3) is selected as the comparison method, as follows:

$$\begin{cases} X_{\text{new}} = [X'_{\text{new}} + \xi \times \text{Rand}_2], \\ X'_{\text{new}} = \text{Rand}_1 \times X_i + (1 - \text{Rand}_1) \times X_j. \end{cases} \quad (8)$$

For convenience, the IBSO-C with (8) is called IBSO-S. Both IBSO-C and IBSO-S use the same parameters shown in Table 2.

TABLE 2: Parameter settings.

Algorithms	Parameter values
ReliefF [43]	The number of nearest neighbors, $k=10$
BPSO [44]	The acceleration coefficients, $c1=c2=0.5$, the inertia weight $w=1$
SaPSO [12]	The selection probability of the strategy $p_j=0.2$, the empirically LP=10, the threshold $\theta=0.6$
BBSO [35]	$P_{\text{cluster}}=0.5$, $P_{\text{gen}}=0.7$, $M=3$
IBSO-C	$P_{\text{cluster}}=0.5$, $P_{\text{gen}}=0.7$, $M=3$

TABLE 3: AC values obtained by IBSO-C and CBSO.

Datasets	AC		FN		Running time	
	CBSO	IBSO-C	CBSO	IBSO-C	CBSO	IBSO-C
GFE01	93.02/0.2198	92.46/1.1901	151.50/3.7621	14.0/0.089	113.89/8.9052	21.4809/3.4756
Yale_64	75.75/0.6004	77.21/0.8129	485.37/8.5049	31.80/0.8367	126.21/5.8376	10.5773/0.5614
Colon	84.94/0.9300	91.53/1.2390	1002.30/8.08	35.87/1.885	95.80/5.8913	0.5667/0.0361
SRBCT	92.77/0.6820	95.78/1.5770	1145.39/14.15	44.15/1.8077	79.55/4.8800	1.1468/0.2899
WrapAR10P	67.43/0.4405	83.37/3.0751	1175.67/6.658	48.625/0.51755	116.07/8.5365	3.9642/0.4815
leukemia_small	98.61/0.5223	100.00/0	1798.56/12.50	59.00/1.069	97.84/7.0991	1.8600/0.3079
DBWorld	93.75/0.8412	94.92/1.3850	2355.86/25.80	49.00/3.0237	102.13/9.8508	1.4812/0.2692
CNS	73.88/1.9208	86.04/4.1702	3577.76/43.69	84.25/0.88641	138.78/13.3634	1.5936/0.0139

FIGURE 2: AC values of IBSO-C and IBSO-S.

Figures 2 and 3 show the AC and FN values obtained by IBSO-C and IBSO-S, respectively. We can see that, (1) for all the datasets except leukemia_small, IBSO-C all obtained the biggest classification accuracy values compared with IBSO-S, with the help of the proposed TVIU strategy. Taking the dataset CNS as example, the AC value of IBSO-C is 5 percentage points higher than that of IBSO-S. (2) In terms of the number of selected features (NF), both IBSO-C and IBSO-S obtained very similar values.

4.4. Comparison Analyses. The proposed IBSO-C algorithm is compared with the four existing algorithms in terms of AC, FN, and the running time. Table 4 lists the average AC values obtained by the five FS algorithms with KNN, and Table 5 shows the average FN values obtained by the five FS algorithms. In addition, we employ the Mann-Whitney U test to investigate whether there is a significant difference between IBSO-C and another algorithm. Here, “+” indicates that IBSO-C is obviously superior to the comparison algorithm, “=” indicates that there is no significant difference between them, and “-” indicates that IBSO-C is obviously inferior to the comparison algorithm.

The following can be seen from the two tables: (1) for 6 out of the 8 datasets, IBSO-C obtained the biggest average AC values; for 4 out of the 8 datasets, i.e., Colon, WrapAR10P, DBWorld, and CNS, the AC values of IBSO-C are significantly superior to that of all the four comparison algorithms. (2) For the dataset GFE01, IBSO-C obtained the third best AV value, while SaPSO had the best AV value. However, the number of features selected by IBSO-C is significantly less than that of SaPSO, where the FN values of IBSO-C and SaPSO are 14.0 and 73.2, respectively. (3) Like the dataset GFE01, SaPSO obtained the best AV value on the dataset SRBCT, but its FN value is significantly bigger than that of IBSO-C. (4) For all the 8 datasets, IBSO-C obtained the smallest FN values compared with the four comparison algorithms.

4.5. Running Time. Table 6 shows the average running time of IBSO-C and the three evolutionary FS algorithms. It reports that the running time of IBSO-C is significantly less than that of BPSO, SaPSO, and BBSO. For all the 8 datasets, the average running time of IBSO-C is 5.33 minutes, while the average running time of BPSO, SaPSO, and BBSO is

FIGURE 3: FN values of IBSO-C and IBSO-S.

TABLE 4: AC values obtained by IBSO-C and the four comparison algorithms.

Datasets	Full set	ReliefF	BPSO	SaPSO	BBSO	IBSO-C
GFE01	87.47 (+)	88.40/0.6874 (+)	92.43/0.2287 (=)	93.97/0.1383 (-)	93.67/0.3680 (-)	92.77/0.2890
Yale_64	64.86 (+)	68.78/0.4922 (+)	73.98/0.7333 (+)	76.68/0.5462 (=)	76.82/0.58 (=)	76.97/0.4098
Colon	72.37 (+)	83.9/1.2789 (+)	89.83/0.3609 (+)	90.34/0.7689 (+)	89.95/0.93 (+)	95.48/1.7664
SRBCT	83.14 (+)	96.71/1.1386 (=)	96.65/0.3035 (=)	100.00/0 (-)	94.80/0.932C0 (+)	96.98/0.6721
WrapAR10P	52.69 (+)	76.19/1.3078 (+)	60.59/0.4882 (+)	80.71/2.7222 (+)	78.46/2.2646 (+)	83.84/1.8540
leukemia_small	95.44 (+)	100.00/0 (=)	100.00/0 (=)	100.00/0 (=)	99.3050/0.4654 (=)	100.00/0
DBWorld	56.67 (+)	58.60/1.0255 (+)	88.83/2.0069 (+)	90.81/0.6859 (+)	93.75/0.78 (+)	96.25/0.008
CNS	55.29 (+)	65.07/4.2603 (+)	76.62/1.7303 (+)	78.32/0.9889 (+)	81.67/1.76 (+)	91.667/26.05

TABLE 5: FN values obtained by IBSO-C and the four comparison algorithms.

Datasets	ReliefF	BPSO	SaPSO	BBSO	IBSO-C
GFE01	121.0/0 (+)	142.2/9.4181 (+)	73.2/9.6799 (+)	120.33/26.12 (+)	14.28/0.0921
Yale_64	410.0/0 (+)	500.2/13.7004 (+)	277.6/9.7108 (+)	310.78/42.23 (+)	32.082/0.8204
Colon	400.0/0 (+)	1005.6/21.4313 (+)	687.6/76.2712 (+)	954.67/27.85 (+)	33.82/2.389
SRBCT	462.0/0 (+)	1145.2/23.5308 (+)	293.6/36.1566 (+)	1109.10/23.80 (+)	44.23/0.7612
WrapAR10P	960.0/0 (+)	1213.4/26.1017 (+)	552.4/117.1593 (+)	231.75/41.183 (+)	48.16/0.9809
leukemia_small	1429.0/0 (+)	1700.0/69.2965 (+)	366.6/54.3765 (+)	1639.0/307.80 (+)	59.07/0.5420
DBWorld	1881.0/0 (+)	23166.6/20.4768 (+)	1368.0/100.1574 (+)	2186.80/20.25 (+)	46.89/3.1452
CNS	2852.0/0 (+)	3560.0/14.6116 (+)	2230.6/115.0535(+)	1601.7/367.35 (+)	84.89/0.6990

TABLE 6: Running time consumed by the four evolutionary FS algorithms (unit: min).

Datasets	BPSO	SaPSO	BBSO	IBSO-C
GFE01	125.55/7.74	127.88/9.35	193.74/7.41	21.4809/3.47
Yale_64	115.96/5.14	115.87/2.38	70.49/12.40	10.5773/0.56
Colon	107.78/4.05	106.37/7.45	18.897/3.07	0.5667/0.03
SRBCT	122.35/6.87	118.01/2.45	25.36/1.70	1.1468/0.28
WrapAR10P	204.94/23.67	127.81/11.27	41.05/12.30	3.9642/0.48
leukemia_small	125.01/7.46	127.09/9.79	29.89/2.78	1.8600/0.30
DBWorld	169.58/41.66	127.72/14.69	52.49/5.41	1.4812/0.26
CNS	192.67/57.79	168.29/19.55	54.02/4.43	1.5936/0.02

145.48, 127.39, and 60.742 minutes, respectively. Overall, IBSO-C is a highly competitive FS algorithm, which can obtain relatively good classification accuracy at less computational cost.

5. Conclusions

This paper studied a new two-phase evolutionary feature selection algorithm, called the clustering-guided integer BSO

algorithm (IBSO-C), for high-dimensional data. In the IBSO-C, the proposed feature clustering strategy in the first phase obviously reduced the search space of the integer BSO in the second phase. Since the number of feature clusters is far smaller than the number of original features, IBSO-C can find an optimal feature subset fast. Moreover, the proposed importance-guided feature clustering method can effectively group features at a relatively small computational cost. The proposed new encoding strategy and the proposed time-varying integer update strategy have improved the search performance of IBSO-C. The proposed IBSO-C was compared with four existing FS algorithms, i.e., relief, BPSO SaPSO, and BBPSO on several datasets. The experimental results showed that IBSO-C is a highly competitive FS algorithm, which can obtain relatively good classification accuracy at less computational cost.

A more sophisticated feature clustering method that does not need to set any threshold or parameters manually will be one of our future research directions. In addition, applying multi- or many-objective evolutionary algorithms to cost-sensitive feature selection problems will be another research direction in the future.

Data Availability

Some or all data, models, or codes generated or used during the study are available from the corresponding upon by request.

Conflicts of Interest

The authors declare no conflicts of interest.

Acknowledgments

This work was supported by the Scientific Innovation 2030 Major Project for New Generation of AI, Ministry of Science and Technology of the People's Republic of China (No. 2020AAA0107300).

References

- [1] K. Mistry, L. Zhang, S. C. Neoh, C. P. Lim, and B. Fielding, "A micro-GA embedded PSO feature selection approach to intelligent facial emotion recognition," *IEEE Transactions on Cybernetics*, vol. 47, no. 6, pp. 1496–1509, 2017.
- [2] M. Liang and X. Hu, "Feature selection in supervised saliency prediction," *IEEE Transactions on Cybernetics*, vol. 45, no. 5, pp. 900–912, 2015.
- [3] J. Xu, B. Tang, H. He, and H. Man, "Semisupervised feature selection based on relevance and redundancy criteria," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 9, pp. 1974–1984, 2017.
- [4] Y. Zhang, Q. Wang, D.-W. Gong, and X.-F. Song, "Non-negative laplacian embedding guided subspace learning for unsupervised feature selection," *Pattern Recognition*, vol. 93, pp. 337–352, 2019.
- [5] D. Gong, J. Sun, and Z. Miao, "A set-based genetic algorithm for interval many-objective optimization problems," *IEEE Transactions on Evolutionary Computation*, vol. 22, no. 1, pp. 47–60, 2018.
- [6] M. Paniri, M. B. Dowlatabadi, and H. Nezamabadi-Pour, "MLACO: a multi-label feature selection algorithm based on ant colony optimization," *Knowledge-Based Systems*, vol. 192, Article ID 105285, 2020.
- [7] X.-F. Song, Y. Zhang, Y.-N. Guo, X.-Y. Sun, and Y.-L. Wang, "Variable-size cooperative coevolutionary particle swarm optimization for feature selection on high-dimensional data," *IEEE Transactions on Evolutionary Computation*, vol. 24, no. 5, pp. 882–895, 2020.
- [8] B. Xue, M. Zhang, and W. N. Browne, "Particle swarm optimization for feature selection in classification: a multi-objective approach," *IEEE Transactions on Cybernetics*, vol. 43, no. 6, pp. 1656–1671, 2013.
- [9] Y. Xue, T. Tang, W. Pang, and A. X. Liu, "Self-adaptive parameter and strategy based particle swarm optimization for large-scale feature selection problems with multiple classifiers," *Applied Soft Computing*, vol. 88, Article ID 106031, 2020.
- [10] Y. Zhang, D.-W. Gong, and J. Cheng, "Multi-objective particle swarm optimization approach for cost-based feature selection in classification," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 14, no. 1, pp. 64–75, 2017.
- [11] B. Tran, B. Xue, and M. Zhang, "A new representation in PSO for discretization-based feature selection," *IEEE Transactions on Cybernetics*, vol. 48, no. 6, pp. 1733–1746, 2018.
- [12] Y. Xue, B. Xue, and M. Zhang, "Self-adaptive particle swarm optimization for large-scale feature selection in classification," *ACM Transactions on Knowledge Discovery from Data*, vol. 13, no. 5, pp. 1–27, 2019.
- [13] U. Mlakar, I. Fister, J. Brest, and B. Potočnik, "Multi-objective differential evolution for feature selection in facial expression recognition systems," *Expert Systems with Applications*, vol. 89, pp. 129–137, 2017.
- [14] Y. Zhang, D.-W. Gong, X.-Z. Gao, T. Tian, and X.-Y. Sun, "Binary differential evolution with self-learning for multi-objective feature selection," *Information Sciences*, vol. 507, pp. 67–85, 2020.
- [15] E. Hancer, "A new multi-objective differential evolution approach for simultaneous clustering and feature selection," *Engineering Applications of Artificial Intelligence*, vol. 87, Article ID 103307, 2020.
- [16] Y. Zhang, S. Cheng, Y. Shi, D.-W. Gong, and X. Zhao, "Cost-sensitive feature selection using two-archive multi-objective artificial bee colony algorithm," *Expert Systems with Applications*, vol. 137, pp. 46–58, 2019.
- [17] Z. Du, D. Han, and K.-C. Li, "Improving the performance of feature selection and data clustering with novel global search and elite-guided artificial bee colony algorithm," *The Journal of Supercomputing*, vol. 75, no. 8, pp. 5189–5226, 2019.
- [18] H. Rao, X. Shi, A. K. Rodrigue et al., "Feature selection based on artificial bee colony and gradient boosting decision tree," *Applied Soft Computing*, vol. 74, pp. 634–642, 2019.
- [19] Y. Zhang, X.-F. Song, and D.-W. Gong, "A return-cost-based binary firefly algorithm for feature selection," *Information Sciences*, vol. 418–419, pp. 561–574, 2017.
- [20] B. Selvakumar and K. Muneeswaran, "Firefly algorithm based feature selection for network intrusion detection," *Computers & Security*, vol. 81, pp. 148–155, 2019.
- [21] H. Faris, M. M. Mafarja, A. A. Heidari et al., "An efficient binary salp swarm algorithm with crossover scheme for feature selection problems," *Knowledge-Based Systems*, vol. 154, pp. 43–67, 2018.

- [22] M. Mafarja and S. Mirjalili, "Whale optimization approaches for wrapper feature selection," *Applied Soft Computing*, vol. 62, pp. 441–453, 2018.
- [23] Y. Shi, "Brain storm optimization algorithm," in *Proceedings of the 2nd International Conference on Swarm Intelligence: Lecture Notes in Computer Science*, vol. 6728, pp. 303–309, Chongqing, China, June 2011.
- [24] S. Cheng, Q. Qin, J. Chen, and Y. Shi, "Brain storm optimization algorithm: a review," *Artificial Intelligence Review*, vol. 46, no. 4, pp. 445–458, 2016.
- [25] Y. Yu, S. C. Gao, S. Cheng, Y. R. Wang, S. Y. Song, and F. Yuan, "A memetic brain storm optimization with chaotic local search," *Memetic Computing*, vol. 10, no. 4, pp. 353–367, 2019.
- [26] P. S. Pal, R. Kar, D. Mandal, and S. P. Ghoshal, "Parametric identification with performance assessment of wiener systems using brain storm optimization algorithm," *Circuits, Systems, and Signal Processing*, vol. 36, no. 8, pp. 3143–3181, 2017.
- [27] S. Cheng, M. Zhang, L. Ma, H. Lu, R. Wang, and Y. Shi, "Brain storm optimization algorithm for solving knowledge spillover problems," *Neural Computing and Applications*, 2021.
- [28] L. Ma, S. Cheng, and Y. Shi, "Enhancing learning efficiency of brain storm optimization via orthogonal learning design," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, pp. 1–20, 2020.
- [29] H. B. Duan and C. Li, "Quantum-behaved brain storm optimization approach to solving loney's solenoid problem," *IEEE Transactions on Magnetics*, vol. 51, no. 1, Article ID 7000307, 2015.
- [30] G. Xiong, D. Shi, J. Zhang, and Y. Zhang, "A binary coded brain storm optimization for fault section diagnosis of power systems," *Electric Power Systems Research*, vol. 163, no. A, pp. 441–451, 2018.
- [31] K. Arai, Y. Fukuyama, T. Iizaka, and T. Matsui, "Dependable multi-population improved brain storm optimization with differential evolution for optimal operational planning of energy plants," *Electrical Engineering in Japan*, vol. 207, no. 4, pp. 25–35, 2019.
- [32] X. Yan, Z. Zhu, Q. Wu, W. Gong, and L. Wang, "Elastic parameter inversion problem based on brain storm optimization algorithm," *Memetic Computing*, vol. 11, no. 2, pp. 143–153, 2019.
- [33] A. Alwaleed and R. Yahya, "Brain storm optimization for electromagnetic applications: continuous and discrete," *IEEE Transactions on Antennas and Propagation*, vol. 67, no. 4, pp. 2710–2722, 2019.
- [34] X.-T. Zhang, Y. Zhang, H.-R. Gao, and C.-L. He, "A wrapper feature selection algorithm based on brain storm optimization," in *Proceedings of the 13th International Conference on Bio-Inspired Computing: Theories and Applications*, vol. 952, pp. 308–315, Beijing, China, October 2018.
- [35] W.-Q. Zhang, Y. Zhang, and C. Peng, "Brain storm optimization for feature selection using new individual clustering and updating mechanism," *Applied Intelligence*, vol. 49, no. 12, pp. 4294–4302, 2019.
- [36] H. Liang, Z. Wang, and Y. Liu, "A new hybrid ant colony optimization based on brain storm optimization for feature selection," *IEICE—Transactions on Info and Systems*, vol. E102.D, no. 7, pp. 1396–1399, 2019.
- [37] F. Pourpanan, C. P. Lim, X. Z. Wang, C. J. Tan, M. Seera, and Y. H. Shi, "A hybrid model of fuzzy min-max and brain storm optimization for feature selection and data classification," *Neurocomputing*, vol. 333, pp. 440–451, 2019.
- [38] F. Pourpanah, Y. Shi, C. P. Lim, Q. Hao, and C. J. Tan, "Feature selection based on brain storm optimization for data classification," *Applied Soft Computing*, vol. 80, pp. 761–775, 2019.
- [39] J. P. Papa, G. H. Rosa, A. N. de Souza, and L. C. S. Afonso, "Feature selection through binary brain storm optimization," *Computers & Electrical Engineering*, vol. 72, pp. 468–481, 2018.
- [40] Q. Qinbao Song, J. Jingjie Ni, and G. Guangtao Wang, "A fast clustering-based feature subset selection algorithm for high-dimensional data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 1, pp. 1–14, 2013.
- [41] Z. Zhan, J. Zhang, Y. Shi, and H. Liu, "A modified brain storm optimization algorithm," in *Proceedings of IEEE Congress on Evolutionary Computation*, pp. 1969–1976, Brisbane, Australia, June 2012.
- [42] Z. Cao, Y. H. Shi, X. Rong, B. Liu, Z. Du, and B. Yang, "Random grouping brain storm optimization algorithm with a new dynamically changing step size," in *Proceedings International Conference on Swarm Intelligence*, pp. 387–364, Beijing, China, June 2015.
- [43] I. M. Robnik and I. Kononenko, "Theoretical and empirical analysis of relieff and rrelieff," *Machine Learning*, vol. 53, no. 1-2, pp. 23–69, 2003.
- [44] L.-Y. Chuang, C.-S. Yang, K.-C. Wu, and C.-H. Yang, "Gene selection and classification using taguchi chaotic binary particle swarm optimization," *Expert Systems with Applications*, vol. 38, no. 10, pp. 13367–13377, 2011.