Particle Swarm Optimisation with Genetic Operators for Feature Selection

Hoai Bach Nguyen, Bing Xue, Peter Andreae and Mengjie Zhang
School of Engineering and Computer Scicene
Victoria University of Wellington, PO Box 600, Wellington 6140, New Zealand
Email: {Hoai.Bach.Nguyen, Bing.Xue, Peter.Andreae, Mengjie.Zhang}@ecs.vuw.ac.nz

***Abstract*—Feature selection is an important task in machine learning, which aims to reduce the dataset dimensionality while at least maintaining the classification performance. Particle Swarm Optimisation (PSO) has been widely applied to feature selection because of its effectiveness and efficiency. However, since feature selection is a challenging task with a complex search space, PSO easily gets stuck at local optima. This paper aims to improve the PSO's searching ability by applying genetic operators such as crossover and mutation to assist the swarm to explore the search space better. The proposed genetic operators are specifically designed for feature selection, which not only improve the quality of current feature subsets but also make the search smoother. The proposed algorithm, called CMPSO, is tested and compared with three recent PSO based feature selection algorithms. Experimental results on eight datasets show that CMPSO can adapt with different numbers of features to evolve small feature subsets, which achieve similar or better classification performance than using all features and the three PSO based algorithms. The analysis on evolutionary processes shows that genetic operators assist CMPSO to evolve better solutions than the original PSO.**

I. Introduction

Classification is one of the most important tasks in machine learning. In a classification problem, a classification algorithm is trained on a number of training instances to predict the class labels of unseen instances, which are known as testing instances. The training process helps the classification algorithm to discover the relationship between the class and the instances' properties, which are called *features*. Therefore, the quantity and quality of features have direct effect on the classification performance. If there are too many features, it might be difficult to build a good classifier due to the "curse of dimensionality" [1]. In a large feature set, there might be some redundant or irrelevant features, which not only do not provide any additional information but also blur useful information from relevant features. The involvement of these features usually causes a longer training time and deteriorates the classification accuracy. In order to overcome the issues of high dimensionality, feature selection is proposed to reduce the number of features and improve the quality of the feature set by removing irrelevant and redundant features.

However, feature selection is not an easy task because of two main reasons. Firstly, given n original features, the total number of possible feature subsets is 2^n, which increases exponentially with respect to the total number of features. In addition, the complex interaction between features is another challenging issue of feature selection. Specifically, two relevant features might become redundant if they provide similar information about the class. Meanwhile, two weakly relevant features, which contain little information about the class, can become significantly useful when they are selected together. Two or more features, which cooperate with each others to improve the overall classification performance, are known as complementary features [2]. A feature selection algorithm usually consists of two main components to deal with the two difficult issues. The first one is a search mechanism, which is responsible for generating candidate feature subsets. The generated subsets are evaluated by a fitness function, which is the second component.

According to the evaluation criterion, feature selection can be divided into two categories: wrapper and filter approaches. In wrappers, a feature subset is evaluated by the classification performance of a specific classification algorithm. Meanwhile, filters assess the feature subset's quality in an independent way of any classification algorithm. Particularly, filter approaches rely on the characteristics of data such as distance, consistency, dependency, information, and correlation. Some examples of filter measures are Fisher score [3], Information Gain [4] or Relief [5]. In comparison with filters, wrappers are usually more expensive in terms of computation cost. However, filter approaches ignore the effects of the selected feature subsets on the performance of the classification algorithm [6]. In addition, wrappers consider the interaction between a set of features, which is usually difficult to capture in the filter measures [7]. Therefore, in this work, a wrapper approach is used to evaluate the candidate feature subsets.

Besides the fitness function, the search mechanism also plays an important role in a feature selection algorithm. Given a huge and complex search space, an exhaustive search, which considers all possible feature subsets, is not applicable. Sequential search methods reduce the computation cost by considering only one feature in each step, which ignores the interactions between multiple features and makes the searching process easily stuck at local optima. Recently, evolutionary computation (EC) techniques have been widely applied to feature selection because of their potential global search ability. Two most popular EC techniques applied to feature selection are particle swarm optimisation (PSO) and genetic algorithms (GAs) [2]. The main reason is their natural representations for feature selection, in which each bit corresponds to an original feature and indicates whether or not the feature is

978-1-5090-4601-0/17/$31.00 ©2017 IEEE

selected. Although it has been shown that PSO is more efficient than GAs in many problems [8], PSO usually suffers from a premature convergence problem. The underlying reason is that in global best PSO, particles move towards a single position. In addition, the fast rate of information flowing between particles results in creating a new swarm with less diversity. On the other hand, GAs are good at exploring the entire search space because of its pseudo-biological operators i.e. mutation and crossover. Therefore, integrating the genetic operators into PSO is expected to avoid premature convergence on problems with complex search spaces like feature selection. However, since PSO's performance is problem-dependent [9], the integrated genetic operators need to be carefully designed according to the characteristics of the problem. This work aims to improve the performance of PSO-based feature selection by proposing novel crossover and mutation operators.

A. Goals

The overall goal of this work is to develop a new PSO-based feature selection algorithm to reduce the number of features while maintain or even improve the classification performance over using all features. In order to achieve this goal, two genetic operators, i.e. crossover and mutation, are designed specifically for feature selection to enhance the exploration ability of a PSO-based feature selection algorithm. Both operators aim to build new candidate feature subsets, whose qualities are better than the current feature subsets selected by the swarm. The proposed algorithm is examined and compared against three other PSO-based feature selection algorithms on eight datasets. Specifically, we will investigate:

- whether the new approach can successfully select a smaller number of features and achieve better classification performance than using all features,
- whether the proposed approach can outperform three other PSO-based features selection algorithms in terms of the number of features and the classification performance,
- whether integrating new genetic operators can enhance the exploration ability of PSO to prevent the premature convergence problem.

II. Background

A. Particle Swarm Optimisation

Particle Swarm Optimisation (PSO) was originally proposed by Kennedy and Eberhart [10], in which a set of particles, called a swarm, are used to solve a problem. Each particle is a candidate solution, which is described by a position in the problem's search space. The position is a vector of real number, whose length is equal to the dimensionality of the search space. Each particle also has its own velocity to move around the search space. In order to follow promising trajectories, particles maintain their best position, called $pbest$, and their neighbours' best position, called $gbest$. The two best positions are used to determine the velocity vector. The position and velocity of the i^{th} particle, denoted by x_i and v_i, are updated according to the following equations:

$$v_{id}^{t+1} = w*v_{id}^{t}+c_1*r_{i1}*(p_{id}-x_{id}^{t})+c_2*r_{i2}*(p_{gd}-x_{id}^{t}) \quad (1)$$

$$x_{id}^{t+1} = x_{id}^{t} + v_{id}^{t+1} \quad (2)$$

where t means the t^{th} iteration in the whole evolutionary process, d is the d^{th} entry in the vectors or the d^{th} dimension of the search space, p_{id} and p_{gd} represents the d^{th} component of $pbest$ and $gbest$, respectively. w is an inertia weight, c_1 and c_2 are two acceleration weights, and r_{i1}, r_{i2} are two random numbers uniformly distributed between [0,1].

There are two main PSO variants, continuous and binary PSO, which have been applied to feature selection. It has been shown that continuous PSO usually results in more promising solutions [11]. Therefore, this work will use continuous PSO to achieve feature selection.

B. Related Work On Feature Selection

1) Traditional Feature Selection: Sequential feature selection is a well-known traditional feature selection algorithm. Sequential search starts from an empty feature subset, called sequential forward search (SFS [12]) or a full set of features, called sequential backward search (SBS [13]). At each step, SFS (or SBS) adds (or removes) a feature to get the best performance improvement. However, once a feature is added or removed, it can not be removed or added later. This problem is addressed in a "$plus-l-take-away-r$" approach proposed in [14]. In this approach, after l forward steps, r backward steps are performed to remove features. Later, Nakariyakul et al. [15] proposed an improved sequential search, which replaces the selected weak features by an unselected feature to enhance the feature subset's quality.

2) PSO based Feature Selection: Recently, PSO has been widely applied to feature selection because of its natural representation, simplicity and efficiency. Researchers focus on many different aspects to improve PSO based feature selection algorithms such as modifying initialisation strategy, developing new representation. For instance, Bharti et al. [16] utilised an opposition strategy to initialise the swarm. The strategy generates and evalutes two opposite positions in the search space, in which the better one is used as an initialisation of a particle from swarm. Xue et al. [17] also developed a new initialisation strategy, which estimated good starting positions for particles by utilising sequential searches. Both initialisation strategies could evolve smaller feature subsets with high classification performance within shorter training times than initialising particles by random positions.

In terms of representation, Vieira et al. [18] embedded support vector machine (SVM) parameters into each particle's position, so that PSO could evolve feature subsets and optimise parameters for SVM simultaneously. This scheme increased the cooperation between SVM and the evolved feature subsets, which results in smaller feature subsets with better performance than GAs and other PSO-based algorihms. However, this representation is longer than the traditional ones. Recently, some work attempted to reduce the representation length in order to make the search space to be less complex. Lane et al. [19] proposed a representation using statistical feature clustering information, which groups similar features

together. In the work, Gaussian distributions are applied to determine the number of selected features from each cluster. Later, Nguyen et al. [20] extended work in [19] by allowing to select a variable number of features from each cluster, which is limited by a predefined number. The proposed algorithm, called PSOR, resulted in better feature subsets with higher classification performance than some other conventional methods. In order to make the fitness landscape smoother, a Gaussian distribution was applied to PSOR to determine which features are selected from a cluster. The proposed method, called GPSOR [21] could select the most important features in different independent runs consistently.

Since PSO converges quite fast, premature convergence becomes its typical problem, especially when the search space is too complex with many local optima like in feature selection. One solution is to reset $gbest$ whenever it is not changed for a certain number of iterations. This idea was proposed by Chuang et al. [22], which was shown to be effective according to the experimental results. Another way is to combine PSO with other EC techniques such as GAs. Ghamisi et al. [23] divided the population into two sub-populations, which were optimised by PSO and GAs simultaneously in each generation. The two sub-populations were then combined together so that the diversity of the population was improved. In [24], genetic operators and new operations in discrete search space were applied to update particles in binary PSO. In other works such as [25]–[28], genetic operators were applied to the swarm after the positions were updated by Eq. (2). The experimental results showed that in different domains like spam detection, cancer classification and gender classification, integrating genetic operators into PSO improved the swarm diversity, which helped to select a smaller number of features with better classification performance than using either PSO or GAs only. However, in the above works, genetic operators were applied only to binary PSO since the binary representation is exactly same as the bit string in GAs. Continuous PSO has been shown to be more effective than binary PSO on feature selection [11], so it would be promising to utilise genetic operators in continuous PSO. To the best of our knowledge, this will be the first time genetic operators are applied to continuous PSO to solve feature selection problems. The aim of the integration is not only to improve the swarm diversity but also to make the continuous representation more meaningful in feature selection, which leads to a smoother fitness landscape.

III. Proposed Feature Selection Approach

Genetic operators, specifically mutation and crossover are useful for PSO to maintain the diversity and avoid being stuck at local optima. This section explains how the operators are applied to continuous PSO to achieve feature selection.

Before explaining how genetic operators are embedded into PSO, it would be helpful to explain how continuous PSO is applied to feature selection, a binary problem. Specifically, each particle's position is a vector of real numbers, which are between 0 and 1. Each position entry corresponds to an original feature and a threshold θ is used to determine whether or not the feature is selected. Particularly, if $x_d > \theta$ then the d^{th} feature is selected. Otherwise, the d^{th} feature is not selected. However, this representation does not make the fitness landscape smooth. For example, suppose that $\theta = 0.7$ and in two different particle's positions, the corresponding position entries of a feature are 0.8 and 0.9. So both values have the same meaning that the feature is selected even though they have different values. By applying genetic operators, the position entry is more meaningful , which is explained later.

In this work, each particle is evaluated by using the fitness function shown in Eq. (3).

$$Fitness = \alpha * \frac{\#Features}{\#AllFeatures} + (1 - \alpha) * ErrorRate \quad (3)$$

where $\alpha \in [0, 1]$ shows the relative importance between the number of selected features and the classification error rate. $ErrorRate$ is the classification error rate obtained by the selected feature subset.

A. Crossover

In order to avoid premature convergence, the crossover operator is performed between a number of particle pairs in each iteration. A roulette wheel selection based on the fitness value is used to select particles, which then are used as the parents of the crossover operator. A uniform crossover is applied to the selected particles to derive a new particle, called a child. The child is evaluated and compared against their parents' current and historical best fitnesses. If its fitness is better than the parent's current fitness value, this child will replace the parent in the swarm. If the child's fitness is even better than the parent's personal best fitness, the parent's personal best position is then replaced by the child's position. The improvement of those parents is then propagated through the swarm during the updating process.

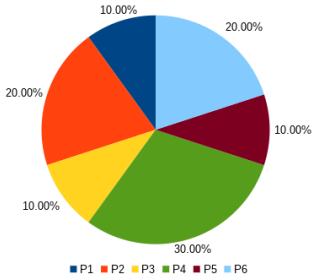
Since the particles within the swarm will be more similar near the end of the run, the crossover will have more impact at the beginning, when the population is just randomly initialised and diverse. Therefore, the number of particles , which is used to apply the crossover operation, is reduced with respect to the increase in the number of iterations (see Eq. 4). By doing so, the computation cost will be reduced, while the crossover performance is maintained.

$$P_i = \lfloor n - i * \frac{n-2}{I} \rfloor \quad (4)$$

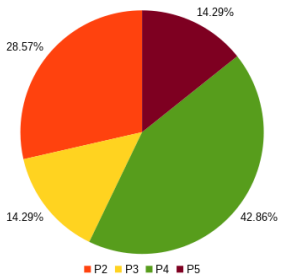
where P_i is the number of selected particles for crossover at i^{th} iteration, and I is the total number of iterations.

An example of typical run-through of selecting particles for crossover operator is outlined below. Suppose the swarm contains 6 particles $\{p_1, p_2, p_3, p_4, p_5, p_6\}$ with these respective fitness values $\{0.1, 0.2, 0.1, 0.3, 0.1, 0.2\}$. We are going to do the crossover operation 2 times, which means $P_i = 4$ particles being selected. The process of performing crossover are shown as follows.

- Step 1: The probabilities of selecting each particle is shown in Fig. 1a. Two particles are chosen via a roulette wheel selection. Since we aim to minimise the fitness



(a) The probability of a particle being selected to be a parent of the crossover operation



(b) The probability of a particle being selected to be a parent of the crossover operation after feature p_1 and p_6 being selected

Fig. 1: Selecting parents for the crossover operation basing on fitness value

value, the lower a particle's fitness value is, the better that particle is. As can be seen from the figure, the particle, which has higher fitness value than the others, has more chance to be chosen. In other words, the worse particle will be more likely to be improved with the crossover operation. Since in a search space, the number of local optima is much smaller than the total number of solutions, particles with low fitness values are usually similar especially at the later iterations. In PSO, the swarm moves towards a single best position causing the premature convergence problem. So by performing crossover on worse particles it is expected that the operator has more effect to assist the swarm to visit unexplored good regions. In this example, we assume that the roulette wheel selection determines that the particle 1 and particle 6 are selected.

- Step 2: After being chosen, particles 1 and 6 are removed from the wheel, so other particles will have chance to improve its fitness via crossover operation. The probabilities of selecting the remaining particles are shown in Fig. 1b. In this iteration, another two particles are chosen as parents of the crossover operation.

B. Mutation

Although the crossover operation can be used to better explore the search space, it has less impact near the end of run because particles in the swarm become very similar. Opposite to crossover, mutation operation has less impact at the beginning of a run, and more near the end [29]. Therefore, integrating a mutation operation into the PSO algorithm would also improve the exploration ability.

Each particle within the swarm records a global best position i.e. $gbest$, which is the best position so far being discovered by the particle and its neighbours. If the $gbest$'s fitness value is not improved after a number of iterations, the particle is probably trapped in local optima. This is when a mutation operation needs to be applied to improve the $gbest$ and pull the swarm out of the local optima.

1) Confidence of a feature's decision: The global best is represented by a vector of real numbers, which are continuous values in the interval [0,1]. Suppose the $gbest$'s position is encoded as $(g_1, g_2, \ldots, g_n)$. These values not only represent the decisions on features but also show how confident the decisions are appropriate. For example, $g_1 = 0.8$ and $g_2 = 0.9$ shows that both feature 1 and feature 2 are selected, since both values are greater than the threshold $\theta = 0.7$. However, it is believed that feature 2 is more deserved to be selected than feature 1, because g_2 is far from θ than g_1. Similarly, if $g_2 < g_1 < \theta$ then it is more confident that the second feature should not be selected than the first feature. In other words, the farther the distance between the feature's value and the threshold, the more the confidence that the feature should be selected or should not be selected.

2) Mutation on gbest: Taking the idea about confidence of a feature's decision, a new mutation method is proposed for $gbest$ position, where the less confident feature is more likely to be mutated. Firstly, the confident rate (CR) of each feature is calculated based on the distance from its value to the threshold θ, which is shown in Eq. (5). As can be seen from the equation, in both cases the numerator is the distance between an entry's value to the threshold θ, which means the more confident a feature, the larger the numerator. The denominator is a normalisation term to ensure that CR is in the range [0.1].

$$CR_j = \begin{cases} \frac{g_j - \theta}{1 - \theta} & \text{if } g_j > \theta \\ \frac{\theta - g_j}{\theta} & \text{if } g_j \leq \theta \end{cases} \tag{5}$$

where CR_j is the confident rate of j^{th} feature, and g_j is the j^{th} feature's value in $gbest$.

A temporary position, called a child, is generated by applying mutation on the $gbest$ position. After calculating the confident rate for a feature, a random number r is generated. If $r < CR_j$, then the j^{th} element in the child's position is set to the corresponding entry in $gbest$. Otherwise, the element is a mutated value of $gbest$'s corresponding entry, which is determined by Eq. (6). The generated child replaces the current $gbest$ if it achieves better fitness value.

$$child_j = 1 - g_j \tag{6}$$

where $child_j$ is the j^{th} position entry value of the $child$, $gbest_j$ is the corresponding entry value in $gbest$. It can be seen that if a feature is selected, it will not be selected after being mutated and vice versa. More importantly, the confident rate CR plays a role in the mutation rate. The more confident a feature's decision, the less likely the decision is changed by the mutation operator.

Algorithm 1 : Pesudo-code of CMPSO

```
begin
randomly initialise the position and velocity of each particle;
while Maximum iteration is not reached do
    evaluate the fitness of each particle;
    for i = 1 to PopulationSize do
        update the pbest of particle i;
    end for
    perform crossover operation on the swarm;
    update the gbest;
    if gbest is not changed for 3 iterations then
        perform mutation operation on the gbest to produce a child;
        if child is better than gbest then
            replace gbest by child
        end if
    end if
    for i = 1 to PopulationSize do
        Update velocity of particle i;
        Update position of particle i;
    end for
end while
end
```

3) Mutation example: Suppose $gbest$'s fitness value is not changed over 3 iterations, we are going to mutate this $gbest$. Assume that:

- The threshold is θ = 0.7
- The total number of features is 4
- The $gbest$ position is [0.14, 0.76, 0.91, 0.21]

According to the Equation 5, the confident rate of each features can be calculated as below:

- $CR_1 = \frac{0.7 - 0.14}{0.7} = 0.8$
- $CR_2 = \frac{0.76 - 0.7}{1 - 0.7} = 0.2$
- $CR_3 = \frac{0.91 - 0.7}{1 - 0.7} = 0.7$
- $CR_4 = \frac{0.7 - 0.21}{0.7} = 0.7$

Suppose the first random number is generated, $r_1 = 0.45$, which is smaller than CR_1. So the child's first position entry, which corresponds to the first feature, is set to $gbest_1 = 0.14$. Similarly, other random numbers are generated for the other features to fully build the child, which is shown as below:

- $r_2 = 0.3 > CR_2$, so $child_2 = 1 - gbest_2 = 1 - 0.76 = 0.24$
- $r_3 = 0.65 < CR_3$, so $child_3 = gbest_3 = 0.91$
- $r_4 = 0.75 > CR_4$, so $child_4 = 1 - gbes_4 = 1 - 0.21 = 0.79$

The mutated child's position is [0.14, 0.24, 0.91, 0.79].

Dataset	#Features	#Classes	#Instances
Wine	13	3	178
Vehicle	18	4	846
Ionosphere	34	2	351
Sonar	60	2	208
Musk1	166	2	476
Arrhythmia	279	16	452
Madelon	500	11	4400
Multiple Features	649	15	2000

TABLE I: Datasets.

C. Overall Algorithm

The pseudo-code of the proposed algorithm, Crossover-Mutation PSO (CMPSO), which embeds genetic operators into a continuous PSO, is given in Algorithm 1.

IV. Experimental Design

A. Benchmark Techniques

In order to examine the performance of the proposed algorithm (CMPSO), three state-of-the-art PSO based feature selection algorithms are used for comparison in the experiments. Firstly, GPSO [19] and PSOR [20] are selected as representatives of binary and continuous PSO based feature selection algorithms, respectively. In GPSOR [21], although feature selection is achieved by using continuous PSO, the fitness landscape is still smooth since the continuous position entry is used to build a Gaussian distribution, which helps to determine the selected features. Therefore, GPSOR is selected to compare against CMPSO. In addition, to analyse the effect of genetic operators, CMPSO's evolutionary process is compared with original PSO (OriPSO)'s one.

Since it has been shown that the three PSO based algorithms already achieve better performance than sequential feature selection algorithms, CMPSO is not compared with sequential algorithms due to the limited space.

B. Datasets and Parameter Settings

In this work, four PSO based feature selection algorithms are compared on eight different datasets from the UCI machine learning repository [30]. The datasets are selected so that they have different numbers of features, classes and instances.

The parameters of the PSO algorithms are set as follows: $w = 0.7928, c_1 = c_2 = 1.49618$, which are recommended settings for PSO [10]. For continuous and binary PSO, v_{max} is set to 0.2 and 6.0, respectively. The population size is 30, the maximum number of iterations is 100 and the threshold θ is set to 0.7 so that the particles are initialised with a small number of features. The α in Eq. (3) is set to 0.02 so that feature selection algorithms focus more on improving the classification performance. Each algorithm is run 30 independent times. A statistical significance test, Wilcoxon test, is performed between the classification accuracies achieved by different algorithms. The significance level of the test is selected as 0.05. In this work, K-nearest neighbour (KNN) is used as the classification algorithm to calculate the $ErrorRate$ in Eq. (3). K is set to 5 so that KNN can avoid noise instances while still maintain its efficiency.

Dataset	Method	Size	AveTrain ± Std	AveTest ± Std	T
Wine	All	13		76.54	+
	GPSO	5.4	96.71 ± 0.77	96.59 ± 2.76	+
	GPSOR	4.60	97.37 ± 0.42	97.70 ± 2.52	=
	PSOR	4.75	95.05 ± 0.58	96.70 ± 3.10	+
	CMPSO	4.70	97.28 ± 0.34	97.24 ± 2.89	
Vehicle	All	18		83.86	+
	GPSO	8.94	86.11 ± 0.20	84.30 ± 0.62	=
	GPSOR	7.30	90.10 ± 0.40	84.74 ± 0.49	-
	PSOR	5.87	84.61 ± 0.56	84.72 ± 0.87	=
	CMPSO	7.57	90.25 ± 0.43	84.49 ± 0.44	
Ionosphere	All	34		83.81	+
	GPSO	7.66	91.59 ± 0.47	89.50 ± 1.68	-
	GPSOR	3.17	93.90 ± 0.67	86.89 ± 1.80	+
	PSOR	9.7	90.04 ± 0.99	88.63 ± 1.68	=
	CMPSO	3.77	93.75 ± 0.86	87.94 ± 2.00	
Sonar	All	60		76.19	+
	GPSO	17.64	86.74 ± 0.94	78.19 ± 4.14	+
	GPSOR	10.17	90.67 ± 1.60	78.25 ± 2.95	+
	PSOR	14.33	87.01 ± 2.00	78.94 ± 4.02	+
	CMPSO	11.60	91.59 ± 1.74	79.42 ± 2.48	
Musk1	All	166		83.92	+
	GPSO	39.64	90.02 ± 0.60	84.95 ± 2.73	=
	GPSOR	38.93	93.22 ± 1.40	83.29 ± 2.48	+
	PSOR	35.03	89.78 ± 1.25	83.12 ± 3.41	+
	CMPSO	39.93	93.47 ± 1.12	85.06 ± 2.49	
Arrhythmia	All	279		94.46	+
	GPSO	45.5	94.87 ± 0.91	94.85 ± 0.34	=
	GPSOR	42.03	95.75 ± 0.18	95.12 ± 0.34	=
	PSOR	44.17	95.11 ± 0.20	94.96 ± 0.38	=
	CMPSO	44.97	95.75 ± 0.19	95.07 ± 0.42	
Madelon	All	500		70.9	+
	GPSO	36.08	85.45 ± 0.73	85.68 ± 1.10	-
	GPSOR	51.13	89.20 ± 1.41	84.06 ± 1.64	-
	PSOR	54.39	83.73 ± 1.74	83.40 ± 2.00	-
	CMPSO	107.2	89.4 ± 0.73	81.57 ± 1.46	
Multiple features	All	649		98.63	+
	GPSO	91.4	99.38 ± 0.38	99.01 ± 0.13	=
	GPSOR	51	99.36 ± 0.07	98.86 ± 0.17	+
	PSOR	51.07	99.17 ± 0.09	98.84 ± 0.18	+
	CMPSO	110.77	99.53 ± 0.05	99.05 ± 0.01	

TABLE II: Experimental Results of CMPSO

V. Experimental Results

This section firstly discusses the performance of CMPSO and the three other PSO based feature selection algorithms (Table II), then compares the search ability between CMPSO and OriPSO.

Table II shows the experimental results of the CMPSO algorithm, where "All" means that all the available features are used for classification. "Size" shows the average number of selected features over the 30 runs. "Ave-Train", and "Ave-Test" illustrate the averages of the training and testing accuracies over the 30 independent runs."Std" is the standard deviation of the corresponding accuracy. T shows the results of the statistical significance tests on the accuracy of CMPSO and other algorithms. "+" or "-" means that CMPSO achieved significantly better or worse classification performance than other algorithms, "=" means there is no significant difference between them.

A. Results of CMPSO

As can be seen from Table II, on all datasets, CMPSO successfully selects feature subsets, which achieve significantly higher classification accuracy than using all features. In addition, on all datasets, the number of selected features is always less than a third of the total number of features. Especially, on datasets with large numbers of features such as Madelon and Multiple Features, CMPSO can reduce around 80% of the features.

The experimental results show that CMPSO can choose a small number of features while improving the classification performance over using all features.

B. Comparisons with PSO based Algorithms

According to Table II, in terms of training accuracy, CMPSO achieves the best mean performance on seven out of the eight datasets. Only on the Ionosphere dataset, CMPSO is ranked at the second position with 93.75% accuracy, which is only 0.25% lower than the highest mean accuracy, achieved by GPSOR.

Compared to the PSOR algorithm, in terms of testing accuracy, CMPSO also achieves similar or better performance on seven out of eight datasets, while the number of selected features remains similar on the small datasets. It is remarkable that on all datasets, CMPSO achieves much higher training accuracy than PSOR. Especially, on Vehicle, CMPSO's training accuracy is about 5.27% higher than PSOR's one.

In comparison with GPSOR, in terms of testing accuracy, CMPSO outperforms GPSOR on three datasets and achieves similar accuracy on three of the remaining five datasets. In addition, on seven out of the eight datasets, CMPSO also achieves higher training accuracy than GPSOR.

Compared to GPSO, CMPSO achieves higher testing accuracy on three out of the eight datasets and similar accuracy on four of the remaining five datasets. However, GPSO performs better than CMPSO on the Ionosphere dataset about 2%. The reason is that CMPSO selects only 3.77 features, which is 2 times less than the number of features selected by GPSO. Furthermore, on each dataset, CMPSO always achieves higher training accuracy than GPSO. For example, on Sonar dataset, the training accuracy of CMPSO is 90.62%, which is 4% higher, while CMPSO selects even a smaller number of features than GPSO.

It can be seen that, on seven out of the eight datasets, CMPSO achieves similar or better testing accuracy than at least two of the three algorithms. Only on the Madelon dataset, the significance test illustrates that CMPSO is worse than all three PSO algorithms. However, CMPSO outperforms all other PSO algorithms in terms of training performance on this dataset. For example, CMPSO's accuracy is around 6% better than PSOR's one. So overfitting may be a problem for CMPSO, where the genetic operators assist CMPSO to find out really good feature subsets for the training set, which is not generalised well on the test set. In addition, as can be seen from the Table II, on the small datasets, CMPSO selects a similar number of features in comparison with the other PSO algorithms. However, on large datasets, CMPSO tends to select more features to maintain high classification performance. So CMPSO can adapt with datasets having different numbers of original features better than the other PSO algorithms.

The results suggest that integrating crossover and mutation operations into PSO improves the search ability of a PSO algorithm, which is clearly shown by the higher training

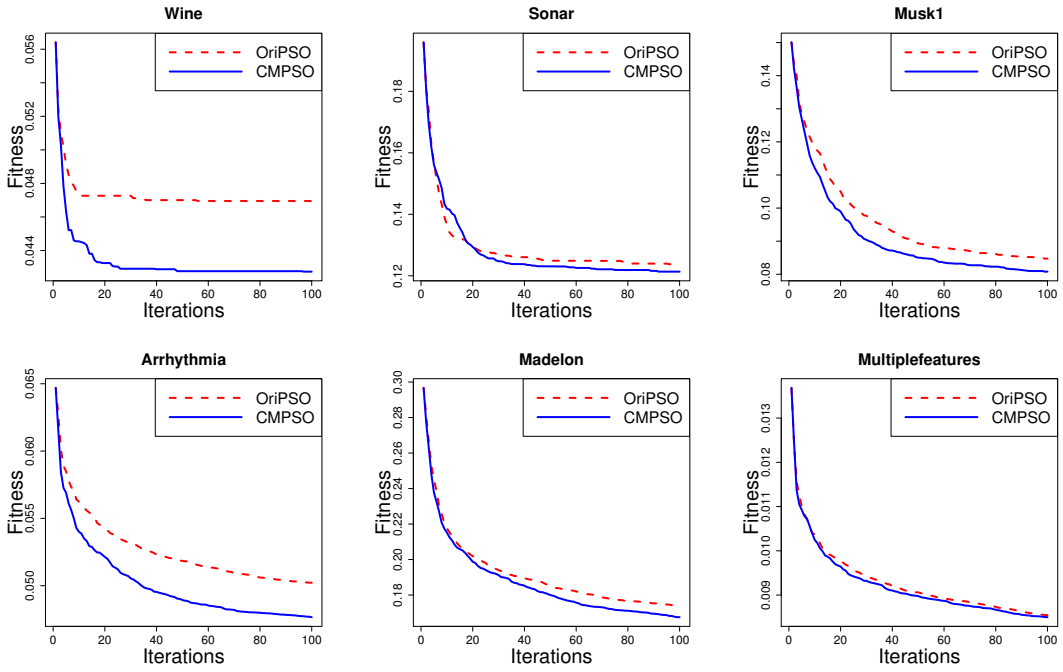


Fig. 2: Fitness evolution figures

accuracy on all datasets. In the next section, a comparison between original PSO (OriPSO) and CMPSO is done to further analyse the effect of crossover and mutation operations.

C. Evolutionary Process

For each dataset, CMPSO and OriPSO are run 30 independent times. The fitness value of OriPSO is also calculated by using the Eq (3). Each run contains 100 iterations, in which the fitness value is improved after each iteration. The average of $gbest$'s fitness values of the 30 independent runs at each iteration is calculated. Those average values are used to evaluate the fitness evolution of the above algorithms. The smaller fitness an algorithm achieves, the better the algorithm is. Fig. 2 illustrates the fitness values in each iteration on 6 datasets. Due to the limited space, the results on Vehicle and Ionosphere are not presented since their numbers of features and patterns of evolutionary processes are similar to the Wine dataset's ones. In the figures, X-axis represents the iteration index and Y-axis represents the average fitness value in each iteration. CMPSO's fitness values are represented by blue lines, while red-dot lines show the fitness values of OriPSO.

As can be seen from Fig. 2, on all datasets, the blue lines are always under the red lines, which indicates that the feature subsets discovered by CMPSO always have better fitness values than the ones discovered by OriPSO. On the small datasets like Wine the difference between those 2 approaches are clearly shown. Although starting from the same initialisation, CMPSO leaves OriPSO behind in the first 10 iterations on the Wine, Musk1 and Arrhythmia datasets. On the Sonar dataset, from the 10^{th} iteration to the 20^{th} iteration, OriPSO has similar or smaller fitness values than CMPSO. However, after that CMPSO is able to catch up and evolves better subsets in the later 80 iterations. On the Arrhythmia dataset, despite of starting at the same position (same fitness value), the gap between the red and blue lines is maintained or even getting bigger with respect to the increment of iterations. On the Musk1 dataset, OriPSO still does not get closer to CMPSO, which is the common pattern on five out the six datasets. Only on the Multiple Features datasets, the distance between two lines is not significant. However, it is still visible that CMPSO is a bit better than OriPSO. The reason is that on the Multiple Features dataset, the classification performance is already very high when using all features (around 98.63%), so it is harder to improve the accuracy on this dataset than on other datasets.

The results clearly show that crossover and mutation operators assist OriPSO to better explore the search space, which is shown by the better fitness values that CMPSO can achieve during an entire run. In addition, CMPSO could reach good points in the search space earlier than the original OriPSO.

VI. Conclusion and Future Work

The goal of this paper is to develop a PSO based feature selection algorithm, which can select a small number of features and increase the classification performance. This goal is achieved by integrating crossover and mutation operators into continuous PSO to improve its search ability. Firstly, crossover operator is applied to a number of particles, which is determined dynamically according to the current iteration. If the generated offsprings achieve better fitness values than the parents, they will replace their parents. After that, the mutation operator is applied to $gbest$ to improve its quality, which in turn guides the swarm towards better positions to avoid being stuck at local optima. The mutation uses position entries to

build confident rates, which can be considered the probability of mutating the entries. By embedding genetic operators deeply into continuous PSO, the proposed algorithm called CMPSO can successfully reduce the number of features and improve the classification performance over using all features. It also achieves similar or better classification performance than three other state-of-the-art PSO based feature selection algorithms on most of the datasets. The experimental results also show that CMPSO can adapt with different numbers of features better than other PSO based algorithms to maintain the classification performance. The genetic operators also assist continuous PSO to evolve better feature subsets during the whole evolutionary process, which is shown in the comparison between CMPSO and the original PSO.

From the results, it can be seen that although genetic operators improve continuous PSO's searching ability, sometimes CMPSO finds feature subsets which overfit the training set resulting in slightly lower testing performance. So it is important to embed a generalisation measure into the crossover or mutation processes to avoid overfitting. In addition, the computation cost of CMPSO is expensive, mainly because the crossover process evaluates its generated children by applying the wrapper approach. In the future, we will investigate on using some filter measures or surrogate models to estimate the children's fitness values, which will enhance the algorithm's scalability. Although the proposed algorithm is designed specifically for feature selection, the idea of using genetic operators can be utilized to avoid PSO's premature convergence issue. We will investigate this direction and compare with other stat-of-the-art PSO algorithms.

REFERENCES

[1] J. Friedman, T. Hastie, and R. Tibshirani, *The elements of statistical learning*. Springer series in statistics Springer, Berlin, 2001, vol. 1.

[2] B. Xue, M. Zhang, W. N. Browne, and X. Yao, "A survey on evolutionary computation approaches to feature selection," *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 4, pp. 606–626, 2016.

[3] Q. Gu, Z. Li, and J. Han, "Generalized fisher score for feature selection," *arXiv preprint arXiv:1202.3725*, 2012.

[4] B. Xue, L. Cervante, L. Shang, W. N. Browne, and M. Zhang, "A multi-objective particle swarm optimisation for filter-based feature selection in classification problems," *Connection Science*, vol. 24, no. 2-3, pp. 91–116, 2012.

[5] R. Fu, P. Wang, Y. Gao, and X. Hua, "A new feature selection method based on relief and svm-rfe," in *2014 12th International Conference on Signal Processing (ICSP)*, Oct 2014, pp. 1363–1366.

[6] M. A. Hall and L. A. Smith, "Feature selection for machine learning: Comparing a correlation-based filter approach to the wrapper." in *FLAIRS conference*, vol. 1999, 1999, pp. 235–239.

[7] H. B. Nguyen, B. Xue, and P. Andreae, "Mutual information estimation for filter based feature selection using particle swarm optimization," in *European Conference on the Applications of Evolutionary Computation*. Springer, 2016, pp. 719–736.

[8] M. Hu, T. Wu, and J. D. Weir, "An adaptive particle swarm optimization with multiple adaptive methods," *IEEE Transactions on Evolutionary Computation*, vol. 17, no. 5, pp. 705–720, 2013.

[9] Y. Zhang, S. Wang, and G. Ji, "A comprehensive survey on particle swarm optimization algorithm and its applications," *Mathematical Problems in Engineering*, vol. 2015, 2015.

[10] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Neural Networks, 1995. Proceedings., IEEE International Conference on*, vol. 4, Nov 1995, pp. 1942–1948 vol.4.

[11] B. Xue, M. Zhang, and W. N. Browne, "Multi-objective particle swarm optimisation (PSO) for feature selection," in *Proceedings of the 14th annual conference on Genetic and evolutionary computation*. ACM, 2012, pp. 81–88.

[12] A. W. Whitney, "A direct method of nonparametric measurement selection," *IEEE Transactions on Computers*, vol. 100, no. 9, pp. 1100–1103, 1971.

[13] T. Marill and D. M. Green, "On the effectiveness of receptors in recognition systems," *IEEE Transactions on Information Theory*, vol. 9, no. 1, pp. 11–17, 1963.

[14] S. D. Stearns, "On selecting features for pattern classifiers." in *Proceedings of the 3rd International Conference on Pattern Recognition (ICPR)*, Coronado, CA, 1976, pp. 71–75.

[15] S. Nakariyakul and D. P. Casasent, "An improvement on floating search algorithms for feature subset selection ," *Pattern Recognition*, vol. 42, no. 9, p. 19321940, 2009.

[16] K. K. Bharti and P. K. Singh, "Opposition chaotic fitness mutation based adaptive inertia weight BPSO for feature selection in text clustering," *Applied Soft Computing*, vol. 43, pp. 20–34, 2016.

[17] B. Xue, M. Zhang, and W. N. Browne, "Particle swarm optimisation for feature selection in classification: Novel initialisation and updating mechanisms," *Applied Soft Computing*, vol. 18, no. 0, pp. 261–276, 2014.

[18] S. M. Vieira, L. F. Mendonça, G. J. Farinha, and J. M. Sousa, "Modified binary PSO for feature selection using svm applied to mortality prediction of septic patients," *Applied Soft Computing*, vol. 13, no. 8, pp. 3494–3504, 2013.

[19] M. C. Lane, B. Xue, I. Liu, and M. Zhang, "Gaussian based particle swarm optimisation and statistical clustering for feature selection," in *European Conference on Evolutionary Computation in Combinatorial Optimization*. Springer, 2014, pp. 133–144.

[20] H. Nguyen, B. Xue, I. Liu, and M. Zhang, "PSO and statistical clustering for feature selection: A new representation," in *Simulated Evolution and Learning*, ser. Lecture Notes in Computer Science. Springer International Publishing, 2014, vol. 8886, pp. 569–581.

[21] H. B. Nguyen, B. Xue, I. Liu, P. Andreae, and M. Zhang, "Gaussian transformation based representation in particle swarm optimisation for feature selection," in *Applications of Evolutionary Computation*. Springer, 2015, pp. 541–553.

[22] L.-Y. Chuang, H.-W. Chang, C.-J. Tu, and C.-H. Yang, "Improved binary PSO for feature selection using gene expression data," *Computational Biology and Chemistry*, vol. 32, no. 1, pp. 29–38, 2008.

[23] P. Ghamisi and J. A. Benediktsson, "Feature selection based on hybridization of genetic algorithm and particle swarm optimization," *IEEE on Geoscience and Remote Sensing Letters*, vol. 12, no. 2, pp. 309–313, 2015.

[24] R. Huang and X. Li, "Band selection based on evolution algorithm and sequential search for hyperspectral classification," in *Audio, Language and Image Processing, 2008. ICALIP 2008. International Conference on*. IEEE, 2008, pp. 1270–1273.

[25] E. Alba, J. García-Nieto, L. Jourdan, and E.-G. Talbi, "Gene selection in cancer classification using PSO/SVM and GA/SVM hybrid algorithms," in *2007 IEEE Congress on Evolutionary Computation*. IEEE, 2007, pp. 284–290.

[26] S. Li, X. Wu, and M. Tan, "Gene selection using hybrid particle swarm optimization and genetic algorithm," *Soft Computing*, vol. 12, no. 11, pp. 1039–1048, 2008.

[27] Y. Zhang, S. Wang, P. Phillips, and G. Ji, "Binary PSO with mutation operator for feature selection using decision tree applied to spam detection," *Knowledge-Based Systems*, vol. 64, pp. 22–31, 2014.

[28] M. Nazir, A. Majid-Mirza, and S. Ali-Khan, "PSO-GA based optimized feature selection using facial and clothing information for gender classification," *Journal of applied research and technology*, vol. 12, no. 1, pp. 145–152, 2014.

[29] R. C. Eberhart and Y. Shi, "Comparison between genetic algorithms and particle swarm optimization," in *Evolutionary Programming VII*. Springer, 1998, pp. 611–616.

[30] M. Lichman, "UCI machine learning repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Sciences," 2013. [Online]. Available: http://archive.ics.uci.edu/ml