Adaptive Multi-Subswarm Optimisation for Feature Selection on High-Dimensional Classification

Binh Tran[1,2], Bing Xue[1], Mengjie Zhang[1]
[1] School of Engineering and Computer Science,
Victoria University of Wellington,
Wellington, New Zealand
[2] Can Tho University, Can Tho, Viet Nam
{binh.tran, bing.xue, mengjie.zhang}@ecs.vuw.ac.nz

ABSTRACT

Feature space is an important factor influencing the performance of any machine learning algorithm including classification methods. Feature selection aims to remove irrelevant and redundant features that may negatively affect the learning process especially on high-dimensional data, which usually suffers from the curse of dimensionality. Feature ranking is one of the most scalable feature selection approaches to high-dimensional problems, but most of them fail to automatically determine the number of selected features as well as detect redundancy between features. Particle swarm optimisation (PSO) is a population-based algorithm which has shown to be effective in addressing these limitations. However, its performance on high-dimensional data is still limited due to the large search space and high computation cost. This study proposes the first adaptive multi-swarm optimisation (AMSO) method for feature selection that can automatically select a feature subset of high-dimensional data more effectively and efficiently than the compared methods. The subswarms are automatically and dynamically changed based on their performance during the evolutionary process. Experiments on ten high-dimensional datasets of varying difficulties have shown that AMSO is more effective and more efficient than the compared PSO-based and traditional feature selection methods in most cases.

KEYWORDS

Feature selection, Particle swarm optimisation, High-dimensional data, Classification

ACM Reference Format:
Binh Tran^{1,2}, Bing Xue^{1}, Mengjie Zhang^{1}. 2019. Adaptive Multi-Subswarm Optimisation for Feature Selection on High-Dimensional Classification. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

High-dimensional data is increasingly involved in machine learning applications. Thousands of collected features bring challenges to the existing learning algorithms due to the curse of dimensionality and the existence of irrelevant and/or redundant features. These features not only unnecessarily increase the search space of the problem but also obscure the effect of the relevant features on showing hidden patterns of the data. Therefore, it is necessary to perform feature selection (FS) on these datasets to select a subset of features that can improve the performance of learning algorithms.

FS is a challenging combinatorial optimisation problem due to its large search space of 2^N possible subsets from N original features. An exhaustive search is impractical even for a problem with hundreds of features. Therefore, heuristic search such as sequential forward or backward selection methods and their variants are used instead [13]. However, these greedy search methods may get stuck into local optima, especially in the large search space of high-dimensional data.

One of the most scalable FS approaches to high-dimensional data is feature ranking [7], where features are first ranked based on a certain criterion and then a predefined number of top-ranked features will be chosen as the final subset. Although its running time is proportional to the number of features, this approach requires a predefined number of features to be selected, which is usually unknown in practice. Feature interaction is another challenge where individually irrelevant features may become useful and relevant features may become redundant when combined with other features. This phenomenon is usually overlooked by this approach. A quick remedy for feature ranking is appending a second stage to remove redundant features from the ranked list [6, 22]. This approach has shown to be effective in many problems. However, since the first stage ranks features individually and the second stage scans features from the most to least relevant features, these methods may be stuck in local optima. Therefore, a more powerful search technique is needed to obtain better solutions.

Using a population-based search to maintain multiple candidate solutions simultaneously, particle swarm optimisation (PSO) is well-known with global search ability. In PSO, each particle represents a candidate solution. By sharing their best found solutions with each other, particles fly towards fruitful areas and explore better solutions. PSO has been applied and shown promise in feature subset selection [20]. However, the performance of PSO for FS on high-dimensional data is still limited due to the huge search space. Different strategies have been proposed to improve its performance on such data. The most common one is the two-stage approach where the first stage uses a criterion to remove irrelevant or less relevant features to reduce the number of features given to PSO in the second stage [3, 11]. This approach has shown to be more effective thanks to the ability of (implicitly) considering feature interaction in the second stage. However, since the first stage does not consider feature interaction and the number of selected features has to be predefined, potentially useful features may be left out. Therefore, instead of removing features before applying PSO to narrow the search space, this study applies the divide and conquer principle to reduce the search space of the problem and shorten the PSO running time.

In this paper, an adaptive multi-subswarm optimisation (AMSO) method is proposed for FS on high-dimensional data where each subswarm searches for solutions in different subspaces of the problem and dynamically changes subspace by shrinking the particle length based on its performance during the evolution. Note that this approach is different from cooperative coevolution, which uses multiple subpopulations, each dealing with a subproblem. Specifically, this study addresses the following research questions:

- How to apply the divide and conquer principle in a flexible and dynamic way that enables PSO to effectively and efficiently search in small subspaces while still covering the whole large search space;
- Whether the proposed algorithm can select very small feature subsets that can achieve better accuracy than using the original feature set; and
- Whether the proposed algorithm can achieve better performance than the compared PSO-based and traditional FS methods in terms of accuracy and running time.

2 BACKGROUND

2.1 Particle Swarm Optimisation

PSO is a population-based algorithm [12]. PSO maintains a swarm of particles where positions are N-dimension real-value vectors representing different candidate solutions of a N-dimension problem. A particle moves in the search space according to its velocity which is updated based on its own inertia, its personal experience about the best position (called personal best or *pbest*), and the best position it has communicated from others (called global best or *gbest* if particles are fully connected). Eqs. (1) and (2) are used to update velocity and position of a particle at time $t+1$, respectively.

$$v_{id}^{t+1} = w * v_{id}^{t} + c_1 * r_{1i} * (p_{id}^{t} - x_{id}^{t}) + c_2 * r_{2i} * (p_{gd}^{t} - x_{id}^{t}) \quad (1)$$

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

where v_{id} and x_{id} are the velocity and position of the i^{th} particle in dimension d. w is the inertia weight showing the importance of following its own moving momentum. *pbest* and *gbest* positions (p_{id} and p_{gd}) are used in the second and third terms of Eq. (1) associated with acceleration constants (c_1 and c_2) and random values (r_{1i} and r_{2i}) to guide particles to search for the best solutions.

When applying PSO to FS, a feature subset is encoded in the position of the particle. Position values range from 0 to 1 in the position vector indicates whether the corresponding feature should be selected or not based on a predefined threshold (e.g. 0.6).

2.2 Related Work

In the last decades, many PSO-based FS methods have been proposed and shown promise on high-dimensional data [19]. Different approaches have been used to improve its performance. For example, to address the problem of early convergence of PSO in the large search space, *gbest* is reset based on the majority voting of all *pbest* if *gbest* does not improve for a number of iterations [11]. Similarly, *pbest* [21] or particle [5] are reset when PSO stagnates. Enhancing the particle updating mechanism is another way to improve its performance on a large search space [16]. To maintain swarm diversity in feature selection on high-dimensional data, Gu et al. [8] applied the competitive swarm optimiser (CSO) [4] where particles learn from randomly selected competitors instead of *gbest* and *pbest*. Another popular approach is to reduce the number of features using a filter measure before applying PSO [11, 16]. Local search is also applied on particles [17], or *pbest* [19], or *gbest* [15] to exploit the fruitful areas detected by PSO.

In general, although the existing PSO-based FS methods have shown to be effective, their performance is still limited when using a fix-length representation for all particles. This prevents PSO from scaling well to higher dimensionality.

3 THE PROPOSED METHOD - AMSO

3.1 The proposed AMSO representation

AMSO aims to divide the whole search space into smaller subspaces so that multi-swarms can search more effectively. To achieve this goal, features are sorted in the descendant order of their relevance, which can be measured by any criterion. In this study, AMSO uses symmetrical uncertainty (SU) to rank features. In other words, individually relevant features are presented in the very first dimensions of the particle representation vector.

Based on this representation, we divide the search space into multiple smaller subspaces so that it is easier for PSO to find better solutions. Specifically, the whole swarm is divided into many subswarms with different particle lengths but all start from the first feature of the representation as shown in Fig. 1. This figure shows an example of AMSO with 5 subswarms for a problem with 2000 features. Particles in each subswarm will have the same length so that they can focus their search on the same set of possible solutions and effectively sharing their experience. Given N as the number of original features (and also the maximal length of particles), and M as the number of subswarms, the length of particles in subswarm s, $ParLen_s$ is calculated based on Eq. (3).

$$ParLen_s = s * \frac{N}{M} \quad (3)$$

As can be seen from the unfilled upper right “triangle” in Fig. 1, this strategy reduces the computation for velocity and position updating by roughly half. It also divides the search space into multiple smaller subspaces, making it easier for PSO to find better solutions. On one hand, since individual relevant features are presented in the very first dimensions of the particle representation, subswarms with short lengths are likely to include reasonably good solutions. On the other hand, AMSO still considers longer solutions in the longer subswarms which may contain less relevant features but may provide a better classification accuracy when combining with other features.

3.2 Subswarm Updating

During the evolutionary process, each subswarm can automatically adjust its particle length to focus its search on better subspaces. This strategy enables AMSO to further reduce its search space to a

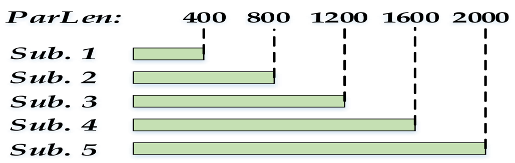

Figure 1: AMSO example with 5 subswarms and N = 2000.

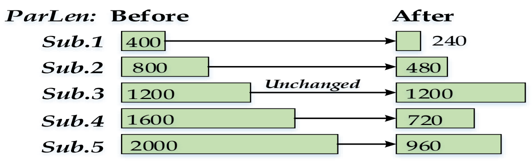

Figure 2: Subswarm updating example with Subswarm 3 containing *gbest*.

smaller and more suitable one, enabling it to reach better solutions in a shorter time. However, if subswarm updating is applied too often, it can negatively affect the AMSO performance. An adequate time should be given for AMSO to learn and converge. Therefore, subswarm updating is only applied when AMSO seems to stagnate in a local optimum which can be detected based on the improvement of *gbest*. If *gbest* does not improve for β iterations, subswarm updating is called. At this point, AMSO will determine the best subswarm which is the one in which the current best solution (*gbest*) is found. The length of this best subswarm will become the new maximal length of AMSO.

Algorithm 1 describes the pseudo-code of the subswarm updating mechanism. The subswarm updating procedure starts with using the length of the current *gbest* particle as the new maximal length. Based on this new maximal length (*MaxLen*), the lengths of each subswarm s is recalculated using $s * \frac{MaxLen}{M}$. The particles will then be shortened by removing the less relevant features at the end of the positions and re-evaluated to be ready for the new iteration.

Fig. 2 shows an example of applying this mechanism on a problem with the maximum length of 2000 and 5 subswarms. Assume the current *gbest* is found in subswarm 3. This subswarm will be kept unchanged, while the others will be cut to fit their new lengths. Note that in order to maintain as much information as possible in the particles, this procedure processes the subswarms in the ascending order of their lengths. This makes sure that the new length is always smaller than the current length. As can be seen from Fig. 2, after subswarm updating, the total length of all particles is significantly reduced.

3.3 Particle Updating

Premature convergence is one of the issues of PSO when applying on high-dimensional problems. Different strategies have been proposed to address this problem. Cheng et al. [4] proposed the competitive swarm optimiser (CSO) in which particles learned from randomly selected competitors. The whole swarm is divided into two groups and pair-wise comparisons are done to determine the winner and the loser. The latter will learn from the former which is kept unchanged (i.e. only half of the swarm is changed in each iteration). *gbest* is also replaced by the mean values of all particle's

Algorithm 1: Subswarm Updating

```
input  : Current swarm
output : New swarm
1  begin
2      $M \leftarrow$ Number of subswarms ;
3      $NewMaxLen \leftarrow$ Length of $gbest$;
4      $k \leftarrow 1$;
5      for Each subswarm s (from the shortest to the longest) do
6          if $ParLen_s \neq NewMaxLen$ then
7              $ParLen \leftarrow k \times NewMaxLen \div M$;
8              Remove the last dimensions of all particles in $s$ to have
                 $ParLen$ dimensions;
9              Calculate fitness of all particles in $s$;
10             $k = k + 1$;
11         end
12     end
13 end
```

position (*Mean*). Eq. (4) shows CSO's updating formula for v_{ld}^{t+1}, which is the velocity of the loser l in dimension d at time $t+1$.

$$v_{ld}^{t+1} = r_{1l} * v_{ld}^{t} + r_{2l} * (x_{wd}^{t} - x_{ld}^{t}) + \phi * r_{3l} * (Mean_{d}^{t} - x_{ld}^{t}) \quad (4)$$

where r_{1l}, r_{2l}, and r_{3l} are random values, x_l and x_w are the loser and winner positions, respectively. ϕ is a weight showing the importance of *Mean* position.

CSO was originally proposed for continuous optimisation problems. Although it has been applied and shown promise in FS [8], its updating mechanism has two drawbacks when applying to FS which is a combinatorial optimisation problem. Firstly, although completely ignoring the personal experience (*pbest*) can increase diversity [4], it may also degrade PSO performance especially on high-dimensional search space. Secondly, when applying CSO to FS [8], the continuous value of a position's dimension is converted into a binary value indicating whether the corresponding feature is selected or not. Therefore, the average position of all particles may not be meaningful, and become misleading. In this study, AMSO uses a simpler updating mechanism that avoids these problems.

Since each subswarm searches for solutions in one subspace, it is more meaningful to share experience among particles of the same subswarm. AMSO applies a similar competitive updating mechanism as in CSO [4] in which particles in each subswarm are randomly divided into two groups and pair-wise comparisons are done to determine the winner and the loser. The latter will learn from the former which is kept unchanged. However, in AMSO, the loser l learns from *pbest* of its winner p_w using a new updating mechanism as shown in Eq. (5) in which w, c, and r are the inertia weight, acceleration constant and random value, respectively. In this updating formula, *pbest* of the winner (p_{wd}) is used to guide the search instead of the winner's current position and the *Mean* position all particles. By learning from the best experience of the winner, the loser has more chance to reach better solutions. Furthermore, applying competition within each subswarm also helps PSO keep its diversity.

$$v_{ld}^{t+1} = w * v_{ld}^{t} + c * r * (p_{wd}^{t} - x_{ld}^{t}) \quad (5)$$

3.4 Fitness Function

Feature evaluation is an important component contributing to the success of a FS algorithm. The two main approaches to evaluating features are wrapper and filter methods, which are based on a learning algorithm and the intrinsic characteristics of the training data, respectively. Therefore, wrapper measures usually obtain better performance than filters with the cost of higher computation time. On the other hand, filters are said to achieve better generalisation than wrappers. AMSO unites the strengths of both approaches by combining K-Nearest Neighbour (KNN) and a distance measure using a weight (μ) as shown in Eq. (6). Because KNN is a distance-based algorithm, the increased computation cost is minor.

$$fitness = (\mu \cdot AvgBalAcc + (1-\mu) \cdot Distance) \tag{6}$$

where *AvgBalAcc* is the average over 10-fold cross-validation on the training set of the balanced accuracy (*BalAcc*) calculated based on Eq. (7).

$$AvgBalAcc = \frac{1}{10}\sum_{fold=1}^{10}(\frac{1}{c}\sum_{i=1}^{c}TPR_i)_{fold} \tag{7}$$

where c is the number of classes of the problem, and the true positive ratio TPR_i is the proportion of correctly identified instances in class i. Since there is no bias to any specific class, the weight for each class is set to $1/c$.

The *Distance* measure [2] is calculated based on Eq. (8). This filter measure is used to maximise the distance between instances of different classes (D_b) and minimise the distance between instances of the same class (D_w). In this study, Manhattan distance is used to measure the distance between two instances $Dis(V_i, V_j)$ since it is more suitable than Euclidean distance for high-dimensional data [1]. Eq.s (9, 10, and 8) show the calculation of D_b, D_w and $Dis(V_i, V_j)$, respectively.

$$Distance = \frac{1}{1 + exp^{-5(D_b - D_w)}} \tag{8}$$

where

$$D_b = \frac{1}{|S|}\sum_{i=1}^{|S|} \min_{\{j|j\neq i, class(V_i)\neq class(V_j)\}} Dis(V_i, V_j) \tag{9}$$

$$D_w = \frac{1}{|S|}\sum_{i=1}^{|S|} \max_{\{j|j\neq i, class(V_i)=class(V_j)\}} Dis(V_i, V_j) \tag{10}$$

$$Dis(V_i, V_j) = \sum_{f=1}^{N} |V_{if} - V_{jf}| \tag{11}$$

3.5 The Overall Algorithm

Algorithm 2 presents the overall algorithm of AMSO. Given the population size *PopSize* and the number of subswarms *M*, all subswarms will have the same number of particles (*SubswarmSize*) equal to *PopSize*/*M*. The length of each particle (*ParLen*) in each subswarm is then calculated based on Eq. 3 each subswarm is then initialised and evolved separately using the updating mechanism as described in Section 3.3. If *gbest* is not changed after β iterations,

Algorithm 2: AMSO Algorithm

```
   input  : $PopSize$, $N$, $M$, $MaxIter$, $\beta$, $\theta$
   output : Selected Features
1  begin
2      Rearrange features in descending order of $SU_C$ ;
3      $MaxLen \leftarrow N$;
4      for Each subswarm s: 1 to M do
5          $SubswarmSize \leftarrow PopSize/M$;
6          $ParLen \leftarrow s \times N \div M$;
7          Randomly initialise $s$ with $SubswarmSize$ particles having
             $ParLen$;
8          Calculate fitness of particles in $s$ using Eq. (6);
9          Update $pbest$, $gbest$;
10     end
11     while (Not reach $MaxIter$) do
12         for Each subswarm s: 1 to M do
13             Randomly divide $s$ into two groups;
14             Apply pair-wise comparision to determine $losers$ and
                 $winners$;
15             Update $losers$' velocity based on Eq. (5);
16             Update $losers$' position based on Eq. (2);
17             Calculate fitness of particles in $s$ using Eq. (6);
18             Update $pbest$, $gbest$;
19             if $gbest$ does not change $\beta$ iterations then
20                 $MaxLen \leftarrow gbest$ length;
21                 Update subswarms based on $MaxLen$   //Sec. 3.2;
22                 Recalculate fitness of particles in $s$ using Eq. (6);
23             end
24         end
25     end
26     Return position indexes in $gbest$ that have values larger than $\theta$;
27 end
```

subswarm updating is applied to automatically change all the subswarms except for the one that has the length of *gbest*. The while loop continues until the maximum iterations is reached, *gbest* is then returned as AMSO's feature subset.

PSO is well-known with its ability in quickly detecting fruitful areas in the search space. However, once there, it is difficult for PSO to perform local search to fine tune its solutions. Although local search adds more computation cost, a careful integration of local search in PSO can effectively and efficiently improve PSO performance on high-dimensional data. Therefore, in this study, AMSO applies the local search procedure using the same way as [19] in the updating *pbest* step at Lines 8 and 17 of Algorithm 2, i.e. after updating each *pbest*.

4 EXPERIMENT DESIGN

Table 1 shows the number of features, instances and classes of the ten high-dimensional datasets [18] used to test the performance of AMSO. The last two columns show the percentage of instances of the smallest class and the largest class, respectively. The datasets are presented in the ascending order of #Features. With thousands of features, very small numbers of instances compared to their dimensionality, and highly unbalanced class distribution, the problems in these datasets become challenging to machine learning algorithms.

Table 1: Datasets

Dataset	#Features	#Ins.	#Class	%Smallest class	%Largest class
SRBCT	2,308	83	4	13	35
Leukemia 1	5,327	72	3	13	53
DLBCL	5,469	77	2	25	75
9Tumor	5,726	60	9	3	15
Brain Tumor 1	5,920	90	5	4	67
Brain Tumor 2	10,367	50	4	14	30
Prostate	10,509	102	2	49	51
Leukemia 2	11,225	72	3	28	39
11Tumor	12,533	174	11	4	16
Lung Cancer	12,600	203	5	3	68

Table 2: Parameter settings

Parameters	Settings
Population Size ($PopSize$)	#features/20 (restriction to 300)
Maximum iterations ($MaxIter$)	100
$c1 = c2$ or c	1.49445
w	$0.9 - 0.5 * \frac{current\ iteration}{max\ iteration}$
θ (Threshold for selecting feature)	0.6
Communication topology	Fully connected (PSO)
μ (fitness weight)	0.8 (PSO, AMSO)
M (number of subswarms)	13 (AMSO)
β (for subswarm updating)	7 (AMSO)

Due to the small number of instances in these datasets, 10-fold cross validation was used to create training and test sets. One fold is used for testing and the remaining nine folds are used to train the FS method. The training set is input to the FS methods to learn a good feature subset. Note that the test set is not seen by all the methods during their FS process. The training and test sets are transformed based on the feature subset and put into KNN (K=1) for performance evaluation. 30 independent runs were conducted with different random seeds on each training set, yielding 300 results (30 runs × 10 folds) for each dataset.

The performance of AMSO was evaluated by comparing the classification accuracy of KNN using the features selected by AMSO, standard PSO, and the CSO-based FS method [8] proposed in 2018. Table 2 shows PSO parameter settings for the three methods. The number of subswarms (M) and the maximum iterations that $gbest$ does not improve to change subswarms (β) are chosen from the ranges [8,15] and [5,10], respectively, based on a grid search on the Brain1 dataset. The best combination of 13 and 7 is used for all datasets. μ is set as 0.8 to bias towards the classification accuracy. AMSO was also compared with CSO since both have a similar updating mechanism as described in Section 3.3. CSO was run with the same PSO settings as AMSO using the author's code. AMSO's source code is available at *https://github.com/tnbinh/AMSO*.

We also compare AMSO with other FS methods including the linear forward selection (LFS) [9], the correlation-based FS method (CFS) [10], and the fast correlation-based FS method (FCBF) [22]. These methods are chosen due to their popularity and the ability to automatically determine the number of selected features as our proposed method. These methods represent typical traditional approaches to FS.

Table 3: Average test results.

Dataset	Method	Time(m)	Size	Best	Mean ± Std	S
SRBCT	Full		2308.0	87.08		+
	PSO	8.2	1119.4	92.50	89.51 ± 1.56	+
	CSO	19.9	85.4	100.00	93.29 ± 3.52	+
	AMSO	**2.0**	**63.7**	100.00	**99.75 ± 0.39**	
Leuk1	Full		5327.0	79.72		+
	PSO	41.2	2615.5	87.36	80.60 ± 2.55	+
	CSO	251.8	170.1	96.81	88.45 ± 3.90	+
	AMSO	**7.6**	**51.5**	97.64	**93.50 ± 1.84**	
DLBCL	Full		5469.0	83.00		+
	PSO	47.6	2681.0	86.33	83.67 ± 1.52	+
	CSO	394.8	**30.1**	100.00	**94.30 ± 4.05**	=
	AMSO	**8.8**	50.6	99.17	93.70 ± 2.59	
9Tumor	Full		5726.0	36.67		+
	PSO	39.2	2811.9	45.00	42.72 ± 1.42	+
	CSO	373.4	220.3	68.33	**59.50 ± 3.72**	−
	AMSO	**6.2**	**52.2**	60.00	55.33 ± 3.46	
Brain1	Full		5920.0	72.08		+
	PSO	66.6	2917.2	77.08	73.73 ± 2.21	+
	CSO	462.1	207.6	86.67	**79.93 ± 3.09**	−
	AMSO	**12.8**	**93.5**	85.00	78.00 ± 3.24	

Table 4: Average test results.

Dataset	Method	Time(m)	Size	Best	Mean ± Std	S
Brain2	Full		10367.0	62.50		+
	PSO	80.5	5117.2	67.08	61.99 ± 2.91	+
	CSO	950.8	90.43	90.83	**80.44 ± 6.28**	−
	AMSO	**12.2**	**62.1**	84.17	74.62 ± 3.87	
Prostate	Full		10509.0	85.33		+
	PSO	160.6	5193.7	88.33	86.00 ± 1.49	+
	CSO	2369.9	357.2	95.17	88.99 ± 2.68	+
	AMSO	**24.3**	**49.9**	95.17	**92.61 ± 1.38**	
Leuk2	Full		11225.0	89.44		+
	PSO	120.6	5535.7	92.22	89.83 ± 1.00	+
	CSO	1845.2	88.6	98.33	91.72 ± 3.16	+
	AMSO	**17.5**	**57.2**	96.67	**94.85 ± 0.99**	
11Tumor	Full		12533.0	71.42		+
	PSO	418.5	6205.0	75.59	71.81 ± 1.75	+
	CSO	6288.6	588.6	84.47	79.52 ± 2.35	+
	AMSO	**94.1**	**324.4**	89.19	**83.86 ± 1.84**	
Lung	Full		12600.0	78.05		+
	PSO	574.2	6234.7	82.72	78.77 ± 1.53	+
	CSO	5565.9	226.4	93.79	87.72 ± 2.93	+
	AMSO	**238.1**	**196.8**	93.46	**89.71 ± 2.24**	

5 RESULTS AND ANALYSIS

Table 5 shows the average results including running time, feature subset size, the best and average test accuracy of KNN (K=1) using AMSO's returned feature subsets from 30 runs. AMSO results are compared with the original feature set ("Full"), PSO and CSO. The last column presents the results of the Wilcoxon statistical test (with 5% significance level) showing whether the proposed method significantly outperforms (+), or has a similar (=) or worse (−) result than the method in the corresponding line. This means that the more "+", the better the proposed method. The smallest running time, feature subset size and the best average test accuracy obtained on each dataset are highlighted in bold.

5.1 AMSO Results

5.1.1 AMSO versus Full. As can be seen from the "Size" column of Table 5, AMSO's feature subsets are two to three orders of magnitude smaller than the original feature sets on all datasets. The highest reduction can be seen on Prostate, where AMSO selects about 50 features from the 10509 features. With a much smaller size, these feature subsets help KNN improve its classification accuracy more than 10% on seven out of the ten datasets with 18.7% highest improvement on 9Tumor. All the "+" signs appeared in the "Full" lines show that using the AMSO selected features, KNN obtains significantly better performance than using the original feature set on all the datasets.

5.1.2 AMSO versus PSO:. Although PSO has reduced the feature set size roughly by half, its feature subsets are still at least two orders of magnitude larger than AMSO's ones on *all* the datasets. When using the feature subsets selected by AMSO, KNN also obtains significantly higher accuracy than using PSO's ones on *all* the datasets with the highest improvement found on Brain1 with 12.6% on average and 17.1% in the best case. For example, while PSO selects 5535 features on Leukemia2, AMSO selects only 55 features, which is a hundred times fewer, to obtain 5% higher test accuracy on average. This shows that AMSO with the proposed mechanisms is much more effective than standard PSO in FS.

5.1.3 AMSO versus CSO:. Compared with CSO, AMSO selects fewer features on all datasets except for DLBCL, obtaining the smallest feature set among the compared methods. On Prostate, AMSO selects less than 50 features on average while CSO selects more than 350 features. In terms of classification accuracy, AMSO obtains significantly better performance on six datasets and similar on one. For example, AMSO selects 19 fewer features on SRBCT than CSO to obtain 6.5% higher accuracy than CSO. On the remaining three datasets, namely 9Tumor, Brain1 and Brain2, CSO obtains significantly higher accuracies than AMSO.

In general, over the 30 comparisons with the three baseline methods, AMSO wins 26, draws 1 and loses 3 in terms of classification accuracy. In terms of the dimensionality reduction, AMSO wins 18 and loses 2 when compared with the two PSO-based FS methods. The results show that AMSO is more effective than standard PSO and CSO in selecting smaller and better feature subsets in almost all cases.

5.1.4 Transformed Data. To further confirm the effect of AMSO in selecting features that can distinguish instances from different classes, we use t-Distributed Stochastic Neighbor Embedding (t-SNE) [14] to visualise the data transformed by each method. t-SNE has shown to be more effective than existing state-of-the-art techniques for visualising high-dimensional data. To apply t-SNE, each dataset was first transformed based on the best feature subset (i.e. giving the highest test accuracy) among 30 different results returned by each method from 30 runs. For datasets with more than 200 features, Principal Component Analysis (PCA) was applied and the first 50 components were used as input of t-SNE [14]. t-SNE was run using the available Python package with perplexity of 20 and 5000 iterations to transform data in 2 dimensions. Due to the page limit, only seven datasets with different number of classes were presented in Figs. 3-9. Each figure has four sub-figures which present the original data (Full), the data transformed by PSO, by

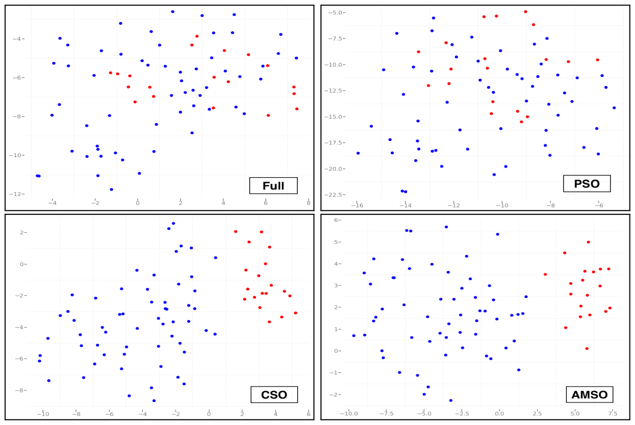

Figure 3: DLBCL (2 classes)

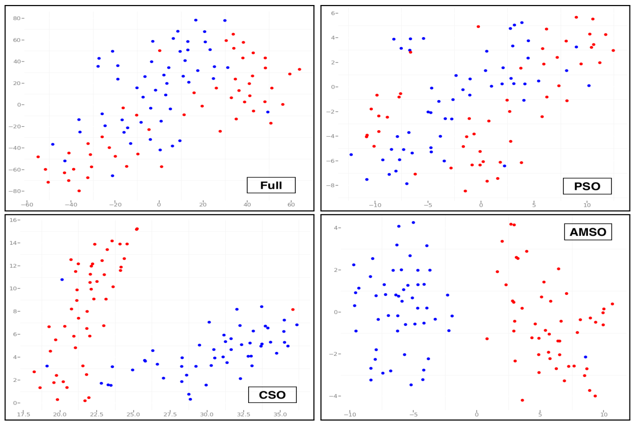

Figure 4: Prostate (2 classes)

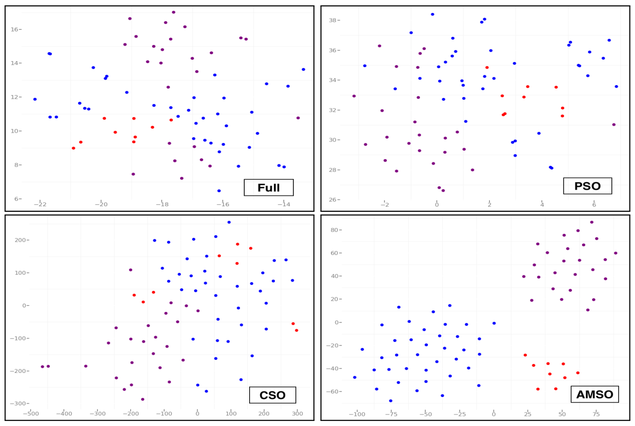

Figure 5: Leukemia1 (3 classes)

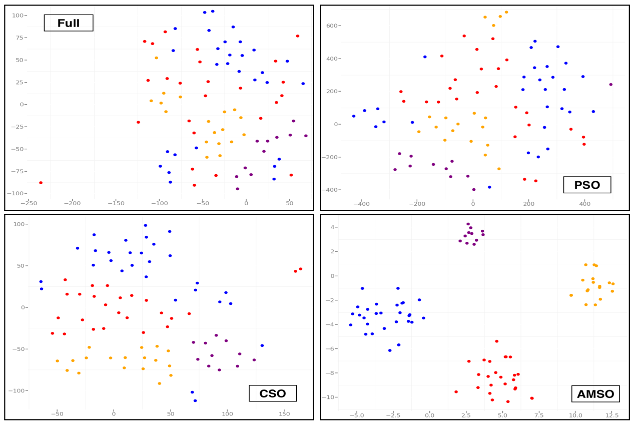

Figure 6: SRBCT (4 classes)

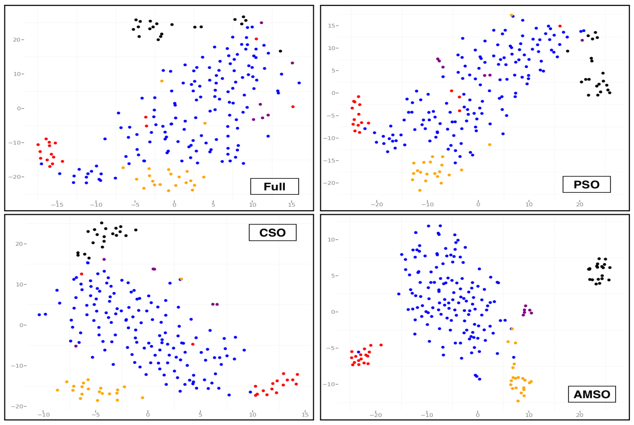

Figure 7: Lung Cancer (5 classes)

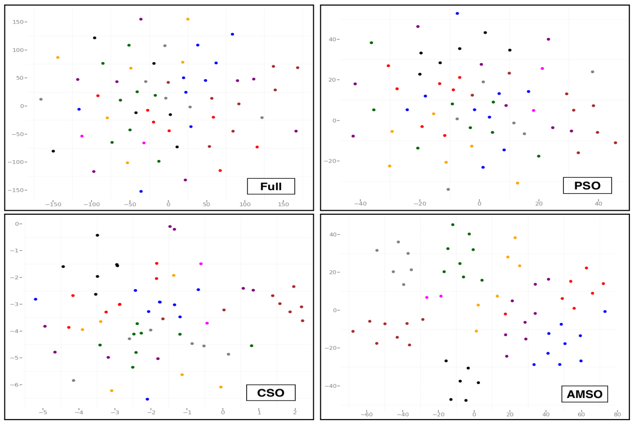

Figure 8: 9Tumor (9 classes)

CSO and by AMSO. In order to show how well each method scales to the complexity of the problem, the results are presented in the ascending order of the number of classes.

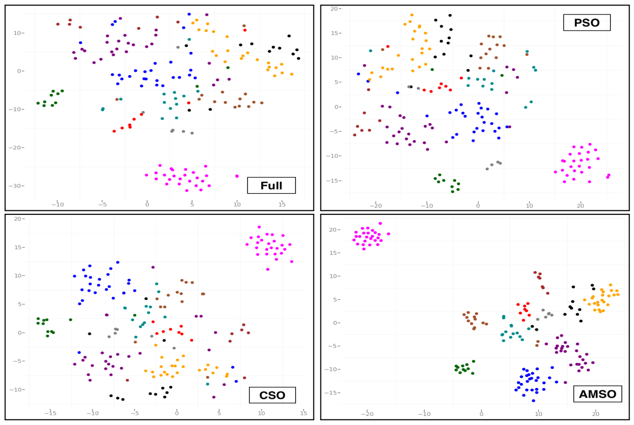

Figure 9: 11Tumor (11 classes)

As seen from Fig. 3, while the two classes of DLBCL have an extensive overlap in the original data and the data transformed by PSO, they are well-separated when using the features selected by CSO and AMSO. This pattern can be seen in all the other datasets. Although KNN obtained the best test accuracy of 92.5% on DLBCL when using features selected by PSO, which is 5% higher than using the original full feature set, the PSO transformed data does not show an obvious difference with the original data. This may be because using the first 50 components transformed by PCA from the 2308 original features and 1119 features selected by PSO, respectively, t-SNE does not have enough information to show the difference between them.

Compared to CSO, the clouds generated by AMSO for different classes usually have smaller size and larger margins between classes, which is clearly shown in Figs.4-7. This may be due to the integration of the distance measure in AMSO's fitness function to bias towards feature subsets that minimise the distance within class and maximise the distance between classes. This strategy helps AMSO attain good performance even on dataset with higher number of classes as in 9Tumor and 11Tumor. As shown in Figs 8 and 9, AMSO selected features provide a better-separated regions for different classes than CSO.

5.2 AMSO Running Time

In terms of the efficiency, as can be seen from Column "Time" of Table 5, AMSO required the shortest running time among the three PSO-based FS methods, while CSO is the slowest with up to 105 times slower than AMSO on Leuk2. This may be due to the strategy of recording the fitness of all the evaluated solutions including feature subsets and its fitness value in a lookup table to avoid re-evaluation. However, using extensive memory and additional time for the lookup table, this strategy may not scale well to larger datasets.

Compared with PSO, AMSO is roughly 2 to 6 times faster. Since both methods use the same fitness function, population size and the number of iterations, they have the same number of PSO evaluations. Note that AMSO spends more evaluations when running local search; therefore, its computation time could be higher. However,

Table 5: AMSO versus traditional methods.

Dataset	Method	Time (s)	Size	Best (Mean) Acc.	*S*
SRBCT	LFS	25.0	**7.1**	91.67	+
	CFS	243.3	112.3	99.17	+
	FCBF	**1.4**	69.0	98.75	+
	AMSO	121.2	63.7	**100.0** (99.75)	
DLBCL	LFS	56.3	**5.9**	83.33	+
	CFS	778.4	86.3	93.00	=
	FCBF	**1.6**	66.1	94.83	−
	AMSO	527.6	50.6	**99.17** (93.70)	
9Tumor	LFS	52.9	**9.7**	26.67	+
	CFS	341.2	44.0	56.67	−
	FCBF	**1.7**	33.7	55.00	=
	AMSO	370.6	52.2	**60.00** (55.33)	
Leuk1	LFS	51.9	**5.4**	85.14	+
	CFS	778.4	79.4	92.08	+
	FCBF	**1.4**	48.5	89.86	+
	AMSO	454.7	51.5	**97.64** (93.50)	
Brain1	LFS	77.9	**12.2**	63.33	+
	CFS	2973.0	151.9	76.67	=
	FCBF	**2.8**	104.6	73.75	+
	AMSO	766.5	93.5	**85.00** (78.00)	

Table 6: AMSO versus traditional methods.

Dataset	Method	Time (s)	Size	Best (Mean) Acc.	*S*
Leuk2	LFS	143.4	**4.7**	89.44	+
	CFS	5653.0	129.5	94.44	+
	FCBF	**4.1**	77.5	95.56	−
	AMSO	1048.6	57.2	**96.67** (94.85)	
Brain2	LFS	113.9	**9.1**	77.50	−
	CFS	3182.2	101.1	77.50	−
	FCBF	**2.7**	66.2	77.50	−
	AMSO	734.9	62.1	**84.17** (74.62)	
Prostate	LFS	158.2	**5.9**	90.17	+
	CFS	2537.4	80.4	92.17	=
	FCBF	**3.4**	66.1	92.17	=
	AMSO	1457.3	49.9	**95.17** (92.61)	
Lung	LFS	358.8	**8.5**	79.62	+
	CFS	85179.1	517.0	**93.76**	−
	FCBF	**56.7**	439.4	92.71	−
	AMSO	14286.2	196.8	93.46 (89.71)	
11Tumor	LFS	309.3	**17.3**	61.76	+
	CFS	57340.7	361.6	80.04	+
	FCBF	**31.1**	349.6	80.57	+
	AMSO	5647.2	324.4	**89.19** (83.86)	

the results show an opposite trend. This means that the computation time saved from having shorter particles, i.e. smaller feature subset sizes, is much more than the time increased by running local search.

5.3 AMSO versus Traditional Methods

Table 6 shows the average running time (in seconds), feature subset size, the best and average (in parenthesis) test accuracy of AMSO compared with LFS, CFS and FCBF. The shortest running time, the smallest feature subset size and the best accuracy are also highlighted in bold.

As can be seen from Column "Size" of Table 6, LFS's subsets are always the smallest among the compared methods on all datasets. However, their classification accuracy is the lowest in almost all cases. Although AMSO selects more features than LFS, it obtains significantly better accuracy than LFS on all datasets except for Brain2 with 3% lower accuracy on average. However, it still obtains 6.7% higher in the best case. On 9Tumor, AMSO selects 43 more features to obtain 29% higher accuracy than LFS on average and 33% higher in the best case. The results show that LFS is trapped in local optima in a very early stage, resulting in a minimal size but low performance feature subsets.

Compared with CFS, AMSO selects a much smaller number of features on almost all datasets to obtain a significantly better performance on four datasets, worse on three and similar on the remaining three datasets. Note that CFS is a filter-based method, which is usually known to be faster than wrapper approaches. However, due to the pair-wise correlation measures, it requires the longest time on all datasets.

In contrast, FCBF has the shortest running time among the compared methods. As can be seen, FCBF and AMSO select a similar number of features in most cases. The average test accuracy of AMSO is significantly better than FCBF on four datasets, worse on four and similar on the remaining two. However, the best accuracy obtained by AMSO on each dataset, meaning that AMSO is always able to find a subset of features that can produce better results than FCBF. For example, AMSO obtain 9% and 12% higher than FCBF on 11Tumor and Brain1, respectively. Note that both FCBF and AMSO start by ranking features based on the same measure. The difference in their performance shows that the heuristic search in the second stage of FCBF is not as powerful as AMSO.

Overall, over the 30 comparisons with the three traditional methods, AMSO wins 17, draws 5, and loses 8. However, on almost all the datasets, AMSO can always find a subset of features that achieve much better accuracy than all these traditional FS methods (see best accuracy in Table 6). The results also show that AMSO scales well to high-dimensional data.

6 CONCLUSIONS

This paper aimed to integrate feature ranking into PSO to combine the strengths of both methods for FS on high-dimensional data. The goal has been achieved by proposing a dynamic PSO algorithm where subswarms are initialised with different particle lengths and automatically change during the evolutionary process based on their performance. The results showed that the proposed method can select a much smaller feature subset with better classification performance in a much shorter time than standard PSO and CSO. By using multiple subswarms to focus on different and smaller subspaces, AMSO not only improved its performance significantly but also reduced its running time, enabling it to scale well to problems with high-dimensionality. AMSO also shows to be more effective than LFS, CFS and FCBF using much shorter running time than CFS in most cases.

Compared with other PSO-based methods, AMSO could be run in parallel, which could further reduce its running time. Wrapper approaches are known to be more effective than filter approaches with the price of longer running time. The ability to run in parallel could help AMSO scalable to big data where both the number of instances and features are large.

Although the proposed dynamic subswarm strategy is proposed for PSO, it can be adapted to other EC methods. In the proposed method, the number of subswarms is fixed during the evolutionary process which may not suitable for all problems. Our future work includes further enhancing AMSO performance by enabling this parameter to be adjusted automatically based on a certain measure on the swarm performance.

REFERENCES

[1] Charu C Aggarwal, Alexander Hinneburg, and Daniel A Keim. 2001. On the surprising behavior of distance metrics in high dimensional space. In *International Conference on Database Theory*. Springer, 420–434.

[2] H. Al-Sahaf, A. Al-Sahaf, B. Xue, M. Johnston, and M. Zhang. 2017. Automatically Evolving Rotation-invariant Texture Image Descriptors by Genetic Programming. *IEEE Transactions on Evolutionary Computation* 21, 1 (2017), 83–101.

[3] Haider Banka and Suresh Dara. 2015. A Hamming distance based binary particle swarm optimization (HDBPSO) algorithm for high dimensional feature selection, classification and validation. *Pattern Recognition Letters* 52 (2015), 94–100.

[4] R. Cheng and Y. Jin. 2015. A Competitive Swarm Optimizer for Large Scale Optimization. *IEEE Transactions on Cybernetics* 45, 2 (Feb 2015), 191–204.

[5] Li-Yeh Chuang, Sheng-Wei Tsai, and Cheng-Hong Yang. 2011. Improved binary particle swarm optimization using catfish effect for feature selection. *Expert Systems with Applications* 38, 10 (2011), 12699–12707.

[6] Chris Ding and Hanchuan Peng. 2005. Minimum redundancy feature selection from microarray gene expression data. *Journal of bioinformatics and computational biology* 3, 02 (2005), 185–205.

[7] Artur J Ferreira and Mário AT Figueiredo. 2012. Efficient feature selection filters for high-dimensional data. *Pattern Recognition Letters* 33, 13 (2012), 1794–1804.

[8] Shenkai Gu, Ran Cheng, and Yaochu Jin. 2018. Feature selection for high-dimensional classification using a competitive swarm optimizer. *Soft Computing* 22, 3 (2018), 811–822.

[9] M. Gutlein, E. Frank, M. Hall, and A. Karwath. 2009. Large-scale attribute selection using wrappers. In *IEEE Symposium on Computational Intelligence and Data Mining*. 332–339.

[10] Mark A. Hall. 2000. Correlation-based Feature Selection for Discrete and Numeric Class Machine Learning. In *Proceedings of the 7th International Conference on Machine Learning*. 359–366.

[11] Indu Jain, Vinod Kumar Jain, and Renu Jain. 2018. Correlation feature selection based improved-Binary Particle Swarm Optimization for gene selection and cancer classification. *Applied Soft Computing* 62 (2018), 203–215.

[12] J. Kennedy and R. Eberhart. 1995. Particle swarm optimization. In *IEEE International Conference on Neural Networks*, Vol. 4. 1942–1948.

[13] Jundong Li, Kewei Cheng, Suhang Wang, Fred Morstatter, Robert P Trevino, Jiliang Tang, and Huan Liu. 2017. Feature selection: A data perspective. *ACM Computing Surveys (CSUR)* 50, 6 (2017), 94.

[14] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of machine learning research* 9, Nov (2008), 2579–2605.

[15] K. Mistry, L. Zhang, S. C. Neoh, C. P. Lim, and B. Fielding. 2017. A Micro-GA Embedded PSO Feature Selection Approach to Intelligent Facial Emotion Recognition. *IEEE Transactions on Cybernetics* 47, 6 (2017), 1496–1509.

[16] M.S. Mohamad, S. Omatu, S. Deris, and M. Yoshioka. 2011. A Modified Binary Particle Swarm Optimization for Selecting the Small Subset of Informative Genes From Gene Expression Data. *Information Technology in Biomedicine* 15, 6 (2011), 813–822.

[17] Parham Moradi and Mozhgan Gholampour. 2016. A hybrid particle swarm optimization for feature subset selection by integrating a novel local search strategy. *Applied Soft Computing* 43 (2016), 117–130.

[18] Alexander Statnikov, Constantin F Aliferis, Ioannis Tsamardinos, Douglas Hardin, and Shawn Levy. 2005. A comprehensive evaluation of multicategory classification methods for microarray gene expression cancer diagnosis. *Bioinformatics* 21 (2005), 631–643.

[19] Binh Tran, Bing Xue, and Mengjie Zhang. 2016. A PSO based hybrid feature selection algorithm for high-dimensional classification. In *Proceedings of IEEE Congress on Evolutionary Computation*. 3801–3808.

[20] B. Xue, M. Zhang, W. N. Browne, and X. Yao. 2016. A Survey on Evolutionary Computation Approaches to Feature Selection. *IEEE Transactions on Evolutionary Computation* 20, 4 (2016), 606–626.

[21] Cheng San Yang, Li Yeh Chuang, Chao Hsuan Ke, and Cheng Hong Yang. 2008. Boolean binary particle swarm optimization for feature selection. In *IEEE Congress on Evolutionary Computation (CEC)*. 2093–2098.

[22] Lei Yu and Huan Liu. 2003. Feature selection for high-dimensional data: A fast correlation-based filter solution. In *Proceedings of the 20th on International Conference on Machine Learning (ICML)*. 856–863.