Using Particle Swarm Optimisation and the Silhouette Metric to Estimate the Number of Clusters, Select Features, and Perform Clustering

Andrew Lensen, Bing Xue, and Mengjie Zhang

School of Engineering and Computer Science,
Victoria University of Wellington, PO Box 600, Wellington 6140, New Zealand
{Andrew.Lensen,Bing.Xue,Mengjie.Zhang}@ecs.vuw.ac.nz

Abstract. One of the most difficult problems in clustering, the task of grouping similar instances in a dataset, is automatically determining the number of clusters that should be created. When a dataset has a large number of attributes (features), this task becomes even more difficult due to the relationship between the number of features and the number of clusters produced. One method of addressing this is feature selection, the process of selecting a subset of features to be used. Evolutionary computation techniques have been used very effectively for solving clustering problems, but have seen little use for simultaneously performing the three tasks of clustering, feature selection, and determining the number of clusters. Furthermore, only a small number of existing methods exist, but they have a number of limitations that affect their performance and scalability. In this work, we introduce a number of novel techniques for improving the performance of these three tasks using particle swarm optimisation and statistical techniques. We conduct a series of experiments across a range of datasets with clustering problems of varying difficulty. The results show our proposed methods achieve significantly better clustering performance than existing methods, while only using a small number of features and automatically determining the number of clusters more accurately.

Keywords: Particle Swarm Optimisation, Clustering,
Feature Selection, Automatic Clustering, Silhouette

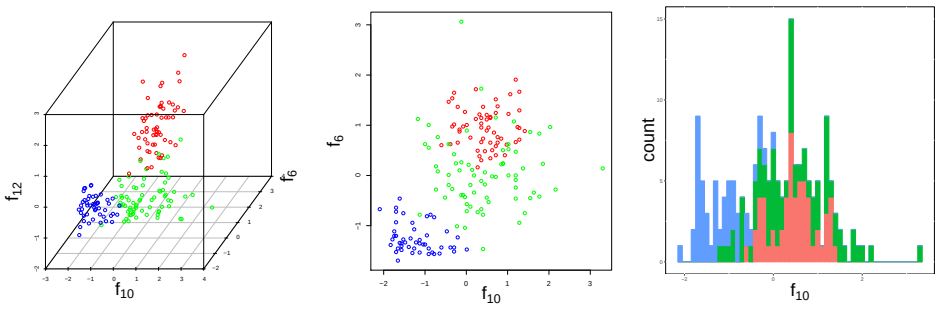
1 Introduction

One of the fundamental challenges in clustering (the task of grouping similar items/instances of a dataset together) is determining the number of clusters (K) to be produced. Many traditional methods such as k-means require K to be pre-defined by the user. Requiring a pre-defined K makes an algorithm less useful, as domain knowledge of a dataset is required to choose an appropriate K. Often, when clustering is used as an exploratory method in the knowledge discovery process, there may be no domain experts available, or there may be no specific goal in mind (so no sensible K can be determined). Hence, there is a need for clustering algorithms that are able to automatically find a suitable K.

Datasets have become increasingly larger in terms of the number of features (m) they contain. Many existing methods such as k-means perform poorly on large feature sets due to the curse of dimensionality. One common technique to reduce dimensionality is feature selection, the process of selecting a subset of features to be used in the data mining process. Performing feature selection removes irrelevant, redundant, and misleading features [1] while reducing the search space size.

Evolutionary Computation (EC) methods are stochastic population-based techniques inspired by natural evolution which produce solutions to difficult problems. EC has been extensively applied to clustering [2] and feature selection [1], but has been rarely used for performing clustering and feature selection simultaneously [3–5]. All of the work in this area uses a single-stage approach where K is either pre-defined [4], or is found during the EC search process [3, 5]. When K is found automatically by the EC method, the search algorithm must optimise three criteria: the number of clusters, the clustering performance, and the number of features used. Such an approach has two significant issues.

The first issue is that a very large search space must be searched on large datasets with many features and instances. If we consider that K is allowed to vary between 2 (a single cluster is not allowed) and K_{max} (often defined as $\sqrt{n}$ for n instances [6]), each value of K is likely to have different optimal feature subsets and different clustering solutions. Even with the powerful population-based search techniques in EC, exploring such a search space thoroughly is difficult.



(a) 3 features: f_{10}, f_6, f_{12}. (b) 2 features: f_{10}, f_6 (c) 1 feature: f_{10}

Fig. 1: Wine dataset projected across varying numbers of features.

There is also an inherent dependency between the number of features selected (m') and K. A larger m' will encourage a larger K and vice versa; the more information (i.e. features) available, the more easily the data can be divided into a larger number of smaller clusters instead of a few big clusters [7]. For example, consider the three plots shown in Fig. 1, which show the Wine dataset (containing three classes, 13 features, and 178 instances) projected using different numbers of features. The three colours represent the three classes of the Wine dataset. When three features are used in Fig. 1a, it is easy to distinguish all three classes as distinct clusters. When one feature is removed in Fig. 1b, the blue class still appears as a homogeneous cluster, however, the red and green classes are much closer and have enough overlap so that they may be considered as a single cluster. When only

a single feature is considered in Fig. 1c, all three classes overlap considerably and it is difficult to choose two thresholds that would split the three classes into three clusters well. As m' is minimised to encourage selecting few features, the evolutionary search will be biased towards picking smaller K, reducing performance on datasets which have large K.

1.1 Goals

In this work, we propose a multi-stage algorithm which extends an existing single-stage approach [5] in order to automatically find K more accurately, improve clustering performance, and decrease the number of selected features. We will:

- Propose a new two-stage approach where an estimate of K (called K_{est}) is computed in the first stage, and then PSO is used to perform clustering, feature selection, and refine K_{est} in the second stage.
- Propose a third stage for fine-tuning the clusters produced in the second stage using a pseudo-local PSO search technique.
- Enhance the existing fitness function [5] to improve feature selection performance and to penalise solutions with K values far away from K_{est}.
- Evaluate our new approach compared to the existing single-stage approach, and to k-means, across a variety of datasets.

2 Background

2.1 Clustering

Clustering is perhaps the most researched of all unsupervised learning tasks [8] with many approaches proposed which are effective on a range of different datasets with different properties and clustering objectives. The most common category of clustering algorithms is partitional clustering algorithms, which divide the dataset into a number of clusters such that each instance lies in exactly **one** cluster. Clusters which are tightly packed and are far away from other clusters are generally regarded to be of high-quality; although other metrics such as connectedness and density have also been used to indicate cluster quality. k-means is the most well-known of the partitional clustering algorithms; it produces compact clusters by repetitively assigning instances to the closest cluster prototype and then recomputing cluster centres to minimise the intra-cluster variance. Other categories of clustering algorithms include density (e.g. DBSCAN), graph-based, and hierarchical (e.g. complete-linkage) algorithms [8].

2.2 Estimating K

A wide range of statistical techniques for estimating the number of clusters (K_{est}) in a given dataset have been proposed [9]. While studies have compared the efficacy of many techniques [9], there is no consensus on the best technique for the general

case. One of the most popular methods is the silhouette criterion, which measures how well a given instance is matched to its cluster. It is defined as follows:

$$Silhouette(i) = \frac{b(i) - a(i)}{max\{a(i), b(i)\}} \tag{1}$$

where $a(i)$ is the average distance between instance i and all other instances in its cluster; $b(i)$ is the *minimum* average distance between instance i and the instances in each other cluster. A silhouette value of 1 indicates an instance is perfectly clustered; a value of -1 indicates it should be in a neighbouring cluster; a value of 0 indicates it is on the border of two clusters. The average silhouette computed across all instances in a partition gives a measure of how good the partition is, and implicitly balances both the intra- and inter-cluster metrics.

The silhouette criterion can be used to give K_{est} by performing clustering for each potential K and then choosing the K for which the average silhouette is highest. This can be computationally expensive due to the need to compute the pair-wise distance between all instances in a dataset. However, this computation only needs to be performed once at the start of the algorithm.

2.3 Feature Selection

EC techniques have been used widely for feature selection [1], with PSO and Genetic Algorithms (GAs) being used for filter, wrapper, and hybrid approaches. Genetic Programming (GP) has also been used for performing embedded feature selection [10]. Wrapper methods are ones which uses a learning algorithm to evaluate the quality of a feature subset and choose the one that gives the highest performance on the learning algorithm. Filter methods take a different approach where the quality of a feature subset is measured more explicitly using a metric such as information gain or entropy [11]. Filter methods tend to give inferior results to wrapper methods, but are usually quicker in terms of computational time required [12]. Hybrid approaches combine both filter and wrapper methods to give better performance than filter methods while being quicker to run than wrapper methods. Embedded approaches perform feature selection as part of the learning algorithm being used, and so can be designed efficiently while being tailored to the algorithm being used; however, they tend to be more problem-specific. While EC has been used extensively for feature selection in classification tasks, little work has used it for clustering, despite clustering generally being regarded as a more difficult task with a larger search space.

2.4 Related Work

NMA_CFS [3] was the first EC method which could simultaneously perform clustering and feature selection while automatically determining K. The authors proposed a single-stage approach using a GA which had a variable length representation based on the number of clusters in a given solution. While this method was shown to give good results, it was only tested on datasets with relatively low K (up to $K = 7$) and small number of features (up to $m = 30$). The variable-length

centroid representation is unlikely to scale well as K becomes larger due to the reasons discussed in [5] which are not repeated here due to space constraints.

Lensen et al. [5] compared a number of medoid- and centroid-based representations for simultaneous clustering and feature selection. It was shown that a medoid representation generally had the best performance over a range of datasets when K was pre-fixed, and also allowed for K to be automatically found by the EC algorithm while maintaining a fixed-length representation (the Dynamic Medoid method, i.e. D-PSO). While the D-PSO method showed promise, it was concluded that it struggled to accurately find K on difficult synthetic datasets which contained a large number of clusters.

Another PSO-based method [4] has also been proposed for simultaneously performing clustering and feature selection when K is pre-fixed. Although the authors used a more advanced fitness function than that of NMA_CFS to improve clustering performance, their reliance on K being known means their approach is not generally comparable to methods which automatically determine K as the latter is a much more difficult problem.

3 The Proposed Method

Our proposed method consists of multiple stages. In the first stage, an estimate of K, called K_{est}, is determined using a statistical measure. The second stage then performs simultaneous clustering and feature selection, while using K_{est} as a guide for finding K. K is still dynamic and so can be optimised by the evolutionary search, but individuals which have a K that varies too far from K_{est} will have their fitness punished correspondingly. As methods used to generate K_{est} in the first stage may not give perfect estimates, allowing minor variations to K_{est} allows the EC method to fine-tune the K value. The (optional) third stage then performs a pseudo-local search using a centroid representation to fine-tune the solution produced by the second stage.

The following subsections discuss the design of each of the stages in detail.

3.1 First Stage

The Silhouette method described in Section 2.2 was used in this study to produce K_{est} in the first stage as it was empirically found to be the most accurate method tested. k-means was used to cluster the data in the first stage for each potential k in the range $[2, \sqrt{n}]$, as suggested in [6], and then the average silhouette for each K was computed. The K with the highest average silhouette is chosen as K_{est}. The silhouette method is non-deterministic and produces a large variation in K_{est} values across different runs. To address this we run the algorithm 30 times and take the median K_{est} to reduce variation, producing more consistent K_{est} values.

3.2 Second Stage

PSO Representation: The output of the first stage is a single K_{est} value. The second stage uses this value as a heuristic to guide the search by PSO for the number of clusters. We use the medoid (an instance that acts as a cluster prototype)

representation introduced in [5], which allows for simultaneous clustering, feature selection and selection of K automatically within a single particle, as shown in Fig. 2. Using such a medoid representation has been shown to give good clustering and feature selection performance, while allowing a fixed-length representation even when K is allowed to vary [5].

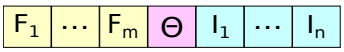


Fig. 2: Medoid representation for simultaneous clustering and feature selection.

The first m dimensions represent whether each of the m features in the dataset is selected. The last n dimensions represent whether each of the n instances is chosen as a medoid. A feature is considered to be selected if its corresponding position value is non-negative. An instance is considered to be a medoid if its position value is greater than Θ, a threshold that is directly encoded (and automatically evolved) in the particle representation. All positions take floating-point values.

Fitness Function: Another novel component of this work is the fitness function to use to evaluate the goodness of solutions produced during the PSO search process. As previously discussed, there are three key criteria required to measure the quality of a given PSO solution: the clustering performance, the number of features used, and how far K deviates from the heuristic K_{est}. As such, we propose that a suitable fitness function should take the following form:

$$\text{Overall Fitness} = \text{Cluster Performance} \times \text{Feature Weighting} \times K \text{ Weighting} \quad (2)$$

We discuss the design of each of the three components below:

(1) Measuring cluster performance: The clustering performance can be measured using many metrics, most of which attempt to minimise the intra-cluster variation while maximising the inter-cluster separation. We use the same metric as in [5], which was shown to give good clustering results. This metric is defined below:

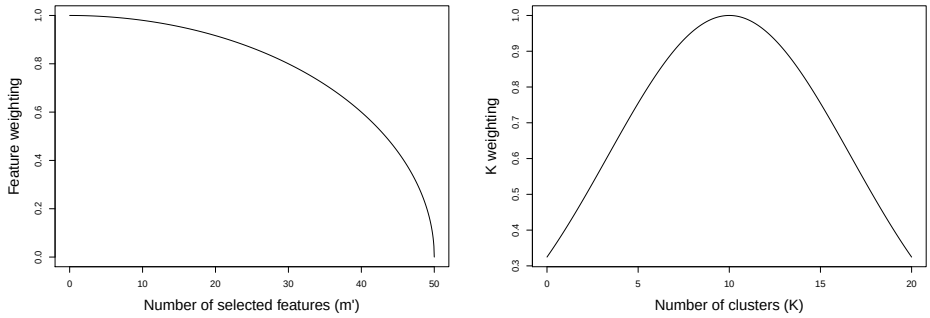
$$\text{Cluster Performance} = \frac{Between_{sum}}{Within_{sum}} \quad (3)$$

$$Within_{sum} = \frac{1}{n}\sum_{i=1}^{K}\sum_{I_a \in C_i} d(I_a, Z_i)^2 \quad (4)$$

$$Between_{sum} = \sum_{i=1}^{K}\frac{|C_i|}{n} d(Z_i, Z^*)^2 \quad (5)$$

where C_i and Z_i represent the i^{th} cluster and the mean of the i^{th} cluster respectively. The dataset mean (Z^*) is the mean across all instances in the dataset. $d(I_a, I_b)$ is the Euclidean distance between two instances I_a and I_b:

$$d(I_a, I_b) = \sqrt{(I_{a1} - I_{b1})^2 + (I_{a2} - I_{b2})^2 + ... + (I_{am} - I_{bm})^2} \quad (6)$$



(a) Elliptical fitness weighting for a dataset containing 50 features.

(b) K fitness weighting for a dataset with K_{est} of 10 using a std. dev. of $\frac{K_{est}}{1.5}$.

Fig. 3: Fitness weightings for balancing the number of features and clusters.

where a_i and b_i give the i^{th} feature value of instances a and b. The distance function considers **all features**, even those that have not been selected by the algorithm. This is done to prevent feature selection introducing a bias towards a low number of clusters (as discussed previously).

(2) Measuring feature selection performance: The most common metric for measuring the goodness of a feature subset is to apply a weighting based on the number of features selected. This is usually expressed as a simple fraction in the form $\frac{m-m'}{m}$ for m total features and m' selected features. Such a weighting applies a linear penalty to the fitness of a given particle with respect to the percentage of features selected. An issue found with this approach is that the search process will tend to over-emphasise minimising m' at the cost of cluster performance — it is usually "easier" to improve fitness by reducing m' than by improving cluster performance. Furthermore, the goal of feature selection is generally to reduce the number of features used to an acceptable level; the user may not differentiate between 5% or 10% features being selected as both values of m' are acceptably small. Hence, a linear weighting mechanism may not be ideal; ideally we would like to apply little penalty when m' is below a threshold and then apply an increasing amount of penalty as m' increases. To achieve this we propose using an elliptical function as shown in Fig. 3a.

The equation used to determine the feature weighting is as follows:

$$\text{Feature weighting} = \frac{1}{m}\sqrt{m^2 - (m')^2} \tag{7}$$

We trial using both this method and the normal linear method for the feature weighting component of the fitness function in our experiments.

(3) Restricting the search space of K: The final decision required is how to penalise particles which have a K value varying significantly from the K_{est} heuristic. As K_{est} is not a perfect estimate, we allow small variations from it without any significant penalty. As the variations increase, we should penalise at a higher rate. The use of a Gaussian function was found to give a good balance of these two

objectives. Fig. 3b shows an example of a Gaussian function with $\mu = K_{est}$ and $\sigma = \frac{K_{est}}{1.5}$ where the output is scaled to give 1 (no penalty) when $K = K_{est}$. As shown, the fitness weighting is small for K values between 8 and 12 or so, but becomes large when K is 5 or 15. The standard deviation must be a function of K_{est} to ensure the function scales effectively; the denominator of 1.5 was chosen empirically. The use of different denominators (e.g. 1 or 2) will increase/decrease the rate of penalty as K varies from K_{est}. We use this Gaussian function as the third component of the fitness function in the second stage.

3.3 Third Stage (pseudo-local search)

One key limitation of a medoid-based representation is that cluster prototypes are restricted to the instances in the dataset. It is possible that better clusters may be formed using cluster prototypes that lie elsewhere in the feature space (e.g. halfway between two instances). To address this limitation, while still maintaining the benefits of a medoid approach, we propose the use of a third-stage where the medoid representation is converted to a centroid representation and then the centroids are fine-tuned using another PSO search process. We call this procedure a "pseudo-local search", as particles are initialised to the best solution found in the second stage, but are allowed to explore the search space freely.

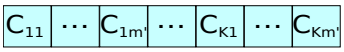


Fig. 4: Centroid representation used in the third stage.

Fig. 4 shows the representation used in the third stage. Each of the K medoids in the best solution from the second stage are used to initialise the position of the particles in the third stage — each medoid is converted to a centroid with length equal to the number of selected features (m') where the centroid contains the feature values for each of the selected features. Each particle's velocity is randomly initialised. Hence, particles will initially spread out in different directions from the second stage's *gbest* before beginning to converge again. It is hoped this will allow fine-tuning of the cluster centres, while focusing the majority of the search in an area which is known to give good performance.

4 Experiment Design

To evaluate the performance of the proposed approach, a number of methods were tested across a range of real-world and synthetic datasets. The methods are:

1. 2-Stage Linear PSO: proposed method using **linear** feature weighting.
2. 2-Stage Elliptical PSO: proposed method using **elliptical** feature weighting.
3. 3-Stage PSO: the 2-Stage Elliptical PSO method plus the third stage (pseudo-local search) for refining the solutions.

4. k_{est}-means: the standard k-means algorithm but using $K = K_{est}$ as computed by the first stage of the proposed approach. This algorithm is used to evaluate how well the proposed approach is able to refine K based on the heuristic and how well it can perform feature selection.
5. k-means: the standard k-means algorithm, initialised with centroids drawing from instances in the dataset. Note that K is **known**, and so this algorithm is being run on a much easier task.
6. D-PSO Scaled: The single-stage medoid approach proposed previously [5].

All methods are non-deterministic and so were run 30 times on each dataset for 500 iterations to ensure search convergence. The PSO methods had a swarm size of 100 and used standard PSO parameters [13]: $w = 0.729844$, $C_1 = C_2 = 1.49618$, and velocity clamped between -6 and 6. A fully connected PSO topology was used; *gbest* after 500 iterations gives the best solution. Feature values were scaled linearly to fall between 0 and 1 based on their minimum and maximum values.

4.1 Evaluation Metrics

We evaluate our proposed methods using two internal metrics (which measure clustering quality based on properties of the clusters produced), and two external metrics (which compare the clusters produced to the known class labels). The internal metrics are the scatter metric, which considers both the within-cluster scatter (S_w, i.e. compactness) and between-cluster variation (S_b, i.e. separability), and the $\sum$ *Intra metric*, which measures the total distance from all instances to their cluster means (i.e. net compactness). The external metrics are class purity, which measures the homogeneity of each cluster using its instance's class labels, and the F-measure, which measures how well pairs of instances agree on their class labels and cluster memberships. Each of these metrics is defined below.

1. Scatter trace criterion:

$$Scatter = trace(S_W^{-1} S_B) \tag{8}$$

$$S_w = \frac{1}{n} \sum_{i=1}^{K} \sum_{I_a \in C_i} (I_a - Z_i)(I_a - Z_i)^T \tag{9}$$

$$S_b = \sum_{i=1}^{K} \frac{|C_i|}{n} (Z_i - Z^*)(Z_i - Z^*)^T \tag{10}$$

   where C_i represents the i^{th} cluster and Z_i and $|C_i|$ are the mean of the i^{th} cluster and the number of instances in the i^{th} cluster respectively. I_a is an instance within cluster C_i. The dataset mean is given by Z^*.
2. Sum intra-cluster distance:

$$\sum Intra = \sum_{i=1}^{K} \sum_{I_a \in C_i} d(I_a, Z_i) \tag{11}$$

3. Class purity: computed according to the following steps:
   (a) For each cluster, find the majority class label of the instances in the cluster.
   (b) Count the number of correctly classified instances, where an instance is correctly classified if it belongs to the majority class.
   (c) Class purity is the fraction of correctly classified instances in the partition.

4. F-measure: We adapt the F-measure used in classification tasks. We consider each pair of instances in turn (it is not possible to directly decide if an instance is in the "right" cluster) and choose which of the following cases apply:
   (a) Same class label, belong to the same cluster: true positive (TP).
   (b) Same class label, belong to the **different** clusters: alse negative (FN).
   (c) **Different** class labels, belong to **different** clusters: true negative (TN).
   (d) **Different** class labels, belong to the same cluster: false positive (FP).
   The F-measure is then calculated in the normal way using the total number of TPs, FPs, and FNs, as follows:

$$\text{F-measure} = 2 \times \frac{precision \times recall}{precision + recall} \tag{12}$$

$$\text{precision} = \frac{TPs}{TPs + FPs} \tag{13}$$

$$\text{recall} = \frac{TPs}{TPs + FNs} \tag{14}$$

4.2 Datasets

To comprehensively evaluate our proposed methods, we selected a variety of real-world and synthetic datasets which are shown in Table 1. The real-world datasets are sourced from the UCI machine learning repository [14], which contains several datasets often used for clustering [3, 5]. These datasets are classification datasets (i.e. class labels are provided), but as is common in the clustering literature, we exclude the class labels from the training process and only use them to evaluate the final cluster against the known classes. The synthetic datasets have been specifically designed for evaluating clustering algorithms, with 10, 20, or 40 clusters and between 1014 and 2893 instances in each dataset. The synthetic datasets with large K (e.g. $K = 40$) and many features (e.g. $m = 100$) are very challenging for traditional methods such as k-means due to the large search space; it is hoped that our proposed methods will show clear improvements on these datasets.

Table 1: Datasets used in the experiments.

Real-World UCI datasets from [14].				Synthetic datasets from [15].			
Name	No. of Features	No. of Instances	No. of Classes	Name	No. of Features	No. of Instances	No. of Classes
Iris	4	150	3	10d10c	10	2730	10
Wine	13	178	3	10d20c	10	1014	20
Movement Libras	90	360	15	10d40c	10	1938	40
				50d10c	50	2699	10
Breast Cancer	9	683	2	50d20c	50	1255	20
				50d40c	50	2335	40
Image Segmentation	18	683	7	100d10c	100	2893	10
				100d20c	100	1339	20
Dermatology	34	359	6	100d40c	100	2212	40

5 Results and Discussion

The results of the experiments are shown in Tables 2 and 3 for the real-world and synthetic datasets respectively. Each table shows the **mean** number of features selected and clusters produced by each method (note that these are constant for k-means) as well as the method's average performance according to a number of evaluation metrics. The $\sum$ *Intra* metric is the only one which should be minimised — it is labelled with a $*$ to indicate this. For each of the proposed methods, each result is labelled with a "+" or a "−" if it is significantly better or worse than the k-means baseline according to a Student's t-test performed with a 95% confidence interval. A lack of a "+" or "−" indicates no significant difference was found. A label of $\uparrow$ or $\downarrow$ indicates a result is significantly better or worse than the existing D-PSO Scaled method [5] according to the same test. The results are analysed on the real-world and synthetic datasets separately in the following subsections, and then some general trends are discussed.

5.1 Real-world Data

Our proposed methods are competitive with k-means on the external metrics and often have superior performance on the internal metrics for the first four real-world datasets, while using a much smaller number of features. The methods also generally outperform D-PSO on the external metrics on four of the six datasets. The proposed methods perform significantly worse than k-means and D-PSO across all metrics on the Image Segmentation dataset due to incorrectly choosing $K = 3$. As the value of K_{est} is 2 on average, PSO is only able to vary K to be 3 without fitness being overly affected. On the Dermatology dataset, the proposed methods achieve a significantly better F-measure value compared to k-means and D-PSO, despite incorrectly estimating K. This is likely due to the estimated K allowing better-formed clusters; on real-world data, class labels are produced by a human expert and may not correspond well to hyper-spherical clusters.

Another important consideration is that a clustering partition that differs from the known classification is not necessarily a "wrong" clustering — there are many ways to group a dataset based on different characteristics (i.e. feature subsets). Hence, it may be better to consider the performance in terms of the internal metrics as a better measurement of how well the proposed approach performs. If we consider the internal metrics, then it is clear that the proposed approach is able to achieve similar to or better results than k-means on the Iris, Wine, Movement Libras, and Breast Cancer datasets, while using a much smaller number of features. On the other two datasets, the performance is far superior to the k_{est}-means method, while again using a small number of features.

To analyse why K was being inaccurately estimated on the Image Segmentation and Dermatology datasets, we visualised these datasets using the principal component analysis (PCA) and t-distributed stochastic neighbour embedding (T-SNE) visualisation methods, as shown in Fig. 5 and 6. Each sub-figure shows the results of applying one of these methods to one of the datasets, where the colours represent the class labels of the dataset.

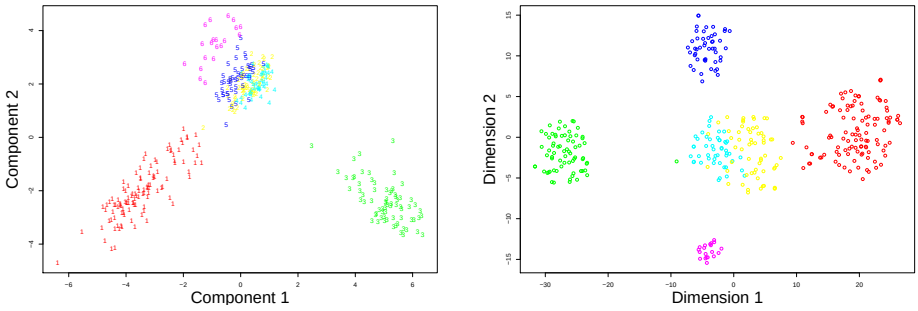
Table 2: Real-world datasets

Dataset	Method	m'	K	Scatter	$\sum$ Intra *	Purity	FM
Iris	2-Stage Linear PSO	1	3	$26.78^{+\downarrow}$	$29.99^{\downarrow}$	$0.9436^{+\uparrow}$	$0.8962^{+\uparrow}$
	2-Stage Elliptical PSO	1	3	$27.4^{+\downarrow}$	$29.96^{\downarrow}$	$0.9493^{+\uparrow}$	$0.9055^{+\uparrow}$
	3-Stage PSO	1	3	$26.9^{+\downarrow}$	$29.99^{\downarrow}$	$0.9449^{+\uparrow}$	$0.898^{+\uparrow}$
	k_{est}-means	4	2	9.8	37.23	0.6667	0.7462
	k-means	4	3	16.37	30.58	0.8404	0.7751
	D-PSO Scaled	1	3.6	35.03	28.29	0.9191	0.8229
Wine	2-Stage Linear PSO	2.27	3.37	$12.53^{\uparrow}$	$89.28^{-\uparrow}$	0.9161^{-}	$0.8152^{-\downarrow}$
	2-Stage Elliptical PSO	3.57	3.4	$13.96^{+\uparrow}$	$88.14^{+\uparrow}$	$0.9418^{\uparrow}$	0.8523^{-}
	3-Stage PSO	3.67	3.4	$14.55^{+\uparrow}$	$87.67^{+\uparrow}$	$0.9541^{+\uparrow}$	$0.8749^{-\uparrow}$
	k_{est}-means	13	2	4.92	104.2	0.6073	0.6357
	k-means	13	3	12.68	88.75	0.9464	0.8947
	D-PSO Scaled	2.27	3	11.14	90.26	0.9167	0.8414
Move. Libras	2-Stage Linear PSO	14.5	17.77	$45.62^{+\uparrow}$	$388.1^{+\uparrow}$	$0.5017^{+\uparrow}$	$0.3423^{\uparrow}$
	2-Stage Elliptical PSO	26.7	17.9	$47.14^{+\uparrow}$	$383.9^{+\uparrow}$	$0.5005^{+\uparrow}$	$0.3501^{\uparrow}$
	3-Stage PSO	26.5	17.73	$49.01^{+\uparrow}$	$380.0^{+\uparrow}$	$0.5012^{+\uparrow}$	$0.3596^{+\uparrow}$
	k_{est}-means	90	12.27	31.28	434.6	0.4226	0.3278
	k-means	90	15	39.01	409.4	0.4705	0.347
	D-PSO Scaled	12.5	6.13	15.21	515.6	0.2859	0.2518
Breast Cancer	2-Stage Linear PSO	1.53	2	$6.062^{-\downarrow}$	$344.2^{-\downarrow}$	0.9407^{-}	$0.8994^{-\uparrow}$
	2-Stage Elliptical PSO	2.5	2	$7.547^{-\downarrow}$	335.6^{-}	$0.9571^{-\uparrow}$	$0.9251^{-\uparrow}$
	3-Stage PSO	2.43	2	$7.807^{-\downarrow}$	334.7^{-}	$0.9573^{-\uparrow}$	$0.9254^{-\uparrow}$
	k_{est}-means	9	2	8.2	332.0	0.9609	0.9313
	k-means	9	2	8.211	332.0	0.9611	0.9316
	D-PSO Scaled	1.6	2.7	10.17	331.0	0.9441	0.8744
Image Seg.	2-Stage Linear PSO	1.33	3	$19.33^{-\downarrow}$	$1245.0^{-\downarrow}$	$0.4251^{-\downarrow}$	$0.4536^{-\downarrow}$
	2-Stage Elliptical PSO	2.13	3	$19.18^{-\downarrow}$	$1242.0^{-\downarrow}$	$0.4275^{-\downarrow}$	$0.4583^{-\downarrow}$
	3-Stage PSO	1.8	3	$19.36^{-\downarrow}$	$1241.0^{-\downarrow}$	$0.4276^{-\downarrow}$	$0.4595^{-\downarrow}$
	k_{est}-means	18	2	4.063	1482.0	0.2857	0.3362
	k-means	18	7	60.86	898.3	0.6426	0.5583
	D-PSO Scaled	2.23	5.23	63.53	984.4	0.6089	0.5725
Derm.	2-Stage Linear PSO	4.53	3.97	68.03^{-}	405.5^{-}	$0.7837^{-\uparrow}$	$0.7886^{+\uparrow}$
	2-Stage Elliptical PSO	7.13	4	$79.26^{-\uparrow}$	$401.6^{-\uparrow}$	$0.8025^{\uparrow}$	$0.8206^{+\uparrow}$
	3-Stage PSO	7.07	4	$81.22^{-\uparrow}$	$400.9^{-\uparrow}$	$0.8083^{\uparrow}$	$0.8311^{+\uparrow}$
	k_{est}-means	34	2.73	55.69	457.3	0.612	0.6113
	k-means	34	6	93.58	387.6	0.8278	0.7351
	D-PSO Scaled	4.37	3.8	68.77	409.4	0.7602	0.7587

On the Dermatology dataset (which has 6 classes), PCA clearly separates the red and green classes into two distinct clusters. The remaining classes appear as one tightly packed cluster, with the pink class potentially a fourth cluster. This gives 3–4 clusters on this dataset, consistent with the 4 average clusters found by the proposed methods. T-SNE more clearly separates the classes into clusters, but there is still overlap between the teal and yellow classes, giving 5 distinct clusters.

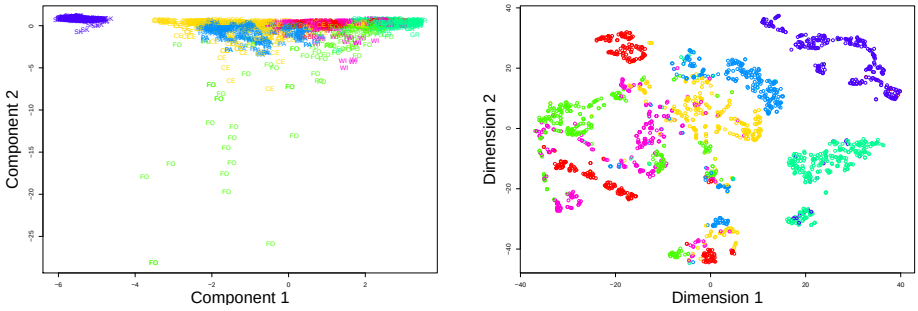
The visualisations on the Image Segmentation dataset are much more unclear; PCA produce a poor visualisation, with only the purple class being clearly separated. T-SNE is clearer — the aquamarine and purple classes are separated well, but the remaining classes all have a fair amount of overlap. This suggests why the proposed methods find 3 clusters — two of the classes fit into two clusters well, and the remaining instances have sufficient overlap to produce a single cluster.

In summary, both the linear and elliptical two-stage methods successfully select a small m' on all real-world datasets. The elliptical method has slightly better performance than the linear method while selecting additional features. If the minimum number of features is desired while achieving good performance, then the linear method is best; however, if better performance is preferred at the cost



(a) PCA (b) T-SNE

Fig. 5: Visualisations of Dermatology dataset.



(a) PCA (b) T-SNE

Fig. 6: Visualisations of Image Segmentation dataset.

of slightly higher complexity, the elliptical method should be used. The 3-stage method has slightly higher performance than the 2-stage elliptical method across all metrics, which shows the pseudo-local search is able to further refine solutions.

5.2 Synthetic Data

Unlike the real-world datasets, the synthetic datasets have classes which map well to hyper-spherical clusters. Thus the external metrics are useful for measuring the performance of the proposed approach, which clearly outperforms k-means and k_{est}-means on all of the synthetic datasets (except 10d10c) while achieving a low m', especially on the datasets with high m. The proposed approach scales to large datasets more effectively than the k-means algorithm, despite not performing only clustering, but also feature selection and determining K in the same search process. It also performs better than k-means on the internal metrics across the 50d and 100d datasets, where it selects the most useful features to improve clustering.

Despite performing well, the proposed methods are inaccurate in predicting K on several of the synthetic datasets, such as 50d20c and 100d20c where they select $35 - 36$ clusters instead of 20. However, compared to the D-PSO method, the proposed methods predict K more accurately on 7 of the 9 datasets. The D-PSO method predicts values of K close to 20 on all the 50d and 100d datasets

Table 3: Synthetic datasets

Dataset	Method	m'	K	Scatter	$\sum$ Intra *	Purity	FM
10d10c	2-Stage Linear PSO	3.63	8.97	$15.04^{-\uparrow}$	$782.6^{-\uparrow}$	$0.8051^{-\uparrow}$	0.763^{-}
	2-Stage Elliptical PSO	4.93	9.57	$15.63^{-\uparrow}$	$750.0^{-\uparrow}$	$0.8604^{-\uparrow}$	$0.8196^{\uparrow}$
	3-Stage PSO	4.9	9.07	$16.11^{-\uparrow}$	$739.9^{-\uparrow}$	$0.8872^{-\uparrow}$	$0.8678^{+\uparrow}$
	k_{est}-means	10	5.43	11.49	815.9	0.7743	0.7786
	k-means	10	10	17.7	715.7	0.9175	0.833
	D-PSO Scaled	3.3	6.97	12.63	817.7	0.7718	0.7481
10d20c	2-Stage Linear PSO	5.07	20.1	$75.32^{+\uparrow}$	$226.9^{+\uparrow}$	$0.9587^{+\uparrow}$	$0.9408^{+\uparrow}$
	2-Stage Elliptical PSO	6.1	20.17	$85.2^{+\uparrow}$	$216.8^{+\uparrow}$	$0.9828^{+\uparrow}$	$0.9721^{+\uparrow}$
	3-Stage PSO	6.07	20	$90.17^{+\uparrow}$	$213.3^{+\uparrow}$	$0.9953^{+\uparrow}$	$0.9928^{+\uparrow}$
	k_{est}-means	10	15.13	53.31	292.7	0.7907	0.7651
	k-means	10	20	70.02	248.5	0.8887	0.8218
	D-PSO Scaled	4.4	14.7	51.65	282.7	0.8249	0.8305
10d40c	2-Stage Linear PSO	5.47	39.73	$67.26^{-\uparrow}$	$452.0^{-\uparrow}$	$0.9182^{\uparrow}$	$0.8699^{\uparrow}$
	2-Stage Elliptical PSO	6.87	39.7	$78.29^{+\uparrow}$	$417.3^{+\uparrow}$	$0.9615^{+\uparrow}$	$0.9437^{+\uparrow}$
	3-Stage PSO	6.87	40.1	$85.66^{+\uparrow}$	$402.9^{+\uparrow}$	$0.9824^{+\uparrow}$	$0.97^{+\uparrow}$
	k_{est}-means	10	29.83	55.58	499.9	0.8385	0.8234
	k-means	10	40	74.5	433.7	0.9219	0.8657
	D-PSO Scaled	3.83	15.6	27.08	756.4	0.5692	0.5477
50d10c	2-Stage Linear PSO	9.3	13.5	$93.48^{+\downarrow}$	$1072.0^{+\downarrow}$	$0.8191^{+\downarrow}$	0.5197^{+}
	2-Stage Elliptical PSO	14.87	13.23	$96.76^{+\downarrow}$	$1071.0^{+\downarrow}$	$0.8174^{+\downarrow}$	0.5172^{+}
	3-Stage PSO	14.23	13.5	$104.5^{+\downarrow}$	$1045.0^{+\downarrow}$	$0.8152^{+\downarrow}$	0.5125^{+}
	k_{est}-means	50	11.53	88.84	1242.0	0.7679	0.4978
	k-means	50	10	72.87	1306.0	0.7426	0.4865
	D-PSO Scaled	9.07	18.5	144.2	930.6	0.8824	0.5196
50d20c	2-Stage Linear PSO	10.87	34.63	$250.2^{+\uparrow}$	$372.4^{+\uparrow}$	$0.8622^{+\uparrow}$	$0.525^{+\uparrow}$
	2-Stage Elliptical PSO	17.43	34.67	$261.8^{+\uparrow}$	$366.1^{+\uparrow}$	$0.8692^{+\uparrow}$	$0.5171^{+\uparrow}$
	3-Stage PSO	17.33	34.6	$283.1^{+\uparrow}$	$356.8^{+\uparrow}$	$0.858^{+\uparrow}$	$0.4835^{+\uparrow}$
	k_{est}-means	50	28.8	211.3	432.5	0.8057	0.4713
	k-means	50	20	137.5	548.6	0.6858	0.3581
	D-PSO Scaled	10.63	17.43	99.65	539.6	0.7165	0.4167
50d40c	2-Stage Linear PSO	13.57	48	$200.4^{+\uparrow}$	$756.9^{+\uparrow}$	$0.7724^{+\uparrow}$	$0.4888^{+\uparrow}$
	2-Stage Elliptical PSO	19.87	48	$211.7^{+\uparrow}$	$738.8^{+\uparrow}$	$0.788^{+\uparrow}$	$0.4949^{+\uparrow}$
	3-Stage PSO	20.43	48	$230.3^{+\uparrow}$	$708.5^{+\uparrow}$	$0.761^{+\uparrow}$	$0.3669^{+\uparrow}$
	k_{est}-means	50	44.7	214.5	822.6	0.7099	0.2841
	k-means	50	40	192.4	871.9	0.6749	0.2586
	D-PSO Scaled	11.33	26.03	104.5	1062.0	0.5546	0.2455
100d10c	2-Stage Linear PSO	20	15.87	$148.3^{+\downarrow}$	$1401.0^{+\downarrow}$	$0.8655^{+\downarrow}$	0.5648^{+}
	2-Stage Elliptical PSO	29.4	15.77	$150.9^{+\downarrow}$	$1395.0^{+\downarrow}$	$0.8676^{+\downarrow}$	$0.5707^{+\uparrow}$
	3-Stage PSO	28.9	15.73	$153.4^{+\downarrow}$	$1376.0^{+\downarrow}$	$0.8567^{+\downarrow}$	0.5557^{+}
	k_{est}-means	100	11.27	115.2	1807.0	0.794	0.5623
	k-means	100	10	103.5	2036.0	0.7436	0.5194
	D-PSO Scaled	17.3	18.97	206.1	1287.0	0.9137	0.5549
100d20c	2-Stage Linear PSO	21.9	36	$358.9^{+\uparrow}$	$559.9^{+\uparrow}$	$0.8945^{+\uparrow}$	$0.5627^{+\uparrow}$
	2-Stage Elliptical PSO	33.4	36	$381.9^{+\uparrow}$	$548.2^{+\uparrow}$	$0.8943^{+\uparrow}$	$0.5466^{+\uparrow}$
	3-Stage PSO	34.17	35.93	$399.9^{+\uparrow}$	$536.1^{+\uparrow}$	$0.8858^{+\uparrow}$	$0.5256^{+\uparrow}$
	k_{est}-means	100	26.2	260.4	707.5	0.7865	0.4505
	k-means	100	20	188.1	841.0	0.7011	0.3799
	D-PSO Scaled	19.53	20.23	162.7	748.6	0.7568	0.4435
100d40c	2-Stage Linear PSO	27.37	47	$304.4^{\uparrow}$	$991.7^{+\uparrow}$	$0.7968^{+\uparrow}$	$0.5116^{+\uparrow}$
	2-Stage Elliptical PSO	39.07	47	$315.8^{+\uparrow}$	$980.4^{+\uparrow}$	$0.8057^{+\uparrow}$	$0.5012^{+\uparrow}$
	3-Stage PSO	37.9	47	$338.6^{+\uparrow}$	$940.5^{+\uparrow}$	$0.7624^{+\uparrow}$	$0.3305^{+\uparrow}$
	k_{est}-means	100	42.87	332.0	1146.0	0.7073	0.2837
	k-means	100	40	301.4	1176.0	0.6923	0.2689
	D-PSO Scaled	22.47	23	142.7	1442.0	0.5426	0.1997

despite K actually varying from 20 to 40. This suggests that D-PSO cannot search for the true K value as effectively as the proposed methods. It is interesting to note that the K values produced by the proposed methods are always higher than the K_{est} generated by the first stage.This suggests that the fitness function

encourages a higher number of clusters, perhaps due to the clustering performance metric used. This behaviour is useful in some cases, where K_{est} is below the actual K (e.g. 10d10c, 10d20c and 10d40c), but on the other synthetic datasets where $K_{est} > K_{actual}$, it means that the proposed methods are not able to correctly lower the K found. It would be useful to investigate changing the clustering performance metric in the fitness function to encourage searching values on both sides of K_{est}.

5.3 Further Analysis

The two variations of the two-stage approach perform better on different datasets: the elliptical approach is best on datasets with small feature sets (Wine, Breast Cancer, Dermatology, 10d) where the linear fitness function selects fewer features at the expense of cluster quality. On the larger feature sets, the elliptical approach selects extra features without increasing performance. This is due to the ellipse used: on large feature sets, e.g. the 50 features seen in Fig. 3a, the feature weighting is close to 1 for m between 0 and 10, and above 0.9 even when 20 features are selected — indeed, on the synthetic datasets with 50 features, this method selects 15 to 20 features. On smaller feature sets, this effect is less noticeable, as only a few features are able to be selected before the feature weighting decreases significantly. Investigating a way of dynamically altering the shape of the ellipse used based on the size of the feature set would ensure that the weighting "drop-off" begins earlier on bigger datasets. The two-stage methods have similar values of K across all datasets, indicating that neither is being overly affected by the correlation between m' and K. If this had occurred, the elliptical approach would have higher K on the datasets where it selected more features than the linear approach.

The three-stage approach is an improvement compared to the two-stage elliptical approach on the internal metrics across all of the datasets (and especially on the hardest synthetic ones). This suggests that fine-tuning the solutions produced with a pseudo-local search is effective, increasing cluster quality. However, the results on the external metrics are much worse for the three-stage approach on the 50d40c and 100d40c synthetic datasets, contrary to that of the internal metrics. It is not obvious why this occurs — one explanation is that the noise present in the synthetic datasets has a significant effect when K is large (i.e. $K = 40$); as a centroid can take any possible co-ordinates (unlike a medoid), it may be much more sensitive to the noise in the dataset, producing overly specific clusters.

6 Conclusion

This work introduced a comprehensive and coherent two-/three-stage approach for performing simultaneous clustering and feature selection while automatically finding the K required. A number of novel techniques were proposed, including: using an estimate of K, K_{est}, to guide the PSO search process; a multi-faceted fitness function which encourages good cluster quality, minimises m', and reduces the search space of K; and a pseudo-local search which refines the clusters produced by the second stage. We showed that our approach gave good performance across several difficult datasets compared to existing methods, while selecting as few features as needed. In particular, our approach was shown to be effective on

datasets with large K, which existing approaches largely fail to address. Our approach was successful at reducing dimensionality on large feature sets, consistently selecting under 40% of features on datasets with 100 features and 10 to 40 clusters.

In future work, we will further refine our fitness function by taking a multi-objective approach in order to allow more intelligent balancing of all three of cluster performance, feature selection, and deduction of K. We would also like to investigate other methods of measuring cluster performance (perhaps using multi-objective techniques), such as connectedness or density. There is also scope for improving performance further with other methods for estimating K, penalising the number of features produced, or using other EC techniques or representations.

References

1. Xue, B., Zhang, M., Browne, W.N., Yao, X.: A survey on evolutionary computation approaches to feature selection. IEEE Trans. Evolutionary Computation **20**(4) (2016) 606–626
2. García, A.J., Gómez-Flores, W.: Automatic clustering using nature-inspired metaheuristics: A survey. Appl. Soft Comput. **41** (2016) 192–213
3. Sheng, W., Liu, X., Fairhurst, M.C.: A niching memetic algorithm for simultaneous clustering and feature selection. IEEE Trans. Knowl. Data Eng. **20**(7) (2008) 868–879
4. Javani, M., Faez, K., Aghlmandi, D.: Clustering and feature selection via PSO algorithm. In: International Symposium on Artificial Intelligence and Signal Processing (AISP), IEEE (2011) 71–76
5. Lensen, A., Xue, B., Zhang, M.: Particle swarm optimisation representations for simultaneous clustering and feature selection. In: Proceedings of the Symposium Series on Computational Intelligence, IEEE (2016) (To appear).
6. Pal, N.R., Bezdek, J.C.: On cluster validity for the fuzzy c-means model. IEEE Trans. Fuzzy Systems **3**(3) (1995) 370–379
7. Alelyani, S., Tang, J., Liu, H.: Feature selection for clustering: A review. In: Data Clustering: Algorithms and Applications. (2013) 29–60
8. Aggarwal, C.C., Reddy, C.K., eds.: Data Clustering: Algorithms and Applications. CRC Press (2014)
9. Chiang, M.M., Mirkin, B.G.: Intelligent choice of the number of clusters in K-means clustering: An experimental study with different cluster spreads. J. Classification **27**(1) (2010) 3–40
10. Muni, D.P., Pal, N.R., Das, J.: Genetic programming for simultaneous feature selection and classifier design. IEEE Trans. Systems, Man, and Cybernetics, Part B **36**(1) (2006) 106–117
11. Guyon, I., Elisseeff, A.: An introduction to variable and feature selection. Journal of Machine Learning Research **3** (2003) 1157–1182
12. Liu, H., Motoda, H., Setiono, R., Zhao, Z.: Feature selection: An ever evolving frontier in data mining. In: Proceedings of the Fourth International Workshop on Feature Selection in Data Mining. (2010) 4–13
13. Van Den Bergh, F.: An analysis of particle swarm optimizers. PhD thesis, University of Pretoria (2006)
14. Lichman, M.: UCI machine learning repository (2013)
15. Handl, J., Knowles, J.D.: An evolutionary approach to multiobjective clustering. IEEE Trans. Evolutionary Computation **11**(1) (2007) 56–76