Particle Swarm Optimisation Representations for Simultaneous Clustering and Feature Selection

Andrew Lensen, Bing Xue, Mengjie Zhang
School of Engineering and Computer Science, Victoria University of Wellington, New Zealand
{andrew.lensen, bing.xue, mengjie.zhang}@ecs.vuw.ac.nz

***Abstract*—Clustering, the process of grouping unlabelled data, is an important task in data analysis. It is regarded as one of the most difficult tasks due to the large search space that must be explored. Feature selection is commonly used to reduce the size of a search space, and evolutionary computation (EC) is a group of techniques which are known to give good solutions to difficult problems such as clustering or feature selection. However, there has been relatively little work done on simultaneous clustering and feature selection using EC methods. In this paper we compare medoid and centroid representations that allow particle swarm optimisation (PSO) to perform simultaneous clustering and feature selection. We propose several new techniques which improve clustering performance and ensure valid solutions are generated. Experiments are conducted on a variety of real-world and synthetic datasets in order to analyse the effectiveness of the PSO representations across several different criteria. We show that a medoid representation can achieve superior results compared to the widely used centroid representation.**

I. Introduction

Clustering is an important part of exploratory data mining which aims to group similar items (instances) of a dataset together into a number of *natural* groups (clusters) [1]. Unlike other common data mining tasks such as classification and regression, clustering is an unsupervised learning task; there are generally no provided labels for the data that can be used for training. Instead, the performance of a clustering solution (partition) is measured by using one or more of a wide range of metrics which give an indication of the quality of a partition [2], [3]. Furthermore, the clustering problem has been shown to be NP-hard when there are more than two clusters [4].

Feature selection (FS), the technique of selecting a subset of effective features (i.e. data attributes) for use in the learning process, is used widely in both supervised and unsupervised learning [5]. Using only a subset of features can improve performance by removing redundant, irrelevant, or even misleading features [6] as well as reducing the search space size. In addition, solutions which use fewer features are generally more comprehensible and thus give a better understanding of the data being explored and solutions produced [7].

Evolutionary Computation (EC) [8] is a field of Artificial Intelligence which contains algorithms that use nature-inspired evolutionary principles to perform a population-based search. EC techniques have been widely applied to NP-hard problems due to their ability to produce high-quality solutions in reasonable computational time [8]. While EC has been widely used for clustering [1], [3], only a few methods have been proposed which simultaneously perform clustering and FS (herein notated as C+FS). Performing both tasks concurrently allows the minimal number of useful features to be chosen for a particular partition. The existing methods use a centroid representation which may limit performance, and have been tested only on datasets with a small number of clusters.

In this paper, we investigate a variety of representations for performing C+FS using Particle Swarm Optimisation (PSO), a commonly used EC method for clustering and FS tasks [6], [9]. In particular, we aim to:

- Present and compare extensions of the centroid and medoid clustering representations which perform C+FS when the number of clusters (K) is known in advance,
- Investigate techniques to improve the centroid representation to prevent invalid solutions being generated,
- Propose a novel technique which uses a medoid representation to find K when it is not pre-defined, and
- Evaluate these techniques across real-world and synthetic datasets, for a range of feature set and dataset sizes.

II. Background

This section begins with an overview of the core concepts involved in PSO, feature selection, and clustering. We then discuss previous work which uses EC techniques for C+FS.

A. PSO

PSO is an EC technique inspired by bird flocking behaviour [10]. In PSO, the swarm (population) consists of a number of particles (individuals) which explore the search space based on their personal experience as well as the swarm's social knowledge of good areas of the search space. Each particle encodes a single solution to the problem being optimised as a position vector which represents the particle's position in the search space. This position vector takes the form $[x_1, x_2, ..., x_D]$, where D is the number of dimensions in the search space. The velocity of a particle encodes its movement through the search space and is represented using a vector of the form $[v_1, v_2, ..., v_D]$. During each iteration of the PSO search process, a particle's velocity is updated based on its previous best position ($pbest$) and the swarm's overall best known position ($gbest$). The particle then updates its position based on its current velocity, giving an updated position and hence a new candidate solution. The position and velocity updates are formally defined as follows:

$$x_{id}^{t+1} = x_{id}^{t} + v_{id}^{t+1} \tag{1}$$

$$v_{id}^{t+1} = w \times v_{id}^{t} + c_1 r_1 \times (p_{id} - x_{id}^{t}) + c_2 r_2 \times (p_{gd} - x_{id}^{t}) \quad (2)$$

where t represents the t^{th} iteration of the PSO process and $d \in D$ represents the d^{th} dimension of the search space. The inertia weight, w, is used to balance the exploration and exploitation behaviour of the particles. c_1 and c_2 are acceleration constants which balance the contributions of the $pbest$ and $gbest$ positions respectively. r_1 and r_2 are randomly generated values in the range $[0, 1]$ which introduce variance to the training process. p_{id} and p_{gd} are used to represent the values of $pbest$ and $gbest$ in the d^{th} dimension. The updated velocity, v_{id}^{t+1}, is restricted to the range $[-v_{max}, v_{max}]$ for some maximum velocity, v_{max}, in order to limit oscillation.

B. Feature Selection

There have been a large number of methods proposed for performing FS, which can generally be classified into one of three categories: filter, wrapper, or embedded approaches [7]. Filter methods perform feature selection as a pre-processing step, independent of the learning algorithm being used. This is the most commonly used approach in traditional clustering methods (e.g. k-means, DBSCAN) as it is the easiest to implement and generally has the lowest computational cost. Wrapper methods utilise the learning algorithm to evaluate the quality of the selected feature subset by running the learning algorithm multiple times using different feature subsets. While wrapper approaches generally give better results, they are much more expensive to run [11]. Embedded approaches perform feature selection directly as part of the training process in the learning algorithm. Embedded approaches can therefore achieve high quality results during only a single run of the learning algorithm, but are generally more difficult to design and specific to the learning algorithm being used.

EC techniques have also been widely used for FS [6] with genetic algorithms (GAs) and PSO being used for both filter and wrapper approaches. These two algorithms are both well-suited for FS tasks as they use a vector encoding, where each feature can be represented by a single value which determines if that feature is selected. Other EC algorithms such as Genetic Programming (GP) and Ant Colony Optimisation (ACO) have also seen some use for FS tasks [6].

C. Clustering

Clustering is a diverse problem which has produced many different types of algorithms that are effective on different domains and datasets [12]. Hierarchical clustering algorithms (e.g. single- or complete-linkage clustering) build a hierarchy of clusters, merging or splitting clusters at different levels of the hierarchy. Density-based clustering algorithms (e.g. DBSCAN) define clusters as being regions of high density and label instances in sparse regions as outliers [12]. Prototype-based clustering algorithms such as k-means use a single feature-vector (called a "prototype") to represent the centre of a cluster and then perform clustering by assigning each instance in the dataset to the cluster with the closest prototype [12]. Other algorithms using distribution, subspace or graph-based models have also been proposed [12].

Most EC techniques have focused on performing hard partitional clustering (where every instance is assigned to exactly one cluster) [3], [13] with the majority of techniques using a prototype-based approach. Clustering performance is generally measured in terms of connectedness (how well similar instances are assigned to the same cluster), compactness (how densely packed a cluster is) or separability (how well clusters are separated from each other) [13].

The *centroid* representation, where the co-ordinates of the cluster prototypes are directly encoded in the solution representation, is the most commonly used by PSO and GA methods for prototype-based clustering. Such an encoding contains K sets of m co-ordinates (where m is the number of features), giving K centroids. Each instance is assigned to the closest prototype using a distance measure (e.g. Euclidean distance). A less commonly used representation is the *medoid*, where instances are used as cluster prototypes. Each instance in the dataset is assigned a value in the solution encoding, and an instance's value determines if it is a cluster prototype. As before, instances are assigned to their nearest prototype.

A number of factors must be considered when performing clustering; we discuss the key issues in this study below.

1) Representations: The centroid and medoid representations each have their share of benefits and limitations. The centroid representation scales well as the number of instances in a dataset grows as Km is independent of n, but may give poor performance when K or m is large compared to n, in which case the medoid representation may be more effective. The medoid representation is restricted to choosing cluster prototypes which are instances in the dataset [13]. As a result, it may not be able to perform exploitation (i.e. local searching) as effectively as the centroid representation, which can produce centroids located at any point in the search space. This is less likely to be an issue on datasets where there is a large number of instances as there will be a higher level of granularity in terms of the medoids available.

It is possible for the centroid representation to produce invalid solutions when a cluster is not assigned any instances. This occurs if all instances have found closer prototypes. This behaviour is especially prevalent on datasets with high K where it is proportionally less likely that a cluster will have any instances assigned to it. This problem does not occur with a medoid representation; at a minimum, the instance acting as a medoid must be assigned to its own cluster.

2) Determining K: Clustering methods can also be broadly categorised based on whether they require K to be pre-defined, or if they are able to automatically determine an acceptable K in the learning process. While the latter case is more flexible and hence is generally more useful, it may happen that a domain expert is available and is able to determine an optimum K, in which case the former methods are likely to perform better as they search a much smaller search space.

3) Simultaneous Clustering and Feature Selection: While a range of EC techniques have been proposed for either clustering or FS, only a few of these techniques perform simultaneous clustering and FS. All of these techniques use

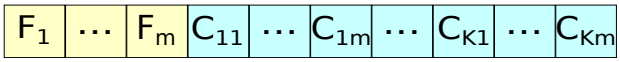


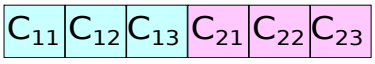
Fig. 1: Centroid representation for simultaneous clustering and feature selection.

a centroid encoding [14], [15] to perform clustering, and a binary encoding for performing FS. An example of such a representation is shown in Figure 1 for m features and K clusters. The first m dimensions correspond to the m features in the dataset. The value for a given feature determines if it has been selected. The remaining Km dimensions correspond to the co-ordinates of the K cluster prototypes.

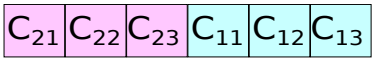
D. Related Work

One of the most influential works in the EC domain for C+FS is the NMA_CFS [14] method, which uses a GA to simultaneously perform FS, find K, and cluster the dataset. The authors use niching and local search techniques to improve the performance and consistency of the method. The niching approach requires a number of parameters to be set in order to optimise performance. NMA_CFS uses a variable-length encoding where the solution size is relative to the number of clusters produced. While the method produced impressive results, the datasets used for testing contained a small number of clusters (a maximum of 6 and 7 in the synthetic and real-world datasets respectively) and a small number of features (m) with a maximum of 20 and 30 on synthetic and real-world datasets respectively. In practice, datasets may have many more clusters and features; it is not obvious how well NMA_CFS would scale as K and m increase.

The use of a variable-length encoding also reduces training effectiveness due to the inability of candidate solutions to fully exchange information when they vary in length. A centroid encoding also introduces an implicit ordering to the clusters in the solution when clusters do not have any ordering in reality. For example, consider the two solutions shown in FIgure 2 – both solutions will produce the same cluster partition, but as their two centroids are in different orders, the training process will incorrectly attempt to move each solution towards the other. In a GA approach, for example, this could mean that crossover produces a solution which contains conflicting or even duplicate centroids.



(a) Solution A



(b) Solution B

Fig. 2: Two centroid solutions with different centroid orderings which produce the same partition.

Javani et. al. proposed a PSO-based method which uses a similar centroid representation to the NMA_CFS [14] approach, but requires K to be pre-fixed [15]. A fitness function was proposed which considers a partition's connectedness, compactness, and separability. The proposed method was shown to generally perform better than NMA_CFS [14] on the synthetic datasets. However, NMA_CFS [14] is able to be used when K is unknown which likely makes it a more useful method in the general case.

In addition to the above work which uses EC techniques to solve the combined C+FS problem, there has also been a number of publications which use PSO for clustering **or** FS and use a non-EC technique to perform the other task. Marinakis et al. [16] proposed a two-phase approach using PSO for feature selection and a greedy randomised adaptive search procedure (GRASP) for clustering. Their second paper [17] uses a multi-swarm version of PSO using a constriction factor in order to improve the feature selection performance compared to the first paper [16]. Kuo et al. [18] used Principle Component Analysis (PCA) to perform dimensionality reduction before using a hybrid GA-PSO approach to perform clustering for an unknown number of clusters. Another method used the Projection Pursuit (PP) model to reduce dimensionality and then used PSO to perform clustering [19].

E. Summary

Existing EC methods for C+FS use centroid encodings which may produce invalid solutions, fail to train effectively due to centroid ordering, and are difficult to use effectively when K is not pre-determined. In this work we propose variations to the centroid representation to prevent invalid solutions and introduce a new medoid encoding which does not produce any ordering and which can be used with a fixed-length encoding when K is not pre-determined.

III. Proposed Methods

This section proposes PSO methods for both the cases where K is pre-defined and where K is automatically determined by the learning process. We also propose a fitness function which can be used to give good C+FS solutions in both cases.

A. Pre-defined K

We use a **centroid** representation (as shown in Figure 1) as a base PSO method for performing C+FS. This is the same representation as in the NMA_CFS approach, but here we only use it when K is known and hence it has a fixed length of $m + Km$, where the first m dimensions correspond to the m features in the dataset. The position value for a given feature determines if it has been selected by the PSO algorithm. A position value greater than 0 indicates a selected feature; less than 0 indicates the feature is not selected. The remaining Km dimensions correspond to the co-ordinates of the K cluster prototypes. Instances are assigned to the cluster with the closest prototype by Euclidean distance based on the selected features only.

To address the issue where the centroid representation could produce invalid solutions, we propose two seeding techniques which ensure at least one of the initial particles in the swarm will contain centroids which produce a valid partition with the correct K. These two techniques are described below.

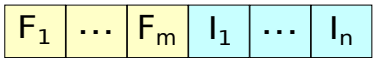


Fig. 3: Medoid representation for simultaneous clustering and feature selection, where K is known.

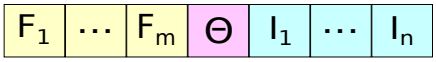


Fig. 4: Medoid representation for simultaneous clustering and feature selection, where K is **unknown**.

The ***k*-means seeded centroid** approach first runs the k-means algorithm on the dataset to give K viable centroids. One particle in the swarm is initialised to contain these K centroids and the remaining particles are initialised as normal (i.e. randomly). For all particles, the FS component of the particle is randomly initialised. This technique ensures at least one viable particle is produced, while also using the result of k-means as a good starting point for the PSO training process.

The **dataset seeded centroid** approach initialises the swarm by randomly choosing instances from the dataset to act as initial centroids. Each particle is initialised by randomly choosing K unique instances and using their feature values as the initial positions of the K centroids. This approach still uses a centroid representation; centroids are not constrained to take instance values in subsequent iterations. This approach ensures that all particles in the swarm will initially give valid solutions.

The **medoid** clustering approach can be extended to perform FS by adding a dimension per feature, as shown in Figure 3. This representation has a length of $m + n$ for n instances in the dataset, and uses the same FS technique as the centroid approach. Instead of using centroids, each instance is assigned a dimension, and the position value of an instance's dimension determines if it is a cluster prototype. The instances that correspond to the K highest position values are chosen to be the K prototypes. As before, instances are assigned to the nearest prototype by Euclidean distance.

B. No pre-defined K

As discussed previously, the use of a centroid encoding when K is not pre-defined requires a variable length encoding (or a technique to turn centroids "on" and "off" [1]). The medoid representation can be easily extended to allow for a flexible number of clusters to be generated while maintaining a fixed-length encoding. When K was fixed, the instances with the K highest position values were chosen as medoids. An alternative technique is to choose all instances where the position values are above some threshold Θ. K will then vary based on the number of instances meeting this threshold. The representation for this **dynamic medoid** approach is shown in Figure 4. This representation includes a single additional dimension corresponding to Θ. The length of this representation is $m+n+1$. While Θ could be a constant value (e.g. 0), we suggest that it may be more effective to allow the PSO process to automatically vary Θ as changing Θ can add or remove multiple clusters from a solution at a time.

C. Fitness Function

In order to perform effective clustering and FS, a fitness function must be used which encourages good clustering performance while minimising the number of features selected. The NMA_CFS [14] algorithm used the J_2 fitness function (shown in Equation 3) which measures clustering performance using the trace scatter metric and applies two penalty functions to minimise the number of features and clusters selected.

$$J2 = trace(S_w^{-1} S_b) \times \frac{m - m^{'}}{m - 1} \times \frac{K_{max} - K}{K - 1} \tag{3}$$

where m and $m^{'}$ are the total number of features and the number of **selected** features respectively. K and K_{max} are the number of clusters and the maximum number of clusters respectively. K_{max} is defined as $\sqrt{n}$ (as is common in the literature [20]) where n is the number of instances in the dataset. S_w and S_b are the within- and between-cluster scatter matrices which measure cluster compactness and separability respectively and are defined as follows:

$$S_w = \frac{1}{n} \sum_{i=1}^{K} \sum_{I_a \in C_i} (I_a - Z_i)(I_a - Z_i)^T \tag{4}$$

$$S_b = \sum_{i=1}^{K} \frac{|C_i|}{n} (Z_i - Z^*)(Z_i - Z^*)^T \tag{5}$$

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^*.

While we base our fitness function on Equation 3, we found that a number of improvements could be made to improve performance. The clusters produced can be improved by punishing outliers more heavily by using the sum of squared Euclidean distances instead of the squared difference to measure intra and inter-cluster dissimilarity. The number of clusters should not be penalised linearly; we should apply only a small penalty to smaller values of K, but apply an increasingly larger penalty as K approaches K_{max}. Doing so allows a wider range of small potential K values to be considered by the learning algorithm, while still penalising K values which are so large as to produce overly specific clusters. This change improved the accuracy of K for the dynamic medoid approach. Our proposed fitness function with these improvements is shown in Equation 6.

$$\text{Fitness} = \frac{Between_{sum}}{Within_{sum}} \times \frac{m - m^{'}}{m - 1} \times (1 - \frac{\log K}{\log K_{max}}) \tag{6}$$

$$Within_{sum} = \frac{1}{n} \sum_{i=1}^{K} \sum_{I_a \in C_i} d(I_a, Z_i)^2 \tag{7}$$

$$Between_{sum} = \sum_{i=1}^{K} \frac{|C_i|}{n} d(Z_i, Z^*)^2 \tag{8}$$

where $d(I_a, I_b)$ is the Euclidean distance between instances I_a and I_b, defined in the standard way:

$$d(I_a, I_b) = \sqrt{(I_{a1} - I_{b1})^2 + (I_{a2} - I_{b2})^2 + ... + (I_{am} - I_{bm})^2} \quad (9)$$

When computing the Euclidean distance between two objects, we use **all** features, not only the selected ones. If only the selected features are used, the fitness function would become overly biased towards a low K and $m^{'}$; selecting fewer features reduces the amount of information available to the clustering method, reducing the extent to which the dataset can be separated into smaller, more specific clusters. By using all features, we ensure the original structure of the dataset is considered when evaluating the performance of a solution.

IV. Experiment Design

We evaluated the performance of the PSO representations on a range of datasets using a variety of evaluation metrics. We also compare our results to those produced by the standard k-means when all features are used. As all of the methods are non-deterministic, we ran each method 30 times and compute the mean of each metric across all runs. Unfortunately we were unable to compare to the NMA_CFS algorithm as the authors were unable to provide their source code and we were not able to reproduce their results when we re-implemented their approach. The remainder of this section will discuss the datasets, evaluation metrics and PSO parameters used in detail.

A. Datasets

We tested the PSO representations across a variety of real-world and synthetic datasets which are summarised in Tables I and II respectively. The real-world datasets are ones which are commonly used in the clustering literature, and all are from the UCI machine learning repository [21]. All of these are classification datasets; it is common practice to use classification datasets for clustering by excluding the class label from the training process. The class labels are then used to evaluate how well the clusters produced match the known classification. The synthetic datasets were generated by Handl et. al. [22] and are also widely used. We used a range of synthetic datasets which contain 10, 50 or 100 features and 10, 20 or 40 clusters. Testing the representations on large m and K shows how they scale as the search space increases. All datasets were normalised so that each feature had zero mean and unit variance to ensure features contribute equally to the clustering process.

TABLE I: Real-World UCI datasets from [21].

Name	No. of Features	No. of Instances	No. of Classes
Iris	4	150	3
Wine	13	178	3
Movement Libras	90	360	15
Breast Cancer	9	683	2
Image Segmentation	18	683	7
Dermatology	34	359	6

TABLE II: Synthetic datasets from [22].

Name	No. of Features	No. of Instances	No. of Classes
10d10c	10	2730	10
10d20c	10	1014	20
10d40c	10	1938	40
50d10c	50	2699	10
50d20c	50	1255	20
50d40c	50	2335	40
100d10c	100	2893	10
100d20c	100	1339	20
100d40c	100	2212	40

B. Evaluation Metrics

We evaluated the performance of each of the representations using five commonly used metrics. These included two internal metrics, which directly measure the quality of a partition. The first of these is the trace scatter metric (as defined in Section III-C) which measures the compactness and separability of a partition. In addition, we compute the sum of the intra-cluster distances which measures compactness directly. This is the metric that the k-means directly optimises.

We also used three external metrics which measure how well the clustering solution matches the dataset's known classification. The class purity metric measures how homogeneous each cluster is in terms of the class labels of the instances it contains. The error rate (ER) is a metric proposed by Cura [23] which measures how well pairs of instances agree on their class labels and cluster memberships. The true positive rate (TPR) measures the proportion of instances in the same class that are also in the same cluster.

Each of the five metrics, asides from the scatter trace metric, are defined as follows:

1) Sum intra-cluster distance:

$$\sum \text{Intra} = \sum_{i=1}^{K} \sum_{I_a \in C_i} d(I_a, Z_i) \quad (10)$$

2) Class purity: computed according to the following steps:
   a) For each cluster, find the majority class label of the instances in that cluster.
   b) Count the number of correctly classified instances, where an instance is correctly classified if it belongs to the majority class.
   c) Find the class purity as the fraction of correctly classified instances across the whole partition.
3) ER [23]:

$$\text{ER} = \Big[\frac{2}{n(n-1)} \times \sum_{a=1}^{n-1} \sum_{b=a+1}^{n} disagree(I_a, I_b)\Big] \times 100\% \quad (11)$$

where $disagree(I_a, I_b)$ is 0 if instances I_a and I_b are either both in the same classes and same clusters or in different classes and clusters, and 1 otherwise.

4) TPR:

$$\text{TPR} = \frac{1}{totalPairs} \times \sum_{i=1}^{\#Cl} \sum_{I_a, I_b \in Cl_i} agree(I_a, I_b) \quad (12)$$

where Cl_i represents the i^{th} class and $\#Cl$ is the number of classes. $agree(I_a, I_b)$ is 1 if I_a and I_b are in the same cluster and 0 otherwise. $totalPairs$ is the sum of the number of pairs of instances in each class.

C. PSO Parameters

Several parameters must be set in the PSO algorithm. We use commonly used parameters [24]: 100 iterations, a swarm size of 30, $v_{max} = 6$, $w = 0.729844$, and $c_1 = c_2 = 1.49618$. These parameters are constant across all datasets.

V. Results and Analysis

The performance of the approaches are shown in Tables III and IV respectively. KMS-Centroid is the k-means seeded centroid approach, DS-Centroid is the dataset seeded centroid approach, and D-Medoid is the dynamic medoid approach. We provide the mean results for each of the evaluation metrics discussed previously, as well as the mean number of features selected ($m^{'}$) and **non-empty** clusters produced (K). We label each metric with a ↑ or ↓ to indicate if it should be maximised or minimised. The best result for each external metric is bolded, and a method is bolded if it has the best performance on a majority of the external metrics. The results will be analysed for the real-world and synthetic datasets in turn.

A. Real-World Results

Most of the pre-fixed representations perform similarly on the easy Iris dataset, with the centroid approach achieving slightly better results on the external metrics. On the more difficult of the real-world datasets, the centroid approach begins to struggle; it is not able to generate 15 clusters successfully on the Movement Libras dataset, and it has the worst performance across the metrics on the Image Segmentation and Dermatology datasets. The seeded centroid approaches (KMS and DS) are able to outperform the normal centroid approach on these datasets and are always able to correctly generate K viable clusters. The medoid approach performs the best of the pre-fixed approaches as it consistently selects a relatively small number of features while achieving competitive performance on both the internal and external metrics. For example, on the Movement Libras dataset it selects only 20.67 features on average while achieving similar results to both KMS and k-means which use 39.57 and 90 features respectively.

k-means outperforms all of the PSO methods on the Wine and Breast Cancer datasets, but performs poorly in comparison on Iris. In general, the PSO methods are able to achieve good performance given that they use many fewer features than k-means. All of the methods struggle on the Movement Libras dataset with TPR and purity values under 50%. This dataset is known to be difficult for clustering algorithms [14] as the class labels do not correspond well to hyper-spherical clusters.

TABLE III: Performance on Real-World Datasets.

(a) Iris

Method	$m^{'}$	K	Scatter ↑	Σ Intra ↓	Purity ↑	ER ↓	TPR ↑
Centroid	1	3	23.57	16.94	**0.9111**	**10.21**	**0.8681**
KMS-Centroid	1	3	22.93	16.92	0.9036	10.99	0.8612
DS-Centroid	1	3	22.50	16.91	0.8987	11.48	0.8568
Medoid	1	3	22.66	16.92	0.9004	11.30	0.8584
D-Medoid	1.100	3.167	23.83	16.64	0.9049	11.36	0.8308
k-means	4	3	12.03	16.93	0.7844	19.97	0.7718

(b) Wine

Method	$m^{'}$	K	Scatter ↑	Σ Intra ↓	Purity ↑	ER ↓	TPR ↑
Centroid	2.600	3	11.05	57.70	0.9045	11.79	0.8212
KMS-Centroid	2.633	3	11.51	57.13	0.9270	9.386	0.8536
DS-Centroid	2.267	3	10.86	57.34	0.9243	9.755	0.8479
Medoid	2.400	3	11.18	57.28	0.9232	9.887	0.8458
D-Medoid	2.733	3	10.11	57.72	0.9077	11.68	0.8237
***k*-means**	13	3	13.69	56.39	**0.9610**	**5.287**	**0.9152**

(c) Movement Libras

Method	$m^{'}$	K	Scatter ↑	Σ Intra ↓	Purity ↑	ER ↓	TPR ↑
Centroid	45.37	10.63	12.52	336.0	0.2489	26.62	0.4164
KMS-Centroid	39.57	15	34.80	240.7	0.4506	9.877	0.3704
DS-Centroid	35.5	15	28.47	259.3	0.4148	11.04	0.3358
Medoid	20.67	15	36.07	237.8	**0.4680**	9.693	0.3820
D-Medoid	23.13	6.267	13.89	300.3	0.2799	18.57	**0.4498**
k-means	90	15	37.24	237.5	0.4640	**9.569**	0.3902

(d) Breast Cancer

Method	$m^{'}$	K	Scatter ↑	Σ Intra ↓	Purity ↑	ER ↓	TPR ↑
Centroid	1.333	2	5.505	142.5	0.9375	11.71	0.9094
KMS-Centroid	1.367	2	5.310	142.6	0.9366	11.87	0.9081
DS-Centroid	1.233	2	5.222	142.8	0.9348	12.18	0.9070
Medoid	1.167	2	5.294	143.0	0.9343	12.27	0.9071
D-Medoid	1.733	2.733	9.778	137.6	0.9414	13.80	0.8417
***k*-means**	9	2	6.774	137.8	**0.9587**	**7.925**	**0.9341**

(e) Image Segmentation

Method	$m^{'}$	K	Scatter ↑	Σ Intra ↓	Purity ↑	ER ↓	TPR ↑
Centroid	6.300	7	35.28	661.3	0.5556	20.37	0.7226
KMS-Centroid	4.533	7	51.60	636.5	0.5845	17.07	**0.7444**
DS-Centroid	4	7	56.69	628.5	0.5857	16.44	0.7124
Medoid	3.800	7	67.51	611.4	**0.6072**	**14.46**	0.6684
D-Medoid	4.5	7.300	55.60	612.0	0.5950	15.20	0.6550
k-means	18	7	50.92	607.1	0.6043	15.30	0.6715

(f) Dermatology

Method	$m^{'}$	K	Scatter ↑	Σ Intra ↓	Purity ↑	ER ↓	TPR ↑
Centroid	11.60	6	46.70	183.6	0.7246	15.69	0.7288
KMS-Centroid	8.833	6	72.87	176.1	0.8024	11.79	0.7788
DS-Centroid	7.867	6	71.46	175.9	0.8044	10.97	0.8035
Medoid	6.5	6	86.21	172.9	**0.8318**	**10.83**	0.7554
D-Medoid	7.867	4.267	61.76	182.7	0.7344	14.84	**0.9088**
k-means	34	6	92.22	178.0	0.8099	12.26	0.7839

The dynamic medoid approach (D-Medoid) performs reasonably well on the Iris, Wine, Breast Cancer and Image Segmentation datasets, generating K within 1 of the actual number of classes and achieving similar results on the metrics to the other PSO methods. It struggles to achieve good results on the Movement Libras dataset where it only finds 6.267 clusters on average and has a correspondingly high error rate. It is interesting to note that on both the Movement Libras and

TABLE IV: Performance on Synthetic Datasets

(a) 10d10c

Method	m'	K	Scatter ↑	Σ Intra ↓	Purity ↑	ER ↓	TPR ↑
Centroid	2.800	10	11.79	803.3	0.6448	13.59	0.5608
KMS-Centroid	4.033	10	14.69	703.3	0.8179	7.271	0.7103
DS-Centroid	3.333	10	14.34	734.3	0.7713	8.442	0.6918
Medoid	3.933	10	15.18	708.0	0.8056	7.864	0.6646
D-Medoid	3.767	7.467	11.83	753.8	0.7458	9.727	0.7379
***k*-means**	10	10	17.50	643.8	**0.9281**	**3.983**	**0.8001**

(b) 10d20c

Method	m'	K	Scatter ↑	Σ Intra ↓	Purity ↑	ER ↓	TPR ↑
Centroid	5.100	12.20	9.875	307.1	0.3854	22.84	0.6632
KMS-Centroid	5.167	20	55.55	159.5	0.8551	2.834	0.7914
DS-Centroid	4.767	20	45.13	174.0	0.8084	3.349	0.7543
Medoid	4.967	20	64.74	143.8	**0.9160**	**1.504**	0.8727
D-Medoid	4.300	12.73	39.17	188.8	0.7527	4.068	**0.9037**
k-means	10	20	70.37	143.8	0.9018	2.095	0.8306

(c) 10d40c

Method	m'	K	Scatter ↑	Σ Intra ↓	Purity ↑	ER ↓	TPR ↑
Centroid	5.367	20.80	8.541	585.8	0.3334	16.13	0.6407
KMS-Centroid	5.300	40	55.08	311.4	0.8388	1.677	0.7411
DS-Centroid	5.233	40	39.87	362.8	0.7447	2.577	0.6260
Medoid	5.667	40	56.81	292.8	0.8734	1.305	0.7821
D-Medoid	3.900	13.10	20.19	502.4	0.4861	6.786	0.8267
***k*-means**	10	40	73.06	261.2	**0.9188**	**0.9848**	**0.8386**

(d) 50d10c

Method	m'	K	Scatter ↑	Σ Intra ↓	Purity ↑	ER ↓	TPR ↑
Centroid	22.97	10	23.72	1330	0.5807	23.94	**0.5782**
KMS-Centroid	17.67	10	56.70	972.9	0.7125	16.82	0.5545
DS-Centroid	16.60	10	52.43	995.2	0.6948	18.53	0.5524
Medoid	13.67	10	62.12	938.9	0.7470	14.36	0.5655
D-Medoid	13.47	18.23	133.5	717.6	**0.8773**	**9.334**	0.4115
k-means	50	10	70.98	967.0	0.7365	16.06	0.5698

(e) 50d20c

Method	m'	K	Scatter ↑	Σ Intra ↓	Purity ↑	ER ↓	TPR ↑
Centroid	24.23	12.47	23.83	813.2	0.3314	29.54	**0.6670**
KMS-Centroid	22.30	20	119.0	480.6	0.6737	11.95	0.5876
DS-Centroid	20.43	20	99.82	529.8	0.6621	8.742	0.5879
Medoid	14.17	20	112.3	453.2	**0.7459**	**7.216**	0.6203
D-Medoid	15.33	17.40	97.53	486.5	0.7115	10.16	0.6655
k-means	50	20	136.0	477.6	0.6895	11.72	0.6021

(f) 50d40c

Method	m'	K	Scatter ↑	Σ Intra ↓	Purity ↑	ER ↓	TPR ↑
Centroid	26.77	22.90	29.76	1520	0.2905	24.19	0.6073
KMS-Centroid	22.87	40	176.2	852.1	0.6696	8.309	0.5551
DS-Centroid	19.47	40	131.6	1001	0.6238	5.387	0.5272
Medoid	14.37	40	156.9	860.9	**0.7053**	**4.135**	0.5687
D-Medoid	14.53	22.80	84.56	1156	0.5042	11.58	**0.6274**
k-means	50	40	190.7	847.2	0.6786	9.822	0.5825

TABLE IV: Performance on Synthetic Datasets cont.

(g) 100d10c

Method	m'	K	Scatter ↑	Σ Intra ↓	Purity ↑	ER ↓	TPR ↑
Centroid	47.5	10	29.04	2099	0.5883	22.00	0.5954
KMS-Centroid	39.53	10	65.85	1533	0.7392	14.04	0.6016
DS-Centroid	39.90	10	63.18	1572	0.7230	15.51	0.6035
Medoid	33.53	10	71.45	1470	0.7808	11.36	**0.6488**
D-Medoid	34.43	19.07	193.3	1060	**0.9092**	**8.459**	0.4302
k-means	100	10	94.17	1592	0.7600	13.76	0.6217

(h) 100d20c

Method	m'	K	Scatter ↑	Σ Intra ↓	Purity ↑	ER ↓	TPR ↑
Centroid	49.80	12.93	27.24	1222	0.3239	30.39	0.6311
KMS-Centroid	46.90	20	149.8	740.1	0.6709	11.42	0.5933
DS-Centroid	45	20	138.8	777.4	0.6514	11.31	0.5852
Medoid	32.17	20	157.2	671.2	**0.7610**	**7.653**	0.6200
D-Medoid	35.63	18.5	146.3	711.8	0.7268	9.342	**0.6414**
k-means	100	20	186.2	733.7	0.6921	12.48	0.6184

(i) 100d40c

Method	m'	K	Scatter ↑	Σ Intra ↓	Purity ↑	ER ↓	TPR ↑
Centroid	49.53	22.77	39.72	2052	0.2868	24.05	0.6019
KMS-Centroid	47.40	40	263.0	1152	0.6742	8.729	0.5618
DS-Centroid	43.60	40	217.9	1277	0.6625	6.074	0.5668
Medoid	32.73	40	246.5	1107	**0.7382**	**4.398**	0.5952
D-Medoid	34.97	22.93	136.3	1439	0.5516	14.14	**0.6705**
k-means	100	40	301.7	1148	0.6874	10.26	0.5903

Dermatology datasets it has the highest TPR value despite having incorrect K; this suggests it is creating a few big clusters each of which contain multiple classes. This highlights a fundamental issue in clustering when K is unknown – it can be difficult to consistently determine across varying datasets when one large cluster should be split into two smaller clusters.

In general, the medoid approach is superior to all other pre-fixed approaches on the real-world datasets, including the centroid approaches which are much more widely used.

B. Synthetic Results

Unlike the real-world datasets, the synthetic datasets are designed specifically for evaluating clustering algorithms. Hence, they may give a better indication of the performance that can be expected on an unlabelled dataset.

The basic centroid approach struggles to find K viable clusters on all of the synthetic datasets that contain 20 or 40 clusters. Of the other pre-fixed K approaches, the medoid approach selects the fewest number of features while achieving the best results across the metrics on all of the synthetic datasets with 50 or 100 features. This approach even outperforms the k-means approach on these datasets despite using many fewer features. This suggests two things: the medoid approach may be able to better explore the search space than the centroid approaches as it can more effectively utilise the social knowledge of the swarm (as theorised in Section II-D), and secondly that k-means fails to scale well for large k or m whereas PSO is able to scale effectively. On the datasets with 10 features, the KMS approach can also perform well relative to the medoid approach. k-means is also able to perform well – it has the lowest error rate and highest purity on 10d10c and 10d40c, as well as the best results for the internal metrics. These datasets have a small number of features and so k-means does not struggle as it does on the other synthetic datasets. Each of the two seeded centroid approaches perform better on different datasets, with the DS approach generally performing more effectively on the larger datasets.

The dynamic medoid approach is inaccurate in terms of the mean number of clusters it produces on all synthetic

datasets asides from 10d10c, 50d20c and 100d20c. On the datasets where it overestimates K (50d10c and 100d10c), it achieves the best scatter, Σ intra, purity and ER results despite obviously performing badly. This highlights the difficulty in measuring good clustering performance and in designing a good fitness function – if we used only scatter fitness as our fitness function, we would likely produce many more clusters than is correct. The purity and ER metrics are also biased towards large K. As the number of clusters increases, the homogeneity of any given cluster is likely to be higher as clusters become smaller and more specific. This leads to an increase in purity. In a similar vein, a higher K also tends to decrease the ER: the more clusters present, the more likely it is that instances from different classes will be assigned to different clusters. As the majority of pairs of instances in a dataset will belong to two different classes when there are more than two classes, the number of disagreements between instance pairs will tend to decrease as K is increased. The TPR metric does not have this issue as it considers only the true positives relative to the number of actual positives; the dynamic medoid approach achieves a low TPR value on 50d10c and 100d10c. These issues demonstrate the need for care when evaluating clustering algorithms, as the number of clusters may have an unexpected effect on how metrics behave.

While the dynamic medoid approach performs poorly compared to the pre-fixed approaches, the case where K is unknown is a much more difficult problem [1]. We believe that the success of the pre-fixed medoid approach and the fact that the dynamic approach can use a fixed-length representation suggests it is worth exploring this direction further.

VI. Conclusion

In this work we developed and compared a number of different PSO representations for simultaneously performing clustering and feature selection. We proposed an extension to the medoid representation that enables it to be used in the case where the number of clusters is unknown. An adaptation of an existing fitness function was also proposed. A comprehensive quantitative comparison of the different representations was conducted across a number of real-world and synthetic datasets with a range of different characteristics. It was found that a medoid representation could achieve the best performance when K is known, as it was able to select the fewest features while outperforming k-means on the hardest synthetic datasets. This finding highlights a gap in the current clustering literature, which has focused primarily on centroid approaches.

While the dynamic medoid approach struggled on the synthetic datasets, we believe it can be improved in future work by finding a more effective way of restricting the search space on the number of clusters. We would also like to explore more effective fitness functions that can better balance cluster performance and feature selection. As clustering in itself is a multi-objective problem, we would like to investigate using a multi- or many-objective approach that may produce better results than the current linear combination of objectives we used. Further work could also investigate more intelligent centroid seeding techniques or better external metrics for measuring cluster performance that are independent of K.

References

[1] A. J. García and W. Gómez-Flores, "Automatic clustering using nature-inspired metaheuristics: A survey," *Appl. Soft Comput.*, vol. 41, pp. 192–213, 2016.

[2] E. Müller, S. Günnemann, I. Assent, and T. Seidl, "Evaluating clustering in subspace projections of high dimensional data," *PVLDB*, vol. 2, no. 1, pp. 1270–1281, 2009.

[3] S. J. Nanda and G. Panda, "A survey on nature inspired metaheuristic algorithms for partitional clustering," *Swarm and Evolutionary Computation*, vol. 16, pp. 1–18, 2014.

[4] E. Falkenauer, *Genetic algorithms and grouping problems*. John Wiley & Sons, Inc., 1998.

[5] H. Liu and H. Motoda, *Feature selection for knowledge discovery and data mining*. Springer Science & Business Media, 2012, vol. 454.

[6] B. Xue, M. Zhang, W. Browne, and X. Yao, "A survey on evolutionary computation approaches to feature selection," *IEEE Transactions on Evolutionary Computation*, 2016.

[7] S. Alelyani, J. Tang, and H. Liu, "Feature selection for clustering: A review," in *Data Clustering: Algorithms and Applications*, 2013, pp. 29–60.

[8] A. E. Eiben and J. E. Smith, *Introduction to Evolutionary Computing*, ser. Natural Computing Series. Springer, 2015.

[9] A. A. A. Esmin, R. A. Coelho, and S. Matwin, "A review on particle swarm optimization algorithm and its variants to clustering high-dimensional data," *Artif. Intell. Rev.*, vol. 44, no. 1, pp. 23–45, 2015.

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

[11] H. Liu, H. Motoda, R. Setiono, and Z. Zhao, "Feature selection: An ever evolving frontier in data mining," in *Proceedings of the Fourth International Workshop on Feature Selection in Data Mining*, 2010, pp. 4–13.

[12] C. C. Aggarwal and C. K. Reddy, Eds., *Data Clustering: Algorithms and Applications*. CRC Press, 2014.

[13] E. R. Hruschka, R. J. G. B. Campello, A. A. Freitas, and A. C. P. L. F. de Carvalho, "A survey of evolutionary algorithms for clustering," *IEEE Transactions on Systems, Man, and Cybernetics, Part C*, vol. 39, no. 2, pp. 133–155, 2009.

[14] W. Sheng, X. Liu, and M. C. Fairhurst, "A niching memetic algorithm for simultaneous clustering and feature selection," *IEEE Trans. Knowl. Data Eng.*, vol. 20, no. 7, pp. 868–879, 2008.

[15] M. Javani, K. Faez, and D. Aghlmandi, "Clustering and feature selection via PSO algorithm," in *International Symposium on Artificial Intelligence and Signal Processing (AISP)*. IEEE, 2011, pp. 71–76.

[16] Y. Marinakis, M. Marinaki, and N. F. Matsatsinis, "A hybrid particle swarm optimization algorithm for clustering analysis," in *Proceedings of the 9th International Conference on Data Warehousing and Knowledge Discovery (DaWaK)*, 2007, pp. 241–250.

[17] ——, "A hybrid clustering algorithm based on multi-swarm constriction PSO and GRASP," in *Proceedings of the 10th International Conference on Data Warehousing and Knowledge Discovery (DaWaK)*, 2008, pp. 186–195.

[18] R. J. Kuo, Y. J. Syu, Z. Chen, and F. Tien, "Integration of particle swarm optimization and genetic algorithm for dynamic clustering," *Inf. Sci.*, vol. 195, pp. 124–140, 2012.

[19] Q. Zhang, X. Lei, X. Huang, and A. Zhang, "An improved projection pursuit clustering model and its application based on quantum-behaved PSO," in *Proceedings of the Sixth International Conference on Natural Computation (ICNC)*, 2010, pp. 2581–2585.

[20] N. R. Pal and J. C. Bezdek, "On cluster validity for the fuzzy c-means model," *IEEE Trans. Fuzzy Systems*, vol. 3, no. 3, pp. 370–379, 1995.

[21] M. Lichman, "UCI machine learning repository," 2013. [Online]. Available: http://archive.ics.uci.edu/ml

[22] J. Handl and J. D. Knowles, "An evolutionary approach to multiobjective clustering," *IEEE Trans. Evolutionary Computation*, vol. 11, no. 1, pp. 56–76, 2007.

[23] T. Cura, "A particle swarm optimization approach to clustering," *Expert Syst. Appl.*, vol. 39, no. 1, pp. 1582–1588, 2012.

[24] F. Van Den Bergh, "An analysis of particle swarm optimizers," Ph.D. dissertation, University of Pretoria, 2006.