Symbiotic Evolutionary Subspace Clustering

Ali Vahdat[1], Malcolm Heywood[1], and Nur Zincir-Heywood[1]

[1]*Faculty of Computer Science, Dalhousie University, Halifax, NS. Canada*

Abstract

New emerging high-dimensional data sets have made traditional clustering algorithms increasingly inefficient. More sophisticated approaches are required to cope with the increasing dimensionality and cardinality of such data sets. Feature selection methods are proposed as a solution to deal with this problem, however they fail for data sets where the attribute support for different clusters is not the same. For this category of data sets subspace clustering algorithms have been introduced over the past decade. We approach this problem from the perspective of Genetic Algorithms by adopting a hierarchical data structure deployed in three stages. 1) a traditional clustering algorithm is applied independently to *each attribute* of the data set, thus defining a grid of potential 1-*d* cluster centroids. 2) describing multi-dimensional cluster centroids indexing 1-*d* cluster centroids. 3) converting the problem of finding the best combination of cluster centroids into that of discrete optimization and applying a multi-objective evolutionary algorithm, which uses group fitness evaluation to give a fitness to a group of clusters, as defined by process 2. Synthetic data sets with different characteristics are generated as the ground truth to evaluate the resulting algorithm for Evolutionary Subspace Clustering (ESC) as well as benchmark against alternative subspace and full-space clustering algorithms. ESC returns competitive accuracy and while typically utilizing less attributes and scaling as attribute count increases.

1 Introduction

Clustering algorithms are increasingly being applied to data sets from fields such as video and text analysis, or bioinformatics. This represents a trend in which high dimensionality – that is minimum of 50 attributes – represents the norm. As the dimensionality of the original data increases, it also becomes more likely that attributes are not of equal utility across all clusters. Thus from the perspective of each cluster, different attributes are either useful or a source of noise. An early approach for attempting to address this problem was to apply an independent feature selection process or apply a transformation; the latter loosing the meaning of the original attributes. Moreover, assuming a common subset of attributes for all clusters represents a bias that may or may not be appropriate. With this in mind, subspace clustering algorithms attempt to provide the flexibility to identify the (possibly unique) subset of attributes for each cluster [16, 17, 14, 15]. Naturally, such algorithms will include the case of all clusters supported through a common subspace (of attributes) as a special case.

In this work we propose an evolutionary approach to subspace clustering – hereafter *Evolutionary Subspace Clustering* or *ESC* – in which cluster-specific attributes of a subspace are identified along with its instances. To do so, a two-population model is assumed (Figure 1) in which a *clustering solution*[1] (CS) defines a candidate solution in terms of some subset of subspace clusters defined by an independent population of *subspace cluster centroids*[2] (SCC). The following 3 stage process summarizes the ESC approach:

- Process 1 – Grid construction: A (traditional) clustering algorithm is applied to the data set on an attribute-by-attribute basis. The outcome will be a grid that establishes the potential location of cluster centroids in terms of 1-*d* clusters defined on each attribute. Such a pre-processing activity is central to turning the process of subspace identification into that of a combinatorial search.
- Process 2 – Subspace cluster centroids: Candidate SCC are declared by assuming a representation that samples from the attributes and 1-*d* clusters defined by process 1. There is no concept of SCC fitness. Theses SCCs are indexed and exploited by process 3 individuals to define candidate clustering solutions.

[1]Hereafter 'Clustering Solution , 'CS', and 'process 3 individual' are used interchangeably.
[2]Hereafter 'Subspace Cluster Centroid' , 'SCC', and 'process 2 individual' are used interchangeably.

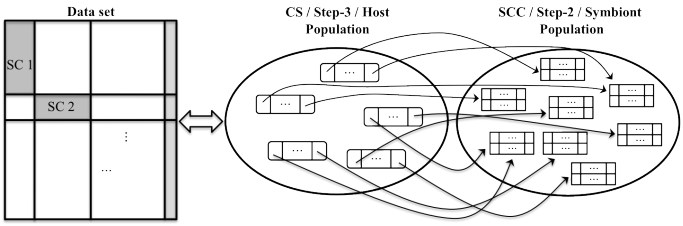

Figure 1: Structure of ESC algorithm.

- Process 3 – Evaluating candidate clustering solutions: A multi-objective genetic algorithm directs the evolution of the best combinations of subspace cluster centroids which will form clustering solutions (CS) of the ESC algorithm.

Processes 2 and 3 form the core of the algorithm and are in constant interaction with each other to evolve SCC individuals concurrently with CS individuals. In fact process 2 prepares the low-level genetic material for process 3, while information from process 3 directs the search for new subspace clustering solutions. Process 1 supports the representation of clusters in terms of a discrete grid, thus permitting application of a combinatorial search process. As such process 1 is only applied once per dataset as a preprocessing step.

The rest of the paper is organized as follows. Section 2 reviews the related literature and in so doing makes a case for utilizing two non-evolutionary subspace clustering algorithms for comparison in the later benchmarking study. Section 3 presents the proposed ESC algorithm. Section 4 details the experimental study, and Section 5 concludes the paper and gives possible future improvements to ESC.

2 Related Works

Subspace clustering has been investigated under different names over the last 10 years including, but not limited to: projected clustering, projective clustering, bi-clustering, and co-clustering. Recent survey papers have established the current capability of subspace clustering algorithms under various frameworks [16, 17, 14, 15]. In particular [14] and [15] conduct empirical evaluations, thus compare different subspace clustering algorithms with respect to various performance measures. In the following we emphasize the works with most relevance to the approach adopted here.

Parsons *et al.* divide subspace clustering algorithms into two categories: bottom-up and top-down [16]. Bottom-up search methods find the dense regions / bins in lower dimensions – starting from 1-*d* bins – and incrementally identify attributes that contain dense bins to make higher dimensional subspace clusters. Top-down approaches on the other hand try to prune irrelevant / noise attributes from an initial full dimensional approximation of the clusters. Using this definition implies that the proposed ESC algorithm falls in the bottom-up category since we generate a grid from which subspace cluster centroids are extracted.

Muller *et al.* assume a different categorization based on three paradigms: cell-based, density-based, or clustering-oriented approaches [15]. They also integrated the implementation for various subspace clustering algorithms into the well-known WEKA machine learning software, and call it OpenSubspace[3]. Properties such as accuracy, cluster distribution, coverage, entropy, F1 measure, and runtime as well as the RNIA and CE metrics defined by [17] are used to compare 10 different algorithms on 23 data sets.

In summary Muller *et al.* conclude that the basic methods from cell-based and density-based categories – CLIQUE and SUBCLU – tend to produce clustering solutions with too many clusters, potentially even more than the number of instances in data set. They also have low clustering quality w.r.t. RNIA and CE metrics – which take relevant attributes as well as instances into consideration – and their runtime overhead makes them infeasible. Conversely, MINECLUS (cell-based category), INSCY (density-based), and PROCLUS (clustering-based) produce the best clustering quality w.r.t. RNIA and CE. However, INSCY fails to scale to data sets with more than 25 attributes, and for most of the data sets, it returns results no faster than twice the time of MINECLUS and PROCLUS. Indeed, MINECLUS and PROCLUS happen to be the fastest algorithms in their respective categories. In general recent cell-based approaches outperform other methods on data sets with low dimensionality; whereas the density-based paradigm tends to suffer from long runtimes, and is not scalable to medium- or high-dimensional problems. Clustering-based methods have the fastest runtime, but this is due to the provision of much a priori

[3]http://dme.rwth-aachen.de/OpenSubspace/.

information than any other approach e.g., correct subspace dimensionality. However, they have inferior accuracy performance than cell-based methods. In short, we assume MINECLUS and PROCLUS as representative baselines for the proposed ESC algorithm in Section 4.2, since they were the algorithms that provided the best balance between cluster quality and runtime efficiency [15].

Most proposals for utilizing Evolutionary Computation in clustering assume full-dimensionality, thus do not address the requirements of subspace clustering. With this caveat in mind, a recent survey summarized contributions to evolutionary clustering with respect to developments in: representation, fitness function, or variation operators [9]. However, as will be come apparent later, multi-objective methods for clustering (e.g., [7]) do represent a generic tool capable of addressing some aspects of the subspace problem. Of the evolutionary methods that are explicitly designed to address all aspects of the subspace clustering task, various limitations appear, either in the form of the provision of suitable prior information or in terms of a limited evaluation. Specifically, recently proposed approaches based on particle swarm [12] and ant optimization [19] required knowledge of the true number of clusters k. The GA based framework of Sarafis et al. emphasized the utility of a rule based representation and corresponding variation operators [20]. Evaluation was limited to artificial data sets with no more than 50 dimensions, and no attempt was made to quantify how well full space clustering performed under the same task. Finally, an earlier instance of the authors work employed a two stage Evolutionary Multi-criteria Optimization (EMO) for the identification of independent clusters and then combinations of clusters [21]. Unfortunately, the independent cluster identification step required the utility of 'surrogate' measures of fitness that attempted to enforce diversity of candidate clusters. This represents a bias on the resulting solution quality. This research will still assume a dual population model representing independent clusters and groups of clusters respectively. However, a serial (endosymbiotic [8, 4]) interaction is assumed with fitness only evaluated at the (cluster) group level (Section 3), thus avoiding any use of surrogate objectives.

3 ESC Algorithm

As remarked in the introduction, ESC assumes a three stage process for building subspace clusters:

1. Defining a grid over which subspace cluster centroids are specified;
2. Indexing elements from the grid to define candidate subspace cluster centroids, and;
3. Specifying groups of candidate cluster centroids as solutions to the overall clustering task.

Naturally, data standardization is assumed to have already been performed. Clustering algorithms generally assume this in order to avoid introducing biases into the distance-metric, however, some applications are also sensitive to the nature of the standardization process [3]. Here assume a linear standardization of attributes to the unit interval $[0, 1]$, where this is common to all algorithms benchmarked in Section 4.2.

3.1 Process 1: 1-d attribute-wise clustering

In order to construct a grid from which candidate clusters can be defined (process 2), a standard clustering algorithm is applied to each attribute alone. Thus, for an N dimensional data set, the standard clustering algorithm is called N times, once per attribute. A grid is then defined as the composition of all 1-d centroids. Any standard clustering algorithm is appropriate for this task so long as it does not require a priori definition for the number of centroids sort e.g., X-means clustering [18] or the EM algorithm [13]. Thus, the number of 1-d centroids per attribute is free to vary as a function of the data distribution. It should also be noted that this process is conducted *once* per data set. In the benchmarking study of Section 4.2 the EM algorithm algorithm is assumed. Multiple runs of the proposed ESC algorithm are performed relative to the same common grid.

3.2 Process 2: Composition of candidate Subspace Cluster Centroids (SCC)

The goal of this process is to provide a diverse population of SCCs for process 3 to combine into groups of Cluster Solutions (CS). The most important contribution of this process is to select the best representation for SCCs; whereas variation will be defined as a hierarchical operator from CS to SCC.

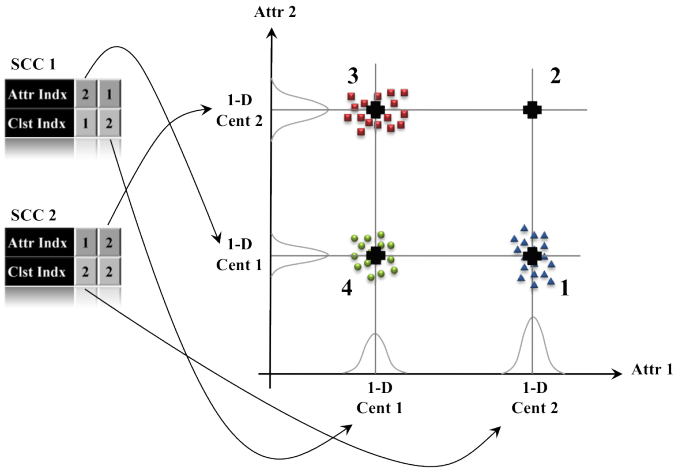

Figure 2: Encoding and decoding of SCC / process 2 individual.

3.2.1 Dual population model

Although most evolutionary clustering algorithms encode the final partitioning in a chromosome, here we adopt a hierarchical approach in which two independent populations are employed (Figure 1). The process 2 population represent candidate Subspace Cluster Centroids (SCC); whereas the process 3 population represent groups of SCC, hence each individual at the process 3 population is a candidate Cluster Solution (CS). Given a suitable representation for a SCC, individuals from CS just take the form of a set of indexes to members of the SCC population. Letting CS individuals assume a variable length representation implies that the number of SCC per CS is free to adapt. Moreover, each CS need only be a different combination, thus the same SCC may appear in multiple CS individuals as directed by natural selection. Fitness is only known explicitly at the CS population as SCC fitness is relative to the other SCC individuals comprising a CS, hence the endosymbiotic nature of the interaction between the two populations [8, 4].

3.2.2 Representation

The encoding of most evolutionary clustering algorithms is coupled with either data set dimensionality or cardinality [9], i.e. a chromosome is a binary / integer / real string that encodes all dimensions of the data set. Here a combination of centroid-based and grid-based encoding is assumed to represent a cluster. Each SCC is represented by a number of integer pairs, or genes, in the form of $\langle a, c \rangle$ where the first integer, a, determines the attribute index, and the second integer, c, determines the 1-d centroid within the attribute.

The encoding and decoding of a SCC is illustrated in Figure 2. The first individual, SCC 1, is composed of two genes (pairs of integers). The first pair $\langle 2, 1 \rangle$ indicates that the encoded SCC uses the 1^{st} 1-d centroid of attribute 2, and the second pair $\langle 1, 2 \rangle$ indicates that it also uses the 2^{nd} 1-d centroid of attribute 1. This individual therefore identifies the subspace cluster centroid in the centre of the triangle-shaped data points on the lower right corner of the grid. Similarly, SCC 2 encodes the candidate subspace cluster centroid on upper right corner of the grid i.e., no data points around it.

3.3 Process 3: Multi-Objective composition of Clustering Solutions (CS)

Process 3 of ESC performs two tasks simultaneously: a) search for the best combination of SCCs to put together to make CS individuals, and b) evolve and adjust the SCCs in relation to the CS they are part of to fine-tune each CS. Fitness is therefore evaluated at the CS level alone. In effect a form of group fitness is applied as it is only within the context of a CS (group) that a SCC (member) is evaluated. SCC individuals 'die' when they do not receive any CS indexes and new SCC individuals appear as a function of the hierarchical variation operators. The size of the SCC population therefore 'floats' whereas the CS population is of a fixed size.

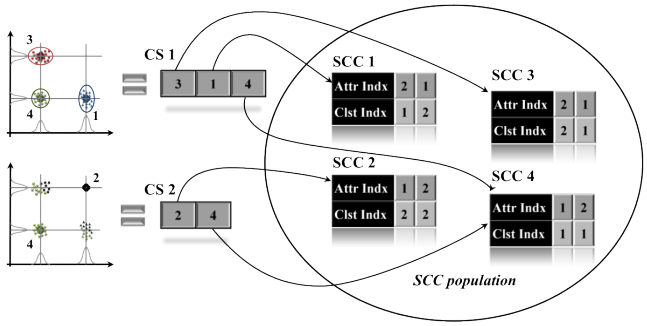

Figure 3: Encoding and decoding of CS / process 3 individual.

3.3.1 Representation

Figure 3 illustrates the encoding of a CS individual. The first individual, CS 1, indexes SCCs 3, 1 and 4 from the SCC population to make a partitioning. This means that CS 1 is a 3-cluster solution for the given data set, with SCCs 3, 1 and 4 specifying the three subspace cluster centroids. Instances belonging to each subspace cluster are determined following a simple nearest neighbourhood (NN) assignment considering only the attributes indexed by each SCC. In other words we calculate the normalized distance of each data point to all SCCs within a CS and assign each data point to its nearest SCC. To reduce the bias from SCCs with less attributes we normalize this distance by the number of attributes indexed by an SCC. Once the nearest neighbourhood assignment is complete, the fitness of a CS can be defined.

3.3.2 Objective Functions

Single objective methods of data clustering can be categorized based on the criterion they optimize. One group of clustering algorithms try to form compact spherical clusters by minimizing the intra-cluster distance between data points and cluster representatives, as in k-means [6], average link agglomerative clustering [22], and self-organizing maps [11]. Another group of clustering algorithms try to form clusters in which data items close to each other fall into the same cluster, hence optimizing connectedness. This category of clustering algorithms can find clusters of arbitrary shape, however they might fail when clusters are close to each other such as single link agglomerative clustering [22], and OPTICS [2].

Since clustering algorithms with a single criterion tend to find only one category of cluster shapes and fail on the other, an EMO might be deployed to increase the chances of finding subspace clusters from both scenarios simultaneously. An earlier version of ESC demonstrated that using an EMO with objectives of *compactness/distortion* (good for recognizing spherical clusters) and *connectivity* (good for arbitrary cluster shape detection) aided in the identification of true cluster count [21].

As indicated above, prior to calculating the compactness of a CS, a nearest neighbour (NN) assignment of data points is performed against the set of SCCs indexed by the CS. This nearest neighbour allocation determines which of the SCCs (within a CS) each instance is assigned to. The distance between an instance and nearest SCC is defined and normalized over the number of attributes supporting the SCC. The sum of all distances between each instance and corresponding SCC is the overall compactness value of a CS:

$$Com(CS) = \sum_{i=1}^{q} \sum_{j=1}^{|SCC_i|} dis(x_j, SCC_i) \tag{1}$$

where SCC_i is the nearest SCC to instance x_j, q is the number of SCCs within CS, and $|SCC_i|$ is the number of instances assigned to SCC_i.

In the case of a connectivity objective, however, the earlier ESC study identified that it also comes at a significant computational overhead when deployed in a subspace clustering context [21]. Specifically, when first proposed by Handl and Knowles, pre-computation was possible relative to the common set of (all) attributes [7]. This is not feasible under subspace clustering, a point we return to with regards to future work.

Since minimizing compactness was previously shown to be correlated with increasing cluster counts [21], we introduce a second objective to reward minimizing the number of clusters. Thus 'tension' between objectives is achieved where this is often required for good performance with EMO frameworks [5]. Naturally, the cluster count objective is efficient to estimate while maintaing post-training performance measures at a satisfactory level.

3.3.3 Variation Operators

The dual population model enables us to conduct a combinatorial search for good candidate clusters while simultaneously maintaining a diverse set of subspace candidate cluster centroids (CS and SCC populations respectively). However, modifying a current SCC is not desirable as it is likely to be shared by several individuals from the CS population. Thus, variation operators are limited to mutation alone and begin by cloning the parent with variation only introduced relative to the (single) child. Parent CS individuals are selected through a tournament against three other CSs. Such a process is explicitly elitist.

Single-Level Mutation The recently-cloned CS goes through three forms of mutation: 1) remove a link to a currently included SCC; 2) add a new SCC link; or 3) replace a current SCC link with that of another. Naturally, the third form of mutation performs in one step what forms 1 and 2 might achieve together. If testing true, the operator is applied again with the threshold of application decreased by an order of magnitude. The process repeating until the test no longer returns true.

Multi-Level Mutation As per single-level mutation a CS parent is selected, cloned, and both parent and clone are retained in the CS population. With uniform probability a SCC within the cloned CS – or parent SCC – is cloned with both parent and clone retained in the SCC population. The newly cloned SCC goes through three mutation steps: 1) remove an attribute, 2) add an attribute, or 3) replace an attribute's 1-d centroid. As per single-level mutation, the case of a mutation operator testing true results in reapplication with the threshold of application decreased by an order of magnitude on each application w.r.t. the same individual.

Finally, as remarked earlier the SCC population size floats. However, a maximum population size is enforced, typically 4 times the initial population size. If the maximum SCC population is reached, the SCCs with smallest SC link count are deleted. To sure that a cloned CS goes through both single- and multi-level mutation, the probability of each operator is set to 1.0.

4 Results

4.1 Evaluation Methodology

In order to evaluate the advantages and disadvantages of subspace clustering algorithms data sets demonstrating specific properties are required. With this in mind the following approach is adopted to building data sets with explicit subspace properties. Subspaces can be characterized as a matrix relating data points to subsets of attributes. Figure 4 illustrates such a process in terms of the embedding of 3 subspace clusters in a larger attribute space. The within-cluster values (values inside coloured boxes) are sampled from distributions specific to each subspace cluster. These distributions can be Gaussian (hyper-spherical subspace clusters), elongated Gaussian (hyper-ellipsoidal subspace clusters), or uniform distribution within a specified range (hyper-rectangular subspace clusters). The values unrelated to subspace clusters (boxes labelled as Uniform Values) are sampled from a uniform distribution within the minimum and maximum range of the data set. Since the within-cluster values and irrelevant values are within the same range in our data sets (between 0 and 1), they therefore overlap, and make the task of distinguishing relevant attributes more difficult. Such a framework follows from that adopted in earlier benchmarking studies [14, 15].

To evaluate the performance of ESC artificial data sets are generated utilizing 50 to 1000 attributes, and 1000 to 4000 data points, Table 1. These data sets contain 20% to 90% irrelevant attributes. ESC is compared against two of the most popular subspace clustering algorithms: MINECLUS [23] and PROCLUS [1], and a full-space clustering algorithm, EM [13], one of the most cited clustering algorithms in the literature. All three baseline clustering algorithms assume their respective WEKA implementations.[4]

We evaluate ESC and its competitors with respect to four categories of performance measure: accuracy, simplicity, relevancy, and efficiency. Each performance category has one or more performance measure(s). The accuracy category includes detection rate, accuracy, and purity. Class-wise detection rate (DR) is simply an improvement upon accuracy measure to give degenerate solutions, which assign all instances to the majority class, lower detection rate. DR is defined as:

$$DR(CS_i) = \frac{\sum_{j=1}^{q} DR_j}{q} \tag{2}$$

[4] http://www.cs.waikato.ac.nz/weka/

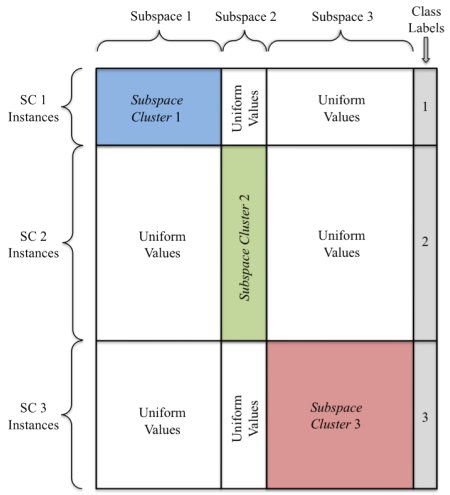

Figure 4: Example visualization of data set with subspaces. The 'Class label' indicates cluster membership and is only used for post training evaluation along with classification style performance metrics such as Detection Rate.

where $DR_j = \frac{TP_j}{TP_j+FN_j}$ is the detection rate of class j. Simplicity is formed by cluster count of a CS, total number of (unique) attributes indexed by a CS, and the number of attributes per subspace cluster. Relevancy includes the ratio of irrelevant attributes over relevant attributes as indexed by a partitioning and also the ratio of irrelevant attributes indexed by a CS over total relevant attributes in a data set. Efficiency is represented by algorithm runtime on a common CPU.

ESC is run 50 times each data set with different seed numbers. The 'centre' individual[5] on the final Pareto front is selected at the end of each run, making a pool of 50 ESC solutions for each data set. We visualize the distribution of values for each performance measure based on this pool using violin plots. A sample set of violin plots is shown in Figure 5. The median values of this distribution provide the basis for the summary performance plots of Figures 6 to 9. The 1-*d* clustering algorithm used in process 1 is EM [13], and the EMO algorithm used in process 3 is the modified NSGA-II developed by Jensen [10]. The selection of the EM algorithm for identifying 1-*d* centroids mirrors the use of the EM algorithm as the representative full-space clustering algorithm.

ESC parameterization assumes a CS population of 100 and a maximum of 1,000 generations. All mutation probabilities assume an initial value of 0.1. ESC also provides for the declaration of a min / max range of attributes and min / max number of SCC per CS: respectively 2 (min) and 20 (max). As such this attempts to guard against long tails in the Pareto front of candidate solutions discovered by NSGA-II. Note also that the parameterization is common to all data sets, and as such better results might well be possible though data specific fine tuning.

To determine the optimal set of parameters for MINECLUS we run an extensive search over the range recommended by MINECLUS authors for α, β, and ω, and select the parameter setting that gives the best results *on each data set*. Since the true number of clusters is unknown in reality we run MINECLUS with different values for k. We use the range [2, 20] for k, thus mirroring that for ESC. Therefore values reported for MINECLUS represent the median of 19 runs on each data set.

PROCLUS requires priori knowledge for the average dimensionality of subspace clusters, l. For data sets where different subspace clusters have a varying number of attributes, we run PROCLUS with different values for l, enumerating attribute counts of all subspace clusters as well as the average dimensionality of all clusters. PROCLUS is also run for different values for $k \in [2, 20]$. In total 10 runs with different seed numbers are performed for each pair of k and l, therefore 19×10 runs are performed for PROCLUS on each data set if subspace clusters have similar dimensionality, and $19 \times 10 \times L$ runs if subspace clusters have different dimensionality; where L defines the number of different values used for l. Given that ESC actually needs to discover these parameters during learning, it appears to be something of a moot point that PROCLUS only requires two parameters.

EM is capable of finding the number of clusters (but assumes that all the attributes appear in each cluster), so there is no explicit need to provide a value for k a priori. However, similar to MineClus, EM has learning parameters that were retuned for each data set. Once the optimized values for all parameters are found (for each data set) we run EM with 10 different seed numbers and report the median value of 10 runs for each performance

[5]The centre of a bi-objective Pareto front (PF) is merely the individual $|F|/2$ where $|F|$ is the number of individuals on the PF and F is the (dominance) ordered set of PF individuals at the last generation.

Table 1: Summary of benchmarking data sets.

Data set	Total Attr # Min Dim #	Total Inst # Max Dim #	Total SCC # Min Inst %	Noise Attr # Max Dim %
50D4C	50	1289	4	10
Spherical	10	10	8	43
200D5C	200	2000	5	50
Rectangular	10	50	10	25
500D4C	500	1286	4	100
Ellipsoidal	100	100	13	32
800D10C	800	3814	10	240
Mixed	10	100	4	16
1000D10C	1000	2729	10	900
Spherical	10	10	3	20

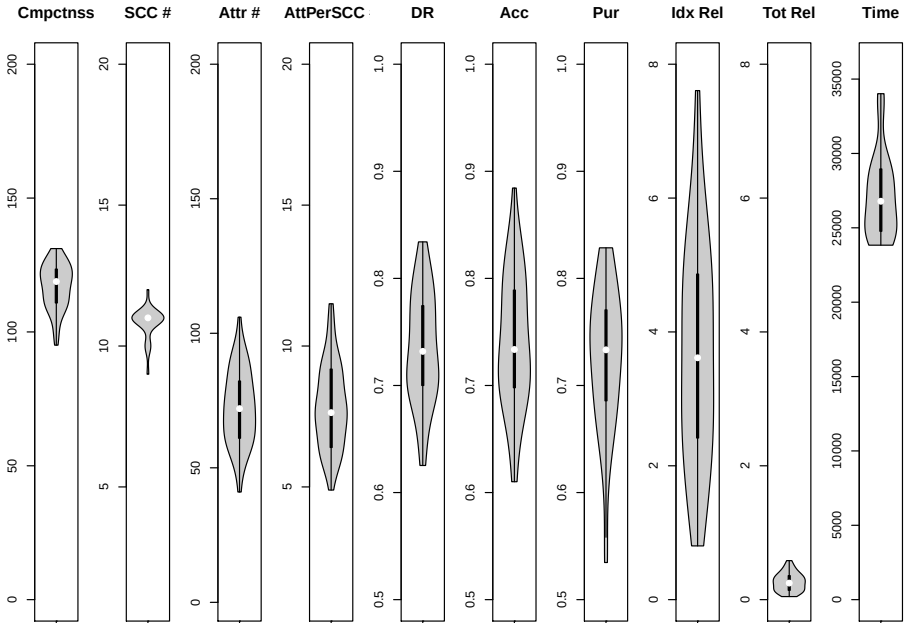

Figure 5: ESC algorithm performance distributions on the 800D10C data set over 50 runs.

measure i.e., EM is reported following the identification of data specific parameterization.

4.2 Experimental Results

All data sets in our work (Table 1) are identified using the convention: $XDYC$, where X represents the dimensionality and Y specifies the cluster count of data set, e.g. 1000D10C is a 1000-dimensional data set with 10 subspace clusters embedded in it. As noted above the data sets have 50 to 1000 dimensions, 1000 to 4000 instances, subspaces with 10 to 100 attribute count, and 3% to 43% coverage per subspace cluster. Within-cluster values are sampled from different distributions making hyper-spherical, hyper-rectangular, hyper-ellipsoidal, and mixed subspace clusters. Table 1 summarizes data sets properties.

A sample violin plot depicting the distribution of each performance w.r.t. the 50 ESC solutions on $800D10C$ is shown in Figure 5. The first column shows the primary (*compactness*) objective values. Columns 2, 3 and 4 report simplicity category measures: subspace cluster centroids count (SCC#), total number of attributes indexed by CS (Attr#), and average number of attributes per SCC (AttPerSCC#). Three accuracy category measures appear in columns 5, 6 and 7: detection rate (DR), accuracy (Acc), and purity of a CS (Pur). The ratio of irrelevant attributes to indexed relevant attributes (Idx Rel), and the ratio of irrelevant attributes to total relevant attributes (Tot Rel) represent the relevancy measures (columns 8 and 9) and are followed by time column representing the efficiency of the algorithm in terms of CPU time in seconds.

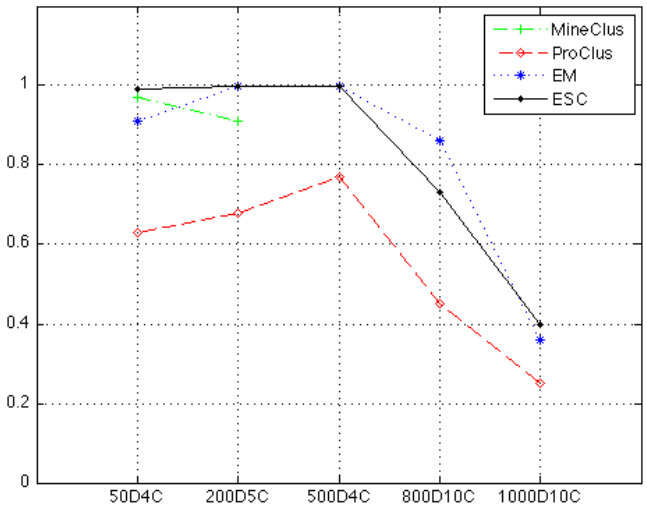

Figure 6: Median of detection rate as a representative of accuracy measures category for different methods.

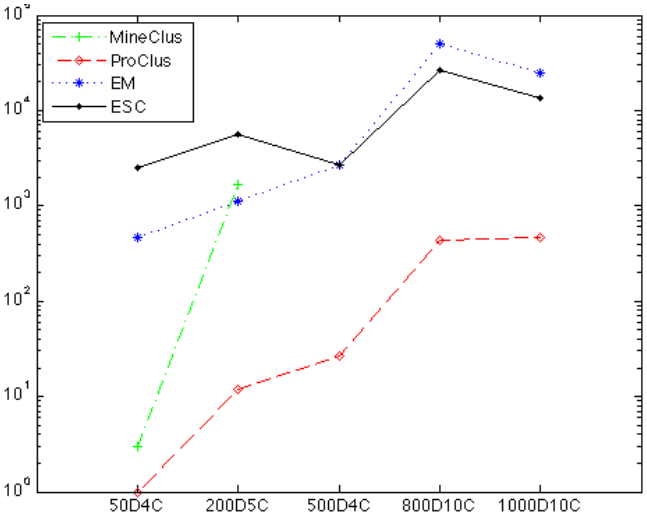

Figure 7: Median of (log) runtime for different methods on a common CPU platform.

In order to summarize this information for a meaningful comparison between the four methods and five data sets we utilize a representative measure from each category, and use the median values to establish a performance point for each method over all data sets. Class-wise detection rate (DR) will represent the accuracy category, total attribute count of a CS (Attr #) represents partitioning simplicity, the relevancy of a CS is determined by the ratio of relevant over irrelevant attributes (Idx Rel), and efficiency of a CS is shown by algorithm runtime. Figures 6 to 9 summarize performance under each of the four performance metrics across the clustering algorithms and data sets.

As Figure 6 shows, out of the five data sets, ESC and EM most consistently return solutions with the strongest DR. Naturally, ESC does so without data set specific parameter tuning (required for EM) and addresses the identification of subspaces (EM solutions utilize all attributes). Conversely, PROCLUS always produces the poorest solutions, and MINECLUS fails to return results for data sets with more than 200 attributes.

The runtime curve of MINECLUS in Figure 7 clearly shows a strong rise as dimensionality increases from 50 to 200 attributes precluding its utility on the larger data sets. PROCLUS is by far the fastest method among the four, however, it returns the least accurate results. ESC appears to scale gracefully as dimensionality increases. Thus, ESC takes almost the same time producing results for the 50 dimensional data set as the 500 dimensional data set. Indeed it appears that the ESC runtime is strongly dependent on data set cardinality rather than dimensionality. This is illustrated by comparing the cardinality of the 800- versus 1000-d data sets: the runtime of the latter is shorter care of its smaller instance count, or 2103 versus 3814.

Cluster simplicity curves are summarized in Figure 8. ESC consistently provides very simple results with no

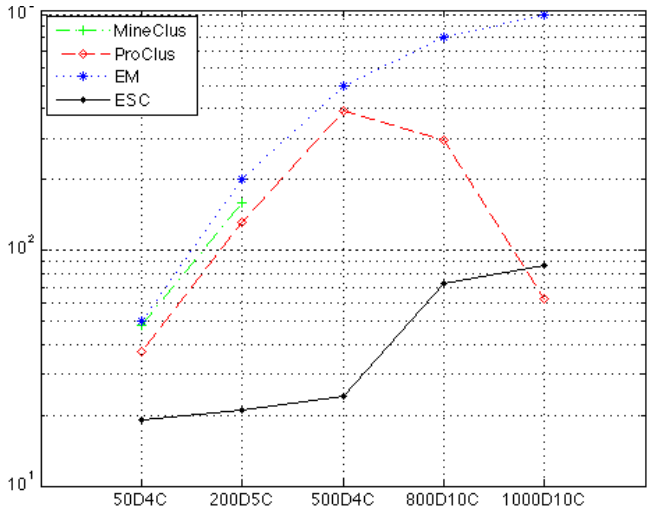

Figure 8: Median of (log) attribute count as a representative of simplicity measures category for different methods.

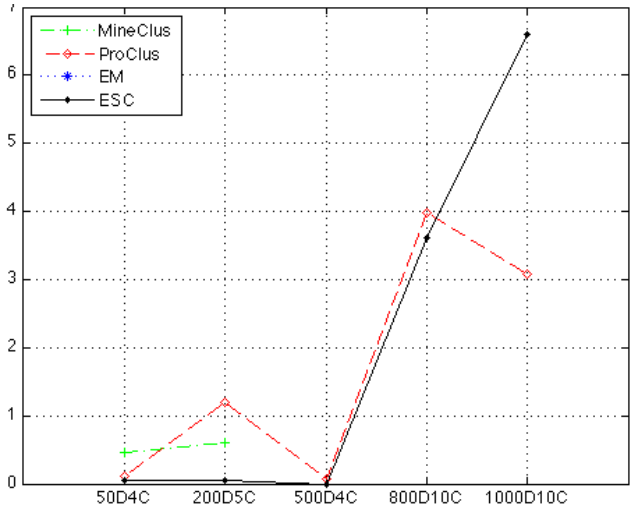

Figure 9: Median of irrelevant attribute count / relevant attribute count as a representative of relevancy measures category for different methods.

more than: 3 attributes per subspace cluster or a maximum of 20% of the total attributes per clustering solution for the lower-dimensional data sets (< 500); and no more than 10 attributes per subspace cluster or a maximum of 10% of total attributes per clustering solution for high-dimensional data sets (> 500). This, however has one exception: when ten 10-*d* subspace clusters are embedded in a 1000-*d* data set along with 900 irrelevant attributes. Since the average dimensionality of subspaces are known to PROCLUS as an input parameter, PROCLUS returns results that are slightly simpler than ESC results; albeit without matching the DR of ESC.

Attribute relevancy (Figure 9) is only evaluated for subspace clustering algorithms since EM uses all attributes for each cluster. Although ESC is the best approach in this regard on data sets between 50 and 800 dimensions; the relevancy is not as good on 800- and 1000-*d* data sets. The challenge in the 800-*d* data set is that the attribute support set of different clusters vary significantly from 10 to 100, whereas the challenge of the 1000-*d* data set is that 90% of attributes are not relevant to the clustering task. These two reasons underlie the inability of all methods to consistently identify relevant attribute subsets.

In addition to the performance measures mentioned above it should also be considered that it usually takes a long time to tune parameters for MINECLUS and EM, whereas parameters for PROCLUS and ESC are straightforward to identify. Moreover, MINECLUS, PROCLUS, and EM parameters are data set dependant; whereas as demonstrated here ESC parameters are robust across a set of data sets.

Another factor to recognize is that values used to generate all curves in Figures 6 through 9 are the medians of performance measures of solutions laying on the centre of the Pareto front from NSGA-II (process 3). We introduced a simple heuristic to sample a single solution from each run, where this has a bias for selecting the solution with the midpoint performing compactness–cluster count. It seems necessary to design a more sophisticated pruning method for ESC Pareto fronts. For example if we manually prune the pool of PF solutions to include only the best 10% of PF solutions, the median DR for 800-*d* data set, for instance, would improve from the current value of 0.73 to 0.84, and from 0.40 to 0.48 for the 1000-*d* data set.

5 Conclusion

A new method for subspace clustering is presented. This work utilizes a hierarchical data structure to assemble subspace cluster centroids from a grid of 1-*d* cluster centroids. The task of finding the best combination of subspace cluster centroids is converted into a combinatorial search. To do so, an endosymbiotic relation is adopted for defining a population of SCC independently from the population of candidate cluster solutions; thus each CS represents a combination of individuals from the SCC population. ESC is applied to a variety of artificial synthetic data sets ranging from 50 to 1000 attributes, and 1000 to 4000 instances, with different characteristics. ESC solutions demonstrated better accuracy and simplicitly compared to its competitors in most cases. Moreover, the computational cost of ESC appears to be a function of exemplar count as opposed to dimensionality.

In [7] connectivity was demonstrated to be an effective secondary objective in interaction with the primary compactness objective. However, estimating connectivity under a subspace clustering context appears to be prohibitively expensive [21]. Specifically, the neighbours of each data instance need to be detected anew for each subspace. Due to this limitation connectivity is not scalable with increasing instance count. One solution to this problem might be to adopt a *subsampling* algorithm to reduce the number of instances involved in evaluating the connectivity objective of a clustering solution. In other words a subset of instances can be used instead of all instances to calculate the connectivity of a clustering solution. Future work will investigate the impact of adopting such a scheme.

Acknowledgment

This work is supported by ISSNet - NSERC Strategic Network and NSERC CRD programs. All research was conducted at the Dalhousie Faculty of Computer Science NIMS Laboratory, http://www.cs.dal.ca/projectx.

References

[1] Charu C. Aggarwal, Joel L. Wolf, Philip S. Yu, Cecilia Procopiuc, and Jong Soo Park. Fast algorithms for projected clustering. In *ACM SIGMOD International Conference on Management of Data*, SIGMOD '99, pages 61–72, New York, NY, USA, 1999. ACM.

[2] Mihael Ankerst, Markus M. Breunig, Hans-Peter Kriegel, and Jörg Sander. Optics: ordering points to identify the clustering structure. In *ACM International Conference on Management of Data*, pages 49–60, New York, NY, USA, 1999. ACM.

[3] C. Bacquet, A. N. Zincir-Heywood, and M. I. Heywood. Genetic optimization and hierarchical clustering applied to encrypted traffic identification. In *IEEE Symposium on Computational Intelligence in Cyber Security*, pages 194–201, 2011.

[4] J. M. Daida, C. S. Grasso, S. A. Stanhope, and S. J. Ross. Symbionticism and complex adaptive systems I: Implications of having symbiosis occur in nature. In *Proceedings of the Annual Conference on Evolutionary Programming*, pages 177–186. MIT Press, 1996.

[5] K. Deb. *Multi-Objective Optimization using Evolutionary Algorithms*. John-Wiley and Sons, 2002.

[6] E. W. Forgy. Cluster analysis of multivariate data : efficiency versus interpretability of classifications. *Biometrics*, 21:768–769, 1965.

[7] J. Handl and J. Knowles. An evolutionary approach to multiobjective clustering. *IEEE Transactions on Evolutionary Computation*, 11(1):56 –76, feb. 2007.

[8] M. I. Heywood and P. Lichodzijewski. Symbiogenesis as a mechanism for building complex adaptive systems: A review. In *EvoApplications: Part 1*, volume 6024 of *LNCS*, pages 51–60. Springer, 2010.

[9] E. Hruschka, R. Campello, A. Freitas, and A. de Carvalho. A survey of evolutionary algorithms for clustering. *IEEE Transactions on Systems, Man, and Cybernetics: Part C*, 39(2):133–155, 2009.

[10] M.T. Jensen. Reducing the run-time complexity of multiobjective EAs: The NSGA-II and other algorithms. *IEEE Transactions on Evolutionary Computation*, 7(5):503 – 515, oct. 2003.

[11] Teuvo Kohonen. *Self-Organizing Maps*, volume 30 of *Springer Series in Information Sciences*. Springer-Verlag, 2001.

[12] Y. Lu, S. Wang, S. Li, and C. Zhou. Particle swarm optimizer for variable weighting in clustering high-dimensional data. *Machine Learning*, 82(1):43–70, 2011.

[13] G. McLachlan and T. Krishnan. *The EM Algorithm and Extensions*. Wiley-Interscience Publication, 1997.

[14] Gabriela Moise, Arthur Zimek, Peer KrÃűger, Hans-Peter Kriegel, and JÃűrg Sander. Subspace and projected clustering: experimental evaluation and analysis. *Knowledge and Information Systems*, 21:299–326, 2009.

[15] Emmanuel Müller, Stephan Günnemann, Ira Assent, and Thomas Seidl. Evaluating clustering in subspace projections of high dimensional data. *International Conference on Very Large Data Bases*, 2:1270–1281, August 2009.

[16] Lance Parsons, Ehtesham Haque, and Huan Liu. Subspace clustering for high dimensional data: a review. *ACM SIGKDD Explorations Newsletter*, 6:90–105, June 2004.

[17] Anne Patrikainen and Marina Meila. Comparing subspace clusterings. *IEEE Transactions on Knowledge and Data Engineering*, 18:902–916, 2006.

[18] D. Pelleg and A. Moore. X-means: Extending k-means with efficient estimation of the number of clusters. In *International Conference on Machine Learning*, pages 727–734, 2000.

[19] T. Piatrik and E. Izquierdo. Subspace clustering of images using ant colony optimization. In *IEEE International Conference on Image Processing*, pages 229–232, 2009.

[20] I. A. Sarafis, P. W. Trinder, and A. M. Z. Zalzala. Towards effective subspace clustering with an evolutionary algorithm. In *IEEE Congress on Evolutionary Computation*, pages 797–806, 2003.

[21] A. Vahdat, M.I. Heywood, and A.N. Zincir-Heywood. Bottom-up evolutionary subspace clustering. In *IEEE Congress on Evolutionary Computation*, pages 1 –8, july 2010.

[22] E. Vorhees. The effectiveness and efficiency of agglomerative hierarchical clustering in document retrieval. Phd thesis, Cornell University, Department of Computer Science, Ithaca, NY, USA, 1985.

[23] Man Lung Yiu and Nikos Mamoulis. Frequent-pattern based iterative projected clustering. *IEEE International Conference on Data Mining*, 0:689, 2003.