On Evolutionary Subspace Clustering with Symbiosis

Ali Vahdat[1] and Malcolm I. Heywood[1]

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

Evolutionary Intelligence 6(4):229–256. Springer 2014
The final publication is available at
`http://link.springer.com/article/10.1007\%2Fs12065-013-0103-1`

Abstract

Subspace clustering identifies the attribute support for each cluster as well as identifying the location and number of clusters. In the most general case, attributes associated with each cluster could be unique. A multi-objective evolutionary method is proposed to identify the unique attribute support of each cluster while detecting its data instances. The proposed algorithm, Symbiotic Evolutionary Subspace Clustering (S-ESC) borrows from 'symbiosis' in the sense that each clustering solution is defined in terms of a *host* (single member of the host population) and a number of coevolved cluster centroids (or *symbionts* in an independent symbiont population). Symbionts define clusters and therefore attribute subspaces, whereas hosts define sets of clusters to constitute a non-degenerate solution. The symbiotic representation of S-ESC is the key to making it scalable to high-dimensional datasets, while an integrated subsampling process makes it scalable to tasks with a large number of data items. Benchmarking is performed against a test suite of 59 subspace clustering tasks with 4 well known comparator algorithms from both the full-dimensional and subspace clustering literature: EM, MINECLUS, PROCLUS, STATPC. Performance of the S-ESC algorithm was found to be robust across a wide cross-section of properties with a common parameterization utilized throughout. This was not the case for the comparator algorithms. Specifically, performance could be sensitive to the particular data distribution or parameter sweeps might be necessary to provide comparable performance. An additional evaluation is performed against a non-symbiotic GA, with S-ESC still returning superior clustering solutions.

1 Introduction

Subspace clustering represents one of the most general forms of the clustering task [28, 29, 24, 25, 17, 18, 36]. Specifically, not only are the number of clusters and type of distribution supporting clusters unknown, but the attribute support for clusters also requires discovery. The underlying motivation for such a requirement is that applications are being increasingly encountered that are attribute rich, potentially resulting in high attribute redundancies. At the very least this can pose drawbacks for distance metrics used to establish the quality of candidate solutions [28]. However, this is only one factor in a set of inter-related factors associated with the curse of dimensionality as viewed from the perspective of subspace clustering [17, 18, 36] (Section 2).

One way of designing subspace clustering algorithms in general is through the concept of frequent itemset mining [3]. The general goal of which is to construct a partitioning of each attribute. This forms the basis for a lattice or grid. Subspace clusters are then constructed "bottom-up" through a combinatorial search conducted with respect to the lattice. In short, decisions are first made with respect to the appropriate attributes, and then clusters formed from the resulting co-occurrences. Such a framework will be assumed in this work and provides the basis for distinguishing between the role of a symbiont and a host. Thus, candidate cluster solutions (hosts) and attributes defining each cluster subspace (symbionts) are defined by independent populations.

From an evolutionary perspective, symbiosis is defined as the close and often long-term interaction between different biological species in which both species benefit from each other [21]. Unlike competitive or cooperative forms of coevolution, symbiosis has been specifically associated with the potential development of higher level organisms from independent species cf., the major transitions of evolution [7]. Thus, the capability to aggregate together 'lower level' symbionts into 'higher level' hosts represents a egalitarian transition capable of describing, for example, the origin of mitochondria within eukaryotic cells [32].[1] Such a process also introduces the concept of "levels of selection" [27] in which fitness is either evaluated at the level of the symbiont or host.

[1]Queller identifies reproductive fission as the second mechanism by which major transitions appear.

In this work, we are not interested in attempting to model the *transition* from groups of (lower level) symbionts into a (higher level) host (see for example [40]), but instead are interested in investigating how the division of duties implicit in the host–symbiont partnership provides a potential solution to the subspace clustering task. Thus, fitness is only ever evaluated at the level of hosts. Here we propose a hybrid bottom-up symbiotic framework for the subspace clustering task, hereafter Symbiotic Evolutionary Subspace Clustering or S-ESC. Each host represents a partitioning or clustering solution (CS), and each cluster centroid (CC) is defined by a symbiont. The same symbiont may exist in multiple hosts. The separation of host and symbiont populations makes explicit the role of different variation operators, whereas the symbiont population size varies under the control of a fixed size host population.

Empirical evaluation will be performed relative to a common set of benchmarks proposed by the study of Moise *et al.* [24]. This provides the basis for conducting a comparison against state-of-art algorithms under common benchmarking conditions. In part due to the comparatively low dimensionality of datasets in [24] we generate 5 more datasets ranging from 50 to 1000 dimensions. Competitor algorithms are chosen from popular full-dimensional and previously proposed subspace clustering algorithms. The resulting empirical evaluation employs a total of 59 datasets (54 datasets from Moise survey and an additional 5 datasets generated to complement those of Moise) and evaluates with respect to cluster accuracy, solution simplicity and computational requirements. Particular attention will be applied to factors such as parameter–dataset sensitivity and the impact of 'noisy' data.

The paper develops as follows: Section 2 begins by providing the background to the subspace clustering task that this research is specifically interested in addressing. To do so we review the categorizations established in recent survey articles; in particular [17, 18, 36]. Relative to this categorization we will then be able to establish: 1) what features the proposed algorithm should have; and 2) where previous approaches from evolutionary computation lie. Details of the S-ESC algorithm as proposed in this paper appear in Section 4. Section 5 details the benchmarking methodology with results following in Section 6. Conclusions are drawn in Section 7 and Appendices provide a summary of the non-symbiotic ESC also employed during the empirical evaluation and additional results.

2 Background and Related Works

Since the publication of CLIQUE by Agrawal *et al.* in 1998 [2] works pertaining to the basic objective of discovering attribute support as well as cluster location have appeared under multiple nomenclature: projected clustering [1], projective clustering [31], bi-clustering [16], and co-clustering [8] to name a few. In the following we aim to be more specific about what precisely we take to imply by the subspace clustering concept.

Developments in subspace clustering have been reviewed multiple times over the last 10 years [28, 29, 24, 25, 17, 18, 36]. In order to develop the topic here we will make use of the framework established by Kriegel *et al.* in particular (e.g., [17, 18]),[2] while noting where this builds on previous surveys. Moreover, we will use the categorization of subspace clustering established by Kriegel *et al.* to discuss previously published approaches that explicitly make use of methods from evolutionary computation, where there is little if any mention of evolutionary methods in the current survey literature. Conversely, the surveys by Moise *et al.* and Muller *et al.* pursue a benchmarking approach to their work [24, 25]. Recommendations from these surveys will be used to inform the selection of comparator algorithms and dataset for benchmarking purposes.

2.1 Curse of Dimensionality Under Cluster Subspaces

The framework of Kriegel *et al.* notes that the frequently used concept of the "curse of dimensionality" actually pertains to at least four different factors when applied to the task of cluster subspace identification:

- **Optimization problem:** the difficulty of approaching a computational task through global optimization increases exponentially with the number of variables.
- **Deterioration of expressiveness:** implies that frequently used similarity measures – such as the L_p norms – suffer from a decrease in their capacity to resolve differences between farthest and nearest points as dimensionality increases (see also [28, 29]).
- **Irrelevant attributes:** are those that do not contribute to the identification of a cluster 'embedded' within a larger dimensional space. As the number of instances, distribution or the number of attributes supporting the identification of such a cluster vary, some clusters are more difficult to locate than others. Such irrelevant

[2]Similar conclusions are drawn in the other recent survey by Sim *et al.* [36] albeit resulting in further refinements in the categorization.

attributes are also referred to as 'noise'. More generally an entire data instance might be an outlier (all attributes are noise). Naturally, as the total number of attributes increases then the cost of co-partitioning attributes and data instances also potentially increases.

- **Correlated attributes:** implies that correlations exist among subsets of data instances among two or more attributes. Potentially, a subspace cluster can now be defined orthogonal to the axis of any of the set of correlated attributes. The number of possible correlations is naturally a function of the total number of attributes.

In summary, Kriegel *et al.* prioritize identifying irrelevant attributes and / or correlated attributes over the remaining two aspects of the curse of dimensionality. Moreover, this has implications for the design of appropriate benchmarking suites for evaluating subspace clustering algorithms [23, 24] (Section 5).

2.2 Defining Types of Subspaces

Following from the properties identified above – that is the role of irrelevant and correlated attributes – then Kriegel *et al.* observe that there are two distinct types of distributions resulting in subspaces, either axis-parallel or arbitrarily oriented subspaces.

Axis-parallel subspaces follow the intuition that the variance of data within irrelevant attributes is high or uniformly distributed. Conversely, in the case of attributes relevant to a subspace, then there are at least some data instances described with a sufficiently small variance to enable detection within a background of other data i.e., those with distributions of a wider variance.

Arbitrarily orientated subspaces represent the more general case in which some subset of data points for two or more attributes define a (typically linear) dependency between the subset of attributes. In this case a subspace is defined orthogonal to the linear dependency and more difficult to detect in terms of the projection on individual attributes.

In addition to the above two scenarios, Kriegel *et al.* recognize a third category, that of 'special cases'. In this case, a wide range of *application specific* algorithms have been developed. For example, as in the case of information retrieval from document repositories or analysis of microarray data. Such algorithms – often referred to as biclustering, co-clustering, two-mode clustering – make use of task specific distance metrics or prioritize a capacity for addressing very sparse attribute spaces (see [17, 18]).

In this research we explicitly assume the axis-parallel case. Such a task definition leads to several algorithmic insights that will guide the approach taken in the proposed S-ESC algorithm. Moreover, there is a wider availability of benchmark datasets and software implementing alternative algorithms for the axis-parallel case which will provide greater scope for empirical evaluation. In the following discussion of background we will therefore concentrate on methods associated with the axis-parallel scenario.[3]

2.3 Categorization of Clustering Methods for Axis-parallel Subspaces

Possibly the simplest categorization of subspace clustering algorithms associated with axis-parallel subspaces is binary: top down vs. bottom up. **Top down** methods assume clusters defined relative to the entire attribute space and attempt to incrementally reduce the dimension of attributes per cluster. In order to minimize the impact of the factors associated with the curse of dimensionality (Section 2.1), heuristics are employed; for example, regarding the sampling of data points. Conversely, **bottom up** methods attempt to make use of attribute specific information such as frequent itemset mining as popularized by the APRIORI algorithm [3], in order to recursively construct groups of attributes with common itemsets. Since the original use of this categorization by [28], greater algorithmic diversity has prompted a more refined categorization [17, 18, 36]:[4]

1. Projected (or hard k-medoid) clustering – a form of top down clustering – attempt to identify tuples (O_i, D_i) where O_i are data instances belonging to cluster i and D_i is the corresponding subset of attributes.[5] Given that the method attempts to construct clusters using such a direct approach, it is necessary to provide sufficient prior information or assumptions to reduce the impact of the curse of dimensionality. Thus, in

[3]PCA and Hough transform based methods appear to currently be the dominant approaches for the case of arbitrarily orientated subspaces [18, 36].

[4]The earlier survey of Patrikainen and Meila recognized axis-aligned and non-axis-aligned categories [29].

[5]k-medoid and k-means algorithms have several similarities. The principle difference however, is that the k-medoid algorithm defines centroids in terms of a sample of data instances. Conversely, the k-means algorithm attempts to define clusters in terms of co-ordinates representing the centroids directly.

the case of PROCLUS [1], one of the most successful and widely utilized cases of projected clustering, it is assumed that the number of clusters is known and average attribute count per cluster is known a priori.

2. Soft projected clustering – are optimization algorithms that employ some form of k-means clustering. As such they incorporate a weight into the distance measure. This implies that no attribute is explicitly excluded i.e., *clusters are always full space clusters* albeit with some attributes with potentially a very low weighting. Kriegel *et al.* note that such methods are variants of EM clustering [17, 18].

3. Subspace clustering – explicitly pursues the goal of finding clusters in all subspaces of the original attribute space. This category of methods generally need to pursue some form of a bottom up approach in order to address the various aspects of the curse of dimensionality. Thus for example, CLIQUE employs a grid based scheme in which axis parallel grids are constructed from equal sized bins [2]. Bins with a sufficient number of data instances are considered to be dense. A cluster is then identified as a set of dense adjacent bins. Difficulties appear in practice when the distributions representing subspaces are not aligned with an attribute axis or as a consequence of trade offs between bin size and search space. Moreover, the anti-monotonic property employed for cluster identification in the APRIORI-like search heuristic central to many grid or density based subspace algorithms does not guarantee that only meaningful clusters are identified [23].

4. Hybrid approaches – The above identified approaches of 'subspace' and 'projection' clustering represent two algorithmic extremes. Hybrid approaches attempt to in some way blend the two approaches. Kriegel *et al.*, note that "the result is neither a clear partitioning [as in projection clustering] nor an enumeration of all clusters in all subspaces [as in subspace clustering]" [18].

The binary and algorithmic categorizations are generally used together, thus projected clustering can have instances that are top-down and others that are bottom-up.

2.4 Benchmarking Surveys

Relative to the approach taken by the survey paper of Muller *et al.*, three categorizations were assumed: *cell-based*, *density-based* or *clustering oriented* approaches [25]. These three categories appear to be synonymous with (APRIORI-like) subspace clustering, projected clustering and soft projected clustering categories respectively. Muller *et al.* also integrated the implementation for various subspace clustering algorithms into the well known WEKA machine learning software, or Open-Subspace.[6] Properties such as accuracy, cluster distribution, coverage, entropy, F-measure, and runtime are used to compare 10 different algorithms on 23 datasets.

Given the algorithm ranking from the Muller survey, we will later adopt MINECLUS [41] and PROCLUS [1] in the baseline evaluation of the S-ESC algorithm in Section 6. Kriegel *et al.* consider MINECLUS and PROCLUS top-down algorithms from their hybrid and projected clustering categorizations [17].

Moise *et al.* made an extensive review of almost 20 subspace clustering methods and compared 10 of them against 6 full-dimensional clustering algorithms [24]. They systematically generated 54 datasets in order to consider the effects of almost 10 different parameters in a dataset e.g., the impact of the distribution that cluster points are drawn from, to the extent of clusters in their relevant attributes. STATPC [23] was shown to dominate all other methods with respect to F-measure on all experiments, while SSPC and P3C were identified as the runner-up algorithms. Kriegel *et al.* consider P3P and SSPC instances of bottom-up and top-down projected clustering algorithms respectively; whereas STATPC is a top-down hybrid clustering algorithm. We will utilize STATPC and Moise datasets for the benchmarking evaluation performed in this work.

2.5 Evolutionary Computation and Subspace Clustering

With regards to methods assuming a framework from evolutionary computation, we identify three algorithms in particular. Sarafis *et al.* proposed a grid-based representation, containing a variable number of non-overlapping rules [34], resulting in a gene length proportional to the overall data dimensionality. A fitness function rewards the maximization of the data covered by the rules of an individual. Specific attention is also placed on how the initial population is initialized and repair operators are necessary to ensure that children remain legal. From the perspective of the categorization of Kriegel *et al.* this would make the approach an example of top-down (i.e., gene length is proportional to data dimensionality) projection clustering.

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

Assent *et al.* describe an evolutionary subspace search approach to detect 'locally relevant' attributes for clustering [4]. In this work, a gene models a specific attribute. Thus a set of genes codes whether subspaces are contained. The population contains all subspaces of the current generation. The fitness function assumes a normalized entropy of subspaces to measure the clustering tendency i.e., fitness of independent subspaces acts as a surrogate for the overall fitness of a set of subspaces forming a clustering solution (see also [37]). To encourage the identification of more than a single local optima, a multi-objective approach is used. Niches formed within in the multi-objective characterization provide the basis for discovering different local subspace optima in the same population. Benchmarking was limited to a few datasets of up to 55 dimensions.

Lu *et al.* propose a particle swarm approach to solve the 'variable weighting problem' in soft subspace clustering [20]. The PSOVW algorithm utilizes a weighting for a k-means style algorithm, on which a change of variable weights is exponentially reflected and adopts three swarms: the weights, the velocity and the cluster centroid midpoints. Each individual takes the form of a $K \times D$ matrix, where K is the number of clusters, provided a priori, and D is dataset dimensionality. The synthetic data used by Lu *et al.* adopts normally distributed values with mean values in the range of [0, 100] for relevant attributes and uniformly distributed values in the range of [0, 10] for the irrelevant attributes. The method therefore corresponds to the (top-down) soft projected clustering categorization of Kriegel *et al.* and therefore does not explicitly result in cluster definitions with dimensions below that of the original task domain.

Zhu *et al.* proposed a multi-objective evolutionary algorithm based on soft subspace clustering (MOSSC) to simultaneously optimize the weighting of within-cluster compactness versus the weighting of between-cluster separation [42]. The final cluster labels of MOSSC are determined by integrating all non-dominated solutions through a clustering ensemble strategy [11]. Similar to PSOVW, a chromosome is represented by a $K \times D$ weighting matrix giving the weight w_{ij} to attribute j for cluster i. As per PSOVM, the number of clusters should be known a priori and provided to the algorithm. Benchmarking utilized the synthetic datasets from Lu *et al.*, with modification to enable the consideration of up to 50 dimensions and 850 data instances. Under the Kriegel *et al.* categorization this would also be a (top-down) soft projected clustering algorithm.

A recent work describes an iterative process for discovering attributes though instance identification using an evolutionary algorithm, given prior knowledge for a common subspace dimension [6]. Once a subspace cluster is found the associated set of attributes are *removed* and the algorithm is re-run in search for a new subspace cluster. Naturally, multiple clusters cannot share the same attribute. The method was evaluated on two datasets from the UCI repository: ionosphere with 34 dimensions and 351 instances, and iris with 4 attributes and 150 instances – thus little can be said for the ground truth of the resulting solutions. Evaluation with synthetically generated data was limited to 9 attributes and 400 instances.

Nourashrafeddin *et al.* propose the 'Esubclus' evolutionary method, mainly for the purpose of finding subspace clusters in text document clustering (i.e. finding relevant keywords to each document category) [26]. The authors provided their source code for comparisons, and we discuss this method further in Section 6.3.

Finally, aside from the need to address the 'subspace' issues, we also note that clustering algorithms with a single criterion tend to find only one category of cluster 'shape' and fail on the others. Thus, an evolutionary multi-objective optimization (EMO) algorithm can potentially increase the chances of finding subspace clusters from both scenarios simultaneously. The MOCK algorithm of Handl and Knowles pioneered the use of EMO in this respect, but assumed that all clusters were full dimensional [13].[7]

An earlier version of the authors approach to evolutionary subspace clustering[8] demonstrated that using an EMO with the MOCK objectives of *compactness* (to detect spherical clusters) and *connectivity* (promising for arbitrary cluster shape detection) aided in the identification of true cluster count [37]. However, the connectivity is much more expensive to evaluate than compactness. The work reported here therefore introduces subsampling and knee detection into the proposed S-ESC algorithm (S-ESC was initially suggested in [38]), and conducts a through benchmarking evaluation on multiple subspace benchmarking datasets.

3 Comparator Methods

Section 2.4 identified three non-evolutionary subspace clustering and one full space clustering algorithm as a representative set of algorithms for comparison, the characteristics of which are summarized below.

[7]For a recent review of evolutionary computation as applied to *full dimensional* clustering see [14].

[8]The earlier evolutionary approach to subspace clustering attempted to first build cluster centroids and then describe clustering solutions through two independent cycles of evolution. This is difficult to do because the 'performance' of cluster centroids is dependent on the clustering solutions (a group of cluster centroids), where such groups are undefined at the point of cluster centroid evolution.

3.1 MINECLUS [41]

MINECLUS is a bottom-up density-based subspace clustering algorithm which assumes hyper-cubic distribution for the subspace clusters. Four basic phases are assumed for building subspace clusters using MINECLUS: 1) frequent itemset mining generates candidate subspace clusters using the DOC algorithm [31]. 2) clusters of 'quality' below the μ value are pruned. 3) if a prior target of k clusters is specified, but an outcome of $> k$ clusters is given, then subspace clusters are merged. The merging heuristic attempts to define a common equivalent cluster based on the shared attributes between two subspace clusters. 4) cluster refinement attempts to reallocate exemplars that represent boarder line cases given knowledge of all the cluster locations. There are three parameters that are essential for defining clusters using MINECLUS: w, α and β. Parameter w is associated with cluster spread i.e., the maximum distance in attributes common to a subspace cluster. Cluster density is defined by $\alpha \in (0, 1]$, thus each cluster must have at least $\alpha \times N$ exemplars, where N is the number of data samples. The β parameter provides the threshold associated with the cluster quality metric as used during step 2 of the MINECLUS algorithm.

3.2 PROCLUS [1]

PROCLUS assumes a three stage algorithm consisting of initialization, iteration and refinement. The basic goal is to locate k exemplars (medoids) as the basis for k subspace clusters. At initialization a superset of $A \times k$ 'well separated' points are sampled. A greedy pruning heuristic reduces this to $M = B \times k$ candidate medoids. During the iterative phase, k medoids are sampled from M, and then the worst medoids are iteratively replaced with other points from M. In a sense a combinatorial search is being performed on the content from M for the best k medoids. Attribute support for each of the medoids is performed as a separate process in which the average distance between medoid and neighbouring points is used to identify the closest $k \times l$ attributes (l is a user specified parameter, along with k). Data can now be assigned to medoids using a hype-spherical distance metric as estimated relative to the subspace associated with each medoid. The final phase, refinement, follows the same process as assumed for MINECLUS.

3.3 STATPC [23]

STATPC approaches the subspace clustering task as that of finding statistically significant regions through optimization. Thus, given a prior significance level, α_0, then a hyper-rectangle (H) in subspace S is said to be statistically significant if more points appear in it than in the case of a uniform distribution. In order for this to be meaningful as the dimensionality increases, a Kolmogorov-Smironov goodness of fit test is used to determine whether points in H are uniformly distributed over all attributes. The key point of STATPC is to define a greedy process for searching through the vast number of possible hyper-rectangles in an efficient way. The scheme assumed for doing this is based on maintaining multiple lists of candidate subspaces that are incrementally refined. The strength in the approach appears through the explicit use of statistical measures of significance, however, key thresholds still need to be defined a priori.

3.4 EM [10]

The Expectation-Maximization (EM) algorithm assumes that the N instances of data are generated by K mixture densities i.e., clusters. The task is now to parameterize the mixture densities. Specifically, multivariate Gaussian densities are typically assumed, in which a Bayes formulation describes the posterior probability of assigning exemplars to a cluster. The goal is now to define an iterative process by which the parameters of the model can be identified from the available data. Much like the k-means algorithm, an initial guess is made and the corresponding 'expected' values for the cluster probabilities. Conversely, the second step attempts to 'maximize' the likelihood of the distribution(s) given the available data. The expectation and maximization steps are repeated until a suitable stopping criteria is identified. From the specific perspective of subspace clustering we note that the dimensionality of the attribute space is not explicitly reduced, but it could be the case that parameters associated with attributes that are in some way redundant might decay. The principle design parameter for prior specification is the correct number of clusters, k.

4 Methodology

As an overview, the symbiotic evolutionary subspace clustering (S-ESC) algorithm consists of four main components and two sub-components. Sections 4.1 to 4.5.2 will then develop the S-ESC algorithm in detail. The overall relationship between the various components is summarized in terms of Algorithm 1.

Algorithm 1 The S-ESC algorithm. H_{size} and P_{size} refer to the host population size and number of points subsampled by RSS respectively. gen is current generation and gen_{max} is the maximum number of generations.

```
1. Generate the grid using 1-d clustering
2. Initialize symbiont (CC) population
3. Initialize host (CS) population, linking hosts to symbionts
4. RSS: Select $P_{size}$ data points as active set points
5. While $gen < gen_{max}$
    (a) For $i = 1$ to $H_{size}$
          i. Find a parent host using a tournament selection of size 4
         ii. Clone the parent host
        iii. Apply SLM operator on cloned host
         iv. Randomly select a parent symbiont, within current host
          v. Clone the parent symbiont
         vi. Apply MLM operator on cloned symbiont
    (b) Evaluate host population
    (c) Sort $2 \times H_{size}$ hosts on fronts
    (d) Delete worst $H_{size}$ hosts
    (e) Delete any symbionts without host membership
    (f) Refresh $P_{size}$ active set points with RSS
    (g) $gen = gen + 1$
6. Apply knee detection to return champion host from among Pareto hosts
```

Component 1: generates a *D-dimensional grid* from the dataset using a regular clustering algorithm as applied to each attribute *independently*. The resulting discrete grid will serve as the starting point for the symbiotic process of subspace cluster identification. The purpose of this step is therefore to convert the task into one that can be addressed through a combinatorial search of a discrete rather than real valued search space. The process is performed *once* relative to the entire training repository. In this work we perform the 1-*d* attribute-wise clustering through the application of the EM algorithm as implemented in WEKA machine learning package [12]. Naturally, any regular clustering algorithm could be used for this purpose.

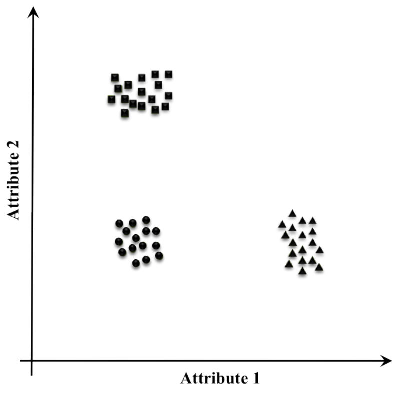
Component 2: of the S-ESC framework denotes the *symbiotic process* for coevolving cluster centroids (symbionts) as well as clustering solutions (hosts) alongside one another through a multi-objective framework. A cluster centroid (symbiont) takes the form of a point, composed from the centres of 1-*d* dense regions, from a cross-section of attributes (care of the grid from component 1). Conversely, a clustering solution (host) is composed of indexes to symbionts, and in essence partitions all data instances into subspace clusters identified by cluster centroids.

Component 3: denotes *fitness evaluation*. Fitness is only performed relative to host individuals (i.e., clustering solutions). Each host represents a candidate group of clusters, or a form of group selection. Fitness evaluation takes the form of a multi-objective Pareto methodology [13]: compactness and connectivity. Without loss of generality the NSGA-II algorithm is assumed [9] although any Pareto framework for multi-objective optimization could be utilized.

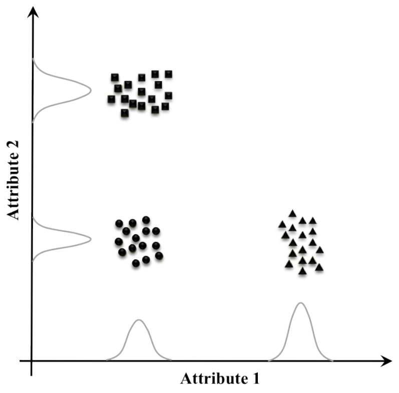
Component 4: assumes the asexual reproduction of hosts, thus variation is based on a set of mutation operators. A single-level mutation (SLM) operator is dedicated to modifying the host individuals, thus conducting a search through different cluster combinations. The second mutation operator, multi-level mutation (MLM), introduces variation within hosts, thus symbionts within a specific host undergo variation. In both cases host / symbiont variation is proceeded by cloning of the corresponding host / symbiont.

Sub-component A: introduces *random subset selection* (RSS) at the beginning of each generation, thus hosts are only evaluated against a subset of data instances (or active set), and not the whole dataset. This is in response to the increased cost of evaluation associated with the connectivity objective under subspace scenarios. Based on the definition of connectivity, M nearest data instances to each data instance need to be identified resulting a $O(N^2)$ cost.

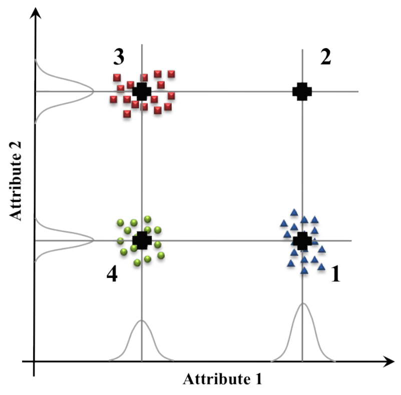
Sub-component B: addresses the post training selection of the most promising solution out of a pool of Pareto non-dominated solutions given by the evolutionary multi-objective process. Under a multi-objective perspective, better solutions on a Pareto front tend to lie where a small change in one objective value makes a big change on the other objective values [35]. Thereafter this post processing activity is referred to as *knee detection*.



(a) Before grid



(b) 1-*d* density identification



(c) After grid

Figure 1: Component 1 – D-dimensional grid generation. Three stages from 1(a) no grid to 1(b) identification of 1-*d* densities on a per attribute basis to 1(c) the resulting enumeration of all possible candidate cluster locations through the D-dimensional grid.

4.1 Grid Generation

In order to construct a grid from which candidate cluster centroids can be defined, a regular 'full space' clustering algorithm is applied to *each attribute alone* (Step 1, Algorithm 1). Thus, for a D-dimensional dataset, the standard clustering algorithm is called D times, once per attribute. A grid is then defined as the composition of all 1-*d* centroids. Figure 1 illustrates the process, thus Figure 1(a) represents the dataset before grid construction; whereas in Figure 1(b) the dense regions of each attribute are identified by attribute specific Gaussians and Figure 1(c) illustrates the superimposition of all 1-*d* centroids forming four 'candidates' for cluster centroid locations (**+** sign and numbered 1 to 4). Naturally, cluster centroids are free to index a variable number of attributes per cluster.

Any standard clustering algorithm would be relevant for this task so long as it does not require a priori definition for the number of centroids.[9] 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 dataset. In the benchmarking study of Section 6 the EM algorithm is assumed. Multiple runs of the proposed S-ESC algorithm

[9]For example both the X-means [30] or EM [22] clustering algorithms would be appropriate choices.

are performed relative to the same common grid. Moreover, since grid generation is only performed once per dataset regardless of the number of times S-ESC is run, grid generation component is considered a preprocessing step.

4.2 Representation and Initialization

S-ESC initialization begins by randomly generating a population of symbionts as the genetic material to be indexed by host individuals (Step 2, Algorithm 1) followed by randomly generating a population of hosts by linking each host to a number of symbionts from symbiont population (Step 3, Algorithm 1) and then selecting a subset of dataset instances (Step 4, Algorithm 1) as active set as will be discussed in Section 4.5.1.

Symbionts index dense regions (1-*d* cluster centroids) *within* attributes, utilizing a series of integer pairs, or $\langle a, c \rangle$; where a determines the attribute index, and c determines the 1-*d* centroid within the attribute. The encoding / decoding of a symbiont is summarized in Figure 2, two symbionts each defined in terms of two attributes. Clearly $CC1$ identifies grid location '1' whereas $CC2$ identifies grid location '2', moreover, it would be expected that the utility of $CC2$ is 'null'. Both symbionts will be treated as potential subspace cluster centroids. Naturally, in higher dimensions the number of integer pairs per symbiont is free to change, implying that CC dimensionally is dynamic. In practice, the search for candidate cluster centroids is limited to the interval $[2, 20]$ in order to avoid declaring 1-*d* cluster centroids (trivially degenerate) or cluster centroids with more than 20 dimensions. Naturally, if prior knowledge indicated that CC with dimensionality greater than 20 would be expected, the range could be modified appropriately. Moreover, such a representation lets a cluster centroid (symbiont) be expressed independently of dataset dimensionality or cardinality.

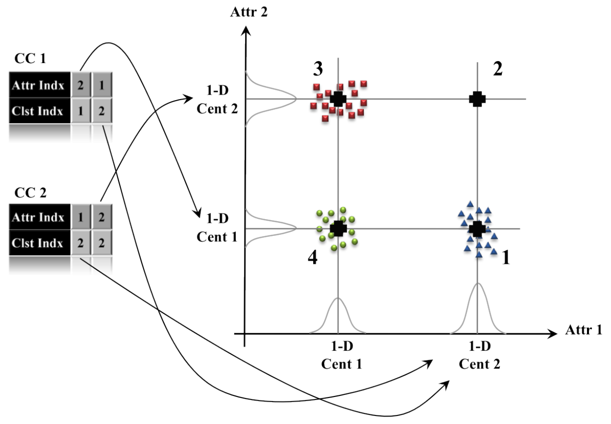


Figure 2: Representation for symbiont or CC individuals relative to a 2-d grid. Two symbionts $CC1$ and $CC2$ are (in this example) expressed in terms of a common pair of attributes, Attr1 and Attr2. $CC1$ consists of two integer pairs: $\langle 2, 1 \rangle$ denoting attribute 2, 1-*d* density 1, and, $\langle 1, 2 \rangle$ denoting attribute 1, 1-*d* density 2. The intersection of each attribute density denotes the symbiont CC location. Thus $CC1$ identifies location '1' in the grid and $CC2$ identifies location '2'.

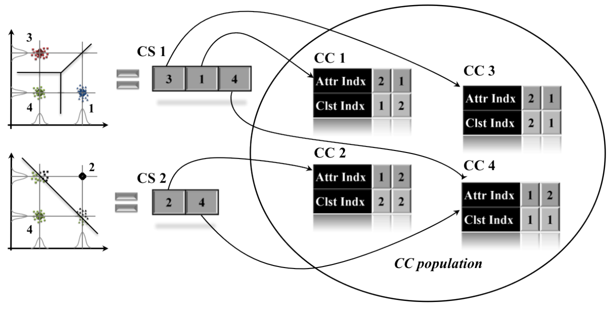


Figure 3: Host or CS representation. Host $CS1$ indexes three symbionts, therefore forming a clustering solution (CS) with three clusters. Host $CS2$ defines a CS from two CC symbionts. Clearly in this case the partitioning established by $CS1$ is significantly better than that of $CS2$ (LHS subfigure).

Hosts identify subsets of symbionts under a variable length representation. Each host is encoded as an integer vector, each integer indexing a symbiont. The same symbiont may appear in multiple hosts. Thus hosts conduct a search for good CC combinations. Two sample hosts, along with a population of four symbionts, are illustrated in Figure 3. Each host defines a partitioning of the data relative to its combination of candidate clusters, care of a nearest neighbour allocation of exemplars to host CC membership. Similar to the representation of symbionts, a minimum and maximum bound is placed on the number of symbionts a host can link to, i.e. the minimum and maximum clusters we expect in a dataset. These parameters are also adjustable based on any prior domain knowledge.

Group selection: Fitness is only evaluated explicitly at the 'level' of host individuals (Steps 5b , Algorithm 1). A host (clustering solution or CS) is formed by selecting a group of symbionts (cluster centroids or CC), therefore group selection is involved. Specifically, the fitness of any team (host) individual can be evaluated in one of two ways: multi-layer selection 1 (MLS1) and multi-layer selection 2 (MLS2) [27]. In MLS1 the fitness of a host is proportional to the sum / average of the symbiont fitness, i.e. each symbiont has a fitness which affects the final fitness of the host. On the other hand, in MLS2 the fitness of a host is the result of the interaction between the symbionts it indexes, regardless of how each symbiont performs individually, i.e. fitness is only ever evaluated at the host level. In this work MLS2 is assumed, thus potentially benefiting from group fitness exceeding the mere sum of its parts. Symbiont 'fitness' is therefore implicit. Should a symbiont *not* receive any host indexes then it is considered useless and deleted. The point at which such checks are made is linked to the breeder style of evolution (Steps 5d and 5e, Algorithm 1).

4.3 Multi-objective Evolutionary Optimization

At each generation μ host (CS) parents produce λ host (CS) offspring, where $\mu = \lambda = H_{size}$. The $\mu + \lambda$ CS individuals are then re-ranked under a Pareto multi-objective co-evolutionary algorithm (Step 5c, Algorithm 1) with only the best half retained as parents for the next generation.

A multi-objective formulation will be assumed based on compactness and connectivity; where this was first proposed by Handl and Knowles in a full dimensional clustering context [13]. Indeed, as will later become apparent in the empirical evaluation, subspace clustering algorithms generally appear to fail to address the significance of cluster shape. Minimizing the *compactness* of a host, in the extreme scenario, results in as many clusters as data instances. Minimizing *connectivity* of a host, on the other hand, tries to put all data instances into one cluster making one 'super' cluster. Therefore there is an obvious tension between these two objectives. This tension is a key factor to estimate the optimal number of clusters in a dataset [13]. The most significant drawback of utilizing a connectivity objective, however, is that under a subspace setting it is necessary to continuously re-evaluate the metric for each subspace [38], whereas under full-space clustering such a metric can be precomputed [13]. Unless this is addressed, it becomes a constraint on scaling to datasets with larger cardinality. With this in mind, a process for data subsampling will be introduced latter (Section 4.5.1).

Section 4.2 established that, prior to calculating the compactness of a host (CS), a nearest neighbour (NN) assignment of data instances is performed against the set of symbionts (CC) linked to the host (Figure 3). This nearest neighbour allocation determines which symbiont (linked to a host) each data instance is assigned to. The distance between an instance and the nearest symbiont is defined and normalized over the number of attributes supporting the symbiont. The overall **compactness** of a host is the sum of all distances between each instance and their nearest cluster centroid (symbiont):

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

where CC_i is the nearest CC to instance x_j, q is the number of CCs within CS, and $|CC_i|$ is the number of instances assigned to CC_i.

Connectivity of each point is a function of the distance to its neighbouring instances as well as their cluster labels. In other words, the farther instances are from instance x_i, the less they contribute to connectivity value, given they have different cluster labels [13]. Punishing connectivity by $1/j$ for j^{th} nearest instance to x_i (which has a different cluster label than x_i) captures this concept, whereas punishing connectivity by a fixed value of 1 implies that all neighbouring instances contribute equally to connectivity of an instance, regardless of their distance to x_i.

$$Con(CS) = \sum_{i=1}^{N} \sum_{j=1}^{M} \begin{cases} \frac{1}{j}; & \text{IF } \theta(x_i, NN_{ij}) \\ 0; & \text{otherwise} \end{cases} \tag{2}$$

where N is the exemplar count of the dataset; M is the number of nearest neighbour exemplars to x_i considered for connectivity which we define as $\max(10, 0.01 \times N)$; NN_{ij} is the jth nearest neighbour exemplar to x_i, and; $\theta(\cdot)$ is a function defined as:

$$\theta(x_i, NN_{ij}) = \nexists CC_k : x_i \in CC_k \cap NN_{ij} \in CC_k \tag{3}$$

A large value for connectivity indicates that there are a lot of neighbouring points placed in different clusters, whereas a small value of connectivity indicates that only a small number of neighbouring instances are assigned to different subspace clusters; thus, connectivity is to be minimized. Naturally, the nearest neighbours of each data point need to be determined over the attributes specific to the subspace cluster centroid. However the attribute support of each symbiont is potentially different in each generation and therefore the nearest neighbours require *rediscovery at each generation*.

Once both host and symbiont populations are initialized, the evolutionary loop of S-ESC begins (Step 5a, Algorithm 1). The S-ESC implimentation benchmarked here assumes the well known NSGA-II formulation for evolutionary multi-objective optimization (EMO) [9], and the code distribution of Jensen in particular [15]. However, any EMO might be assumed for this purpose.

NSGA-II uses a ranking system based on individual's Pareto front number and crowding distance [9]. At the end of each generation all (parent and offspring) individuals get sorted on different fronts in the objective space (Step 5c, Algorithm 1). Individuals on the Pareto front (rank 1) are kept to evolve further in next generations i.e., elitist archiving. If there is room for more individuals in the population, individuals from fronts ranked 2, 3, ... are also included in the population for next generation. If there is not enough room to include all individuals of a given rank, a *crowding* heuristic is invoked. Such a heuristic helps NSGA-II select the most diverse set of individuals by picking individuals that have the least number of neighbours on the same front.

4.4 Variation Operators

Variation operators maintain population diversity and introduce new genetic material to both host and symbiont populations (Steps 5(a)i to 5(a)vi, Algorithm 1). However, modifying a symbiont (CC) might not be desirable as it is likely to be shared by multiple host (CS) individuals. Thus, an asexual model is assumed in which a parent is first cloned and variation is only introduced relative to the cloned child. Parent CS individuals are selected by a tournament against three other CSs. Such a process is explicitly elitist. Two mutation operators are utilized in S-ESC; Single-Level Mutation (SLM) and Multi-Level Mutation (MLM). SLM modifies individuals only at the host 'level'; thus, it is only the links to symbionts within the cloned host that undergo variation. On the other hand MLM modifies the symbiont links within a host *as well as* one of the symbionts linked to the host under modification.

4.4.1 Single-Level Mutation

SLM (Step 5(a)iii, Algorithm 1) consists of 4 phases. A host (CS) parent is selected by tournament selection against three other hosts. The selected host (CS) is then cloned. Then the recently-cloned host (CS) potentially goes through three forms of mutation: *i*) remove a link to a currently included symbiont, *ii*) add a new symbiont link, and *iii*) replace a current symbiont link with a link to a symbiont from symbiont population. Naturally, the third form of mutation performs in one step what forms '*i*' and '*ii*' might achieve together.

The latter three phases (interchangeably called **atomic operators**) are applied with a probability of 1.0. These three atomic operators are applied again with a probability of 0.1, and while the test is true they are applied again with the threshold of application decreased by an order of magnitude, i.e. $0.1^2, 0.1^3, \ldots$. The process repeats until the test no longer returns true.

4.4.2 Multi-Level Mutation

MLM (Step 5(a)vi, Algorithm 1) unlike SLM acts on individuals from both host and symbiont populations. It has 6 phases and, as per SLM, starts by selecting a host (CS) parent among four hosts through tournament selection. Such a CS parent is selected, cloned, and both parent and clone are retained in the host population (Phase 1). With uniform probability a symbiont (CC) within the cloned host (CS) is selected (Phase 2) and cloned with both parent and cloned child retained in the symbiont population (Phase 3). The newly cloned symbiont goes through three mutation steps (alternatively **atomic operators**): *i*) remove an attribute (Phase 4), *ii*) add an attribute (Phase 5), and *iii*) replace an attribute's 1-*d* centroid (Phase 6). 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.

4.5 Pragmatics

The four components discussed above are the main building blocks of S-ESC [38]. However, two additional components are necessary for scaling S-ESC with respect to dataset cardinality (subsampling) and facilitating the post training identification of a 'champion' solution from a pool of equally-fit Pareto non-dominated solutions (knee detection). Neither of these issues were addressed in the earlier summary of the S-ESC algorithm [38]. The resulting code distribution is available for research purposes.[10]

4.5.1 Subsampling

At each generation the nearest neighbours of each data instance should be determined for the connectivity objective (Step 5f, Algorithm 1). Furthermore the nearest neighbours for each data instance x_i should be determined relative to the possibly unique attribute set of each symbiont. Data subsampling will therefore be introduced. Instead of including all N dataset instances when calculating connectivity (and compactness) objective(s) M data instances are selected with uniform probability and objectives are calculated relative to the selected subset. At each generation, the subsampling of M data instances is repeated; hereafter the entire process is denoted Random Subset Selection (RSS). Naturally, fitness evaluation is now decoupled from the cardinality of the original dataset, at the possible expense of accuracy associated with any single evaluation.

4.5.2 Knee Detection

EMO methods based on Pareto dominance are generally expected to provide a diverse and distributed set of non-dominated solutions. All these (non-dominated) solutions are potentially equally important, which makes selecting a single solution a nontrivial task. There is an abundance of literature on mechanisms for focusing EMO on specific areas of the Pareto front and / or how to select the winner solution [35, 33].

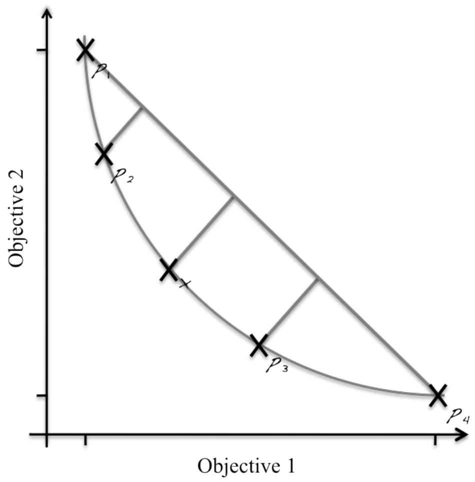


Figure 4: Example of Pareto front with champion identification through knee identification. The Pareto set $\{p_1, p_2, p_x, p_3, p_4\}$ denote the front of candidate solutions from which $\{p_1, p_4\}$ are the tails. The champion solution in this case is p_x

Knee detection represents one of the most established heuristics for identifying a single representative champion from the non-dominated set (Step 6, Algorithm 1). We adopt the method proposed by Schutze *et al.* [35] in which a three step process is assumed: 1) Find the two extreme 'tail' individuals of the Pareto front. 2) Form a diagonal line between them from which the tangent distance to all other individuals are evaluated. 3) The member of the non-dominated front with greatest (perpendicular) distance from the diagonal is promoted as the knee and assumed to be the champion individual. Figure 4 illustrates the process relative to a set of Pareto front individuals.

[10]http://web.cs.dal.ca/~mheywood/Code/S-ESC

The tail individuals p_1 and p_4 establish the diagonal and the individual with greatest distance from the diagonal, individual x, denotes the champion. Note, also, that although cluster validation metrics have been developed for the post training assessment of solutions from clustering algorithms, they assume a common attribute space. Such a constraint is not satisfied or desirable in the general case of subspace clustering as assumed in this work. Adopting a knee detection algorithm for this purpose is therefore more appropriate.

5 Evaluation Methodology

5.1 Data Standardization

Clustering algorithms generally assume attribute standardization in order to avoid introducing biases into the distance-based objectives, however, some applications are also sensitive to the nature of the standardization process [5]. Here we assume a linear standardization of attributes to the unit interval $[0, 1]$, where this is common to all algorithms benchmarked.

5.2 Why Artificial Datasets?

In order to evaluate the effectiveness of subspace clustering algorithms datasets with specific structure are required. This is generally a problem in the case of 'real-world' datasets. With this in mind, researchers usually design and generate artificial datasets of their own to show the various properties of subspace algorithms [28, 29, 24, 25].

Two different sets of synthetic datasets will be utilized in the following study. Firstly we employ the datasets provided by Moise *et al.* in their benchmarking survey of more than 10 different (axis parallel) subspace clustering algorithms on more than 50 datasets [24]. Specifically 9 properties were considered of relevance to subspace clustering in general (Table 1). Thus, the impact of:

1. assuming different distributions of cluster points in the relevant subspaces albeit limited to the case of Gaussian or Uniform distributions (we will return to the significance of this decision later);
2. assuming clusters with an equal number of relevant attributes versus clusters with varying numbers of relevant attributes;
3. changing the average number of relevant attributes per cluster;
4. the database dimensionality or total number of attributes (D);
5. dataset cardinality or total number of instances (N);
6. varying the number of clusters (K);
7. varying the average number of instances per cluster.
8. varying the extent of relevant attributes within the unity range;
9. varying the overlap between the extent of clusters in common relevant attributes, and;

Table 1 summarizes how the different datasets were designed to assess the above nine subspace clustering impact factors, resulting in a total of 54 datasets, hereafter called 'incremental' benchmarks. Labels are also declared for identifying each set of experiments. Also note that we first use the original incremental datasets with one modification. Since S-ESC (or for that matter all but the STATPC algorithm proposed by Moise *et al.*) do not explicitly filter for outliers[11], we remove the outliers in all datasets before commencing the evaluation.[12] Naturally, the resulting outlier free datasets still retain the noise terms associated with attributes not associated with a subspace. A separate set of experiments will be conducted to assess the impact of including the pure outlier data. For a more detailed description of the datasets refer to [24].

Although the incremental datasets of Moise *et al.* are designed to evaluate the effect of different parameters pertinent to subspace clustering, we see two shortcomings in particular. Firstly, the data dimensionality–cardinality is relatively low in most cases e.g., dimensionality $<= 50$ and cardinality ≈ 300. Applications increasingly

[11]Outliers are data instances for which all attribute data values represent noise.

[12]Outlier points are labeled as an extra cluster in incremental datasets and are therefore straightforward to explicitly remove.

have cardinalities in the thousands and / or hundreds of dimensions. Secondly, the generating distribution of points in the relevant attributes is limited. In the Moise experiments all clusters within a dataset use the same type of distribution for all clusters, e.g. when Gaussian distribution is being used, it is used for all clusters. These limitations motivate us to design datasets with higher dimensions, larger cardinality, and varying cluster distribution within the *same* dataset. Moreover, we also note that the uniform or Gaussian distributions assumed by Moise were of similar aspect ratios, thus favouring compactness style objective functions as opposed to rewarding the utility of a connectivity style objective.

A further 5 datasets are therefore introduced with between 50 and 1000 dimensions and 1000 to 4000 data instances with up to 10 clusters per dataset (Table 2); hereafter referred to as the 'large-scale' datasets. Attribute support for different clusters within a dataset ranges between 10 and 100. In the case of the '1000D' dataset 90% of attributes are irrelevant leaving only 10% of attributes actually relevant to the clustering task. Datasets are highly unbalanced e.g., in the '50D' dataset the smallest cluster has 8% of instances while the largest cluster has 43%. Under the '800D' dataset the smallest cluster support is 10 attributes while the largest cluster support is 100 attributes. Finally, different cluster distributions with different parameterizations are used to generate a dataset with mixed cluster distributions (800D). These datasets are available for research purposes.[13]

Table 1: Summary Properties of the incremental datasets of Moise. Parenthesis after dataset identifier denote the number of variants of a dataset. GE denotes Gaussian distribution with an equal number of attributes per cluster; GD denotes Gaussian distribution with a different number of attributes per cluster; Likewise for UE and UD but in terms of the uniform distribution.

Data Set	**Distrib-ution**	**Cluster Shape**	*D*	*N*	*K*	**Irr. Attr#**	**Irr. Attr%**	**Min Dim#**	**Max Dim#**	**Min Inst%**	**Max Inst%**
GE (7)	Gaus.	Sphere	50	240	5	40,30,20, 10,0,0,0	80,60,40, 20,0,0,0	2,4,6,8, 10,15,20	2,4,6,8, 10,15,20	17%	25%
GD (7)	Gaus.	Sphere	50	240	5	40,30,20, 10,0,0,0	80,60,40, 20,0,0,0	2,4,6,8, 10,15,20	2,4,6,8, 10,15,20	17%	25%
UE (7)	Uni.	Square	50	240	5	40,30,20, 10,0,0,0	80,60,40, 20,0,0,0	2,4,6,8, 10,15,20	2,4,6,8, 10,15,20	17%	25%
UD (7)	Uni.	Square	50	240	5	40,30,20, 10,0,0,0	80,60,40, 20,0,0,0	2,4,6,8, 10,15,20	2,4,6,8, 10,15,20	17%	25%
D (5)	Uni.	Square	20,35,50, 75,100	240	5	0,15,30, 55,80	0,33,60, 73,80	4	4	17%	25%
N (5)	Uni.	Square	50	80,240,400, 820,1650	5	30	60	4	4	17%	25%
K (4)	Uni.	Square	50	250	2,3, 4,5	42,38, 34,30	84,76, 68,60	4	4	20%	50%
Cluster Size (4)	Uni.	Square	50	145,200, 240,265	5	30	60	4	4	14%	25%
Extent (4)	Uni.	Rect.	50	240	5	30	60	4	4	17%	25%
Overlap (4)	Uni.	Square	50	250	2	42	84	4	4	17%	50%

Table 2: Summary Properties of the Large-scale datasets.

Data Set	**Distrib-ution**	**Cluster Shape**	*D*	*N*	*K*	**Irr. Attr#**	**Irr. Attr%**	**Min Dim#**	**Max Dim#**	**Min Inst%**	**Max Inst%**
50D4C	Gaussian	Spherical	50	1289	4	10	20%	10	10	8%	43%
200D5C	Uniform	Rectangular	200	2000	5	50	25%	10	50	10%	25%
500D4C	Gaussian	Ellipsoidal	500	1286	4	100	20%	100	100	13%	32%
800D10C	Mixed	Mixed	800	3814	10	240	30%	10	100	4%	16%
1000D10C	Gaussian	Spherical	1000	2729	10	900	90%	10	10	3%	20%

[13]http://web.cs.dal.ca/˜mheywood/Code/S-ESC

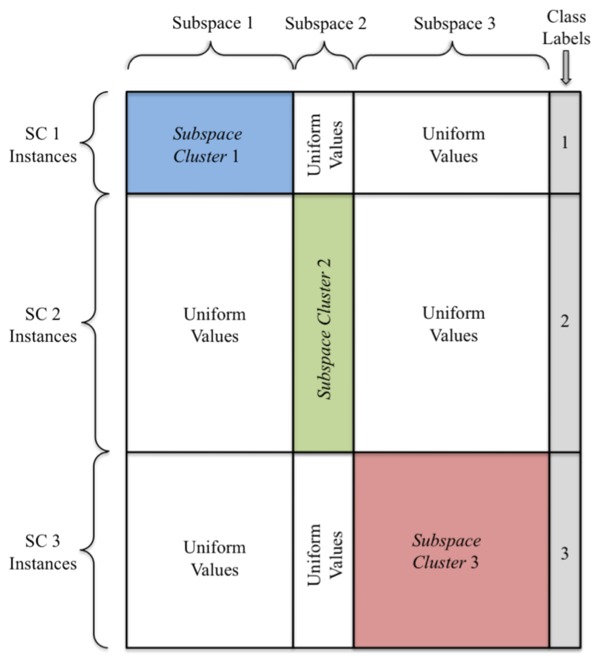


Figure 5: Example visualization of dataset with subspaces. The 'Class label' indicates cluster membership and is only used for post training evaluation along with classification style performance metrics such as F-measure.

5.3 What is The Data Generation Routine?

In order to generate datasets with known (axis parallel) subspace structure we take the following approach. Figure 5 shows a sample dataset with 3 subspace clusters embedded in it. The cluster points in relevant subspaces (or values inside coloured boxes) are sampled from different distributions with randomly generated parameters. These distributions can be Gaussian (which forms hyper-spherical subspace clusters), elongated Gaussian (producing hyper-ellipsoidal subspace clusters) or uniform distribution within a specified range (making hyper-rectangular subspace clusters). The values unrelated to subspace clusters (cluster points in irrelevant attributes or white boxes labelled as Uniform Values) are sampled from a uniform distribution within the minimum and maximum range of the dataset. Data points with relevant attributes and those with irrelevant attributes are within the same range [0,1] and therefore overlap non-trivially.

5.4 What Are The Post-training Performance Measures?

We evaluate S-ESC and its comparators with respect to *accuracy* through micro F-measure, *simplicity* through unique attribute count and algorithm *runtime* during cluster identification.

Accuracy: Similar to [24] we denote the embedded clusters in a dataset *input* clusters and the clusters found by a clustering algorithm as the *output* cluster. For each output cluster i, we determine the input cluster j_i with which it shares the largest number of data points. The precision and recall of each pair of matched input and output clusters are then calculated. F-measure of an output cluster i is the harmonic mean of its precision and recall. In this work we adopt the more specific case of the *micro* F-measure. This implies that not only are the clusters 'pure' but the exact number of clusters are found. Conversely, the more typically employed *macro* F-measure would 'group' all clusters of the same majority 'label' and merge the resulting F-measure. Thus, there might be many more clusters than actually required, but as long as they are pure, the F-measure would still be high. Previous researchers tended to assume use of the macro as opposed to the micro F-measure.

Simplicity: Simplicity of a solution is simply determined by the total number of unique attributes indexed, counting each attribute only once.

Runtime: refers to the time to execute the clustering algorithm on a common computing platform.[14] This does not include the cost of parameter sweeps. However, in the case of the EM algorithm a ten-fold cross validation is performed to identify the relevant cluster count and this is included in the estimate of runtime.

[14]16 core Intel Xeon 2.67Ghz server, 48GB RAM, Linux CentOS 5.5.

5.5 Parameterization

S-ESC is run with a fixed set of parameters for all datasets. The host population is set as 100. Symbiont population size is initialized at 100, however, this population is allowed to expand up to 5 times its initial size. Only 100 points are selected at each generation by RSS. Naturally, given priori knowledge of the cardinality (N) a user could choose to increase subset size to maintain fitness evaluation relative to a fixed fraction of the dataset. The minimum and maximum number of allowed attributes in a symbiont is set as 2 and 20 respectively with the same values defined for minimum and maximum number of clusters in a solution. Probabilities of atomic actions within SLM and MLM operators start with 1.0, and decrease to 0.1, 0.01, ... for the second, third, and further rounds of application.

S-ESC is run for 1,000 generations 50 times on each dataset, using different initializations. The knee individual of Pareto front represents the champion from each run; therefore we report the distribution of results for 50 knee solutions, each chosen from an independent run. S-ESC takes advantage of EM [10, 22] for 1-d density estimation (grid construction) where this takes the form of the implementation from WEKA [39]. Some experiments are shown in Section 6.4 to illustrate the parameter insensitivity of S-ESC.

Competitor algorithms: Benchmarking is performed against both full-dimensional and subspace categories of clustering algorithms. The EM algorithm is assumed as the representative of full-dimensional clustering algorithms [10, 22]. Properties that make it particularly appealing include the wide utility of the algorithm, robustness, and relatively low number of parameters e.g., a user need not a priori define the number of clusters required. MINECLUS [41], PROCLUS [1] and STATPC [23] are selected as representatives of subspace clustering algorithms with established levels of performance. MINECLUS and PROCLUS performed well in the benchmarking survey of Muller *et al.* [25], whereas the STATPC algorithm performed very consistently in the Moise *et al.* survey [24]. Refer to Section 2 for the properties and characterization of PROCLUS, MINECLUS, and STATPC.

In all cases, the WEKA compatible *OpenSubspace* implementations for MINECLUS, PROCLUS and STATPC as developed by Muller *et al.* will be assumed [25]. Developers of OpenSubspace claim that they used the original implementation of MINECLUS provided by its authors and re-implemented PROCLUS and STATPC according to the original papers.

In the case of the incremental datasets (Table 1), the specific parameterization procedure adopted for the comparator algorithms is summarized as follows:

- MINECLUS: Earlier benchmarking studies recommended the parameterization of: $\alpha = 0.1$, $\beta = 0.25$ and $\omega = 0.3$ whereas k was enumerated over the range [2, 20] – the same range as S-ESC. MINECLUS has the property that although k clusters might be a priori specified, a clustering solution need not use all k clusters. Hence, presenting results as quartiles will give some insight as to how effective this property is in practice.

- PROCLUS: The true number of clusters and average dimensionality of clusters requires a priori specification, implying that PROCLUS is only capable of answering a simpler question in comparison to the other algorithms included in the study. The algorithm is run 10 times with different seed numbers. Variance in result distribution should therefore be significantly lower than the other algorithms as much more information is given a priori.

- STATPC: As recommended by authors the following parameterization was first assumed $\alpha_0 = 10^{-10}$, $\alpha_K = \alpha_H = 10^{-3}$ [23]. However, considerable sensitivity existed. Hence it was actually necessary to perform a sweep over all three parameters and use the post training performance metric (F-measure) to select the top 20 results out of the 216 parameter combinations tested.[15] All STATPC distributions therefore make use of such a process, thus biasing results in favour of STATPC.

- EM: as implemented in WEKA conducts cross validation thus resolves the cluster count k. Per dataset tuning is still necessary for the iteration limit and target standard deviation. Having established specific values for these two parameters the EM algorithm is run with 10 different seed numbers.

In the case of the large-scale datasets (Table 2), parameter sweeps were utilized for MINECLUS. Specifically, the α, β and ω parameters of MINECLUS were first determined and then 19 runs performed with $k \in [2, 20]$. It is these results which are reported below i.e., k is unknown. PROCLUS again requires the correct number of clusters as well as their average dimensionality to be declared a priori. In the case of STATPC the parameter values producing the best solutions with respect to F-measure on the incremental datasets were assumed. This was necessary as the larger datasets precluded the use of parameter sweeps for computational reasons. Likewise

[15]Six parameter values $\{10^{-15}, 10^{-12}, 10^{-9}, 10^{-6}, 10^{-3}, 10^{-1}\}$ were considered for each of the three STATPC parameters, or a total of $6 \times 6 \times 6 = 216$ parameter settings.

for EM the best parameter setting from incremental datasets were selected and EM is run 10 times with different seed numbers.

5.6 Simplified Version of Evolutionary Subspace Clustering

None of the comparator approaches utilized in this work utilize evolutionary computation. Moreover, either the code for the few evolutionary subspace clustering algorithms identified in Section 2 are not publicly available or they only perform soft projected clustering i.e., there is no dimension reduction. We therefore also included a genetic algorithm to perform the subspace clustering task. The baseline method is a simplified version of S-ESC algorithm where the symbiotic (dual-population) representation is replaced by a single 'flat' population representation of subspace clusters. In other words, each individual is responsible for describing all aspects of a clustering solution (including all cluster centroids within the clustering solution) while assuming the same 1-*d* attribute specific preprocessing activity (Component 1 of the S-ESC framework, Section 4) and evolutionary multi-objective formulation of fitness (cf. MOCK [13]). Hereafter the resulting algorithm will be referred to as 'flat' Evolutionary Subspace Clustering (F-ESC.) For a detailed description of F-ESC see Appendix (Section 8.1).

6 Results

Results are presented in separate subsections for the 'incremental' benchmarks of Moise *et al.* and our own synthetic 'large-scale' datasets (as defined in Section 5). All results are in terms of the quartile statistic and therefore significantly more robust to outliers than mean–variance based metrics. When plots are employed the coloured bar represents the median and the error bars show the first and third quartiles.

The x-axis in all figures reflects the design variable (Table 1). For example, in the GD and UD experiments, the x-axis represents the average dimensionality of clusters embedded within a dataset which varies from 2 to 20, and for large-scale benchmark it is the dataset dimensionality. The y-axis in all 'F-measure' figures such as Figures 6 and 7 is the *micro* F-measure which ranges between 0 and 1 with larger values preferred. The y-axis in 'Attribute count' figures such as Figure 8 is the average number of attributes indexed by a final clustering solution and a smaller value is preferred.

Statistical significance tests are performed to support or reject the hypothesis that the S-ESC distribution is the same as each of the candidate clustering algorithms. These tests and their results will be discussed in Appendix 8.2.

6.1 Moise *et al.* Datasets

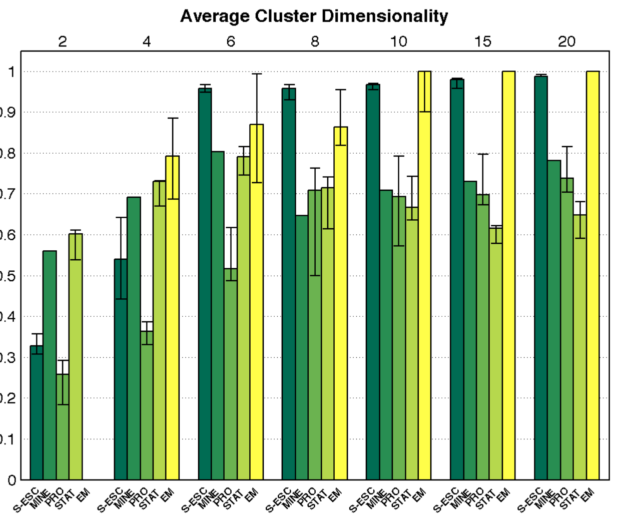
Table 1 defines a total of 10 different categories of incremental dataset. However, the GE–UE comparison did not produce significant performance variation; thus results from the GD and UD datasets are emphasized in the following.

Cluster embedding / data distribution experiments S-ESC and EM tend to dominate other methods with respect to F-measure for GD and UD datasets (Figures 6(a) and 6(b)). However, EM fails to return results when the average dimensionality of clusters is 2. MINECLUS had a distinct preference for data with a uniform distribution (Figure 6(b)). STATPC performs well when the cluster dimensionality is 4 or less, but did not maintain this level of performance as cluster dimension increased.

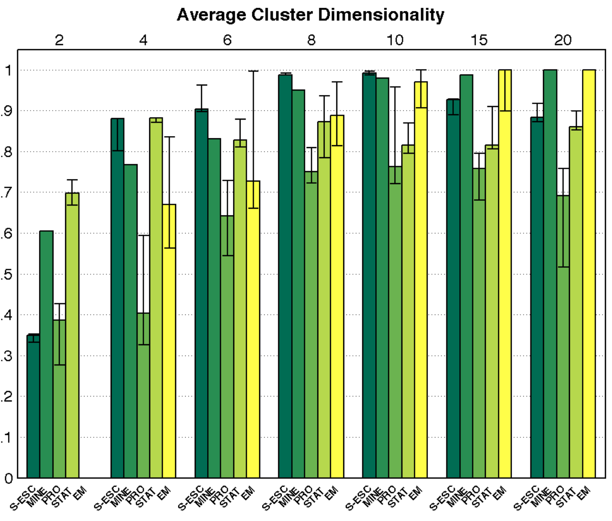
Dataset dimensionality experiments or (D) both S-ESC and STATPC provide the most consistent results, Figure 6(c). Conversely, the accuracy of EM drops as the dimensionality increases, and even fails to return results when dataset dimensionality reaches 100.[16] The next experiment evaluates effect of **dataset cardinality** (N). Here EM provides the stronger performance after initially failing on the smallest dataset. S-ESC and STATPC represent the most consistent subspace clustering algorithms (Figure 6(d)). Under the **cluster count variation** experiments (K), the cluster count increases from 2 to 5 and S-ESC, STATPC and EM perform equally well (Figure 6(e)). Conversely, PROCLUS deteriorates as the number of clusters increased beyond 3, an interesting result given that PROCLUS is given the correct cluster count a priori. The next experiment evaluates the effect of varying the number of data **instances per cluster** ($ClusterSize$) within a dataset (Figure 6(f)). S-ESC and STATPC dominate in all but the case of largest cluster support.

The next Moise experiment varies the **attribute spread** ($Extent$) of the relevant attributes (Figure 7(a)). All methods deteriorate rapidly as distributions on relevant attributes get wider and more diverse, thus more difficult

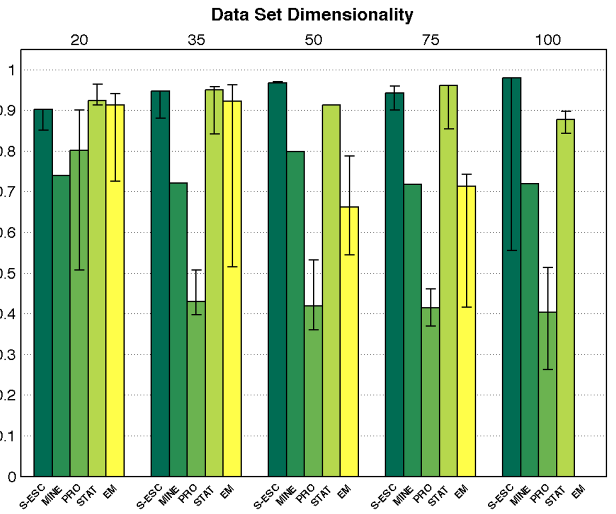
[16]This is to be expected given that EM is the only full-space clustering algorithm, however, it does not preclude better results following additional parameter optimization.



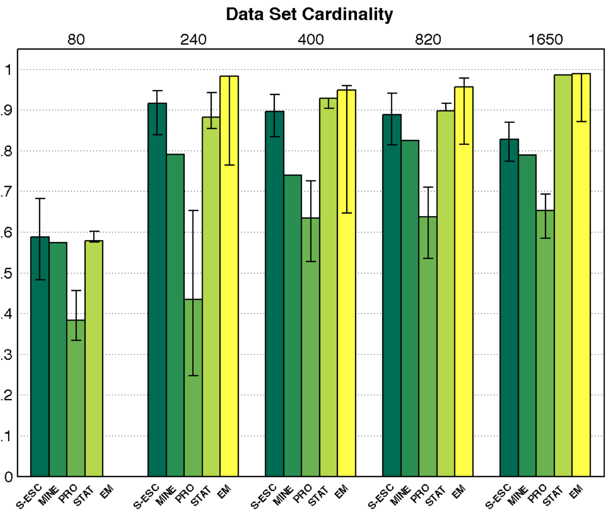
(a) *GD* experiment



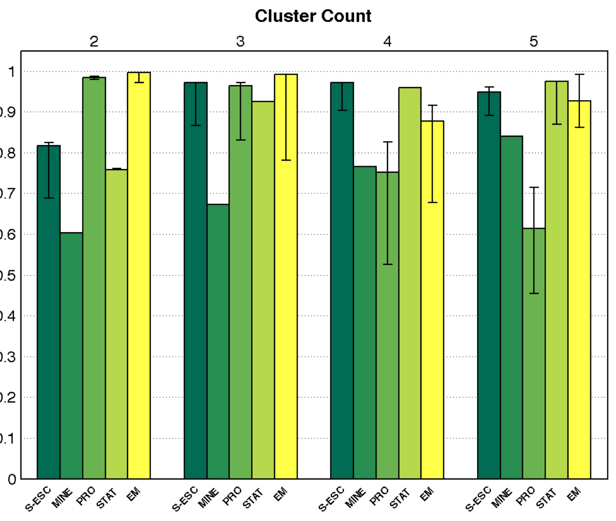
(b) *UD* experiment



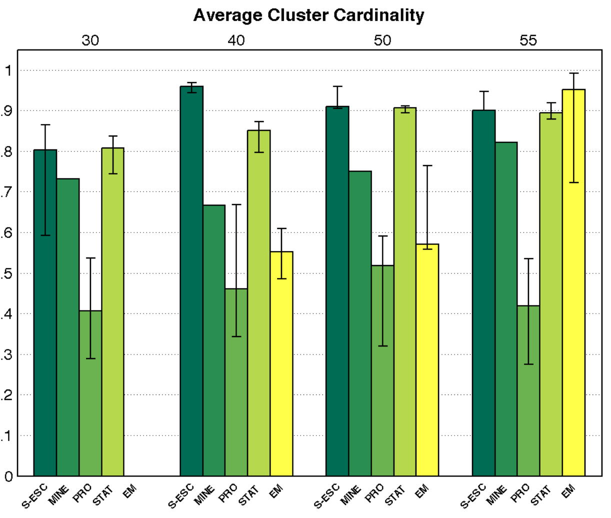
(c) *D* experiment



(d) *N* experiment

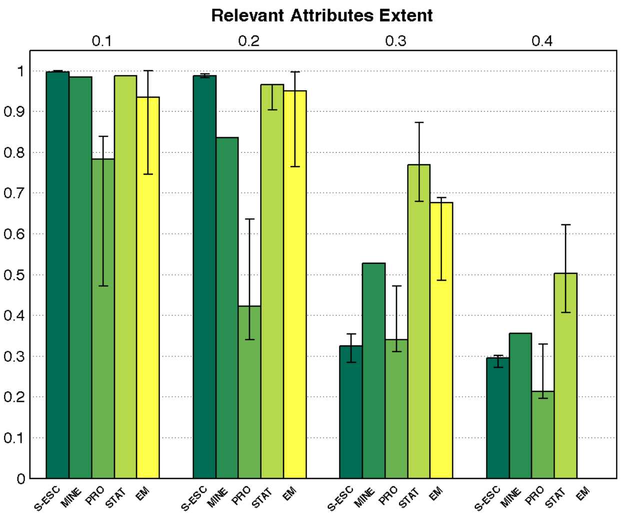


(e) *K* experiment

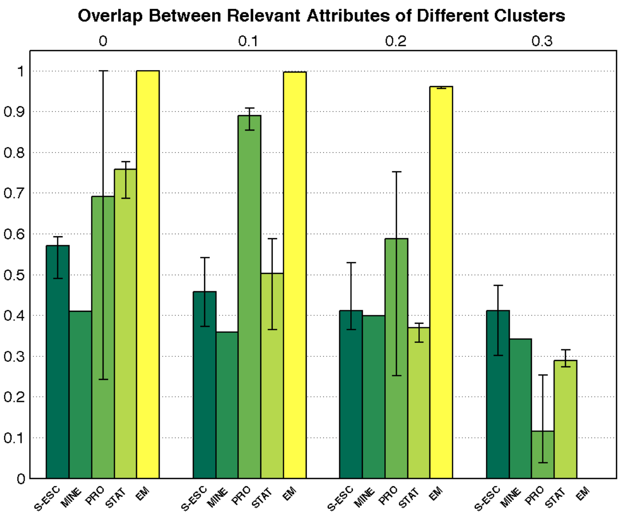


(f) *ClusterSize* experiment

Figure 6: F-measure for Moise *GD* (6(a)), *UD* (6(b)), *D* (6(c)), *N* (6(d)), *K* (6(e)) and *ClusterSize* (6(f)) experiments. Bar order per dataset: S-ESC, MINECLUS, PROCLUS, STATPC, EM.



(a) *Extent* experiment



(b) *Overlap* experiment

Figure 7: F-measure for Moise *Extent* (7(a)) and *Overlap* (7(b)) experiments. Bar order per dataset: S-ESC, MINECLUS, PROCLUS, STATPC, EM.

to distinguish from the noise associated with non-relevant attributes. However, STATPC degrades more gracefully than the others; albeit with F-measure used to select the appropriate STATPC parameterizations. EM fails to return results for the last dataset where the relevant attributes have a spread of 0.4 out of the possible [0,1] range. Varying the **overlap** between relevant attributes of different clusters (*Overlap*) also has a strong effect on all methods (Figure 7(b)). EM performs more consistently in all but the last dataset, possibly indicating a need for re-parameterization. However, the EM algorithm will naturally not be able to select the correct attribute space, whereas PROCLUS appears to be the stronger of the subspace methods.

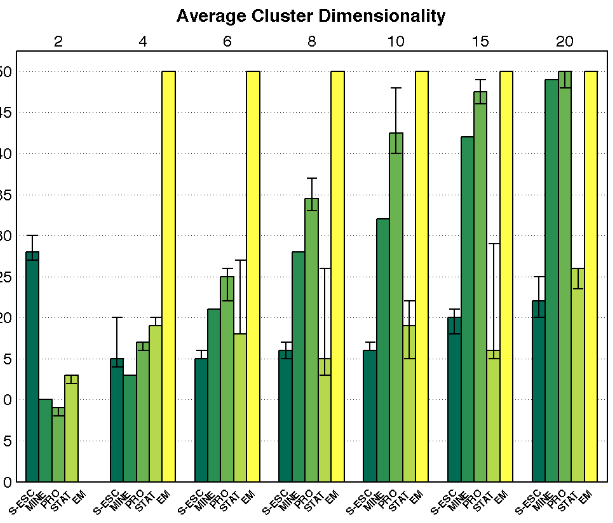
The above summary evaluates the outcome of the Moise experiments in terms of accuracy alone, the motivation being that the purity of clusters (as measured by F-measure) represents the primary objective. However, as a secondary objective we assume that the resulting clusters should also be simple. That is to say, solutions should employ the fewest attributes possible to perform the clustering task without compromising cluster purity. Therefore the second performance measure is the total number of (unique) attributes indexed by a solution and should be *minimized*. For sake of brevity, we emphasize results under Moise experiments with significant trends.

Starting with the specific case of the *UD* **Cluster embedding / data distribution** experiment (Figure 8(a)) S-ESC falls behind all subspace methods on the smallest number of attributes per cluster. However, once the average cluster dimensionality grows beyond two, S-ESC very consistently provides the most simple solutions. Conversely, MINECLUS and PROCLUS show a tendency to index an increasing number of attributes as the average cluster dimensionality increases; resulting in almost all attributes being indexed. STATPC, on the other hand, performs comparatively well, bettering S-ESC on two datasets. EM is only plotted to show the maximum number of attributes per dataset i.e., will always include all attributes. Plots for the equivalent *GD* experiment show similar trends, this time with a consistent trend in better S-ESC performance appearing after an average cluster dimensionality of 4.

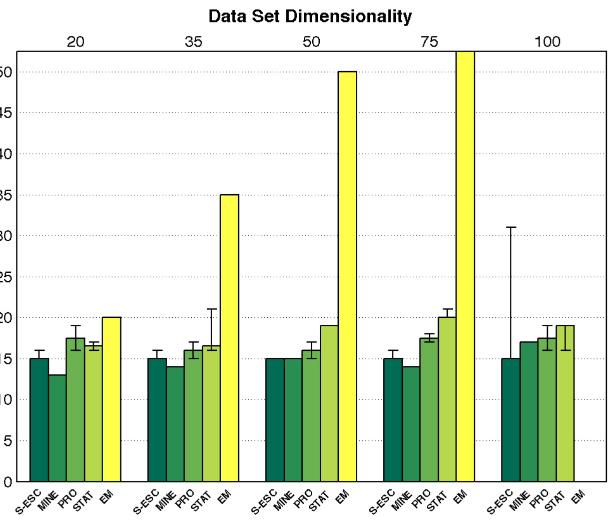
On the **dataset dimensionality** *(D)* experiment all subspace clustering methods perform equally well and index between 13 and 20 (unique) attributes per dataset (Figure 8(b)), therefore a winner can not be selected here. Almost the same trend holds true for the **dataset cardinality** *(N)* and **cluster count variation** *(K)* experiments (Figures 8(c) and 8(d)) respectively; however, MINECLUS and S-ESC most consistently utilize the least number of attributes.

6.2 Large-scale Datasets

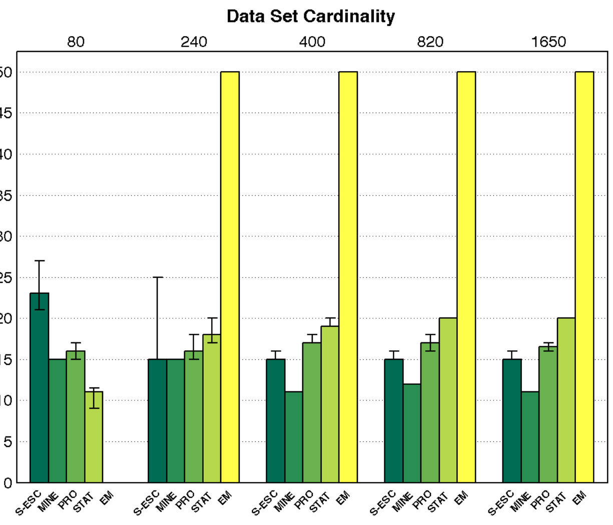
The Moise benchmarking dataset used above covers a wide range of properties. However, as noted in Section 5, there are two basic shortcomings: 1) the cardinality and dimensionality remains low, and 2) the cluster embedding assumes only one data distribution per dataset and aspect ratios that are not particularly diverse. For example, the 200D and 800D datasets from the large-scale test suite (Table 2) are the datasets that has both low- (10D) and high-dimensional (50D in case of former and 100D in case of the latter dataset) clusters within the same dataset. Moreover, the implanted clusters use different distributions (both Gaussian and uniform) at different



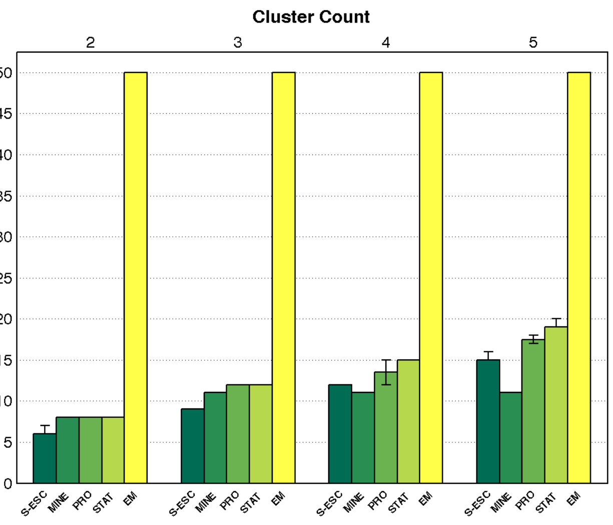
(a) *UD* experiment



(b) *D* experiment

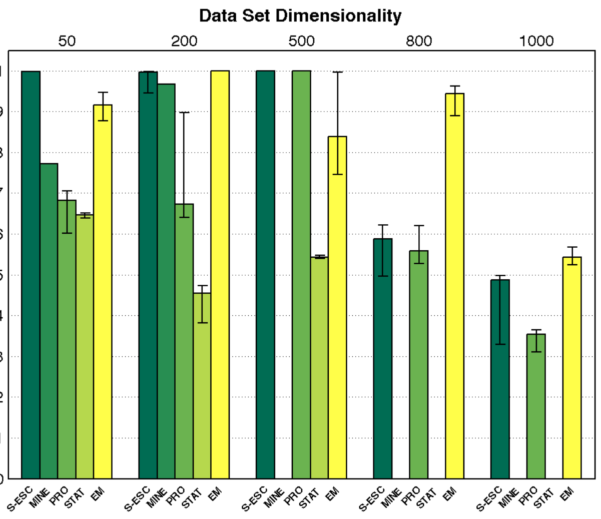


(c) *N* experiment

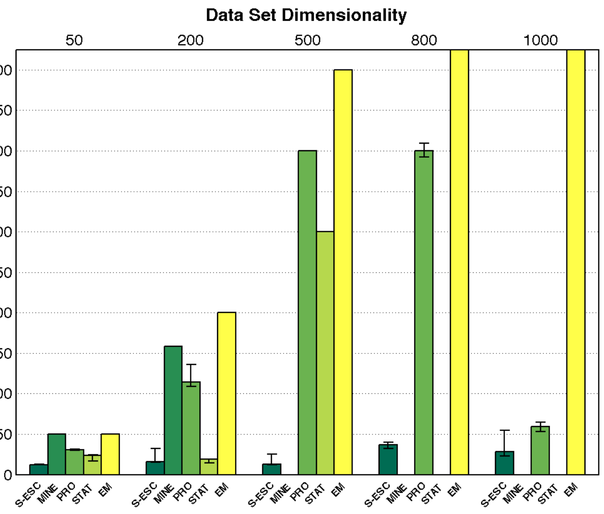


(d) *K* experiment

Figure 8: Attribute count for Moise *UD* (8(a)), *D* (8(b)), *N* (8(c)) and *K* (8(d)) experiments. Bar order per dataset: S-ESC, MINECLUS, PROCLUS, STATPC, EM.



(a) Accuracy (F-measure) experiment



(b) Simplicity (Attribute count) experiment

Figure 9: Accuracy (F-measure) and simplicity (attribute count) for large-scale benchmark. Bar order per dataset: S-ESC, MINECLUS, PROCLUS, STATPC, EM. For visualization purpose EM attribute count under 800 and 1000 dimensional datasets is cropped at 550.

aspect ratios. The 1000D dataset also embeds ten 10D clusters, thus only 10% of the attributes are relevant and the rest are irrelevant. Benchmark results are therefore presented for the 5 additional large-scale datasets of Table 2. Given the larger size of these datasets it was necessary to enforce a computational limit of 24 hrs per run. If results were not returned within this time then the implementation / algorithm was deemed inappropriate for the task dimension / cardinality.

Figure 9(a) summarizes how F-measure varies as a function of each dataset (identified in terms of data dimensionality). S-ESC betters all other methods on the first three datasets, and is the runner up method for 800D and 1000D datasets (w.r.t. full-space EM clustering), thus consistently bettering all other subspace methods. MINECLUS fails for datasets with 500 dimensions or more. PROCLUS matches S-ESC under the specific case of the 500D dataset. It is also the only implementation of a subspace clustering algorithm – beside S-ESC – that returns solutions for 800D and 1000D datasets. The STATPC implementation produces the least accurate results for the first three datasets, and fails altogether for datasets beyond 500 dimensions i.e., results are not produced within 24 hrs. Part of this might be due to the sensitivity of parameterization. However, as the scale of a task increases, then the cost of parameter sweeps also increases significantly.

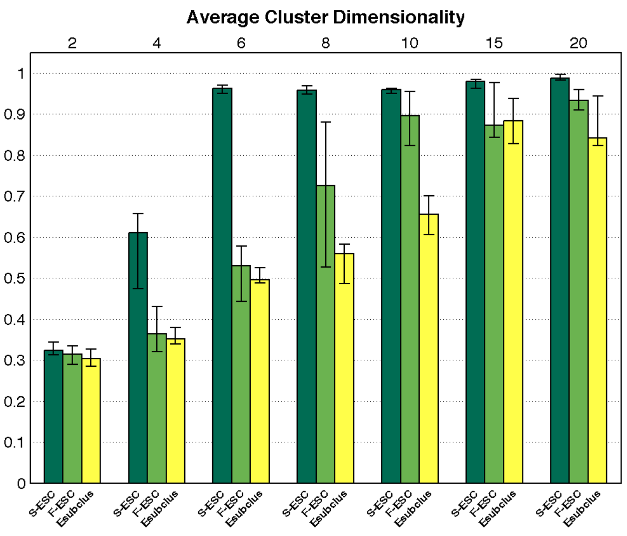
Bar plots for the number of unique attributes included (solution complexity) clearly illustrates the efficiency of S-ESC with respect to parsimony (Figure 9(b)). S-ESC uses less than 50 attributes for all datasets. Medians are 12, 16, 13, 37 and 28 for the five datasets respectively, using the most attributes for 800D and 1000D datasets. STATPC employs 24, 19 and 300 attributes for the first three datasets. MINECLUS indexes all 50 attributes for 50D dataset and 158 for the 200D dataset. PROCLUS indexes 36 and 112 for the first two datasets, and then jumps to use almost 400 attributes for 500D and 800D datasets.

6.3 S-ESC vs. Alternative Evolutionary Methods

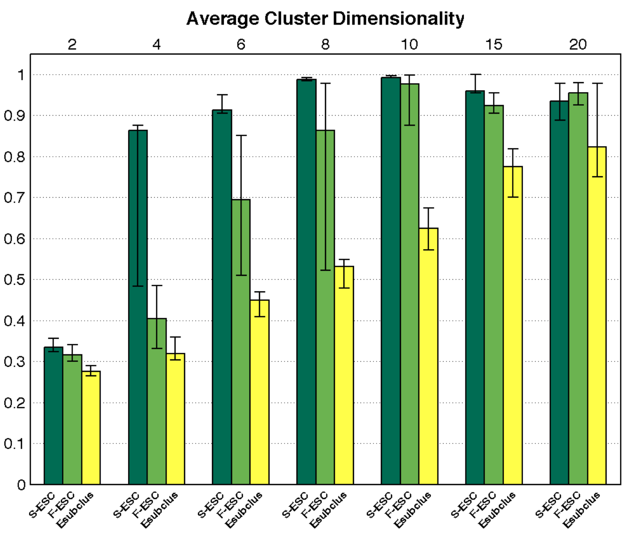
In order to benchmark S-ESC against evolutionary methods, we assume one of the most recent methods in subspace clustering (Esubclus) [26] as well as a non-symbiotic ('flat') version of the ESC algorithm.

Esubclus proceeds in two phases. Given the correct number of clusters, k, the authors use a a multi-objective genetic algorithm (NSGA-II) to provide (low-dimensional) groups of representative / relevant attributes for each cluster. Once the relevant attributes are determined the objects are clustered in the space spanned by the centroids of those groups.

Individuals from NSGA-II define subsets of attributes, with variation operators adding or deleting attributes. X-means clustering is then assumed for building clusters within the subset of attributes identified by each individual. Fitness evaluation is performed relative to two objectives: minimizing cluster variation and maximizing diversity. Evolution under NSGA-II is applied a total of k times in order to produces k Pareto fronts. A final stage of the process searches through the possible combinations of sub-space clusters to identify a final complete cluster



(a) *GD* experiment



(b) *UD* experiment

Figure 10: F-measure comparison of S-ESC against F-ESC and Esubclus methods on incremental benchmarks with Gaussian (10(a)) and Uniform (10(b)) distributions. Bar order per dataset: S-ESC, F-ESC, Esubclus.

solution (choose *k*-clusters).

Esubclus does not require extensive parameter tuning. Being an evolutionary method it requires standard evolutionary parameters such as population size, generation count, mutation and crossover probabilities. Given the number of clusters, *k*, the primary NSGA-II is run sequentially *k* times, each time with a population size of 20 and for 50 generations. Crossover probability is set to 90% and mutation is set to 10%. All these parameters are set based on recommendations by authors.

We also compare the results of performing the search for subspace clusters using a 'flat' GA representation that otherwise retains the evolutionary multi-objective formulation of fitness and 1-*d* preprocessing step as the proposed approach, or F-ESC (Appendix, Section 8.1). F-ESC is compared against S-ESC with respect to the large-scale datasets as well as selected incremental benchmarks. Rather than just assuming S-ESC parameter values, preliminary experiments are performed to optimize evolutionary parameter setting for F-ESC. These parameters include crossover and mutation probabilities, and the population size. S-ESC does not use the crossover operator and relies only on mutation operators. The probability of mutation is 100% in S-ESC meaning that all individuals go through the mutation process with an annealing schedule decreasing the rate of mutation (Section 4.4). For F-ESC on the other hand, four experiments are run to identify the best probabilities for crossover and mutation rates. The best results were achieved when both crossover and mutation probabilities are set to 100%; albeit again note that this is discounted by the annealing schedule of Section 4.4. For the sake of brevity we will not present any plots for this experiment.

With regards to population size the situation is somewhat different. S-ESC uses two distinct populations in which the host population is a fixed size (100) and the symbiont population size varies as a natural effect of group selection (Section 4.2) and the variation operators (Section 4.4); whereas F-ESC assumes a single fixed size population. In order to characterize an appropriate F-ESC population size, S-ESC is applied to a sample of the benchmarking tasks and the variation in symbiont population size recorded. Experiments show that there is some variation between profiles of different datasets of Table 2, however, the minimum and maximum bounds are very close. Symbiont population size for the 200d dataset varies between 284 and 392, while more complex dataset, 800d, utilizes between 297 and 436 symbionts. Similar trends can be observed for the other three datasets. In general symbiont population size never exceeded 4.5 times the host population size (100).

With this in mind, F-ESC benchmarking will assume three different values for population size: 100, 200 and 500. It appears that the overall performance of F-ESC marginally improves as the population size increases, however, this comes with the price of an increased computational overhead. Moreover, there does not appear to be any statistical significance to the variation in population size for F-ESC. Therefore, we choose to run F-ESC with a fixed population of 100 individuals for the evaluation against S-ESC and Esubclus.

Figures 10(a) and 10(b) compare S-ESC against F-ESC and Esubclus in terms of F-measures for the *GD* and *UD* benchmarks. It can be observed that there is only one dataset on which the S-ESC median performance is dominated by one of its evolutionary comparators. This happens on the last dataset of *UD* experiment, a 5-class, 50-dimensional dataset with Uniform distribution in which the average dimensionality of clusters are 20 (last column of Figure 10(b)). In all other cases S-ESC outperforms both F-ESC and Esubclus. In most cases S-ESC dominates F-ESC with significant margin.[17] Moreover, these correspond to the more difficult scenarios in which the number of supporting attributes per cluster subspace is very low. Results conducted on the other 38 datasets of the incremental datasets (not shown for the sake of brevity) also support this general trend. Also note that F-ESC dominates Esubclus in most cases and is outperformed by Esubclus only in one dataset.

Figure 11 compares the three methods against large-scale datasets. Again S-ESC performs more consistently and dominates the other evolutionary methods i.e., statistically significantly better in 4 of 5 of the datasets. F-ESC is the runner up in 3 cases, providing very similar results on 50- and 500-dimensional datasets. Esubclus outperforms F-ESC on 200- and 1000-dimensional datasets which are more complex than 50- and 500-dimensional datasets but fail to provide comparable solutions for the less-complex benchmarks.

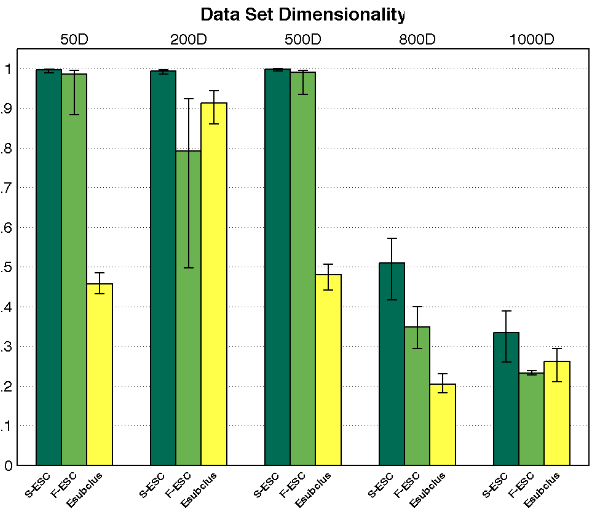


Figure 11: F-measure comparison of S-ESC against F-ESC and Esubclus on large-scale datasets.

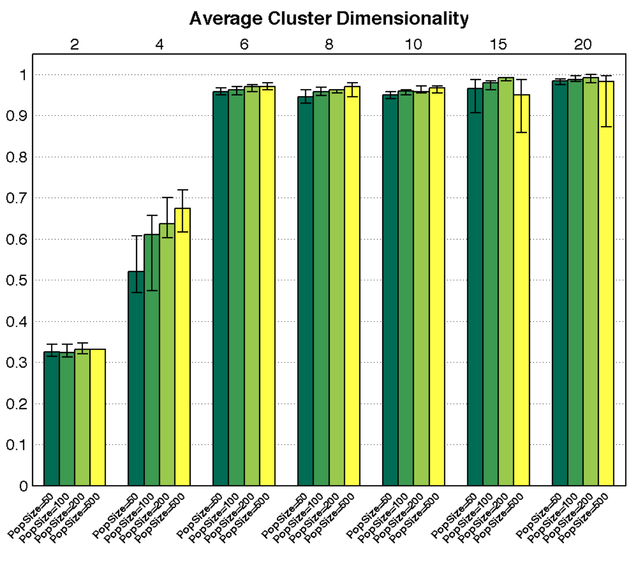
6.4 Parameter Insensitivity

As mentioned in Section 5.5, S-ESC is used with the same set of parameters on both incremental and large-scale datasets. A fixed-sized host population of 100, a random population of points refreshed in the beginning of each generation by RSS with a size of 100, mutation operators with probability of 1.0, and a minimum of 2 and maximum of 20 for the number of attributes per symbiont as well as symbiont per host. In this section we provide tests to two of these parameters being host population and point population sizes.

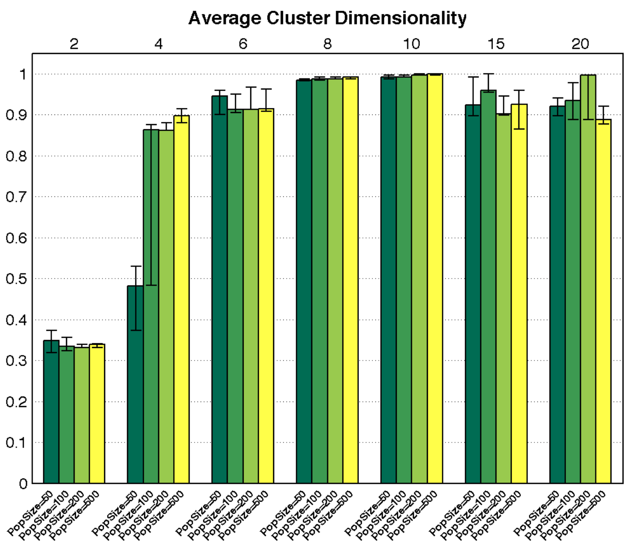
Figures 12(a) and 12(b) show the effect of increasing host population size from 50 to 500 on F-measure. *GD* and *UD* experiments of incremental datasets are assessed here. The four bars for each datasets represent the distribution of F-measures for the case with 50, 100, 200 and 500 hosts respectively. Recall that the symbiont population is always initialized to the size of host population but is allowed to increase as large as 5 times of the host population. It is easily observed that there is not a significant difference between the performance of S-ESC with different population sizes specially once the average dimensionality increases beyond 6.

Similarly we run the same experiment to evaluate the effect of increasing RSS point population on the performance of S-ESC. Figures 13(a) and 13(b) indicate that again there is only marginal variation. Similar to host population and point population size we ran experiments to see the effect of modifying mutation probabilities and how they might affect S-ESC performance. We ran experiments with different mutation probabilities and concluded that a mutation probability of 1.0 for both single-level and multi-level mutation operators maximizes the diversity and provides best solutions. Due to space limitations these plots are not included here.

[17]Statistical significance holds for 6 of 7 datasets in Figure 10(a), 5 of 7 in Figure 10(b).

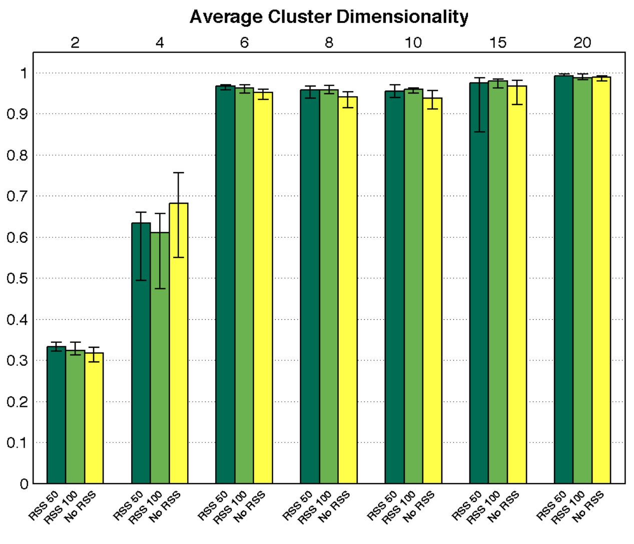


(a) *GD* experiment

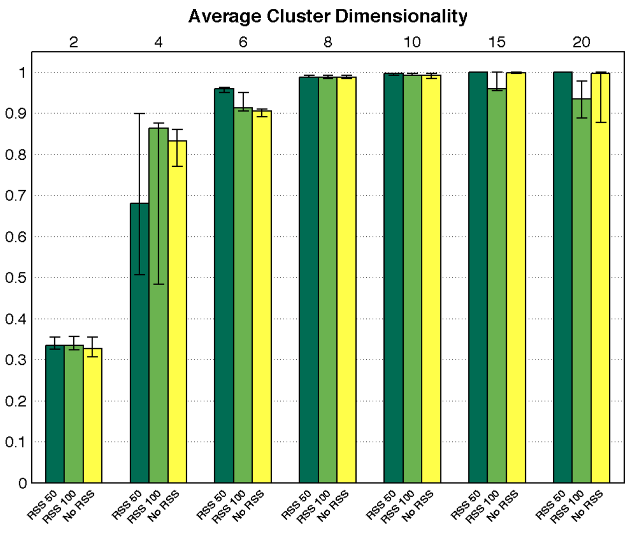


(b) *UD* experiment

Figure 12: F-measure comparison of S-ESC with different host (and symbiont) population sizes on incremental benchmarks with Gaussian (12(a)) and Uniform (12(b)) distributions. Bar order per dataset: Popsize=50, Popsize=100, Popsize=200, Popsize=500.



(a) *GD* experiment



(b) *UD* experiment

Figure 13: F-measure comparison of S-ESC with different RSS point population sizes on incremental benchmarks with Gaussian (13(a)) and Uniform (13(b)) distributions. Bar order per dataset: RSS=50, RSS=100, RSS=*N*.

6.5 Outlier Effect

S-ESC has no integrated outlier detection process, where outliers represent data instances whose attributes all represent noise. As noted in Section 5.2, only STATPC explicitly includes an outlier detection capability. In the following the incremental benchmarks of Table 1 are used in their original form with outliers included. From the perspective of S-ESC we can also use this as an opportunity to assess the impact of outliers on different components of the S-ESC framework. Thus we ask: 1) what impact does outliers have on the initial grid construction alone; 2) what impact does outliers have on the identification of S-ESC subspace clusters alone, and; 3) what is the combined impact of outliers on the entire S-ESC framework.

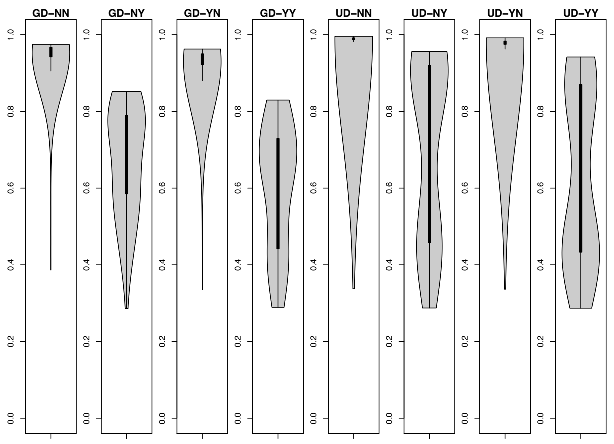
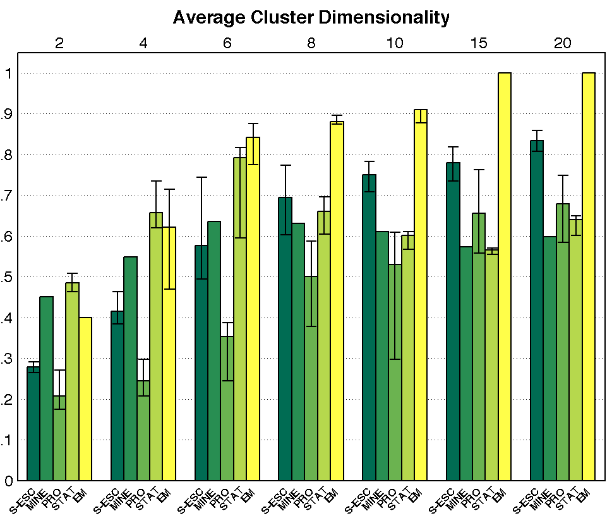


Figure 14: F-measure with outlier datasets before and after grid generation process. GD–XX denote experiments on the Gaussian dataset and UD–XX the Uniform dataset; XX–NN are the base case of no outliers; XX–NY implies no outliers during the preprocessing step of grid generation but are included during the S-ESC evolutionary identification of subspaces; XX–YN implies only during grid generation are outliers present; and XX–YY implies outliers are present throughout.

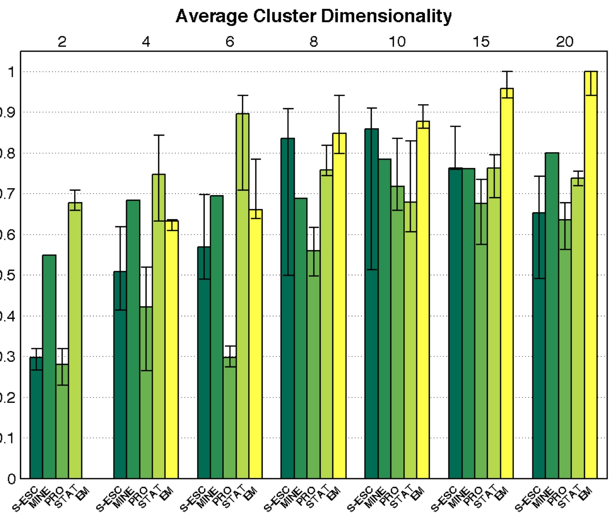
Two datasets with different data distributions (Gaussian (GD) vs. Uniform (UD)) with 5 legitimate clusters and 8 attributes per cluster will be utilized for the outlier experiments. Figure 14 summarizes performance using a combined violin–box plot quantifying the distribution of F-measure across 50 runs per experiment. The violin establishes the nature of the distribution – thus to what extent a bimodal distribution is present – while the box plot defines the quartile statistics (1st, 3rd quartile and median). For reference purposes the corresponding no outlier results are also included (GD–NN and UD–NN).

It is apparent that the preprocessing step of grid generation is least affected by the introduction of outliers (compare GD–NN with GD–YN and UD–NN with UD–YN), whereas a 20% reduction in F-measure appears under the Gaussian distribution once outliers appear in the process of subspace cluster identification (compare GD–NN with GD–NY and GD–YY). However, under a uniform distribution a bimodal distribution results with half of the solutions returned with an F-measure of 80% or more when noise is introduced to the S-ESC cluster identification process (UD–NY and UD–YY).

Figures 15(a) and 15(b) show the effect of introducing outliers on S-ESC and comparator methods on 14 datasets from the incremental benchmarks (*GD* and *UD* experiments). As these figures illustrate (compared to results from same methods on datasets without outliers, Figures 6(a) and 6(b)) the accuracy of all methods drop. Results from the EM algorithm remain strong as long as there are a sufficient number of attributes per cluster. However, once cluster attribute support drops below eight then the performance of one or more subspace algorithms are significantly better. Under the Gaussian distribution, S-ESC remains the stronger algorithm for cluster dimensionalities above 6. At lower dimensions STATPC is preferred but must undergo extensive parameter sweeps to do so. In the case of experiments w.r.t. the Uniform distribution (Figure 15(b)) MINECLUS was the most consistent subspace algorithm, followed by S-ESC. STATPC appeared to be still sensitive to parameterization, with the lowest ranked performance under 4 out of the 7 experiments. That all clustering methods found the outlier task more difficult under the Uniform distribution is understandable given that both the subspace clusters and the outliers are from a Uniform distribution.



(a) *GD* experiment



(b) *UD* experiment

Figure 15: F-measure comparison of S-ESC against comparators on incremental benchmarks with Gaussian (15(a)) and Uniform (15(b)) distributions. Bar order per dataset: S-ESC, MINECLUS, PROCLUS, STATPC, EM.

6.6 Algorithm Runtime

Runtime cost is assessed by running each algorithm on a common computing platform. Given that we are only interested in the execution time to complete tasks of increasing size, a single run is performed on each algorithm using each of the large-scale datasets (Table 2). Naturally, some variation can occur between different runs, however, even in the case of S-ESC the principal algorithmic factors are constant across all runs i.e., host population size (100), RSS sample (100) and generation count (1,000). The major sources of variation come from the cardinality and dimensionality of the benchmarks. Figure 16 summarizes the respective runtime for each of the clustering algorithms.

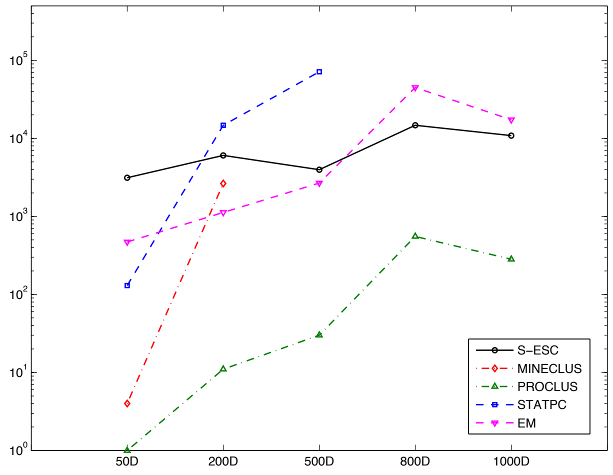


Figure 16: CPU runtime in seconds on Large-scale datasets. Lines between data points are of illustrative purposes alone. Results reflect the cost of single runs and do not reflect the cost of performing any parameter sweeps.

The profiles for MINECLUS and STATPC are not complete on account of neither algorithm returning results after 24 hours of CPU time for the larger datasets. Taken at face value PROCLUS is always the faster algorithm by one or two orders of magnitude. However, PROCLUS also requires the correct number of clusters and attribute support to be given by the user a priori. Hence, the other algorithms are answering a more difficult question from the outset i.e., the prior information provided to PROCLUS reduce the impact of the curse of dimensionality (Section 2.1). Both of the explicitly stochastic algorithms – EM and S-ESC – scale to all the tasks and follow a similar profile; albeit with EM beginning at approx. an order of magnitude shorter runtime. As the datasets increase however, this advantage is lost, with EM taking longer on the larger datasets. This is an artifact of the RSS component of S-ESC effectively decoupling the effect of cardinality i.e., fitness is only evaluated with respect to the content of the RSS sample. Thus, between smallest and largest task, S-ESC undergoes the least increase in runtime or less than an order of magnitude; whereas all other algorithms experience more than an order of magnitude increase.

S-ESC computational cost can also be expressed algorithmically. In this case we note that there are two factors that contribute to the cost of subspace clustering under S-ESC: NSGA-II and distance estimation. As noted earlier, we assume the code distribution of Jensen for NSGA-II [15], thus for the case of a bi-objective EMO the cost is $O(GP \log P)$; where G is the number of generations and P is the population size. In the case of the distance estimation, the cost of compactness objective is linear, whereas the cost of the connectivity objective is $O(N \times N_s)$ where N is the total number of exemplars and N_s is the size of the RSS sample [13]. Further simplifications have been proposed, but these are again implementation specific.

6.7 Discussion

Each clustering algorithm presents different requirements for prior information: MINECLUS requires knowledge of the relevant number of clusters; PROCLUS requires both the relevant cluster count and (average) dimensionality; EM and STATPC benefit from parameter tuning, STATPC in particular. Conversely, S-ESC was used throughout with a common parameterization, and prior knowledge was limited to assuming that the cluster count and attribute support lied somewhere between 2 and 20. Various previous benchmarking studies have introduced specific datasets for evaluating subspace clustering algorithms e.g., [29, 24, 25]. Here we therefore began by taking the datasets from one of the most extensive previous studies and concentrated on comparing performance relative to cluster purity (as measured through micro F-measure) and a secondary cluster metric of simplicity.

As per any empirical evaluation, several pragmatic benchmarking decisions need to be taken. Specifically, with respect to parameterization. This was particularly problematic in the case of STATPC as although parameter sweeps could be made, it was then necessary to choose some representative subset of solutions. To this end the purity metric was used to identify the best 10%, thus a bias in favour of better STATPC solutions. PROCLUS has a stochastic component, hence could be reinitialized given the correct prior information. MINECLUS was used with multiple values for k (cluster count), mimicking the range of values over which S-ESC evolves.

Overall, both S-ESC and EM performed well the most consistently, albeit with EM unable to provide any information regarding cluster attribute support. Moreover, it was necessary to explicitly sweep EM parameters to achieve this. STATPC when provided with the (generally unknown) information regarding cluster purity was also effective on the Moise benchmarks, but was not effective under the large-scale datasets. Indeed, neither STATPC nor MINECLUS scaled to all of the datasets in the large-scale suite of tasks. Part of this might be attributed to the specifics of an implementation. However, the second factor present in the large-scale experiment was the use of more variation in the cluster properties, particularly with respect to the use of non-spherical distributions. MINECLUS was very consistent in the results, there being no variation in the median, 25th or 75th percentile throughout the experiments. Thus, for MINECLUS results are either weak and sit below the 1st quartile due to improper choices of k or are all equally good and sit *between* the 1st and 3rd quartiles.[18] PROCLUS benefitted from the most prior information, but the only real benefit this appears to have was in terms of not failing to return results under the large-scale experiments. The EM algorithm performed well as long as cluster attribute support was sufficiently high e.g., Figures 6(a), 6(b), 6(d) and 6(f).

The proposed S-ESC algorithm was robust across the widest range of datasets. Moreover, this was achieved under a common parameterization throughout. Factors supporting such an outcome include the use of a multi-objective fitness function, and the separation of centroid identification from composing clustering solutions care of host–symbiont coevolution. Adopting a subsampling scheme ensured that the algorithm scaled with increased cardinality, while competitive results on the smaller incremental datasets indicated that there was little if any penalty for this.

7 Conclusion and Future Works

A multi-objective evolutionary subspace clustering algorithm is introduced in which symbiosis provides the metaphor for the coevolution of both cluster centroids and clustering solutions (symbionts and hosts respectively). In dividing the representation between two populations we are more effectively able to model the sharing of common centroid between multiple clustering solutions while also promoting specialization. Moreover, variation

[18]Recall that when k exceeds the actual number of clusters, MINECLUS need not use all the k clusters specified a priori.

operators can be designed such that they focus on distinct aspects of the combinatorial search in each population. Adopting a multi-objective approach enables the S-ESC algorithm to model clusters with a wide range of distributions; a property that the comparator algorithms were not able to achieve. Moreover, the attribute support for the resulting solutions was generally very low. Subsampling was employed to decouple the cost of fitness evaluation, with little impact on the resulting quality. Benchmarking followed the approach established in previous research, with the study of Moise *et al.* pursued in particular (54 datasets). In the vast majority of cases the S-ESC algorithm provided either the best or runner-up performance. Additional experiments designed to assess the impact of higher dimensions / cardinality and clusters designed to include multiple factors also emphasized the general robustness of the proposed S-ESC algorithm.

Aside from applying the algorithm to other sources of data, future work might consider the following:

- S-ESC depends on the initial grid to convert the continuous search space into a discrete combinatorial optimization problem. Finding dense regions in each dimension individually comes with the price of S-ESC failing to provide solutions to datasets with non-axis-aligned clusters. This effect could possibly be alleviated by exploring other methods of grid generation such grids constructed from fractal dimensions [19].

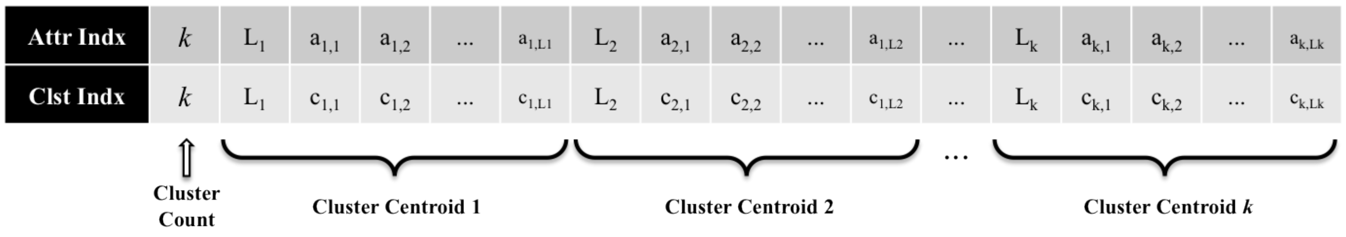


Figure 17: Flat Chromosome: A flat chromosome is composed of *k* cluster centroids (similar to a S-ESC symbiont representation), each supporting a (potentially) unique attribute support. Similar to S-ESC symbiont, an F-ESC individual is composed of two integer strings, one for attributes, and the one for 1-*d* centroids.

- The framework in its current form assumes a batch model of cluster construction. Conversely, the advent of 'big data' implies that more incremental forms of cluster maintenance are becoming necessary. In such scenarios the concept comprising different classes / clusters might change over time, therefore method should be capable of adapting to the dynamic nature of the data. A natural approach to address this would be to adapt the grid over time with the RSS sample now taking the form of a sliding window.

8 Appendix

8.1 Flat Evolutionary Subspace Clustering

Flat evolutionary subspace clustering (F-ESC) is a simplified version of S-ESC in which the two-level (hierarchical) symbiotic representation is replaced with a single-level flat representation, thereby eliminating the symbiotic relationship. Apart from the symbiotic process and replacing the single level mutation operator with a crossover operator everything is shared. The grid generation, multi-objective evolutionary optimization using compactness and connectivity objectives, atomic mutation operators to remove and add attributes and modify the 1-*d* centroid within an attribute, subsampling and knee detection are all used in F-ESC as per S-ESC. This section characterizes the main differences between the two variants of the ESC: representation and crossover operator.

8.1.1 Representation

The two-level representation of S-ESC is simplified (and condensed) into a flat single-level representation as shown in Figure 17. Here, each individual encodes all *k* cluster centroids necessary for a partitioning, therefore the chromosome is composed of *k* cluster centroids laid out into a single string of integer pairs. Similar to a symbiont representation in S-ESC, F-ESC chromosome has two connected integer strings; one for indexing attributes, and one for indexing 1-*d* cluster centroids, within each attribute. Both integer strings start with the number of cluster centroids the individual is encoding (*k*). Then *k* cluster centroids are encoded similar to a S-ESC symbiont representation, with a minor change. The first integer of each cluster centroid is the number of attributes the cluster centroid supports in both attribute and 1-*d* centroid strings. In other words L_1, L_2 and L_k in Figure 17 represent the attribute count for cluster centroid 1, 2 and k respectively. $a_{1,1}$ and a_{1,L_1} are the first and last attributes for cluster centroid 1 whereas $c_{1,1}$ and c_{1,L_1} are the first and last 1-d centroids for cluster centroid 1.

The same limits are set for F-ESC with respect to minimum and maximum number of clusters in a dataset and attributes per cluster. The range [2,20] is selected for both constraints, which means that a single individual's length can vary between 4 and 400 in length.

8.1.2 Crossover

The second main difference between S-ESC and F-ESC – due to the 'flat' representation – is the use of a crossover operator replacing the single-level mutation operator utilized in S-ESC. The single-level mutation operator in S-ESC is responsible for removing / adding / swapping symbionts (CC) from / to / between hosts (CS). The crossover operator in F-ESC essentially performs the same modification between two flat individuals. The variation operator is a 2-point crossover operator which swaps one or more cluster centroids from parent a with one or more cluster centroids from parent b. There is also a repair mechanism to make sure the offspring meets the cluster limit constraints.

8.1.3 Similarities

The remaining components and sub-components of F-ESC are the same as S-ESC. Grid generation is the pre-processing component with its output being the genetic material used by evolutionary process. F-ESC employs the same EMO algorithm utilizing both compactness and connectivity objectives. The selection operator is a tournament process between four individuals where the best individual is selected as parent a and the runner up as parent b; thus elitist. The same atomic mutation operators are implemented to remove and add attributes to a randomly selected cluster centroid within an individual with a third mutation operator modifying the 1-d centroid of a randomly selected attribute within a cluster centroid.

To account for robustness against dataset cardinality, the same subsampling process is used in F-ESC. M points are randomly selected anew at each generation and individuals' objectives are evaluated against this set instead of the whole dataset. This set (called active set) is refreshed at the end of each generation. Once the evolutionary process provided a pool of solutions the knee detection procedure as suggested by [35] identifies the knee solution as the champion solution.

8.2 Statistical Tests

Tests were performed to support or reject the hypothesis that the performance of the S-ESC solutions are drawn from the same distribution as that of the comparator methods. The following tables return the p value with a confidence level of 99%. Hence values smaller than 0.01 imply that the distributions are independent with a confidence of 99%. It does not say wether S-ESC is outperforming the comparator method or vice versa, only the fact whether they are statistically different or not, with a confidence level of 99%. Results in *bold* indicate that it is *not possible* to reject the hypothesis (i.e. neither S-ESC nor the comparator in the test is being outperformed by the other method), whereas in most cases the hypothesis is rejected (i.e. one of the methods in the test is being outperformed by the other method).

Some care is necessary in the case of distributions about the extreme values. Thus, the * symbol is used to denote the use of a single tailed test rather than a double-tailed test. Similarly in the case of the outlier datasets a Normal distribution could not be assumed, thus the Krushal-Wallis non-parametric hypothesis test was used in place of the student t-test. The 'NaN' values imply that the comparator algorithm failed to provide any results for the task within the given time.

Out of $54 \times 4 = 216$ tests between S-ESC and comparator methods on the incremental datasets of Moise *et al.*, there were 9 cases (approximately 4%) in which the comparator algorithms (MINECLUS, STATPC and EM) do not return results (NaN's in Tables 3) and 32 cases (approximately 15%) in which there is no statistically significant difference between S-ESC and the comparator results (the bold cases in Tables 3). There are 134 cases (approximately 62%) in which S-ESC outperforms the comparator method in a statistically significant way, and only 41 cases (approximately 19%) in which S-ESC is outperformed by a comparator algorithm.

Note that the numbers in parentheses define the specific dataset to be tested. For the case of the GE, GD, UE and UD experiments, it is the average dimensionality of the dataset. For the D, N and k experiments, it is the dimensionality, cardinality and cluster count of the dataset, respectively. For the $Extent$ experiment, it is the spread of values for relevant attributes. For the $Overlap$ experiment, it is the overlap between the relevant attributes of the different clusters, and for the $ClusteSizes$ experiment, it is the average instance count of the clusters.

Table 3: The t-test *p* values for F-measure significance of the incremental benchmarks of Moise *et al.* (Section 5, Table 1). Values in bold indicate that there is no statistically significant difference between S-ESC and the comparator method.

Data Set	MINECLUS vs. S-ESC	PROCLUS vs. S-ESC	STATPC vs. S-ESC	EM vs. S-ESC
GE(2)	2.42E-27	0.009	6.00E-27	NaN
GE(4)	2.02E-24	4.70E-06	6.45E-42	7.66E-34
GE(6)	1.74E-83	0	9.48E-20	**0.06**
GE(8)	2.03E-09	0	4.39E-19	**0.64**
GE(10)	2.68E-22	2.95E-05	2.89E-26	**0.56**
GE(15)	1.83E-23	**0.02**	1.05E-23	2.48E-30
GE(20)	1.65E-12	7.71E-05	7.18E-58	8.68E-66
GD(2)	3.75E-16	0.006	8.53E-36	NaN
GD(4)	2.04E-15	0	1.17E-35	2.98E-23
GD(6)	0	3.57E-06	3.59E-29	3.31E-16
GD(8)	1.99E-24	0	7.37E-14	5.70E-13
GD(10)	1.06E-15	0	1.31E-16	**0.37**
GD(15)	1.02E-32	1.31E-06	6.18E-33	1.90E-69
GD(20)	1.27E-09	7.59E-05	1.09E-19	5.04E-49
UE(2)	5.08E-15	**0.86**	1.41E-44	NaN
UE(4)	5.60E-07	0	9.80E-25	3.14E-10
UE(6)	1.91E-06	0	4.60E-05	1.51E-20
UE(8)	1.95E-06	0	1.32E-23	3.57E-10
UE(10)	0	**0.12**	1.10E-10	1.07E-22
UE(15)	**0.35**	0	1.11E-08	0
UE(20)	**0.01**	**0.01**	0.008	6.57E-56
UD(2)	8.37E-39	**0.59**	5.15E-29	NaN
UD(4)	0.009	0	1.76E-30	1.19E-06
UD(6)	6.92E-53	5.40E-05	4.91E-07	8.89E-12
UD(8)	0	3.99E-05	3.05E-12	3.23E-23
UD(10)	**0.03**	0	3.16E-12	1.46E-09
UD(15)	**0.16**	0	**0.57**	2.27E-25
UD(20)	**0.02**	0	**0.22**	1.04E-89
D(20)	4.55E-08	**0.06**	2.49E-14	**0.01**
D(35)	1.93E-145	3.06E-05	1.30E-22	1.67E-07
D(50)	2.11E-298	5.85E-06	1.31E-12	4.42E-24
D(75)	5.91E-84	1.65E-07	1.53E-09	2.43E-12
D(100)	0	5.41E-06	3.01E-08	NaN
N(80)	**0.78**	6.02E-06	**0.03**	NaN
N(240)	1.04E-08	0	8.77E-14	**0.59**
N(400)	5.59E-20	0	1.48E-60	0.007
N(820)	**0.02**	0	7.94E-211	0
N(1650)	4.67E-06	0	8.90E-182	2.29E-13
K(2)	1.10E-26	5.07E-110	0	4.06E-109
K(3)	1.05E-22	**0.36**	**0.01**	**0.3**
K(4)	6.88E-11	0	1.12E-16	3.67E-16
K(5)	1.33E-07	9.91E-05	3.29E-36	**0.27**
Ext(0.1)	6.55E-02	5.07E-03	2.45E-19	1.23E-12
Ext(0.2)	6.00E-09	7.70E-06	9.64E-53	1.18E-07
Ext(0.3)	3.22E-52	**0.15**	9.25E-28	2.61E-19
Ext(0.4)	7.24E-30	**0.13**	2.05E-16	NaN
Over(0.0)	1.04E-16	**0.49**	1.27E-32	5.43E-53
Over(0.1)	2.57E-09	3.93E-06	0	3.08E-56
Over(0.2)	0	**0.35**	0	1.11E-70
Over(0.3)	1.67E-05	**0.05**	9.47E-07	NaN
ClsSz(30)	**0.86**	8.45E-05	1.69E-11	NaN
ClsSz(40)	1.26E-13	3.54E-05	1.83E-13	9.85E-25
ClsSz(50)	7.92E-10	0	1.29E-05	4.40E-40
ClsSz(55)	1.96E-07	1.13E-06	0	**0.02**

Table 4: The t-test p values for F-measure significance in the large-scale benchmark (Section 5, Table 2). Values in bold indicate that there is no statistically significant difference between S-ESC and the comparator method.

Data Set	MINECLUS vs. S-ESC	PROCLUS vs. S-ESC	STATPC vs. S-ESC	EM vs. S-ESC
50D	1.05E-08	1.14E-07	1.59E-70	9.41E-11
200D	**0.98**	0.009	1.01E-24	**1***
500D	NaN	**1***	2.36E-28	8.62E-09
800D	NaN	**0.77**	NaN	9.48E-14
1000D	NaN	**0.04**	NaN	2.30E-15

Out of the $5 \times 4 = 20$ tests on the large-scale datasets of Table 2 there are 5 cases in which comparator methods (MINECLUS and STATPC) fail to produce a result (NaN's in Table 4) and 5 cases in which the results are not significantly different (the bold cases in Table 4). In 8 cases S-ESC outperforms the comparator methods and only in 2 cases is S-ESC outperformed by comparators methods. The * symbol is used to denote the use of a single tailed test rather than a double-tailed test.

Acknowledgment

The authors gratefully acknowledge support from the NSERC Discovery grant program (Canada).

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*, pages 61–72. ACM, 1999.

[2] Rakesh Agrawal, Johannes Gehrke, Dimitrios Gunopulos, and Prabhakar Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. *SIGMOD Rec.*, 27:94–105, June 1998.

[3] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules. In *ACM International Conference on Very Large Data Bases*, pages 487–499, 1994.

[4] Ira Assent, Ralph Krieger, Andreas Steffens, and Thomas Seidl. A novel biology inspired model for evolutionary subspace clustering. In *Proc. Annual Symposium on Nature inspired Smart Information Systems (NiSIS)*, 2006.

[5] Carlos Bacquet, A. Nur Zincir-Heywood, and Malcolm 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.

[6] Lydia Boudjeloud-Assala and Alexandre Blansché. Iterative evolutionary subspace clustering. In *International Conference on Neural Information Processing (ICONIP)*, pages 424–431. Springer, 2012.

[7] Brett Calcott, Kim Sterelny, and Eörs Szathmáry. *The Major Transitions in Evolution revisited*. The Vienna Series in Theoretical Biology. MIT Press, 2001.

[8] Hyuk Cho, Inderjit S. Dhillon, Yuqiang Guan, and Suvrit Sra. Minimum sum-squared residue co-clustering of gene expression data. In *SIAM International Conference on Data Mining*, 2004.

[9] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182 –197, apr 2002.

[10] Arthur P. Dempster, Nan M. Laird, and Donald B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977.

[11] Xiaoli Zhang Fern and Carla E. Brodley. Solving cluster ensemble problems by bipartite graph partitioning. In *Proceedings of the 21st international conference on Machine learning*, pages 36–. ACM, 2004.

[12] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The weka data mining software: an update. *SIGKDD Explorations*, 11(1):10–18, November 2009.

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

[14] Eduardo R. Hruschka, Ricardo José Gabrielli Barreto Campello, Alex Alves Freitas, and AC Ponce Leon F. De Carvalho. A survey of evolutionary algorithms for clustering. *IEEE Transactions on Systems, Man, and Cybernetics: Part C*, 39(2):133–155, 2009.

[15] Mikkel 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.

[16] Yuval Kluger, Ronen Basri, Joseph T. Chang, and Mark Gerstein. Spectral bi-clustering of microarray data: co-clustering genes and conditions. *Genome Research*, 13:703–716, 2003.

[17] Hans-Peter Kriegel, Peer Kröger, and Arthur Zimek. Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering and correlation clustering. *ACM Transactions on Knowledge Discovery from Data*, 3(1):1–58, 2009.

[18] Hans-Peter Kriegel, Peer Kröger, and Arthur Zimek. Subspace clustering. *WIREs Data Mining and Knowledge Discovery*, 2:351–364, 2012.

[19] L. Liebovitch and T. Toth. A fast algorithm to determine fractal dimensions by box counting. *Physics Letters*, 141A(8), 1989.

[20] Yanping Lu, Shengrui Wang, Shaozi Li, and Changle Zhou. Particle swarm optimizer for variable weighting in clustering high-dimensional data. *Machine Learning*, 82(1):43–70, 2011.

[21] Lynn Margulis and René Fester. *Symbiosis as a Source of Evolutionary Innovation*. MIT Press, 1991.

[22] Geoffrey McLachlan and Thriyambakam Krishnan. *The EM Algorithm and Extensions*. Wiley-Interscience Publication, 1997.

[23] Gabriela Moise and Jörg Sander. Finding non-redundant, statistically significant regions in high dimensional data: a novel approach to projected and subspace clustering. In *ACM International Conference on Knowledge Discovery and Data Mining*, pages 533–541. ACM, 2008.

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

[25] 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.

[26] Seyednaser Nourashrafeddin, Dirk Arnold, and Evangelos Milios. An evolutionary subspace clustering algorithm for high-dimensional data. In *Proceedings of the ACM Genetic and Evolutionary Computation Conference Companion*, pages 1497–1498, 2012.

[27] Samir Okasha. Multilevel selection and the major transitions in evolution. *Philosophy of Science*, 72:1013–1025, 2005.

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

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

[30] Dan Pelleg, Andrew W Moore, et al. X-means: Extending k-means with efficient estimation of the number of clusters. In *International Conference on Machine Learning*, pages 727–734, 2000.

[31] Cecilia M. Procopiuc, Michael Jones, Pankaj K. Agarwal, and T.M. Murali. A monte carlo algorithm for fast projective clustering. In *ACM International Conference on Management of Data*, SIGMOD '02, pages 418–427, New York, NY, USA, 2002. ACM.

[32] David C. Queller. Relatedness and the fracternal major transitions. *Philosophical Transactions of the Royal Society of London B*, 355:1647–1655, 2000.

[33] Lily Rachmawati and Dipti Srinivasan. Multiobjective evolutionary algorithm with controllable focus on the knees of the pareto front. *IEEE Transactions on Evolutionary Computation*, 13(4):810–824, 2009.

[34] Ioannis A. Sarafis, Phil W Trinder, and Ali M.S. Zalzala. Towards effective subspace clustering with an evolutionary algorithm. In *IEEE Congress on Evolutionary Computation*, pages 797–806, 2003.

[35] Oliver Schütze, Marco Laumanns, and Carlos A. Coello Coello. Approximating the knee of an MOP with stochastic search algorithms. In *Parallel Problem Solving from Nature*, volume 5199 of *LNCS*, pages 795–804, 2008.

[36] Kelvin Sim, Vivekanand Gopalkrishnan, Arthur Zimek, and Gao Cong. A survey on enhanced subspace clustering. *Data Mining and Knowledge Discovery*, 26:332–397, 2012.

[37] Ali Vahdat, Malcolm I. Heywood, and A. Nur Zincir-Heywood. Bottom-up evolutionary subspace clustering. In *IEEE Congress on Evolutionary Computation*, pages 1371–1378, 2010.

[38] Ali Vahdat, Malcolm I. Heywood, and A. Nur Zincir-Heywood. Symbiotic evolutionary subspace clustering. In *IEEE Congress on Evolutionary Computation*, pages 2724–2731, 2012.

[39] Ian H. Witten and Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, 2 edition, 2005.

[40] Shelly Xiaonan Wu and Wolfgang Banzhaf. A hierarchical cooperative evolutionary algorithm. In *ACM Genetic and Evolutionary Computation Conference*, pages 233–240, 2011.

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

[42] Lin Zhu, Longbing Cao, and Jie Yang. Multiobjective evolutionary algorithm-based soft subspace clustering. In *Evolutionary Computation (CEC), 2012 IEEE Congress on*, pages 2732–2739, 2012.