SYMBIOTIC EVOLUTIONARY SUBSPACE CLUSTERING (S-ESC)

by

Ali R. Vahdat

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy

at

Dalhousie University
Halifax, Nova Scotia
November 2013

To my mother who made all this possible.

Table of Contents

List of Tables

List of Figures

Abstract

Application domains with large attribute spaces, such as genomics and text analysis, necessitate clustering algorithms with more sophistication than traditional clustering algorithms. More sophisticated approaches are required to cope with the large dimensionality and cardinality of these data sets. Subspace clustering, a generalization of traditional clustering, identifies the attribute support for each cluster as well as the location and number of clusters. In the most general case, attributes associated with each cluster could be unique. 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* (a 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 data sets, while an integrated subsampling process makes it scalable to tasks with a large number of data items. A bi-objective evolutionary method is proposed to identify the unique attribute support of each cluster while detecting its data instances. Benchmarking is performed against a well-known test suite of subspace clustering data sets with four well-known comparator algorithms from both the full-dimensional and subspace clustering literature: EM, MINECLUS, PROCLUS, STATPC and a generic genetic algorithm-based subspace clustering. 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 a particular data distribution or parameter sweeps might be necessary to provide comparable performance. A comparison is also made relative to a non-symbiotic genetic algorithm. In this case each individual represents the set of clusters comprising a subspace cluster solution. Benchmarking indicates that the proposed symbiotic framework can be demonstrated to be superior once again. The S-ESC code and data sets are publicly available.

List of Abbreviations Used

ACO	Ant Colony Optimization
CC	Cluster Centroid
CS	Clustering Solution
EC	Evolutionary Computation
EM	Expectation Maximization
EMO	Evolutionary Multi-objective Optimization
F-ESC	Flat Evolutionary Subspace Clustering
GA	Genetic Algorithm
ML	Maximum Likelihood
MLM	Multi-Level Mutation
NN	Nearest Neighbourhood
NP	Non-deterministically Polynomial
NSGA	Non-dominating Sorting Genetic Algorithm
PCA	Principal Component Analysis
PF	Pareto Front
PSO	Particle Swarm Optimization
RSS	Random Subset Selection
S-ESC	Symbiotic Evolutionary Subspace Clustering
SLM	Single-Level Mutation

Acknowledgements

I would never have been able to finish my dissertation without the guidance of my supervisor, help from friends, and support from my family.

I would like to express my deepest gratitude to my supervisor, Dr. Malcolm Heywood, for his excellent guidance, caring and patience. He provided me with an excellent atmosphere for doing research and patiently corrected my writing and financially supported my research.

I would also like to thank my examining committee; Dr. Nur Zincir-Heywood, Dr. Dirk Arnold and Dr. Nawwaf Kharma. Thanks for taking interest in my work. I definitely value your insights. I would also like to recognize the contributions of my fellow office mates, technical and otherwise.

Special thanks goes to my mother and my younger brother and sister. It was definitely difficult going through these years without them next to me, but I was always close to their hearts and minds. My mom has always supported, encouraged and believed in me, in all my decisions and endeavours and I am very grateful for this.

Chapter 1

Introduction

1.1 Clustering

The clustering task is used as a mechanism for summarizing properties of data in general and appears frequently in (unsupervised) exploratory data mining tasks, i.e. when there is no label information as is required for supervised learning tasks such as classification. Clustering was characterized by Berkhin as “a division of data into groups of similar objects” [12, 13]. The quality of the resulting clusters is impacted by the assumptions used to answer a series of ‘design questions’ which guide the application and/or specification of a clustering algorithm. Such a set of design questions might be summarized in the following way:

1. Training context: data exists which is sufficient for building a data summary which is representative of trends of wider interest to a user community. Moreover, there might well be a need to perform some data pre-processing, cleaning and/or standardization, before presenting data to a clustering algorithm.

2. Representation: a priori decisions are made regarding how to describe candidate clusters. Sample representations might require coordinates to define the location for each cluster centroid (in a centroid-based representation) or assume a medoid-based representation, in which case cluster prototypes are defined in terms of data instances. Moreover, a priori information may be available which potentially simplifies the task, e.g. the solution should consist of a specific number of clusters.

3. Cost function: the mechanism by which candidate solutions are ranked. This function determines which candidate solution is ‘better’ or ‘worse.’ Sample metrics might characterize inter- and/or intra-cluster distance and therefore establish the overall cost of partitioning the data.

4. Credit assignment: the mechanism for modifying current candidate solution(s) in proportion to the information provided by the cost function. Depending on whether a greedy or stochastic credit assignment process is assumed, solutions which are worse than those discovered previously may or may not be penalized.

5. Post-training assessment: the methods and metrics by which the generalization properties of the candidate solution are summarized. In the case of clustering this is often referred to as cluster validation.

Part of what makes the clustering task interesting is the unsupervised nature of the task. The unsupervised nature of the clustering task implies that its structural characteristics (number, distribution, shapes and orientations of clusters, if any) are not known [39]. A clustering task can be made even more challenging if multiple data types (binary, discrete, continuous and categorical) need to be dealt with. Outlier data points[1] and irrelevant or noisy attributes[2] can worsen the situation further by misguiding the clustering algorithms [49, 50].

Most clustering algorithms make some assumptions regarding the nature of the data and hence make prior assumptions regarding cluster distribution (usually normal or uniform) and/or cluster shape (usually hyper-spheres or hyper-rectangles). Therefore, an algorithm which performs well on one data set might perform poorly on a different data set depending on the underlying data distribution/behaviour.

The anticipated outcome from a clustering algorithm would be for similar objects to be grouped together and dissimilar objects to appear in different groups. Multiple ambiguities are implicit in such a high-level definition. Central to this ambiguity is the inability to identify a single unifying definition for objective similarity/dissimilarity. In a real D-dimensional space the similarity of objects could be approximated indirectly, e.g. by their distance. Similar objects have lower dissimilarity/distance values [50]. Thus, as soon as a specific distance/similarity/dissimilarity measure is adopted, potentially different cluster definitions will result [49, 118]. Each similarity/dissimilarity measure has certain advantages/disadvantages and/or bias, hence there is no single measure which is optimal for all clustering applications [47].

[1]Instance, object, exemplar and data item are all interchangeable terms used in the literature for a data point in a data set.

[2]Attribute, dimension and feature are all used interchangeably in the literature.

In the case of cost function, most algorithms utilize some type of distance-related criterion based on either the distance between the data items of a cluster (within-cluster/intra-cluster distance) or the distance between data items from different clusters (between-cluster/inter-cluster distance.) Assuming one over the other results in a specific bias to the form of clusters constructed. For example, algorithms which minimize the distance between the data items of a cluster, e.g. k-means, tend to find spherical clusters. Conversely, algorithms which optimize a criterion which enforces the grouping of proximate data items into the same cluster tend to find elongated clusters.

Since a single criterion cannot detect different cluster shapes optimally, recent works from evolutionary computation have employed multiple criteria to increase the chance of detecting clusters of arbitrary shapes, with different objectives trying to focus on different data distributions, and hence different cluster shapes. Hence, multi-objective optimization methods have been applied to clustering such as [43, 24]. We will come back to the concept of multi-objective optimization later in Section 4.3.

The pre-processing steps (e.g. data standardization and feature selection) and post-processing steps (e.g. cluster validation) can be potentially effective in the operation of a clustering algorithm. Data standardization is required in most clustering approaches because optimality criteria are usually based on some definition of distance, which can be biased toward attributes of larger or smaller range/extent depending on the optimality criterion being minimized or maximized. Feature selection, not necessarily a part of all clustering algorithms, tries to decrease the dimensionality of data sets before or during the clustering process. A typical data set might contain a number of redundant, irrelevant or even noise features. While redundant features are undesirable because they add to the computational cost of the algorithm, irrelevant (or noise) features can potentially misguide the clustering algorithm. Feature selection; therefore, potentially contributes to the identification of more informative features for the clustering task. Solution simplicity is another benefit of feature selection. Removing unnecessary attributes makes the final clustering solutions simpler and therefore easier to analyze and interpret for the end user.

On the other hand, cluster validation, which is a post-processing step, evaluates the final cluster structure and determines how trustworthy the end results of the algorithm are. Naturally, a cluster validation metric shares many of the properties assumed for a good distance metric, i.e. qualification of the intra-/inter-cluster properties. In this work extensive use will be made of label information; however, this information is not available during cluster construction. Indeed, such information is generally not available, but from the perspective of empirical benchmarking it does provide a very robust metric for cluster validation. In summary, improper choice of pre-processing methods might result in poor performance of the core clustering approach, whereas adopting a poor approach to post-processing can lead to accepting mediocre to degenerate solutions or even rejecting effective solutions.

In addition to the above five basic design questions, increasingly, clustering algorithms are expected to address a variety of additional challenges, as summarized below:

1. Scale to data sets with high dimensionality and large cardinality: Most approaches to the clustering problem are dependent on either the number of data items in a data set or the number of attributes in it, hence making them impractical for application to high-dimensional or large-scale data sets.

2. Simplify results: The ultimate goal of a clustering algorithm is to find the structure of data being analyzed to get a better understanding of the data structure. The simpler the results, the easier it is for users to understand, interpret and learn from the results. Moreover, it might even be possible to visualize the results in order to get an intuitive understanding of the data structure.

3. Self determining cluster count without prior knowledge: One of the main differences between a classification/supervised and a clustering/unsupervised task is the lack of knowledge regarding the true or optimal number of clusters, k. Not knowing the true number of clusters in a data set, a naïve distance-based criterion would assign each data point to a new cluster, making singleton clusters. A good distance measure is, therefore, a function of the number of clusters, but the meaningfulness of distance-based criterion functions potentially decreases with increases in cluster count; therefore, determining the optimal number of clusters is quite crucial. Recent works on data clustering attempt to estimate the optimal number of clusters in a data set by using cross validation, for example.

4. Decrease data-specific parameter count and sensitivity: The fewer parameters an algorithm needs to tune, the simpler it is for users with little or no domain knowledge to use the algorithm. Some algorithms require a large number of data-specific parameters which are not always easy to provide. Sensitivity to parameter tuning is also an issue. In this work the view is taken that an algorithm with a large number of parameters with low sensitivity is preferable to an algorithm with fewer high sensitivity parameters.

5. Detect arbitrary cluster shapes: The ideal scenario is to design and use a criterion function which is independent of the cluster shape. However, since most optimality criteria are distance-based and hence relative to data distribution and cluster shape, this is difficult to achieve in practice. One solution to this issue is to utilize more than one objective in a multi-objective metaphor in an attempt to detect different arbitrary cluster shapes.

6. Detect noise attributes and outlier instances: Noise attributes and outliers are almost inevitable in real data sets. Irrelevant/noise attributes can misguide the clustering algorithm whereas outlier instances, if frequent enough, might appear as a new cluster if not detected and treated properly. Prior to the introduction of subspace clustering algorithms, clustering algorithms would implicitly assume that all attributes were equally informative. Naturally, a 'filter' could be applied prior to clustering in order to perform dimension reduction (e.g. singular value decomposition). All clusters would then share a common 'reduced' attribute space.

7. Insensitivity to instance order: In an extreme case a clustering algorithm which is sensitive to the order of data instances provides different solutions each time instances are presented to it in a different order and therefore, it lacks robustness. A robust clustering algorithm would be expected to identify the 'same' solution regardless of the order in which instances are presented to it.

8. Simplify clustering of new unseen instances: Once a suitable partitioning of the initial set of data is reached, new data should preferably result in dynamic and incremental modifications to the current solution. This capability is particularly important under streaming data scenarios.

It is not the plan to tackle all the issues listed above directly. Indeed, Xu and Wunsch [118, 119] claim that: "There is no clustering algorithm that universally solves all problems." Rather, the initial focus will be on tackling the first three points. In other words, the plan is to design a clustering algorithm for high-dimensional large-scale data sets, while trying to keep clusters as simple and interpretable as possible while automating the determination of the optimum number of clusters embedded in the data. Also, this requires the ability to address the problem of noise attributes and assess the impact of outliers (point 6). The sensitivity to instance ordering (point 7) will be addressed by the approach assumed for scaling to data sets with larger cardinality. As for detecting arbitrary cluster shapes (point 5), a previously-proposed bi-objective scheme will be adopted and multiple cluster shapes are included in the empirical benchmarking; hence it is not considered to be a unique contribution of this work. Decreasing the data-specific parameter count and sensitivity (point 4) and simplifying the clustering of new unseen instances (point 8) are considered to be explicitly outside the remit of this work. Given these motivations, the goal will be to design a subspace clustering algorithm. Section 1.2 will articulate more formally the challenges of such a task.

1.2 High-dimensional Clustering

Although clustering in general is one of the more insightful data mining methods, it is only since 1998 that methods have been designed to deal with the new challenges put forward by modern automatic data generation and acquisition approaches. Usually this new emerging data has a large amount of attributes, and hence is deservedly called high-dimensional data [62]. Challenges posed by high-dimensional data are different from those posed by low-dimensional data and therefore potentially require different approaches to addressing them. Typical examples of high-dimensional data in which task-specific methods have been proposed include: gene expression analysis, text document analysis, video analysis and customer recommendation systems.

1.2.1 Curse of Dimensionality

Simply put, the 'curse of dimensionality' states that as the number of dimensions increases, the concept of distance as measured between points become less and less meaningful. That is to say, the points will appear increasingly to be equally distant from each other as the number of dimensions increases. In other words as the dimensionality increases the slight distance differences between pairs of points becomes more and more negligible.

This concept is illustrated in Figure 1.1, adopted from a review paper by Parsons *et al.* [87]. 20 points are distributed randomly in one dimension over a distance of two units. It can be observed in Figure 1.1(a) that there are approximately 10 points in each 1-*d* bin. In Figure 1.1(b), a second dimension is added and therefore there are approximately 5 points in each 2-*d* bin. Adding the third dimension as illustrated in Figure 1.1(c) stretches the data points further and approximately 2.5 points reside in each 3-*d* bin. Analogously, by adding more and more attributes, the points are spread out along other axes, pulling them further apart until they are almost equally far from each other making the utility of a distance metric less meaningful.

In their extensive review paper Kriegel *et al.* note that the frequently-used concept of the curse of dimensionality goes beyond this intuitive view and actually pertains to at least four different factors [63]:

- *Optimization problem* states that the difficulty of approaching any computational task through global optimization increases exponentially or at least polynomially 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 the farthest and nearest points as dimensionality increases (see also [87, 88]).

- *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 become more difficult to locate than others. For any given data object an irrelevant attribute not only does not help the clustering algorithm, it might even interfere with the performance of the similarity measure and disrupt the efforts to find a group to which the data item could belong. Therefore, these irrelevant attributes for a given data object

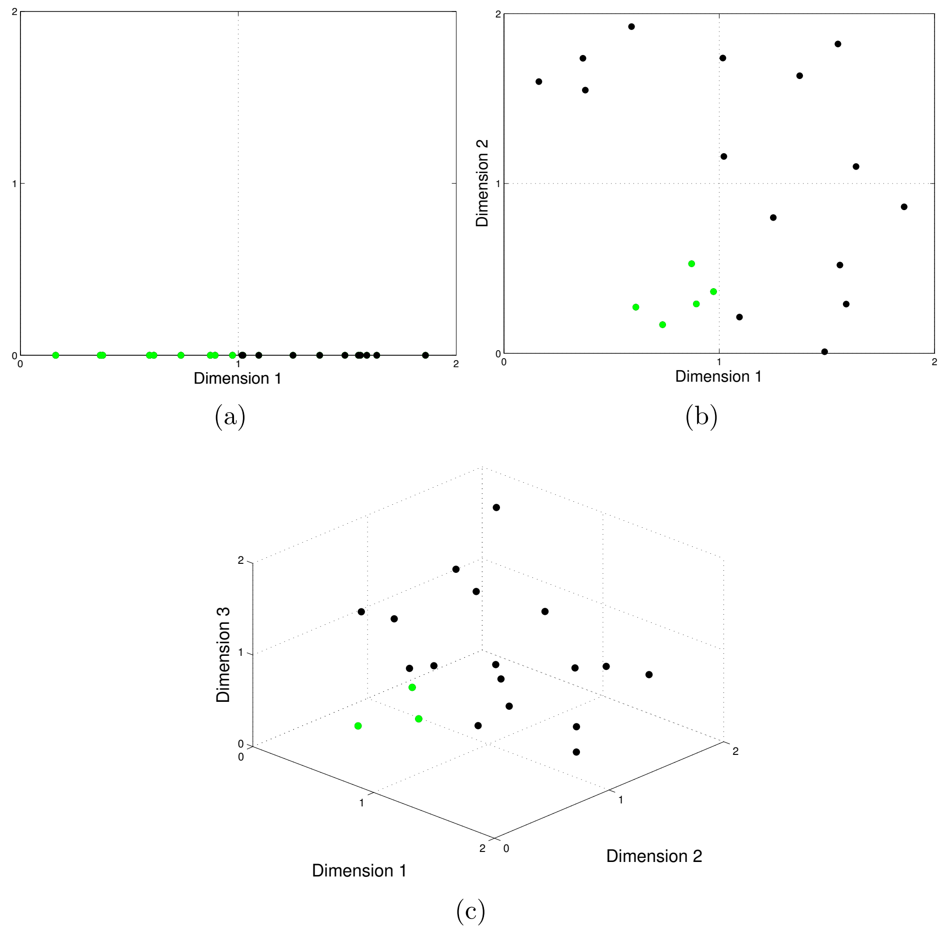


Figure 1.1: Curse of dimensionality. Points projected on (a) 1, (b) 2 and (c) 3 dimensions. Reproduced from Parsons *et al.* [87].

are referred to as 'noise' attributes as well. Naturally, as the total number of attributes increases, the cost of co-partitioning attributes and data instances can increase as well.

- *Correlated attributes* implies that correlations exist among subsets of data instances between two or more attributes. A feature reduction method might keep one (or a linear combination of some of these attributes) and disregard the rest. This, however, might lead to losing an important source of valuable information which can be insightful for the expert domain user. Potentially, a subspace cluster can now be defined orthogonally 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 [78, 80] (Section 5).

Dimensionality reduction. A traditional and common method to deal with high dimensionality is dimensionality reduction in the form of feature selection or feature transformation. A feature selection or feature transformation method reduces the original feature space so that the data set is clustered more accurately under the new reduced feature space [81]. Feature selection approaches find a single subset of attributes to group all data points, which can be different than reality for a lot of applications, as will be discussed shortly.

On the other hand, feature transformation finds linear combinations of attributes from the original feature space and replaces a subset of features with a single attribute representing the feature subset [69]. The resulting reduced feature space might improve clustering performance (or any other data mining task), but the solution does not convey any further useful information once the original feature space is lost. Also similar to feature selection methods, feature transformation approaches find a single subspace for all data set instances thus failing to identify subspace structure when it exists.

Subspace definition. However, there are cases in which not all instances are easily described by a single attribute subset (alternatively called a 'subspace'). As mentioned earlier, in high-dimensional data sets each instance is described using many features. However, all attributes might not be relevant for identifying a particular instance/data object, and the attribute subset (subspace) relevant to describing such an instance might not be relevant to another data object. Therefore, different subspaces can be relevant to different clusters. Since feature selection and dimensionality reduction methods such as PCA are not useful in these cases alternative methods are required to deal with this new challenge.

Problems associated with finding subspaces: Finding relevant subspaces in the search space of possible subspaces is known to be a NP-complete[3] task [104, 36]. Enumerating all possible subspaces in a D-dimensional space is in the order of $O(2^D)$.

[3]A decision problem L is NP-complete if it is in the set of NP problems and also in the set of NP-hard problems.

Furthermore, this comprises only half of the final task of a clustering algorithm dealing with high-dimensional data. The other half is detecting final clusters, i.e. grouping together data points which use the same subspace. Note as well that these two tasks need to be solved simultaneously, so heuristics are generally considered for both tasks.

1.2.2 Defining Types of Subspaces

Following from the properties contributing to the curse of dimensionality – that is the role of irrelevant and correlated attributes – Kriegel *et al.* observe that there are two distinct types subspaces; axis-parallel and arbitrarily oriented subspaces [63].[4]

Axis-parallel subspaces. The first type of subspaces follow the intuition that the data within irrelevant attributes are uniformly distributed, i.e. the variance of data is high. Conversely, in the case of attributes relevant to a subspace, 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. If data points are visualized in full-space (including all relevant and irrelevant attributes,) cluster points form a hyper-plane parallel to the irrelevant attributes. Kriegel *et al.* call methods dealing with this category of subspaces as ‘subspace clustering’ or ‘projected clustering.’ Cluster 3 in Figure 1.2 is an axis-parallel subspace.

Arbitrarily-oriented subspaces. Second type of 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 orthogonally to the linear dependency and is more difficult to detect in terms of the projection on individual attributes. If data points are visualized in full-space (including all relevant and irrelevant attributes), cluster points form a hyper-plane orthogonal to the hyper-plane made by the linear combination of correlated attributes making the arbitrarily-oriented subspace. Kriegel *et al.* call methods dealing with this category of subspaces as ‘correlation clustering.’ Clusters 1 and 2 in Figure 1.2 are in an arbitrarily-oriented subspace.

Special cases. 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

[4]Patrikainen and Meila also recognized the axis-aligned (parallel) and non-axis-aligned categories [88], whereas Parsons *et al.* [87] investigate only different methods within the axis-parallel category.

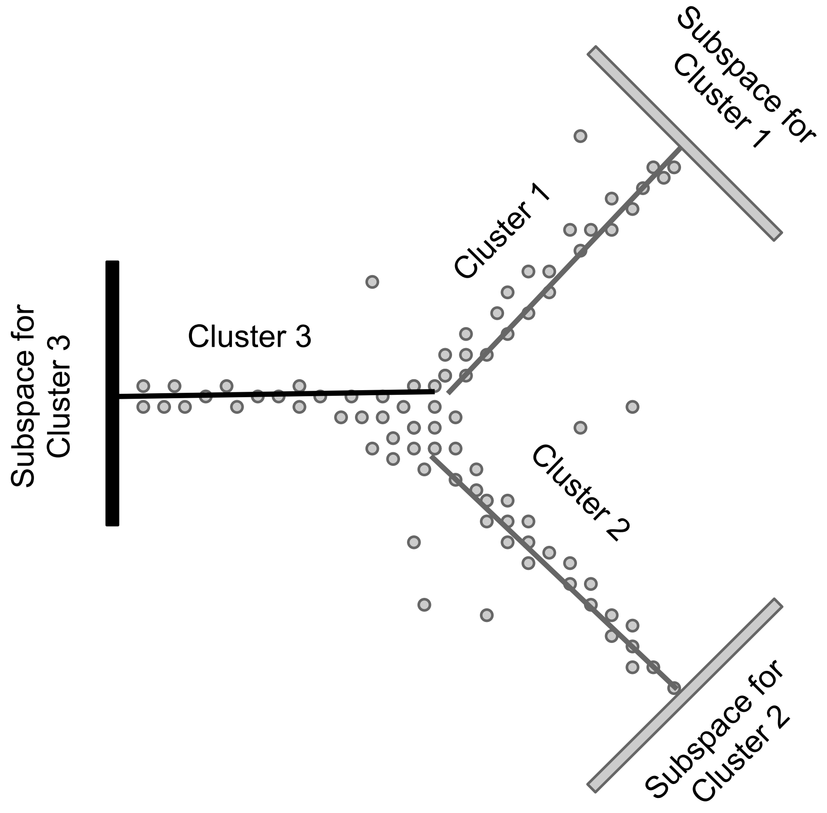


Figure 1.2: Axis-parallel vs. arbitrarily-oriented subspaces. Cluster 3 exists in an axis-parallel subspace, whereas clusters 1 and 2 exist in (different) arbitrarily-oriented subspaces. Reproduced from Kriegel *et al.* [62].

algorithms has been developed. Typical examples include information retrieval from document repositories or the analysis of microarray data. Such algorithms – often referred to as bi-clustering, co-clustering, two-mode clustering or pattern-based clustering – make use of application-specific distance metrics or prioritize a capacity for addressing very sparse attribute spaces (see [62, 63]).

In this research axis-parallel clusters are assumed. Such a task definition leads to several algorithmic insights which will guide the approach taken in the proposed algorithm. Moreover, there is a wider availability of benchmark data sets and software implementing alternative algorithms for the axis-parallel case which will provide greater scope for empirical evaluation. Therefore, in the following background discussion the focus will be on methods associated with the axis-parallel scenario. PCA and Hough transform-based methods appear currently to be the dominant approaches for the case of arbitrarily-oriented subspaces [63, 103].

1.3 Subspace Clustering

Apart from the accuracy of results, the main motivation behind any clustering method dealing with high-dimensional data – as it is for feature selection and feature transformation – is 'parsimony.' End users want both accurate and easy-to-interpret solutions; thus, a highly accurate and complicated solution is not always preferred to a less-accurate yet simple solution. Although simplicity is by no means an excuse to sacrifice accuracy; however, a slight decrease in accuracy, in many situations, can be disregarded to gain simplicity, interpretability and even less computational cost to run the built model.

This is where subspace clustering methods step in to provide solutions where feature selection and feature transformation approaches address only the simpler case of dimension reduction using a common subspace. In the situation mentioned earlier methods which are able to detect the (potentially) unique subset of attributes which form each cluster are needed. In other words there is a subset of attributes, also called a *subspace*, under which instances from one particular cluster are easily distinguished from other instances. This subspace can – but not necessarily will – be unique to that cluster. Thus, when different clusters use a different (and potentially unique) reduced set of attributes, they are called *subspace clusters*. Subspace clustering is a generalization of traditional clustering which seeks to find clusters in different (attribute) subspaces while simultaneously distinguishing the instances belonging to each cluster [87, 88, 80, 83, 62, 63, 103]. Finally, in subspace clustering not only are the number of clusters and distribution supporting clusters unknown, but also the attribute support for clusters requires discovery.

The principal benefit of following such an approach is the insight which the resulting solution provides. For example, recently the data mining of the genes correlated with breast cancer has gone from an attribute space of 700 to just 3 without compromising the resulting model accuracy [67]. Naturally, such a result provides the insight necessary to inform more successful clinical practice.

One way of designing subspace clustering algorithms in general is through the concept of frequent itemset mining the general goal of which is to construct a partitioning of each attribute [4]. This forms the basis for a lattice or grid. Then subspace clusters are constructed 'bottom-up' through a combinatorial search conducted with respect to the lattice. In short, decisions are made first with respect to the appropriate attributes and then clusters are formed from the resulting co-occurrences. As reviewed in Section 2, such a framework is likely to be effective for axis-parallel clusters (the focus of this research), but less effective under arbitrarily-oriented subspaces or subspace clustering algorithms designed for specific applications, e.g. bi-clustering, co-clustering, two-mode clustering or pattern-based clustering [62, 63, 103]. In particular, the focus here is in the case of dense as opposed to sparse data sets. One implication of this is that a subspace clustering algorithm must adopt a 'hard' as opposed to 'soft' approach to attribute removal; whereas under sparse high-dimensional data sets adopting a 'soft' approach to attribute removal is sufficient. Different approaches to the subspace clustering task and their respective implications are investigated further in Section 2.

The approach proposed by this thesis will make the axis-parallel assumption and assumes dense as opposed to sparse multi-dimensional spaces. Moreover, the approach adopted – Symbiotic Evolutionary Subspace Clustering (S-ESC) – is based on genetic algorithms and a symbiotic approach to coevolution. In short, two populations coevolve with one representing candidate cluster centroids and the second attempting to group them into effective cluster partitions. Also, fitness evaluation is based on a bi-objective cost function. What follows is a discussion of the motivation and background for adopting such an approach.

1.4 Evolutionary Computation

Looking at clustering as a special case of an optimization problem (known to be a particular kind of NP-hard grouping problem [36]) has motivated researchers to use meta-heuristic methods. Evolutionary Computation (EC) has a good record for producing feasible solutions to NP-hard problems with little or no prior information. Over the past decade EC has been applied to various forms of the clustering problem [47]. Evolutionary computation maintains multiple candidate solutions (or population) and therefore addresses the central machine learning issue of credit assignment through the concepts of: 1. (parental) selection and replacement; and 2. variation operators, i.e., the mechanism by which new variants of parent individuals are obtained. Such processes are often inspired by biological mechanisms of evolution. However, it is the use of a population which results in a significant departure from mainstream approaches to machine learning/optimization[5] and therefore an opportunity for pursuing alternative problem solving capabilities. In addition, the frequently stochastic nature of the credit assignment process provides more freedom in the representation assumed. For these reasons the utilization of EC has been chosen for the development of an algorithm for subspace clustering, with the objective of benchmarking against currently available non-EC methods.

1.4.1 Evolutionary Computation for Clustering

Among the different approaches to evolutionary computation both Genetic Algorithms (GA) and Particle Swarm Optimization (PSO) have been used widely as the basis for clustering algorithms in general [47]. In the work described by this thesis a GA basis is assumed, where this reflects an underlying assumption regarding the use of a discrete representation. Most GA-based clustering algorithms tend to encode a partitioning solution into each individual resulting in a chromosome corresponding to the size or dimensionality of the data set at hand. These partitioning individuals are then altered and evolved through selection, recombination and variation operators until a goal condition or stop criterion is satisfied.

Early evolutionary works on full-space clustering tried simply to evolve the parameters of a traditional clustering algorithm (e.g. k-means) and used the clustering algorithm inside the evolutionary loop as a fitness function [8, 64, 56, 110]. Hruschka *et al.* [47] assembled a recent and comprehensive review of evolutionary algorithms for clustering, but no subspace or projected clustering approaches were identified. Full-space and subspace evolutionary clustering methods will be discussed in Section 2.3.

The following highlights two of the more recent developments from EC which form key components of the proposed framework followed by a summary at a high level of the proposed Symbiotic Evolutionary Subspace Clustering algorithm or S-ESC.

[5]The use of stochastic as opposed to gradient information during credit assignment might be considered a fundamental difference. However, gradient information can be employed as well in evolutionary methods whereas stochastic credit assignment does not require the adoption of a population-based framework.

1.4.2 Multi-objective Evolutionary Algorithms

Single-objective clustering. Traditional methods of data clustering using a single objective can be categorized based on the criterion they optimize [43]. One group of clustering algorithms tries to form *compact* spherical clusters by minimizing the intra-cluster distance between data points and cluster prototypes, as in k-means [38, 74], average link agglomerative clustering [114] and self-organizing maps [59]. Another group of clustering algorithms tries to form clusters in which data items close to each other fall into the same cluster, hence optimizing *connectedness*. This category of clustering algorithms can find clusters of arbitrary shape, but they might fail when clusters are close to each other. Two examples of this group of clustering algorithms are the single link agglomerative clustering [114] and OPTICS [5].

Pathologies. In addition, there are problems associated with optimization functions with respect to single objective clustering algorithms; or pathologies. One such example can be given in the case where only within-cluster distance is used in a clustering algorithm to evaluate different clusterings. In this case the best solution (according to the performance function) is a solution with as many singleton clusters as the number of instances, i.e. clusters with only one instance. This solution minimizes the within-cluster criterion globally, but obviously fails to cluster the data set. Clearly, this case penalizes solutions with a large number of clusters and which favours clusterings with a small cluster count. On the other hand, adopting a between-cluster criterion potentially results in another pathology, one with only one cluster containing all instances. What is needed in this case is a mechanism to penalize solutions with a small number of clusters and favours clusterings with larger cluster support. The likelihood of encountering such pathological cases can be reduced significantly by introducing a second objective which contradicts the first objective; in the simplest form an objective to minimize cluster count for the former example and an objective to maximize cluster count for the latter example.

Multi-objective Evolutionary Optimization. Methods maintaining multiple solutions, such as evolutionary computation, provide the opportunity for retaining multiple equally 'good' candidate solutions simultaneously. Thus, under a multi-objective setting the concept of Pareto dominance or Pareto front (PF) has been employed frequently to identify and retain all the individuals which are equally good in how they trade the multiple objectives off against each other. From the perspective of the clustering task in general, this represents an effective framework for addressing the possible conflicting aspects of distance metric selection. The seminal work of Handl and Knowles [43] illustrated the effectiveness of such a scheme under full attribute space clustering. This issue will be presented in more detail in Section 4.3.

1.4.3 Symbiosis

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 [75]. A popular example of symbiosis in nature is the interaction between clownfish and sea anemones. On the one hand, the clownfish consumes invertebrates which can potentially harm the sea anemone and the fecal matter from the clownfish provides the main source of nutrients for the sea anemone. On the other hand, the sea anemone protects the clownfish with its stinging cells to which the clownfish is immune [26].

Unlike competitive or cooperative forms of coevolution – in which two populations interact with one another – symbiosis has been associated specifically with the potential development of higher-level organisms from independent species (cf., the major transitions of evolution) [18]. Thus, the capability to aggregate together lower-level 'symbionts' into higher-level 'hosts' represents an egalitarian transition capable of describing, for example, the origin of mitochondria within eukaryotic cells [92].[6] In addition, such a process introduces the concept of 'levels of selection' [86] in which fitness is either evaluated at the level of the symbiont or host.

The work developed by this thesis assumes a symbiotic evolutionary approach toward subspace clustering. Two separate – yet interacting – populations are coevolved simultaneously; the lower-level population of cluster centroids and the higher-level population of clustering solutions respectively. The cluster centroid population defines the location for a single (subspace) cluster. The clustering solution population indexes some subset of individuals from the cluster centroid population. In the context of symbiosis cluster centroids are the lower-level *symbionts*, whereas clustering

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

solutions are the higher-level *hosts*. Therefore from this point on in this text 'cluster centroid' and 'symbiont' are used interchangeably for individuals of the lower-level population whereas 'clustering solution' and 'host' are used interchangeably for individuals of the higher-level population. Occasionally 'member' and 'team' terms might be used to refer to these two entities respectively.

Adopting such a symbiotic framework permits the use of different representations for defining cluster centroids and clustering solutions. Moreover, this recognizes explicitly that there are two search processes under way; that for the cluster centroids and that for clustering solutions. Various algorithmic rules of thumb are also apparent. For example, the rate at which individuals turn over at the host level should be greater than that at the symbiont level.

1.5 S-ESC in a Nutshell

The proposed approach, hereafter referred to as Symbiotic Evolutionary Subspace Clustering (S-ESC)[7], assumes a symbiotic evolutionary approach to the subspace clustering problem. S-ESC is composed of three major components and two minor sub-components which are summarized here, illustrated in Figure 1.3, and detailed later in Section 4.

Starting from the upper right corner of Figure 1.3 the data set is first projected onto each attribute one by one. A classical clustering algorithm such as X-means or EM is then applied to each attribute independently. The clustering algorithm summarizes each attribute by a set of 1-d cluster centroids (or 1-d centroids for short). Super-imposition of these 1-d centroids forms the grid on the lower right corner of Figure 1.3. This reduces the task of building subspace clusters to that of conducting a discrete combinatorial search over the candidate 1-d centroids as defined by the grid; albeit requiring the axis-parallel assumption – where it is a constraint associated with the vast majority of current subspace clustering algorithms [88, 62, 63, 103].

A population of cluster centroids, as seen in the middle lower of Figure 1.3, is created each supporting a (potentially unique) subset of attributes (a.k.a. subspace);

[7]The S in S-ESC is hyphenated because a flat (as opposed to hierarchical), non-symbiotic version of the algorithm was introduced later on as a baseline method for the sake of comparison to get a better understanding of the effect of symbiosis involved in the S-ESC on the performance results. This non-symbiotic version is called Flat Evolutionary Subspace Clustering (F-ESC).

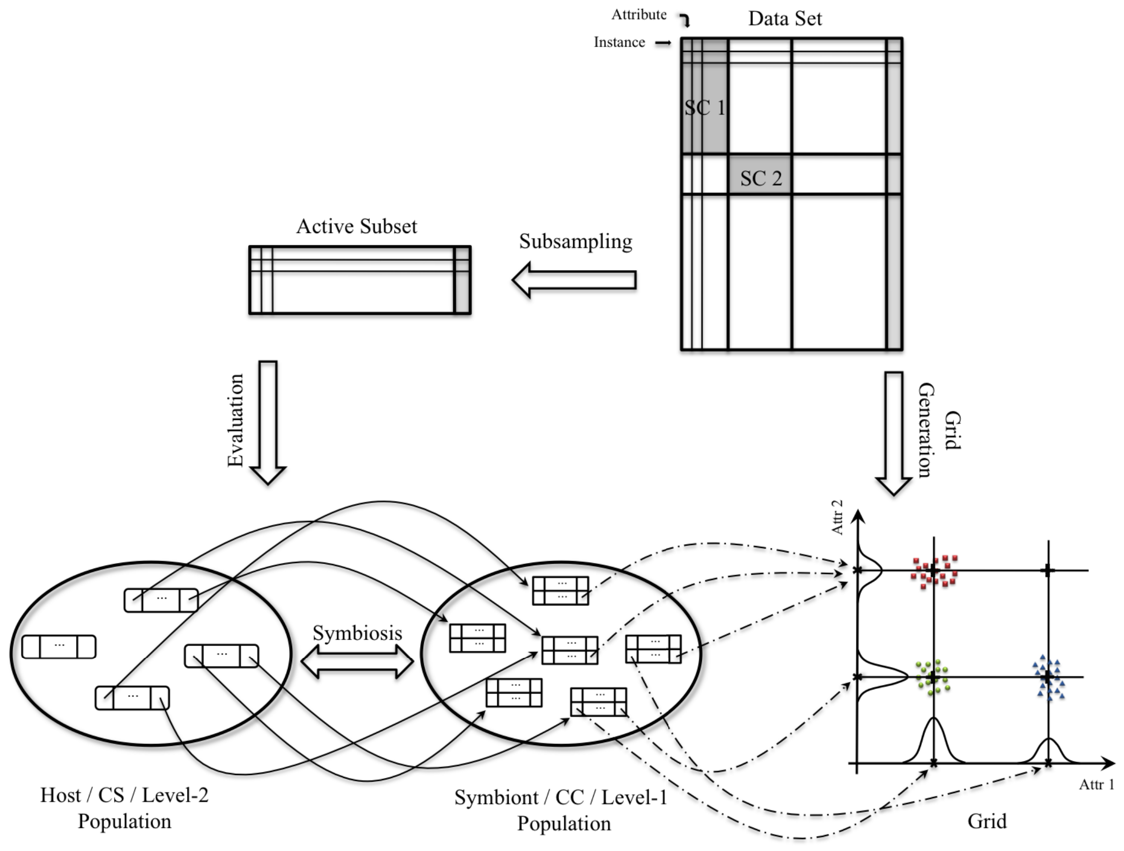


Figure 1.3: S-ESC components and sub-components and their interaction in summary.

hereafter referred to as the cluster centroids (CC) or symbiont in the case of the symbiotic metaphor. Next a population of different combinations of cluster centroids is created, hereafter referred to as a clustering solution (CS) or host under the symbiotic metaphor. This is seen in the lower left corner of Figure 1.3. The CC/symbiont population and CS/host population will coevolve to determine the optimal partitioning of the data.

A cluster centroid (CC) is a point in the attribute space and, as the name suggests, acts as the centroid of a subspace cluster once it is linked to a clustering solution (CS). It is created by putting together a number of 1-d centroids from different attributes. A candidate CS, on the other hand, is formed by linking a number of cluster centroids from the CC population. Each CS is evaluated based on two objectives after a nearest neighbour (NN) assignment process which assigns each data point to the nearest CC, considering only the attributes supported by the corresponding CC. In other words performance of solutions can be defined only at the CS level where each CS is expressed in terms of a group of CCs.

At each generation μ CS parents produce $\lambda = \mu$ CS offspring. The $\mu + \lambda$ CS individuals are then re-ranked under a Pareto multi-objective coevolutionary algorithm with only the best half retained as parents for the next generation. While the CS individuals evolve toward the optimal combination of CCs, cluster centroids themselves go through a parallel diversifying process to fine-tune each CC within the CS it is a part of, care of level-specific variation operators.

Within any generation, a subsampling process defines the subset of data points against which fitness evaluation is performed. The motivation for the subsampling process is the computational cost of evaluating the objective functions against all data instances. Specifically, connectivity, one of two S-ESC distance-based objectives (Section 4.3), has a time complexity of $O(N^2)$, where N determines data set cardinality. The subset of instances selected by the subsampling process is called an 'active subset' and is shown in the upper left corner of Figure 1.3. The potential drawback of such a scheme is that errors are introduced into the fitness ranking process.

Once the evolutionary loop is over, the bi-objective evolutionary algorithm provides a pool of clustering solutions (hosts) which are equally good from a Pareto evolutionary point of view. However, in reality, some of these solutions try to minimize one of the objectives regardless of whether the other objective makes them useless for partitioning the data. Some of these solutions are easy to find manually but there are also solutions which are very similar and very difficult to decide whether one is better than the other. The proposed approach uses an automated process to find a single solution to represent the pool of solutions which is called the 'champion' or 'winner' solution.

Empirical evaluations will be performed relative to a common set of benchmarks proposed by the study of Moise *et al.* [80]. This provides the basis for conducting a comparison against state-of-the-art algorithms under common benchmarking conditions. In part due to the comparatively low dimensionality of data sets in [80] 5 more data sets are generated ranging from 50 to 1000 dimensions. Comparator algorithms are chosen from popular full-dimensional and subspace clustering algorithms. EM represents the only full-dimensional clustering algorithm in these comparisons [30, 77]. The studies by Moise *et al.* introduce STATPC [78, 80] as the preferred subspace clustering method, whereas the benchmarking survey of Muller *et al.* recommends MINECLUS [122, 123] and PROCLUS [1] as the most consistent methods among subspace clustering algorithms [83]. The empirical evaluation reported here compares S-ESC against all 4 methods on 59 data sets (54 data sets from Moise *et al.* survey and an additional 5 large-scale data sets generated to complement those of Moise *et al.*) and evaluates them with respect to cluster accuracy and solution simplicity. In addition assessments are made of the impact of including outliers in the data and the proposed S-ESC is compared with a variant in which the symbiotic relationship is removed and the two-level hierarchical representation is compressed into a 'flat' chromosome (each individual has to express both CS and CC in a single chromosome, hence only one population). The benchmarking studies are summarized in broad terms below:

- the comparator algorithms required a lot of effort to tune the parameters to each configuration of a task or required significantly more prior knowledge to configure. By contrast, S-ESC assumed a common parameterization throughout.

- two of the comparator algorithms failed to return results for the larger scale tasks within a 24 to 96 hour computational period. Indeed, S-ESC was affected the least by increases to data set cardinality or original task dimensionality.

- S-ESC and EM consistently returned the best results across all of the 59 data sets.

- assuming a coevolutionary host–symbiont representation as opposed to a flat representation (S-ESC versus F-ESC) provided the basis for a more capable algorithm, particularly as the number of attributes supporting a subspace cluster decreased.

This thesis proceeds in the following manner: Chapter 2 provides a survey of the related literature on variants of subspace clustering and evolutionary clustering; Chapter 3 explains the four comparator algorithms against which S-ESC is compared; details of the S-ESC algorithm appear in Chapter 4; Chapter 5 explains the benchmarking methodology with the results following in Chapter 6; Lastly conclusions and potentially useful future work are summarized in Chapter 7.

Chapter 2

Background and Related Work

The fact that different points may cluster better using different subsets of dimensions was observed for the first time by Agrawal *et al.* and published under the name CLIQUE in 1998 [3]. Since the publication of CLIQUE works pertaining to the basic objective of discovering attribute support as well as the location of different clusters within the same data set has appeared under multiple terms: subspace clustering [3], projected clustering [1], projective clustering [91], bi-clustering [58], co-clustering [23], two-mode clustering, correlation clustering and pattern-based clustering, to name a few. Developments in subspace clustering have been reviewed multiple times over the last ten years [87, 88, 80, 83, 62, 63, 103].

In order to develop the topic here the framework established by Kriegel *et al.* [62, 63] will be used, while noting where this builds on previous surveys.[1] As established in Section 1.2.2, Kriegel *et al.* distinguish two basic types of subspace: axis-parallel and arbitrarily-oriented. This work will concentrate on the axis-parallel case, as does the majority of published research. Section 2.1 will comprise a review of subspace clustering algorithms under the guide of the categorization established by Kriegel *et al.*

Conversely, the surveys by Moise *et al.* and Muller *et al.* pursue a benchmarking approach in their work [80, 83]. Recommendations from these surveys will be used to inform the selection of comparator algorithms and data sets for benchmarking purposes. Comparator methods will be discussed in more detail in Section 3. Evolutionary methods used for clustering will be reviewed, specifically, a few full-space clustering algorithms using evolutionary methods which are pertinent to the development of the S-ESC algorithm and then the works using evolutionary methods for subspace clustering.

[1] Patrikainen and Meila also divide subspace clustering algorithms into ‘axis-aligned’ and ‘non-axis-aligned’ methods [88].

2.1 Categorization of Clustering Methods for Axis-parallel Subspaces

Parsons *et al.* published one of the first surveys on existing methods in 2004 and introduced the simplest categorization of subspace clustering algorithms associated with axis-parallel clusters [87]. They divided different methods into two groups relative to the algorithmic properties: *top-down* and *bottom-up*.

Top-down. These methods start by assuming that all attributes contribute to each cluster and propose metrics for identifying the more dense subspaces within the original attribute space, attempting to reduce incrementally the dimension of attributes per cluster. In order to minimize the impact of the factors associated with the curse of dimensionality heuristics are employed (e.g. regarding the sampling of data points). Other works determine a random set of attributes for a random set of points so that the points meet the cluster criterion once projected to the given attribute set. The attribute set and point set are iteratively refined until the final subspaces and cluster points are found.

There are many variants of top-down subspace clustering algorithms in the literature, PROCLUS [1], ORCLUS [2], FINDIT [116], δ-clusters [120] and COSA [40], to name a few. PROCLUS and COSA will be summarized, being representative of the top-down category, in part because these algorithms have repeatedly performed well in various benchmarking studies [83].

PROCLUS (PROjected CLUStering) [1] is a top-down partitional subspace clustering algorithm. Partitional clustering algorithms tend to focus on the final clustering results rather than on the individual clusters independently. PROCLUS finds the best set of medoids using a hill climbing process similar to CLARANS [84], but adapted to work on subspaces. PROCLUS works in three phases. In the first phase it randomly samples $A * k$ instances of the data set. Then it greedily prunes this set until only the $B * k$ more promising instances are left. The objective is to ensure that at least one instance is selected from each true subspace cluster. Once an optimal subset of medoids is found, an iterative phase then selects k instances as medoids, finds the instances assigned to each medoid, determines the relevant attributes for each medoid, and randomly replaces 'bad' medoids iteratively until there is no change in the clustering for a number of iterations. Instances assigned to medoid m_i are those which fall within a hyper-sphere around m_i with a radius equal to the distance between

m_i and the closest medoid. Once instances assigned to each medoid are determined, attributes related to each medoid are determined according to within-cluster dispersion. The average of the Manhattan segmental distance between the cluster medoid and its assigned instances are used to identify attributes with the smallest average distance while satisfying the constraint that at most $k * l$ attributes can be selected where l is the input parameter specifying the average dimensionality of the subspace clusters. A refinement phase at the end of the algorithm tries to tune the instances and attributes assigned to each medoid.

The drawback of PROCLUS is that it needs a lot of prior knowledge about the data set such as the true/desired number of subspace clusters as well as the average dimensionality of subspace clusters. In addition PROCLUS tends to form hyper-spherical subspace clusters which have approximately the same number of attributes per cluster, which could potentially limit the generality of the algorithm.

COSA (Clustering On Subsets of Attributes) [40] is also a top-down subspace clustering algorithm. It is unique in that it assigns a weight to each attribute for each instance, not each cluster. Initially the k-nearest neighbours (kNN) of all instances are determined with all attributes having same weight. These neighbourhoods are used to calculate the weights of all attributes for each instance where these attribute weights will in turn be used to re-calculate the pairwise distance of instances during the kNN evaluation. The process repeats until all attribute weights are fixed. While attributes relevant to a cluster gain larger weights, the neighbourhood for each instance accepts more and more relevant instances. The output of the algorithm is a matrix of distances based on the weighted inverse exponential distance. This matrix forms the basis for the application of a distance-based clustering algorithm of the user's choice. Once the clustering is performed attribute weights for each cluster member are compared and an importance measure for each attribute per cluster is determined.

COSA parameterization assumes that a scalar (λ) is defined a priori that defines a relative weighting for large versus small attribute subspaces. As with PROCLUS, COSA is more apt to find clusters with similar attribute count. In addition, COSA calculates the attribute weights for single instances as well as for pairs of instances, and not for each cluster. Therefore, once clustering is done, the relevant dimensions must be calculated based on the attribute weights assigned to cluster members.

Bottom-up. These methods attempt to make use of attribute-specific information such as frequent itemset mining as popularized by the APRIORI algorithm [4], in order to recursively construct groups of attributes with common itemsets. They start by constructing low-dimensional – usually 1-d – partitions of each attribute; where the most recent methods do so using density information. This forms the basis for a lattice or grid. Then subspace clusters are constructed bottom-up through a combinatorial search conducted with respect to the lattice.

A property which bottom-up methods use is that a k-dimensional bin/cell/region is dense if and only if there is a dense bin in all of its $(k-1)$-dimensional projections. Therefore attributes with no dense bins can be omitted easily from the process and dense higher-dimensional bins can be composed from multiple dense lower-dimensional bins. Finding the dense bins in each attribute provides a grid over which the dense higher-dimensional bins are easy to locate. The grid generation process can be done statically (with fixed-sized bins) or dynamically (with variable-sized bins). In the case of either grid generation method, it might be expected that the quality of the initial grid will influence the resulting quality of the subspace clusters. Other than grid size (in static grids), a density threshold may also be tuned for a bottom-up approach. A small value for density threshold may result in a large number of clusters, whereas a large value might result in labelling some instances belonging to a cluster as outliers.

Methods like CLIQUE [3] and ENCLUS [21] use fixed pre-determined locations for bins in each attribute, whereas other bottom-up methods like MAFIA [41], CBF [20], CLTree [68], DOC [91], FASTDOC [91] and MINECLUS [122, 123] use data-driven techniques to define the location of the bins. What follows is a summary of CLIQUE and MINECLUS as representatives of the bottom-up category.

CLIQUE [3] the first widely known subspace clustering algorithm is a bottom-up method which uses pre-defined static bins. The first step is to find dense subspaces using an APRIORI-style technique. The subspaces then get pruned and the ones with the highest coverage are kept. The next step finds the dense bins which are close to each other and merges them to make subspace clusters using a greedy growth technique. There is then a pruning step at the end which attempts to identify and remove redundant regions. Using a static grid, CLIQUE is capable of identifying hyper-rectangular clusters in disjoint or sometimes overlapping subspaces. CLIQUE also allows data instances to belong to multiple clusters. Conversely the principal disadvantages of CLIQUE are that parameter sweeps are necessary before application and current implementations of the algorithm do not appear to scale well with high dimensional clusters in data sets [83, 80].

MINECLUS [122], which is an improvement on the popular DOC [91] method, is a bottom-up density-based subspace clustering algorithm which assumes a hyper-cubic shape for subspace clusters. DOC computes subspace clusters one-by-one. Each time a random instance – hereafter called pivot point or p – from a data set is selected. Then a subset P of instances is selected randomly and considered to be in the same subspace cluster as the pivot point. Relevant attributes are then determined based on the distance between the projection of the pivot point and the projection of P points on the attribute being evaluated. If all P projections are within the distance w of the projection of the pivot point, then the attribute is considered relevant. This process is repeated for $2/\alpha$ pivot points (where α is a fraction of data set instances), and for each pivot point m potential subspace clusters are tried. The best subspace cluster (determined with a μ function which is a function of two user-input parameters, α and β) is then added to the list of subspace clusters in the solution. MINECLUS adds a method on top of DOC to find the subspace cluster once a pivot point is selected randomly and introduces some refinement strategies to parameter settings. Later, it will be noted that parameter sensitivity is still evident (Section 5.3.2) and the assumption regarding the universal applicability of hyper-cube cluster 'shapes' may or may not be appropriate.

Since the original use of this categorization by [87], greater algorithmic diversity has prompted a more refined categorization [62, 63, 103]:

1. *Projected* (or hard k-medoid) *clustering* – a form of top down clustering – attempts 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.[2] 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

[2] k-medoid and k-means algorithms have several similarities. The principal difference is that the k-medoid algorithm defines centroids in terms of data instances. Conversely, the k-means algorithm attempts to define clusters in terms of coordinates representing the centroids directly.

curse of dimensionality. Thus, in the case of PROCLUS [1], one of the more successful and widely utilized cases of projected clustering, it is assumed that the number of clusters and average cluster dimensionally are known a priori.

2. *Soft projected clustering* are optimization algorithms which 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 a potentially very low weighting. Kriegel *et al.* note that such methods are variants of EM clustering [62, 63].

3. *Subspace clustering* explicitly pursues the goal of finding clusters in all subspaces of the original attribute space. This category of methods generally needs to pursue some form of a bottom-up approach in order to address the various aspects of the curse of dimensionality. For example, CLIQUE employs a grid-based scheme in which axis-parallel grids are constructed from equal-sized bins [3]. 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 [78].

4. *Hybrid approaches* attempt to in blend the two previously identified approaches of ‘subspace’ and ‘projection’ clustering, which represent two algorithmic extremes. 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)” [63].

The original binary and later more refined algorithmic categorizations are generally used together, so projected clustering can have instances which are top-down and others which are bottom-up.

2.2 Benchmarking Surveys

In addition to the survey papers of Parsons *et al.* [87] and Kriegel *et al.* [62, 63] – in which the emphasis was on establishing a framework for categorizing different algorithmic approaches to the subspace clustering task – there are other works in which benchmarking experiments are carried out on synthetic data sets and recommendations made regarding the observed empirical properties. These review papers make an effort to apply different subspace clustering algorithms to data sets with different properties to evaluate and compare the performance of different methods and identify their qualities and caveats.

Similar to Kriegel *et al.*, Patrikainen and Meila [88] divide subspace clustering algorithms into axis-aligned and non-axis-aligned methods. They categorize FASTDOC [91], PROCLUS [1] and HARP [121] as examples of axis-aligned and ORCLUS [2] as an example of a non-axis-aligned method. They define new measures as well – Relative Non-Intersecting Area (RNIA) and Clustering Error (CE) – based on relevant instances as well as attributes assigned to each subspace cluster to evaluate a subspace clustering algorithms.

Patrikainen and Meila do not run any experiments on synthetic data, rather they use the results of [121] and only calculate their proposed measures based on the results of that work. They conclude that of the four methods HARP produces the best qualitative results, whereas FASTDOC and PROCLUS have a strong parameter dependency.

Relative to the approach taken by the survey paper of Muller *et al.*, three categorizations were assumed: *cell-based*, *density-based* or *clustering-soriented* approaches [83]. 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[3]. Properties such as accuracy, cluster distribution, coverage, entropy, F-measure and runtime are used to compare 10 different algorithms on 23 data sets. In the benchmarking study reported in this thesis, extensive use will be made of the Open-Subspace repository.

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

Cell-based approaches such as CLIQUE [3], DOC [91], MINECLUS [122, 123] and SCHISM [100] find static- or dynamic-sized grids in each dimension, and merge them to make subspace clusters. This category is very similar to the bottom-up category of Parsons *et al.* [87]. CLIQUE and MINECLUS methods have already been described as bottom-up subspace clustering algorithms. DOC can be considered as an earlier version of MINECLUS. The FP-tree (Frequent Patterns)[4] structures help MINECLUS achieve better runtimes compared with DOC. SCHISM on the other hand tries to enhance CLIQUE by adapting the density threshold to the subspace dimensionality.

Cell-based methods only adjust the properties of single subspace clusters, which have little effect on the overall clustering solution. The first two cell-based approaches, CLIQUE and DOC, also tend to produce solutions which over-estimate the number of clusters and in some extreme cases provide solutions with more clusters than data instances.

Density-based methods such as SUBCLU [55], FIRES [61] and INSCY [6] start by finding dense instances instead of dense bins in each attribute. In other words they count the instances in the neighbourhood of a specific instance instead of counting the instances in a unit bin in each attribute. A cluster in a density-based approach is defined as a chain of dense instances and because of this structure these methods are capable of finding arbitrarily-shaped clusters. SUBCLU [55] is the first density-based approach and is an extension to the DBSCAN [33] clustering algorithm for subspaces. Density-based approaches require expensive computation to find dense instances; therefore they are usually computationally infeasible. FIRES uses 1-d histograms to jump directly to interesting subspaces instead of going bottom-up. INSCY is an extension to SUBCLU which processes subspaces and eliminates non-promising low-dimensional subspace clusters, achieving better runtimes.

Similar to cell-based approaches, density-based methods only have control over individual clusters and not the overall clustering solution. Parameter optimization is also a challenging task for density-based methods. Also, the cluster count of SUBCLU solutions sometimes exceeds the data set cardinality.

[4]The FPtree is a compact data structure which stores all patterns which appear in the database. For each transaction, the frequent items (in descending frequency order) are inserted as a path in this tree, where each node corresponds to an item and paths with common prefixes are compressed [123].

Clustering-based approaches focus on optimizing the overall clustering results rather than finding separate subspace clusters. These methods define the properties of the overall clustering approach rather than defining a single subspace cluster. Therefore, they have much more control over the final clustering result than any method in any other category. For example, they can define the total number of clusters in a data set as well as the average dimensionality of clusters (as in PROCLUS [1]). Other examples are P3C [79] and STATPC [78]. PROCLUS has already been discussed as a top-down subspace clustering method. P3C uses the χ^2 statistical test and expectation-maximization to find the optimal final clustering solution. STATPC defines the statistically significant density of a cluster to eliminate redundant clusters.

Below is a summary of the empirical conclusions from Muller *et al.*:

1. Quality: Basic methods from cell-based and density-based categories (CLIQUE and SUBCLU) have a low clustering quality with respect to clustering error [88]. They also tend to produce clustering solutions with too many clusters. MINECLUS, from the cell-based category, INSCY, from the density-based algorithms and PROCLUS, from the clustering-based category produce the best clustering quality with respect to clustering error, however INSCY fails to scale to data sets with more than 25 attributes.

2. Speed: The runtime of the basic methods (CLIQUE and SUBCLU) make them infeasible. CLIQUE, for example, returns runtimes of up to 114 hours on the *glass* data set, which is a 9-dimensional data set with 214 instances on a computer cluster with 4 quad-core Opteron 2.3 GHz CPUs. SUBCLU runtimes can get as long as 4 hours which is much better than CLIQUE, but still far behind more practical methods. MINECLUS and PROCLUS happen to be the fastest algorithms in their respective categories. INSCY, however, returns results no faster than twice the time of MINECLUS and PROCLUS for most of the data sets.

In general, Muller *et al.* concluded that recent cell-based approaches outperform other methods on data sets with low dimensionality. Density-based paradigms may suffer from long runtimes, and therefore might not scale as dimensionality increases. Clustering-based methods have the fastest runtime, but this is due to the fact that they are provided with *much more* a priori information than any other approach regarding the data set they are trying to cluster. They are easier to parameterize because the parameters are straightforward, however they have lower performance measures than cell-based methods.

Given the algorithm ranking from the Muller *et al.* survey, MINECLUS [122, 123] and PROCLUS [1] be adopted for 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 categories [62].

Moise *et al.* [80] made another extensive survey on subspace and projected clustering. They divide methods into two groups of *projected clustering* and *subspace clustering* algorithms. A projected cluster is defined as a pair (I, A) where I is the subset of instances and A is the subset of attributes so that the instances in I are close when projected onto the attributes in A, but are not close when projected onto the remaining attributes. Subspace cluster approaches, on the other hand, search for all clusters of points in all subspaces of a data set according to their respective cluster definitions. They reviewed 19 methods and compared 10 of them against 6 full-dimensional clustering algorithms [80]. They systematically generated 54 data sets in order to consider the effects of almost 10 different parameters in a data set, e.g. the impact of the distribution that cluster points are drawn from, to the extent of clusters in their relevant attributes. From a benchmarking perspective this resulted in the most systematic of studies conducted to date. These plus some additional data sets will be adopted to address further issues of relevance to the subspace clustering task (Section 5).

Projected clustering approaches are divided further into partitional, hierarchical and density-based subcategories. PROCLUS and SSPC fall into the partitional group, HARP is a hierarchical method; and PreDeCon, DOC, FASTDOC, MINECLUS, EPCH, FIRES and P3C are density-based approaches. A total of nine Subspace Clustering methods were reviewed in the survey by Moise *et al.*: CLIQUE, nCluster, ENCLUS, MAFIA, COSA, SUBCLU, SCHISM, DUSC and DiSH.

Moise *et al.* choose 10 (from the 19 methods reviewed) and compared them on real and synthetic data focusing on different characteristics of the data sets, such as data distribution (uniform vs. Gaussian distributions), data set dimensionality, data set cardinality, cluster count, clusters with equal vs. different numbers of attributes, average cluster dimensionality, average cluster size, the spread of distribution from which instances are drawn and the overlap between relevant attributes of clusters. They also add 6 full-dimensional clustering algorithms to compare the performance of subspace clustering methods with that of full-dimensional algorithms.

While evaluating the effects of average cluster dimensionality, Moise *et al.* noted that STATPC, SSPC and HARP produce the best results in detecting both instances and attributes relevant to each cluster. Almost all methods performed slightly better on data sets with cluster instances sampled from a uniform distribution rather than a Gaussian distribution. Also all methods performed better on data sets with an equal number of attributes per cluster, compared with data sets with a different number of attributes per cluster. In evaluating the effects of data set dimensionality, cardinality and cluster count, STATPC consistently returns the best results. The accuracy of PROCLUS and MINECLUS drops with increasing dimensionality and cluster count and rises with increasing size. Once clusters have overlapping relevant attributes, the accuracy decreases because the clusters tend to become more identical.

Moise *et al.* applied their methods to some real data sets and notice that differences between quality measures are less noticeable on real data sets. They attribute this to the fact that class labels are used instead of cluster labels, since there might not be a strong mapping and correspondence between classes and hidden clusters. Moreover, by the very nature of assuming a "real" data set, it becomes impossible to establish a ground truth in the way that this is possible to achieve in the case of an artificial data set. While evaluating different methods, they make some conclusions about the scalability of approaches to different aspects of data sets, as follows:

1. MINECLUS fails on data sets with higher than 200 dimensions; therefore it is not scalable to data set dimensionality.

2. HARP, DiSH and FIRES are not scalable to data set cardinality, and fail to return results for data sets with more than 1,000 instances. Also PreDeCon cannot produce results for data sets with more than 10,000 instances.

3. However, almost all approaches are scalable to the number of clusters embedded in a data set.

STATPC [78] 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. STATPC and the Moise *et al.* data sets will be used for the benchmarking evaluation performed in this work.

Moise *et al.* conclude their work by making three recommendations for future work. They suggest that future works on subspace clustering should be focused on:

1. Finding low-dimensional clusters, instead of finding clusters with a large number of supporting attributes;

2. Limiting parameters and avoiding parameters which require a knowledge of the data structure in the data set to be evaluated; and

3. Taking into account the statistical significance of clusters.

The approach of this thesis addresses the first two recommendations directly and will assume a post-training assessment based on the micro F-measure. Doing so will result in a more informative evaluation than reported in the past when the ground truth of the data set is known a priori, i.e. as in a data set created through artificial means.

2.3 Evolutionary Computation for Clustering

Most works to-date have assumed deterministic solutions to the subspace clustering task and little, if any, mention of evolutionary computation (EC) methods is found in any of the review papers dealing with subspace clustering [87, 88, 80, 83]. A beginning will be made by discussing some full-space evolutionary clustering algorithms pertinent to the development of the proposed S-ESC approach and then reviewing briefly some of the recent works on subspace clustering with evolutionary methods.

2.3.1 EC-based Full-space Clustering

Hruschka *et al.* published an extensive survey paper on evolutionary algorithms for clustering [47]. They provide a recent overview of clustering methods based on different families of evolutionary computation such as genetic algorithms and particle swarm optimization. In addition, they cover recent developments such as multi-objective and ensemble-based evolutionary data clustering.

They divide EC-based data clustering methods into three separate groups: hard partitional clustering, soft-partitional (overlapping) clustering and methods using ensemble and multi-objective approaches. Under hard partitional clustering they further divide approaches into methods with a fixed cluster count (k) a-priori and those with a variable cluster count. Also under soft partitional clustering they mention fuzzy clustering approaches.

Different EC-based clustering methods using centroid-based, medoid-based, label-based, tree-based and graph-based representations as well as binary vs. integer vs. real encodings are exemplified in this survey. Three different types of variation operators are identified and different methods utilizing each type of operator are introduced. Some forms of fitness function as well as small sections on multi-objective and ensemble-based evolutionary data clustering are included in this review. What follows is a brief summary of each important component of an EC-based algorithm in a data clustering context.

Representation/Encoding

Most evolutionary algorithms encode the final clustering/partitioning solution in a single individual[5]. This representation can be encoded using binary, integer and real data types. The choice of representation and encoding in most evolutionary clustering algorithms is coupled either with data set dimensionality or cardinality. This means that each chromosome is a binary/integer/real string in which the encoding is directly proportional to the total number of attributes or (even worse) defines how each data instance is assigned to a cluster [15, 43, 66].

In a binary representation a binary string of size equal to the data set size encodes the clustering, where each binary position corresponds to a data set instance. If the value of a gene is 1, then the corresponding instance is marked as a prototype for a cluster within the overall clustering. Methods in which instances from the data set

[5]'Individual' and 'Chromosome' are interchangeable terms in an evolutionary computation context.

appear as cluster prototypes are also called medoid-based representations as in [66]. However, this representation, assumes a common ordering of the instances.

Integer encoding is typically implemented in two ways. In the first case a chromosome has as many genes as instances in the data set, and each gene stores the cluster label of the corresponding instance. However, this encoding is redundant since, for example, the two clusterings represented by [1 1 2 2 2] and [2 2 1 1 1] for a data set with 5 instances are the same. The second case assumes a medoid-based representation in which each chromosome has only k elements. In this representation each element stores the index of an instance which is the prototype of a cluster. For example, the chromosome [4 7 17] decodes into a clustering in which the 4th, 7th and 17th instances are each a prototypes/medoids for a cluster. This representation also assumes the instances in the data set are ordered as in [101].

Real valued encoding methods represent each clustering by the coordinates of all cluster prototypes (usually centroids instead of medoids) and are referred to as centroids-based representations. For example, for a 3D data set with $k = 2$, the chromosome [2.5, 1.6, 6.3, 7.1, 5.4, 3.9] decodes into two 3-d cluster centroids of location [2.5, 1.6, 6.3] and [7.1, 5.4, 3.9]. This representation is independent of data set cardinality and instances need not be ordered; however, it is dimensionality dependent. Also note that the real valued encoding is not limited to encoding cluster centroids. They can encode cluster medoids as well by encoding the medoid's coordinate values [76].

The grid-based or rule-based representation uses grids/rules to encode a hyper-rectangular cluster. Each gene (rule) decodes into the lower and upper bounds to define the boundaries of a cluster in one dimension [97]. There are several benefits associated with this method: 1. ability to discover homogeneous clusters of arbitrary density, geometry and data coverage; 2. the ability to discover clusters embedded in arbitrary subspaces of high dimensional data; 3. the potential support for outlier detection, since outliers tend to be located in low-density regions away from the dense regions which are part of the clusters; and 4. it is straightforward to apply it to categorical data.

Distribution-based representation is another method of encoding a clustering task. In this method each gene of a chromosome represents a pre-determined distribution

– e.g. a normal distribution – by its parameters, mean and variance. This representation assumes clusters in a hyper-elliptical shape [27].

Graph-based encoding is used in Handl and Knowles [43]. In this encoding each individual is represented by an integer string of data set size, with each gene corresponding to an instance. A value j in position i shows a connection or link between instances i and j. The caveat of this method is that instances are assumed in order, and this order should be maintained throughout the application of the algorithm.

Most representations for clustering algorithms with a known k are applicable to clustering algorithms without a given k with no or slight changes. For example, a binary representation can be used to encode a partitioning with or without an optimal k value. The first type of integer string can be used to represent a partitioning with unknown k with no change as in [48]. However, in the case of a medoid-based integer representation, the string length should be made dynamic to make room for the different number of clusters in a partitioning. Variable length chromosomes are also necessary under real-valued partitioning [76] or grid- or distribution-based representations to accommodate clustering algorithms with unknown k.

Fitness Function

There is a variety of clustering validity criteria used for evaluating partitions, some of which can be used as a fitness function in an evolutionary algorithm. Since credit assignment is not generally based on gradient information, a lot of flexibility exists for adopting metrics which could be of an application-specific nature. Conversely, most evolutionary clustering algorithms utilize some form of distance as their fitness function which has the caveat of imposing a specific structure on the data set, such as a bias towards spherical clusters when using a Euclidean distance.

Obvious distance-based choices as fitness functions for an EC-based clustering algorithm are those metrics based on intra- versus inter-cluster variation. Intra-cluster (a.k.a. within-cluster) distance sums the distances between each data set instance and the closest cluster prototype (i.e. cluster centroid or medoid). Inter-cluster (a.k.a between-cluster) distance measures the distance between different clusters. This could be in the form of measuring the distance between two cluster prototypes or the distance between a cluster prototype and the closest instance from other clusters. While minimizing intra-cluster variance yields dense spherical clusters, maximizing inter-cluster variance produces clusters which are well separated.

Within-cluster distance applied to a medoid-based encoding is used in [73], i.e. distances between a cluster medoid and instances of the corresponding cluster are summed over k clusters. Other works use the same functions to evolve clusters [34], [101]. The authors of [70] re-formulate the hierarchical clustering as an optimization problem which tries to find the closest ultrametric distance [44] for a given dissimilarity with Euclidean norm. This work proposed an order-based GA to solve the problem. The work in [76] suggests minimizing the sum of the squared Euclidean distances of the instances from their respective cluster means, and [64] minimizes a modified version of the within-cluster distance using cluster centroids. This is basically the centroid-based version of the medoid-based method mentioned earlier. Other works use the same distance-based fitness function – sometimes called distortion function – to evolve clusters [64], [72].

A popular fitness function used in different evolutionary-based clustering algorithms [66], [15] is:

$$J_m = \sum_{i=1}^{k} \sum_{j=1}^{N} \mu_{ij}^{m} ||x_j - v_i||^2 \tag{2.1}$$

where x_j is the jth instance of the data set, v_i is the prototype for ith cluster C_i, μ_{ij} denotes the membership of object x_j to cluster C_i and m is a user-defined parameter ($m < 1$). J_m is an extension to the within-cluster distance which can be used for both hard clustering ($\mu_{ij} \in \{0, 1\}$) and soft clustering ($\mu_{ij} \in [0, 1]$).

Two obvious drawbacks to within-cluster distances are: 1. bias toward finding spherical-shaped clusters; and 2. a bias toward higher number of clusters, since in an extreme case of $k = N$ the within-cluster distance will be optimally minimized (i.e. zero).

The clustering algorithm introduced in [110] takes advantage of a single-linkage algorithm in order to divide data into small subsets, prior to GA-based clustering. This is done in an effort to reduce the computational complexity. The fitness function was designed to adjust the different effects of the within-class and between-class distances. The criterion function in [102] was constructed as a weighted sum of six cluster validity functions which is optimized further by the hybrid niching genetic algorithm (HNGA). The niching method is developed in order to avoid premature convergence. In another work the Variance Ratio Criterion (VRC) employs both intra- and inter- cluster distances [19]:

$$VRC = \frac{trace\ B/(k-1)}{trace\ W/(N-k)} \tag{2.2}$$

where B and W are the between-cluster and within-cluster sums of square (covariance) matrices, N is the data set cardinality and k is the number of clusters in an individual.

Recent fitness functions focus on finding clusters in high-density regions of data space which are separated by low-density regions. Methods using this group of fitness functions fall under the category of density-based clustering algorithms and can be more flexible in finding clusters with arbitrary shapes. The method proposed in [27] – technically an Estimation of Distribution Algorithm (EDA) – uses a density-based fitness function. The density is defined as the number of instances in a cluster normalized by the volume of the cluster and the fitness function is the total sum of cluster density values normalized by the number of clusters within an individual.

Variation Operators

Variation operators act on individuals to introduce diversity through reproducing offspring. They make guided or stochastic changes to individuals in order to exploit and explore the search space for better solutions. Variation operators usually fall into the two categories of crossover and mutation. A crossover operator takes advantage of the embedded genetic information inherited from two parents and therefore might be considered to be more exploitive in the credit assignment mechanism. Mutation, on the other hand, introduces stochastic changes relative to a child;[6] potentially exploring an area of the search space not previously present in the population (i.e. explorative).

Hruschka *et al.* categorize variation operators under cluster-oriented vs. non-oriented, guided vs. unguided and context-sensitive vs. context-insensitive operators [47]. A cluster-oriented operator deals with a cluster in a chromosome by copying, merging or splitting clusters, whereas a non-oriented operator is a conventional evolutionary operator which makes changes in chromosomes regardless of their phenotypic

[6]Resulting from sexual recombination (crossover) or the asexual cloning of a parent.

decoding. Guided operators, as opposed to unguided operators, utilize some form of information about the performance of a cluster or the quality of a clustering solution. Context-sensitivity is only defined for crossover operators. A context-sensitive crossover operator is a cluster-oriented operator which does not create a new clustering once it is given two (possibly different) chromosomes encoding the same clustering solution.

In methods where k is fixed a priori, all variation operators should be context-sensitive. However, most traditional variation operators manipulate gene values without considering their connection with other genes. As a consequence, a repair mechanism is necessary to correct (if possible) or remove the offspring with a cluster count different from k. Such repair mechanisms are expensive, e.g. requiring further fitness evaluation; thus, it is more efficient to design cluster-oriented diversity operators specifically for clustering from the outset. Various context-sensitive diversity operators for clustering algorithms have been suggested in the literature. For example, [101] and [64] introduce cluster-oriented crossover and mutation operators for integer representation and [76] introduces similar operators for real encoding.

Naturally, variation operators need to reflect the constraints of a fixed versus variable length representation. Fixed length individuals require the provision of the necessary routines to make sure that the offspring bred as a result of a variation operator has the same number of clusters as its parents, whereas these constraints are relaxed in variable length individuals. In other words, two individuals with the same number of clusters need not necessarily breed an offspring with same number of clusters and mutation operators can be designed to add clusters to or remove them from an individual.

Selection and Initialization

Selection: Different selection methods have been utilized for evolutionary methods applied to the clustering task. However elitist variants of selection [57, 19] appear to be more popular than proportional variants [76, 110].

Initialization: The random assignments of instances to clusters [64], and the random selection of a subset of instances as k medoids [76] are among the more popular methods for initializing the population for an evolutionary clustering algorithm.

Multi-Objective Approaches

Since there is no standard definition for clustering and there is no ground truth (i.e. cluster labels)[7] to indicate how well a clustering algorithm is performing, researchers have started using multiple criteria to evaluate a clustering. This helps to remove the biases imposed by any single criterion while evaluating a clustering.

As opposed to a single-objective method which returns one best solution, using a Pareto-based multi-objective evolutionary algorithm usually provides a set of optimal solutions [106, 46, 28, 25]. Each solution on the Pareto front can be considered the best solution to a single objective formed by a scalar weighting of multiple objectives. This includes solutions on the tails of a Pareto front which tend to optimize only a single objective among multiple objectives. There are also solutions on the center of Pareto front which try to optimize multiple objectives simultaneously. These are the solutions which are of current interest. Therefore, a mechanism is required for selecting the best solutions among a Pareto set once the training process is finished.

Multi-objective evolutionary clustering algorithms also tend to assume distance metrics; albeit metrics which are in some way measuring different properties. Thus, improving one (metric) does not necessarily lead to the improvement of the other. In one of the more highly cited multi-objective evolutionary clustering works, Handl and Knowles [43] use compactness (distortion) and connectedness (connectivity) as the objectives of their multi-objective evolutionary clustering algorithm. Compactness measures the intra-cluster variation over all clusters, i.e. the summation of the distance between each cluster centroid and other instances within that cluster. Connectivity, on the other hand, is punished if two adjacent instances are grouped in different clusters, i.e. connectivity measures how connected the clusters are. At one extreme compactness tries to create as many singleton clusters as there are instances in the data set, while on the other hand, connectivity tries to group all data set instances in one connected super-cluster. The interaction of the two objectives not only balances the cluster count, but also avoids the trivial solutions and searches for

[7]Most clustering algorithms get evaluated on data sets with class labels which are in the most part designed for evaluating classification methods. However, this has the disadvantage of preferring those algorithms which find clusters matching data set classes, potentially leading to misleading conclusions. This is also one of the reasons researchers design their own synthetic data sets with known clusters in them.

interesting regions in the search space.

The multi-objective evolutionary clustering algorithm described in [60] uses the intra-cluster variation and cluster count as its objectives. This results in a pool of solutions with the smallest intra-cluster variation given different cluster counts. In the multi-objective evolutionary fuzzy clustering algorithm presented in [10] both objectives are in some form or another distance-related. One objective takes the form of the Xie-Beni (XB) index (the intra-cluster variation with different membership values for each pair of objects and clusters) [117] and the second objective is the J_{FCM} measure (the ratio between intra-cluster variation and the distance between the closest clusters) explained by [14].

A similar work [82] uses intra-cluster variation and the total summation of distances between all pairs of clusters. In [94], as in other multi-objective clustering methods, the primary objective is intra-cluster variation. However, here the intra-cluster variation is normalized by the number of clusters in a solution. The second objective here is the average separation between any pair of clusters.

The authors of [56] are among the first researchers utilizing multi-objective evolutionary approaches with data clustering and attribute reduction in mind. In this work a set of attributes is selected for a k-means clustering algorithm, rather than directly clustering the objects. They use four objectives for their multi-objective algorithm. While cluster cohesiveness (i.e. some form of intra-cluster distance) and separation between clusters (i.e. some form of inter-cluster distance) appear, cluster count and attribute support count are used as the remaining objectives.

Once a Pareto set of solutions has been evolved, post-training selection of a single 'champion' solution is necessary. Thus, either domain knowledge (some form of visualization) or some form of automatic selection mechanism is necessary. There is a whole category of multi-objective evolutionary literature dedicated to this area. One of the more prominent and widely used techniques is to detect those individuals for which a small change in one objective causes (comparatively) larger changes in other objectives. These individuals are called 'knees' with methods designed for either post-training identification of the knee [17, 99] or schemes for focusing the algorithm during evolution for producing more individuals on knee regions [93, 11].

Ensemble Methods

Instead of simultaneously optimizing multiple objectives, ensemble methods apply some form of diversity maintenance in combination with a consensus objective (cf., a clustering validity criterion). Diversity mechanisms might take the form of employing different clustering algorithms [109], performing multiple runs with the same clustering algorithm or using the same algorithm with different partitions of the training data [43].

The consensus function is usually based one of the following four options [125], [124], [35]: co-association, graph, mutual information and voting. Methods based on co-association focus on keeping together instances which are grouped together in different clusterings. Techniques based on mutual information try to maximize the mutual information between the labels of the initial clusterings and the labels of the final consensus clustering. Graph-based approaches use well-known graph partitioning methods to find the optimal consensus clustering. Voting methods, being the simplest of all, assign each instance to a cluster based on the number of times the instance was assigned to different clusters among the initial clusterings.

Yoon *et al.* start off by evolving a population of partitions. They select pairs of partitions which share a large overlap among cluster objects and apply a crossover operator on them, reproducing new partitioning solutions [124]. The authors claim that their ensemble method takes characteristics from the individual algorithms and the data set into account during the ensemble procedure. Faceli *et al.* perform the same task with minor changes [35]. The initial partitions come from different qualities and the clustering method behind each of them is also different. The initial partitioning population also has a large diversity with respect to cluster types. Their goal is to evolve a set of partitions which represent different views of the data set.

Handl and Knowles [43] argue that although ensemble methods can provide more robust methods compared to results produced by non-ensemble methods; however, they will be outperformed by multi-objective methods. The reason, they believe, is the fact that the data partitions are found by single-objective methods and therefore the final consensus of the ensemble method will be outperformed by the results of multi-objective methods, which are more capable of directing the search process.

2.3.2 EC-based Subspace Clustering

A few algorithms have been identified with regard to methods assuming a framework from evolutionary computation.

Sarafis *et al.* use a non-binary, rule-based representation. Their representation contains a variable number of non-overlapping rules, in which each rule consists of a fixed number of d intervals, one for each attribute [96]. This results in a gene length proportional to the overall data dimensionality. The left boundary (lb) and right boundary (rb) of the intervals are drawn from discrete domains. They automatically quantize the attribute space into a multidimensional grid. In order to deal effectively with subspace clustering the aforementioned encoding scheme is further enhanced by introducing an additional field for each feature in each rule called status (st) where st_{ij} is a flag indicating whether the ith feature of the jth rule is relevant (active or '1') or irrelevant (inactive or '0') to the clustering of patterns which are covered by the jth rule. The fitness function in this work rewards maximizing the data covered by the rules of an individual or in other words, maximizing the coverage. From the perspective of the categorization of Kriegel *et al.* this would make the approach an example of top-down projection clustering (i.e. gene length is proportional to data dimensionality).

Sarafis *et al.* also employ different operators such as seed discovery, generalization, mutation, recombination and repair operators. Seed discovery finds clusters existing in 2-d projections of the original attribute space to initialize the population instead of using random initialization. The generalization operator replaces two adjacent rules with a single and more generic rule, and helps in detecting inactive attributes. There are two mutation operators; growing-mutation for increasing the size of existing rules in an attempt to cover as many instances as possible, but with a minimum number of rules, and seed-mutation which is complementary to the previous mutation operator in cases where the latter fails to fine-tune due to strict requirements. The recombination operator in this work, called imported rule crossover (IRC), works by importing rules into offspring rather than exchanging rules between the parents. Therefore, mutation operators act at the attribute level, whereas a recombination operator is applied at the rule level. The design of a repair operator is motivated by the observation that clusters become separated because of the different extent of condensation of instances between them. In other words, a repair operator finds the borders between clusters to achieve precision in capturing their shapes.

Synthetic data sets are used in the evaluation of this work. Data sets are produced using a modified version of the generator described in the CLIQUE paper [1]. The domain of each dimension is set to [0, 100]. Each data set consists of 10 clusters and the non-outlier points were distributed among them as follows: 20%, 15%, 15%, 10%, 10%, 10%, 8%, 6%, 4% and 2%. Outliers were scattered uniformly throughout the entire feature space while the level of noise was set to 10% of the total number of points. For all the non-bounded dimensions of a cluster the points are uniformly distributed over the entire domain. On the other hand, in the bounded dimensions the values of the points are drawn either from a normal or from a uniform distribution at random. In both cases the points are spread in the bounded dimensions within a small range (i.e 10) in a way that no overlapping between clusters occurs. Note that all clusters are embedded in the same number of dimensions. Therefore, the hyper-volume of clusters is roughly the same and because the points are not equi-distributed among clusters, the density between them varies considerably.

The authors apply their algorithm on 4 synthetically-generated data sets for each experiment and use the results to show that their algorithm runtime grows linearly with data set dimensionality, cardinality and cluster attribute support. With respect to accuracy they report that except for a few experiments where cluster attribute support is high, their algorithm is able to detect all clusters; however, they only report the fitness of individuals (coverage) as a measure of performance for their algorithm and claim a 60 to 90% coverage for 50-dimensional data sets with 100,000 instances, 10 clusters and 5 to 20 dimensional clusters. One of the drawbacks of the work by Sarafis *et al.* is that apart from ordinary evolutionary parameters,[8] there are five other parameters involved in this work which need to be fine-tuned prior to experiments – which is not a trivial task.

In a work by Assent *et al.* an evolutionary subspace search approach is used to detect locally relevant attributes for clustering [7]. In this work individuals are described by their genes. A gene models a specific attribute. Thus a set of genes encodes whether subspaces are contained. The population contains all subspaces of

[8]Population size, max number of generations, probability of crossover and mutation operators.

the current generation. The fitness function uses normalized entropy of subspaces to measure the clustering tendency. To ensure that algorithm escapes local optima, a multi-objective approach is used based on biologically inspired evolutionary niches. These niches allow for evolutionary generation of different local subspace optima in one population. Although the method sounds appealing the algorithm is only tested on very few number of data sets of up to 55 dimensions and therefore the consistency and scalability of the work is in doubt.

Piatrik *et al.* propose a novel approach for the subspace clustering of images based on the learning process of ant colony optimization [90]. The optimization function in this work (SC-ACO) is the summation of the weighted (similarity) distance between image feature values and cluster centroids. In other words, the problem of finding the best clustering is converted into the problem of finding the best attribute-weighting vector. As such, the algorithm does not remove attributes explicitly. With respect to the categorization of Kriegel *et al.* this makes the algorithm a soft projected clustering method.

It is assumed that the required number of clusters is known to the SC-ACO algorithm. Each ant represents a cluster centroid and the ants are initialized far from one another at the beginning of the process. Each ant assigns each image to its corresponding cluster with a probability based on a pheromone level, which carries criterion information between ants. Attribute weights are updated based on the average distance from the centroids and images are assigned to each centroid. Then new centroids are computed according to the clustering of images, and the pheromone levels are updated after all ants have done their clustering.

It is important to note that this method does not cluster images based solely on the similarity distance between centroids and images. The pheromone level carries the criterion information from each ant to the others and also plays an important role in the partitioning of images. The authors test their method and compare it against other methods on 3 image data sets with 600, 6000 and 500 images respectively. The dimensionality of these data sets is not mentioned, and SC-ACO is given the number of clusters in each data set. The average error rate is the only reported performance measure, which is between 0.30, 0.38 and 0.40 for the three data sets respectively. No computational complexity or runtime information is available.

Sangeetha *et al.* [95] assume a grid-based approach to discover subspace clusters where the data set is partitioned into a number of non-overlapping units by dividing each attribute into d equal-length intervals. A k-dimensional unit is defined as the intersection of one interval from each of the k attributes, and is considered dense if it contains more than a certain number of points. If two dense units have a common face or if there is a third dense region which is connected to them, they are considered connected as well.

Lu *et al.* propose a particle swarm approach to solve the variable weighting problem in subspace clustering of high-dimensional data [71]. In their algorithm, PSOVW, the authors are interested in soft subspace clustering which finds an attribute weight vector for each cluster, as opposed to hard subspace clustering which aims at finding the exact attribute subspace for each cluster. They employ a special weighting k-means function – with k known a priori – which calculates the sum of the within cluster distances for each cluster along its own subspace, i.e. the similarity between each pair of objects is based on weighted attributes. They also utilize a simple non-normalized weight representation method and transform the constrained search space into a redundant closed space, which largely facilitates the search process. The attribute weights transform distance so that the associated cluster is reshaped into a dense hyper-sphere and can be separated from other clusters.

Finally, instead of employing local search strategies, they make full use of the particle swarm optimizer to minimize the given objective function which is a k-means weighting function and calculates the sum of the within-cluster distance for each cluster, preferably along relevant dimensions to irrelevant dimensions. The method therefore corresponds to the (top-down) soft projected clustering categorization of Kriegel *et al.* and does not explicitly result in cluster definitions with dimensions below that of the original task domain.

PSOVW utilizes three swarms: 1. the position swarm of attribute weights which are set to random numbers distributed uniformly within a certain range; 2. the velocity swarm of attribute weights, which are set to random numbers uniformly distributed in the range $[-maxv, maxv]$; and 3. the swarm of cluster centroids which are k different data objects chosen randomly from all the data objects. In all three swarms, an individual is a $k \times D$ matrix where k is the number of clusters and D is data set dimensionality; therefore, representation is subject to data set 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. PSOVW recovers clusters correctly over the data sets with 4 well-separated clusters on each trial; however, it occasionally missed one entire cluster on data sets with more complicated 10 clusters. It sometimes fails to differentiate two very close groups, whose relevant dimensions overlap, and merges them into one cluster. One reason for the high accuracy of these results is that the synthetic data sets are not noisy, and noise-free data sets are appropriate for the purpose of this experiment. Moreover, PSOVW is explicitly given the number of clusters in all experiments, which reduces the search space.

Zhu *et al.* proposed a multi-objective evolutionary algorithm based on soft subspace clustering (MOSSC) to optimize simultaneously the weighting of within-cluster compactness versus the weighting of between-cluster separation [127]. NSGA-II [29] is used to evolve a set of near-Pareto-optimal solutions. As with 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. The soft subspace clustering version of Xie-Beni [117] and a subspace-modified version of J_{FCM} [14] are utilized as objectives in this work. Under the Kriegel *et al.* categorization this would also be a (top-down) soft projected clustering algorithm.

Once the non-dominated cluster weighting values and corresponding cluster membership values are retuned by NSGA-II, MOSSC uses cluster ensembles to find the super-solution. They adopt the Hybrid Bipartite Graph Formulation (HBGF) algorithm [37] to produce the super-solution from the near-Pareto-optimal non-dominated set of solutions. HBGF treats both instances and clusters as vertices of a bipartite graph. The cluster ensemble problem is then reduced to a graph partitioning technique. The final cluster labels of MOSSC are determined by integrating all non-dominated solutions. Benchmarking utilized the synthetic data sets from Lu *et al.*, with modification to enable the consideration of up to 50 dimensions and 850 data instances.

In a method developed by Boudjeloud *et al.* individual clusters are iteratively found one by one in form of a subset of instances of the data set and a subset of its relevant dimensions [16]. At each iteration once a cluster is found the set of relevant attributes are removed and the algorithm is re-run in search for a new cluster. The search continues as long as there are attributes left, which appears to do extra iterations once the hidden clusters are found and only irrelevant attributes remain and it might find degenerate clusters. The number of relevant attributes in each cluster is fixed and multiple clusters cannot share the same attribute. The method utilizes a steady state evolutionary algorithm.

An individual is composed of two parts; attribute support of a cluster and a cluster medoid in form of a simple index to the specified instance. Once a cluster is formed (using a subset of attributes and a cluster medoid) all data set instances are sorted based on their distance from the cluster medoid with the first instance being the closest distance. Closest instances are incrementally added to the cluster until an abrupt increase in the distance is observed. Authors propose two methods to evaluate clusters individually rather than the final partitioning.

The method is only evaluated against two real data set; ionosphere with 34 dimensions and 351 instances, and iris with 4 attributes and 150 instances. It is also evaluated against synthetically generated data set with 9 attributes and 400 instances, although no performance measure is given in this case. Therefore the consistency and scalability of the work is in doubt.

Nourashrafeddin *et al.* develop an evolutionary method called 'Esubclus' mainly for the purpose of finding subspace clusters in text document clustering (i.e. finding relevant keywords to each document category) [85]. Esubclus proceeds in two phases. Given the number of clusters, k, authors use fuzzy C-means clustering along a multi-objective genetic algorithm (NSGA-II) to provide (low-dimensional) groups of representative / relevant attributes for each cluster. Once relevant attributes are determined the objects are clustered in the space spanned by the centroids of those groups.

Each individual in the evolutionary algorithm encodes a limited, small number of attributes. Corresponding groups are formed by averaging the attributes encoded in the individuals with their nearest neighbours, where similarity of attributes is defined in terms of cosine similarity of their corresponding columns in the data matrix. To determine the fitness of an individual, the entries of the column vectors of the data matrix within the attribute groups are averaged, resulting in an m-dimensional vector for each group of attributes. X-means with the maximum number of clusters set to two is applied to each one of those vectors individually, potentially identifying two one-dimensional clusters one of which consists of indices corresponding to data objects partially identified by the attribute group. The fitness function favours tightly clustered groups of attributes and objects with large degrees of separation from other groups. The evolutionary algorithm is terminated if no improvement has been made in a number of generations.

Objects, which are characterized by combinations of relevant attribute groups, are assigned to different clusters in the second phase. Due to the small number of attribute groups, this goal can be simply accomplished by interactive thresholding in one pass.

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 providing a Pareto front with each individual on the pareto front being a candidate for the current cluster. A second single-objective GA then searches the space of different combinations of clusters from k Pareto fronts for the best k-cluster solution with one cluster from each Pareto front.

Other methods such as [107, 108] are also examples of EC-based methods which are related to subspace clustering but with minor contributions.

An earlier version of this proposed evolutionary subspace clustering (Bottom-up Evolutionary Subspace Clustering) attempted first to build cluster centroids and then describe clustering solutions through two independent cycles of evolution [111]. 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. It was also apparent that using a multi-objective evolutionary method with the MOCK [43] objectives of compactness (to detect spherical clusters) and connectivity (promising for arbitrary cluster shape detection) aided in the identification of the true cluster count [111]. However, the connectivity objective is much more expensive to evaluate than compactness. The proposed S-ESC algorithm (S-ESC was initially suggested in [112]) is needed to address both of these limitations.

Chapter 3

Comparator Clustering Algorithms

To compare S-ESC against state-of-the-art algorithms under common benchmarking conditions comparator algorithms are chosen from popular full-dimensional and subspace clustering algorithms. This Section describes the methods with which S-ESC is being compared, while the benchmarking data (data generation and their characteristics) as well as the evaluation methodology will be discussed later in Chapter 4.

Moise *et al.* introduce STATPC [78] as the dominant subspace clustering method following comparison with a large number of other subspace clustering algorithms [80], whereas Muller *et al.* recommend MINECLUS [122, 123] and PROCLUS [1] as the most consistent methods among other subspace clustering algorithms [83]. These three methods, then, are the best choices for comparison with this work's S-ESC approach. Although all three methods have been mentioned previously in Chapter 2, here an in-depth look into each one will be undertaken.

The EM algorithm is assumed to be representative of full-dimensional clustering algorithms [30, 77]. The properties which make it particularly appealing include the wide utility of the algorithm, its robustness and the relatively low number of parameters, e.g. a user need not provide a priori definitions for the number of clusters required.

3.1 MINECLUS

MINECLUS proposed by Yiu *et al.* [122, 123] – which is an improvement to the popular DOC [91] method – is a bottom-up density-based subspace clustering algorithm which assumes a hyper-cubic shape for subspace clusters. DOC defines a projected cluster as a set of points C and a set of dimensions relevant to those points D. Three parameters are also essential to the definition of clusters in DOC; w, α and β. The first parameter, w, controls the range or spread of clusters in the sense that the distance between instances of a cluster in each relevant attribute is w. The second parameter, $\alpha \in (0, 1]$, controls the density of clusters, i.e. each cluster should have at least $\alpha \times N$ points, where N is the total number of instances in the data set. The last parameter, $\beta \in (0, 1]$, reflects the importance of the size of the subspace over the size of the cluster.

DOC computes subspace clusters one by one. At each step it selects a random instance – hereafter called pivot point p – from the data set and tries to find the cluster for which p is the medoid. To this end a subset X of points are selected. Relevant attributes are then determined based on the distance between the projection of the pivot point and the projection of X points on the attribute being evaluated. If all X projections are within the distance w of the projection of the pivot point, the attribute is relevant. The relevant attributes set is called D. Once relevant dimensions to p are determined, a bounding box of width $2w$ around p and in the relevant dimensions form cluster C and all points within this box belong to cluster C. This process is repeated for $\frac{2}{\alpha}$ pivot points, and for each pivot point m potential subspace clusters are tried. In DOC the cluster quality measure is defined as $\mu(a, b) = a \cdot (1/\beta)^b$, where a and b are the number of points and the dimensionality of C respectively. The cluster with the highest quality is selected, the points in it are removed from the sample and the process is applied iteratively to the rest of the points. DOC has the advantage of automatically finding k; however, it produces approximate results and it is very computationally expensive.

MINECLUS has four phases. In the first phase, the iterative phase, the frequent itemset mining process is applied to generate clusters one by one iteratively. The second phase is the pruning phase which removes clusters with a quality measure (μ value) smaller than the others. A sorting process divides clusters into two categories of strong and weak clusters based on their μ values and the weak clusters are removed. The merging phase, the third phase, is applied only when the user prefers at most k clusters. If this is the case and if there are more strong clusters than k, the strong clusters are merged to create only k clusters. A cluster formed by merging two clusters contains all the points from both clusters but only utilizes those dimensions which are present in both clusters, i.e. the intersection of their attribute sets. The refinement phase, the last phase, takes advantage of the same refinement process used in PROCLUS to go over all the unassigned instances of the data set and assign them to those clusters already established.

Although MINECLUS is built on DOC to improve its sensitivity to higher dimensions; however, it is not tested on data sets with a dimensionality higher than 80. MINECLUS also suffers from parameter sensitivity in that parameter optimization needs to be performed on each data set. In addition, the assumption that a projected cluster is a hyper-cube may not be appropriate in real applications.

3.2 PROCLUS

PROCLUS (PROjected CLUStering), proposed by Aggrawal *et al.* [1], is a top-down partitional subspace clustering algorithm. Partitional clustering algorithms tend to focus on the final clustering results rather than on a single cluster independently. PROCLUS defines a 'projected cluster' as a subset C of data points together with a subset D of dimensions such that the points in C are closely clustered in the subspace of dimensions D. Aggarwal *et al.* define the projected clustering to be a two-fold task: locating the cluster centroid and identifying the dimension subset which supports each cluster. Rather than using cluster centroids, they take advantage of medoids[1]. They use the local search approach of the CLARANS algorithm [84] to generate a set of medoids. The cluster count, k, and the average dimensionality of the clusters, l, are provided to PROCLUS a priori.

The algorithm has three major phases; an initialization phase, an iterative phase and a cluster refinement phase. At a higher level the general idea of PROCLUS is to improve a set of medoids iteratively in a manner similar to CLARANS with modifications for dealing with subspaces. The initialization phase reduces the set of points going through the iterative phase while trying to maintain at least one point from each cluster. The iterative phase is a hill climbing process which tries to find a good set of medoids while identifying the subset of dimensions under which the cluster points are more easily separated from those of other clusters. Finally, the third phase improves the quality of clustering by refining clusters and their points with one pass over the data set.

The initialization phase of PROCLUS looks for a set of k points – each point from

[1]For a description about cluster centroids and medoids refer to Section2.1.

a different cluster. This set is called a 'piercing' set and although it is very unlikely to find this set in the first try, the initialization phase tries to find the smallest superset of the piercing set to feed into the next iterative phase. PROCLUS starts by randomly selecting $A * k$ well separated points. This guaranties that at least one point is drawn from each cluster. Then a greedy approach is applied to prune the size of this set further to include only the more promising $B * k$ points, where B is a small constant. This final set is called $M = \{m_1, m_2, ..., m_k\}$ where m_i is the medoid for cluster i.

Finding medoids. The iterative phase starts by selecting k medoids randomly from M. It finds the bad medoids iteratively and replaces them with new medoids from M. If the quality of clusters improves by introducing the new medoids, the newly tested medoids are kept, otherwise they will be discarded and new medoids will be tested again. This process repeats until there is no change in the clustering for a number of iterations. The current set at this stage will be returned as the final cluster medoids.

Finding dimensions. To find the most relevant dimensions for cluster i, all points in the vicinity of medoid m_i with a distance smaller than the distance between m_i and its closest medoid are considered and called L_i. Then the average distance between m_i and those points with a distance smaller than L_i along each dimension j is computed, $X_{i,j}$. The dimensions along which the average distance $X_{i,j}$ is the smallest are selected as relevant dimensions for cluster i given that the total number of relevant attributes for all clusters is constrained to $k*l$, where l is the input parameter specifying the average dimensionality of the subspace clusters. This produces a dimension set D_i for cluster i.

Forming clusters. Once the medoids and relevant dimensions are found, the points are assigned to medoids in one pass over the data set. Points are simply assigned to their nearest medoid; however, the distance is calculated only over the dimensions relevant to cluster i. In other words, points assigned to medoid m_i are those which fall within a hyper-sphere around m_i with a radius equal to the distance between m_i and the closest medoid.

A refinement phase at the end of the algorithm tries to tune the instances and attributes assigned to each medoid. The refinement is started by re-discovering the relevant dimensions of each medoid m_i. However, this time, instead of using the locality points for each medoid, points assigned to the cluster are considered. Once the new dimensions are discovered, points get re-assigned to clusters based on the new set of dimensions.

PROCLUS suffers from the fact that it needs a lot of prior knowledge about the data set such as the true/desired number of the subspace clusters as well as the average dimensionality of subspace clusters. Also, the A and B choices could affect the robustness of the algorithm. To ensure robustness the algorithm requires a sufficient number of points in the locality of each medoid in order to determine the relevant dimensions for each cluster. In addition, PROCLUS tends to form hyper-spherical subspace cluster which have approximately the same number of attributes per cluster, which could potentially limit the generality of the algorithm.

3.3 STATPC

STATPC proposed by Moise *et al.* [78] in 2008 is motivated by two facts: firstly that the objectives utilized in any specific subspace clustering method are not independent of the method itself and secondly, that there is always some form of user input about the point density threshold. STATPC tries to detect axis-aligned subspace clusters which are statistically significant, i.e. which contains significantly more points than expected. The task of finding a statistically significant region is transformed into that of an optimization problem. Considering R to be the set of all statistically significant regions, STATPC tries to find a set $R^{reduced} \subset R$ which is small enough to search efficiently and then determine a smallest set P^* which explains all the elements in $R^{reduced}$.

Statistical significance. Assume that H is a hyper-rectangle in a subspace S. Also assume that α_0 is the significance level and θ_{α_0} is the critical value computed with the significance level α_0. H is a statistically significant rectangle if $AS(H) > \theta_{\alpha_0}$ where the probability is computed using $AS(H) \sim Binomial(N, vol(H))$, where $AS(H)$ is the total number of points in hyper-rectangle H, N is the total number of instances in the data set and $vol(H)$ is the volume of hyper-rectangle H. In simple words, a statistically significant hyper-rectangle H has significantly more points than expected if the distribution of the points is assumed to be uniform or the probability of observing $AS(H)$ points in H, when the N data points distributed uniformly in subspace S are fewer than α_0.

Note as well that as the dimensionality of H increases the volume, $vol(H)$ decreases toward 0 which moves the critical value θ_0 toward 1. In other words by increasing the dimensionality, even a hyper-rectangle with a very small number of points might be statistically significant.

Relevant vs. irrelevant attributes. Assuming that H is a hyper-rectangle in subspace S, an attribute a is called relevant for H if all points within the hyper-rectangle H are not distributed uniformly over the active range of a, i.e. between the lower and upper bounds of the hyper-rectangle on attribute a. The Kolmogorov-Smirnov goodness of fit test is used to determine whether or not all the points of H are distributed uniformly over each attribute. Obviously, attributes which are not relevant to H are called irrelevant attributes.

Although the number of hyper-rectangles in a certain subspace might be infinite; however, Moise *et al.* are interested only in all unique Minimum Bounding Rectangles (MBRs) formed with the data points. A subspace cluster is defined as an MBR which is statistically significant and has relevant attributes. However, the number of unique MBRs with at least 2 points can be somewhere between $choose(N, 2)$ and $choose(N, 2) + choose(N, 3) + \cdots + choose(N, 2 \times dim(S))$. Even after pruning, the entire set of statistically significant subspace clusters is too large and too computationally expensive to detect.

Moise *et al.* try to find a reduced set $P^{opt} \subset R$, where R is the set of all subspace clusters in a given data set, so that every subspace cluster in R can be 'explained' by a subspace cluster in P^{opt}. Also, this reduced set should be the smallest set with such a property. The 'explain' relationship tries to say that a set of subspace clusters P, plus background noise explains a subspace cluster H if the observed number of points in H is consistent with this assumption and not significantly larger or smaller which expected.

Moise *et al.* claim that given a data set of N D-dimensional points, a non-empty set $P^{opt} \subset R$ can be found which has the smallest number of a subspace clusters possible so that P^{opt} explains all subspace clusters $H \in R$. They claim that the reason their method provides a solution to this problem is based on two facts: that their objective is formulated as an optimization problem, which is independent of any particular algorithm, as well as the fact that their definition of subspace cluster is based on statistically significance principles and thus it can be trusted that the resulting solution stands out in the data in a statistical way, and is not an artifact of the method.

In short the final objective is to find a reduced set $R^{reduced}$ which is small enough to search in it for the optimal set P^{opt}. To achieve the former, STATPC constructs subspace clusters by analyzing the subspaces and local neighbourhoods around the individual data points Q.

To detect relevant attributes around data point Q, STATPC considers all 2-d rectangles around the given data point with Q in the center and side length $2 \times \delta$ where $\delta \in [0, 0.5]$. These 2-d rectangles are ranked based on the number of instances in them. Then a set of attributes, called 'signaled' attributes, which are – with high probability – relevant to one of the true subspace clusters around Q are selected. Assuming that S^0 is a set of 'signaled' attributes, an iterative refinement of S^0 will give a candidate subspace around Q, called S. To determine whether a subspace cluster around Q exists in S, STATPC builds a series of MBRs, called R^{local}, in S, starting from Q, and adding in each step to the current MBR the point where is closest to the current MBR in subspace S. Since a cluster contains typically only a fraction of the total number of points, STATPC builds only $0.3 * N$ MBRs around Q in subspace S. The question now is to select one of these MBRs which is locally optimal in the sense that it explains all subspace clusters in R^{local} better than any other subspace cluster in R^{local}.

The process explained above will give a set $R^{reduced}$ around the data points. While the first point is selected randomly, subsequent points are selected randomly from the points which do not belong to the detected subspace clusters in the previous steps. Building $R^{reduced}$ terminates when no data point can be selected for further subspace cluster search. Although the cardinality of $R^{reduced}$ is smaller than R, searching the $R^{reduced}$ space with an optimization problem is still too computationally expensive. Therefore a greedy process prunes $R^{reduced}$ to include only a subset of subspace clusters from $R^{reduced}$ which can explain all the clusters within $R^{reduced}$. This process copies subspace clusters from $R^{reduced}$ into a temporary set P^{sol} one by one until the subspace clusters in P^{sol} can explain all the subspace clusters in $R^{reduced}$.

Moise *et al.* evaluate their subspace clustering approach on both synthetic and real data sets. The synthetic data sets designed by Moise *et al.* will be described in Section 5 since these data sets will be used to evaluate S-ESC and its comparators. These data sets generally have approximately 50 dimensions, 300 instances and 5 clusters. The real data sets in this work range between 6 to 34 attributes, 200 to 800 instances and they all are two-class data sets.

3.4 EM

Expectation-Maximization (EM) is based on the idea that there exists an analytic model for the data, and that the functional form of the model is known. However, the values for the parameters which characterize this functional form are unknown. EM is well-known for its ability to optimize a large number of variables, and the ability to predict good estimates for missing information. It can be tailored to provide hard clusters as well as soft clusters whenever needed. Observed data does not need to be complete – i.e. missing values are accepted and in some cases EM can determine an estimate for the missing value. An example of missing information includes but is not limited to cases where a sensor fails to return a value at a desired time.

EM is an example of mixture density-based clusterings. In such a view, each instance is assumed to be generated from one of K underlying probability distributions, i.e. each probability distribution generates instances of a cluster. Probability distributions can take any form such as multivariate Gaussian or t-distributions. It is also possible to have the same probability distribution with different parameters. Since the forms of mixture densities are known, the problem is reduced to estimating the parameters of the distributions.

Assuming that the N instances of a given data set are generated by K mixture densities (clusters), then the mixture probability density for the entire data set can be given as:

$$p(\mathbf{x}|\theta) = \sum_{i=1}^{K} p(\mathbf{x}|C_i, \theta_i) P(C_i) \tag{3.1}$$

where $p(\mathbf{x}|C_i, \theta_i)$ is a class-conditional probability density for cluster C_i with unknown parameter vector θ_i, $P(C_i)$ is the prior probability of instance $\mathbf{x}$ coming from cluster

C_i, $\theta = (\theta_i, \cdots, \theta_K)$ and $\sum_{i=1}^{K} P(C_i) = 1$. Both the class-conditional probability density and the prior probability are known. Also although mixture densities can take any form in theory, they are usually assumed to be multivariate Gaussian densities.

Using the Bayes formula the posterior probability for assigning an instance to a cluster is:

$$P(C_i|\mathbf{x}, \hat{\theta}) = \frac{P(C_i)p(\mathbf{x}|C_i, \hat{\theta}_i)}{p(\mathbf{x}|\hat{\theta})} \tag{3.2}$$

where $\hat{\theta}$ is the vector of estimated parameters.

Once the component densities $p(\mathbf{x}|\theta_i)$ take the form of multivariate Gaussian distributions, mean vectors (μ) and covariance matrices (Σ) replace θ in parameter estimation likelihood equations:

$$p(\mathbf{x}|\theta_i) = \frac{1}{(2\pi)^{D/2}|\Sigma_i|^{1/2}} exp\{-\frac{1}{2}(\mathbf{x} - \mu_i)^T \Sigma_i^{-1}(\mathbf{x} - \mu_i)\} \tag{3.3}$$

To estimate the parameters further, EM uses maximum likelihood (ML) which is a statistical approach which finds the best estimate $\hat{\theta}$ to be the one which maximizes the probability of producing all instances of $\mathbf{X} = \{\mathbf{x}_1, \cdots, \mathbf{x}_N\}$. ML returns the best estimate by solving the log-likelihood equations using the prior known probabilities. Readers are referred to [30, 32, 119] for more details on the mathematical proofs.

The solution to likelihood equations are not available analytically in most cases. In these cases iteratively suboptimal approaches are the popular method to approximate ML estimates [119]. Among these methods, the Expectation-Maximization (EM) algorithm is the one of the more popular approaches [30, 77].

EM does not require complete data. In fact EM regards the data set as incomplete and divides each instance $\mathbf{x}_j$ into two parts ($\mathbf{x}_j = \{\mathbf{x}_j^g, \mathbf{x}_j^m\}$) – observable features $\mathbf{x}_j^g$ and missing data $\mathbf{x}_j^m = (\mathbf{x}_{j1}^m, \cdots, \mathbf{x}_{jK}^m)$. The missing data for each instance is a binary vector of size K with its values being 1 or 0. A 1 in the ith position determines that $\mathbf{x}_j$ belongs to component i. Therefore, the complete data log-likelihood can take the form:

$$l(\theta) = \sum_{j=1}^{N} \sum_{i=1}^{K} x_{ji}^m \ log[P(C_i)p(\mathbf{x}_j^g|\theta_i)] \tag{3.4}$$

The EM algorithm generates a series of parameter estimates $\{\hat{\theta}^0, \hat{\theta}^1, \cdots, \hat{\theta}^T\}$ iteratively until the termination criterion is met. The termination criterion is usually a number of iterations (T). After initializing $\hat{\theta}^0$ at $t = 0$ the two main steps of EM keep alternating until the termination condition is reached. The two steps are described below:

1. The E-step. Compute Q , which is the conditional expectation of the complete data log-likelihood, given the current estimate,

$$Q(\theta, \hat{\theta}^t) = E[logp(\mathbf{x}^g, \mathbf{x}^m|\theta)|\mathbf{x}^g, \hat{\theta}^t], \tag{3.5}$$

   and

$$E[x_{ji}^m|\mathbf{x}^g, \hat{\theta}^t] = \frac{\hat{P}(C_i)p(\mathbf{x}_j^g|\hat{\theta}_i^t)}{\sum_{l=1}^{K} P(C_i)p(\mathbf{x}_j^g|\hat{\theta}_l^t)} \tag{3.6}$$

2. The M-step. Select a new parameter estimate that maximizes the Q-function,

$$\hat{\theta}^{t+1} = arg\ max_{\theta} Q(\theta, \hat{\theta}^t) \tag{3.7}$$

It should be noted that as with other iterative optimization methods, the EM algorithm suffers from sensitivity to parameter initialization, convergence to the local optima, the effect of a singular covariance matrix, and a slow convergence rate [77].

Chapter 4

Methodology

The various components comprising the proposed framework for Symbiotic Evolutionary Subspace Clustering (S-ESC) were summarized in Chapter 1. This chapter begins by re-introducing the S-ESC components. Algorithm 1 presents the chronological order of the components and Figure 4.1 illustrates the relationship and interaction between them.

Component 1 generates a *D-dimensional grid* by applying a regular clustering algorithm to each data set dimension 'independently.' The resulting discrete grid will serve as the starting point for the evolutionary process of subspace cluster identification under the assumption of axis-parallel clusters as established in Chapter 2. The purpose of this step is to convert the task into one which can be addressed through a combinatorial search of a discrete – rather than continuous – search space. The process is performed *only once* relative to the entire training repository. However, if the data set is inhibiting, sampling can be considered. In this work the 1-d attribute-based clustering is performed through the application of the EM algorithm [77] as implemented in the WEKA machine learning package [42]. Naturally, any regular clustering algorithm which does not require a-priori knowledge of k could be used for this purpose, i.e. the clustering activity does not require dimension reduction, but should minimize the need for additional arbitrary assumptions.[1]

Component 2 of the S-ESC framework denotes the *symbiotic process* for coevolving *cluster centroids* (CC) as well as *clustering solutions* (CS) alongside one another through a bi-objective framework. Specifically, a subspace coordinate is defined by sampling from the available attributes and corresponding 1-d dense regions as defined by the grid from component 1. The resulting subspace coordinate denotes the cluster

[1]Other candidate algorithms that potentially satisfy this goal include X-means [89] and the potential function method [22]. The EM algorithm was used here because the WEKA implementation makes use of cross-validation to aid in the optimization of parameters such as k.

centroid. Conversely, a clustering solution is composed of indices (links) to some subset of the available cluster centroids. Cluster centroids associated with a clustering solution establish a partitioning of all the data instances through a nearest neighbour allocation. In the context of symbiosis a cluster centroid is the *lower-level symbiont* and a clustering solution is the *higher-level host* which is formed by grouping an arbitrary number of symbionts. In a coevolutionary context a host is a team whereas a symbiont is a member.

Component 3 denotes the *fitness evaluation*. Fitness is only performed relative to clustering solution individuals (i.e. the higher-level individuals). Each clustering solution represents a candidate group of clusters – denoted by their centroids – hence a form of group selection is in effect. Fitness of a group can be evaluated either by summing/averaging the fitness of its members or it could be the outcome of interaction between members. In the case of S-ESC, the fitness of a clustering solution is the result of a *group interaction* between cluster centroids forming the clustering solution. Thus, the quality of a partitioning is a function of the nearest neighbour allocation of data instances to cluster centroids defined by each clustering solution. It is not possible to estimate the quality of cluster centroids independently. Fitness evaluation takes the form of a bi-objective Pareto methodology [43] using compactness (distortion) and connectivity (connectedness) objectives. Without loss of generality the NSGA-II algorithm is assumed [29], although any Pareto framework for evolutionary multi-objective optimization could be utilized.[2]

Component 4 is responsible for *diversity maintenance* through the asexual reproduction of clustering solutions based on a set of mutation operators. Diversity is encouraged in both lower-level and higher-level populations, hence two separate mutation operators are assumed. A single-level mutation (SLM) operator is dedicated to modifying the higher-level clustering solution individuals, thus conducting a search through different cluster combinations. The second mutation operator, multi-level mutation (MLM), introduces variation within lower-level cluster centroids as well as higher-level clustering solutions. In both cases the clustering solution/cluster centroid variation is proceeded by the cloning of the corresponding individual. Moreover, as both operators potentially modify the content of the clustering solution population,

[2]A decision made in part by the availability of a particularly efficient implementation of the NSGA-II algorithm [51].

the rate of change of the clustering solutions is greater than that of the (lower-level) cluster centroid population. The key to this is that clustering solutions are defined in terms of the cluster centroid population. Hence, it is necessary for the clustering solution content to turn over faster than that of the cluster centroid content.

Sub-component A introduces *subsampling* in the form of *random subset selection* (RSS) at the beginning of each generation so that clustering solutions are evaluated only against a subset of data instances (called the active set), rather than the whole data set. This is in response to the computational cost associated with evaluating the connectivity objective under subspace clustering scenarios. Thus, sample-based evaluation is being employed increasingly in evolutionary computation in response to the cost of fitness evaluation in general, e.g. with regards to policy search [31] or genetic programming [105].

Sub-component B addresses the post-training selection of the most promising solution out of a pool of solutions. That is to say, assuming a Pareto evolutionary multi-objective process such as NSGA-II results in a set of non-dominated solutions (i.e. they are all equally good in the Pareto sense), whereas a single 'champion' solution is required per run. One approach to champion selection might be to visualize the Pareto front. However, for benchmarking purposes this is not practical and somewhat arbitrary. From a bi-objective perspective, better solutions on a Pareto front tend to lie where a small change in one objective results in a much larger change in the other objective. This is referred to intuitively as a *knee* on the Pareto front; the solution is referred to as a *knee solution* and the method is hereafter referred to as *knee detection* [99].

The rest of this chapter expands on each of the S-ESC components with more details.

4.1 Component 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*. Thus, for a D-dimensional data set, the standard clustering algorithm is called D times, once per attribute (Step 1, Algorithm 1). The values of a data set for each attribute can also be considered as the values of the data set as projected on the attribute – albeit

Algorithm 1 The S-ESC algorithm. CC and CS denote the cluster centroid (host) and clustering solution (symbiont) populations respectively. H_{size} and P_{size} refer to the host population size and the number of points subsampled by RSS respectively. gen is the current generation and G_{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 $gen < G_{max}$
   (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 the SLM operator on the cloned host
       iv. Randomly select a parent symbiont, within current host
       v. Clone the parent symbiont
       vi. Apply the MLM operator on the cloned symbiont
   (b) Evaluate the host population
   (c) Sort $2 \times H_{size}$ hosts on fronts
   (d) Delete the 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 the champion host from among Pareto hosts
```

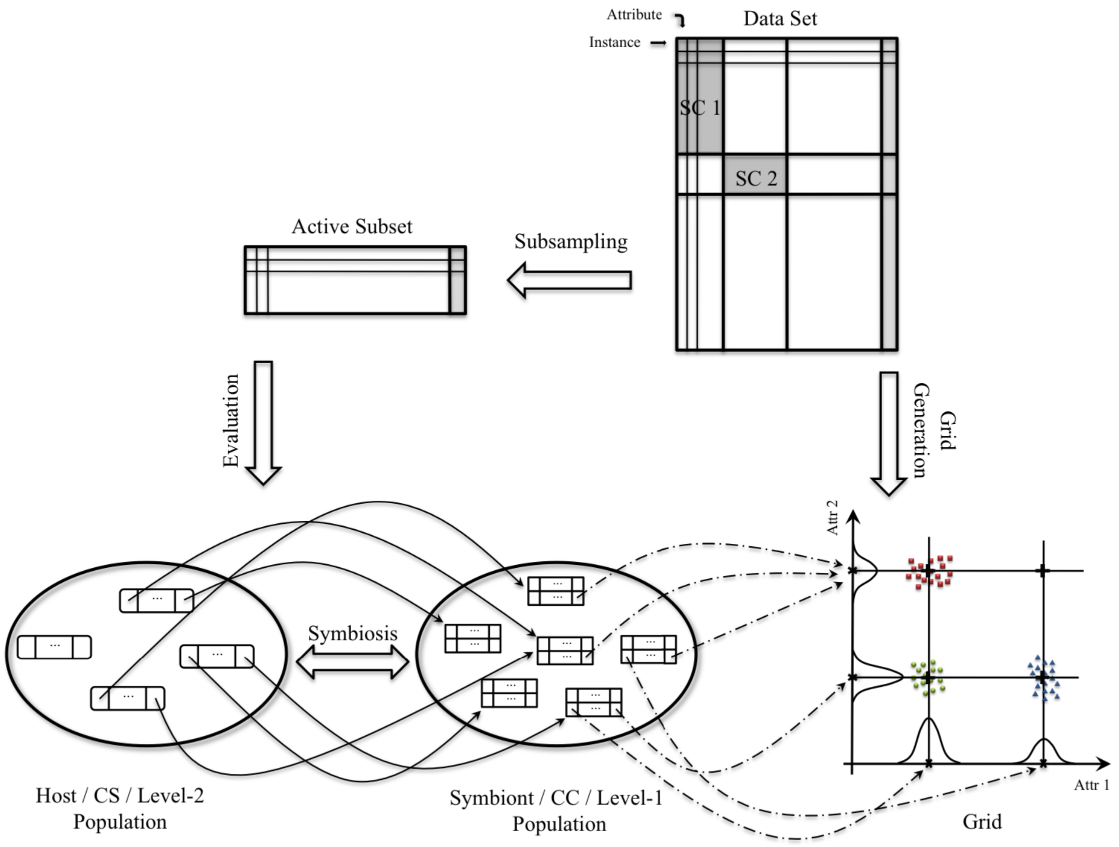


Figure 4.1: S-ESC components and sub-components and their interaction in summary.

under the axis-parallel subspace assumption. In effect, rather than describe each data instance, $\overrightarrow{x}(i)$, by its location with respect to all attributes simultaneously, each data instance is decomposed instead into the set of projections onto each attribute, j. Thus, $x(i)_j$ – or simply x_{ij} – is the projection of data point i onto attribute j.

A traditional clustering algorithm such as k-means [45], X-means [89] or EM [30, 77] may be used to find the centers associated with each dense region of each attribute. Although most traditional clustering algorithms can potentially be considered for this application, methods which require less a-priori information are preferable. EM and k-means – as traditionally defined – require knowledge regarding the number of clusters or k selection [45, 77]. Similarly, X-means requires the minimum and maximum number of clusters in a data set and then uses a maximum description length metric to guide the selection of a specific k value [89]. Alternately, some form of search process could be employed to select k values on a task-by-task basis. Examples include cross-validation and genetic algorithms. The WEKA implementation for EM clustering assumes the process of cross-validation. Hereafter, EM clustering with cross-validation will be assumed for the identification of 1-d centroids with respect to each attribute.[3]

Figure 4.2 illustrates the process for a 2D data set.[4] Figure 4.2(a) represents the data set in its original format. In Figure 4.2(b) dense regions of each attribute are identified by the clustering algorithm of choice as illustrated by attribute-specific Gaussians. The centroids of each cluster (i.e. dense region) as projected on each axis are marked by a '$\times$' symbol. Figure 4.2(c) illustrates the superimposition of all axis-specific centroids forming the final 2D grid. The corresponding set of 'candidate' cluster centroid locations is identified by $+$ signs and numbered 1 to 4. Naturally, the resulting cluster centroids ($+$) are free to index a variable number of attribute-specific clusters ($\times$) – albeit under the constraint that only one attribute-specific cluster be sampled per attribute.

It should also be noted that this process is conducted *once* per data set. In the benchmarking study in Section 6 the EM algorithm [77] is assumed. Multiple runs of the proposed S-ESC algorithm are performed relative to the same common grid. Moreover, since grid generation is performed only once per data set regardless of the number of times S-ESC is run, the grid generation component is considered a pre-processing step. On completion of component 1 a D-dimensional grid is formed by the superimposition of all attribute-specific (1-d) centroids. Note that not all grid locations correspond necessarily to valid cluster locations, thus adding additional ambiguity to the task.

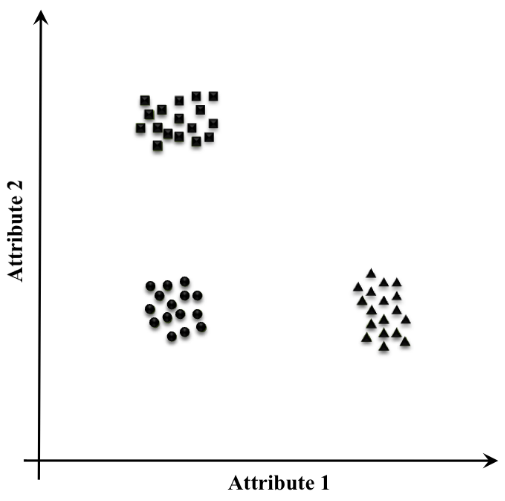
4.2 Component 2: Symbiotic Relationship and Representation

S-ESC decomposes the overall task of subspace clustering into two search processes: finding candidate cluster centroids (using different attribute subsets); and finding the best combination of candidate cluster centroids to define an overall solution to the subspace clustering task.

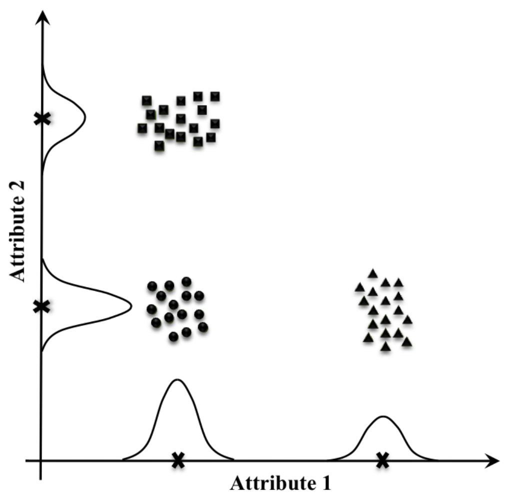
A cluster centroid is basically an S-dimensional coordinate where S is the number

[3]Some comparative benchmarking was performed to reach this conclusion. X-means returned solutions faster, but was still much more sensitive to factors such as the order of data presentation during the clustering process.

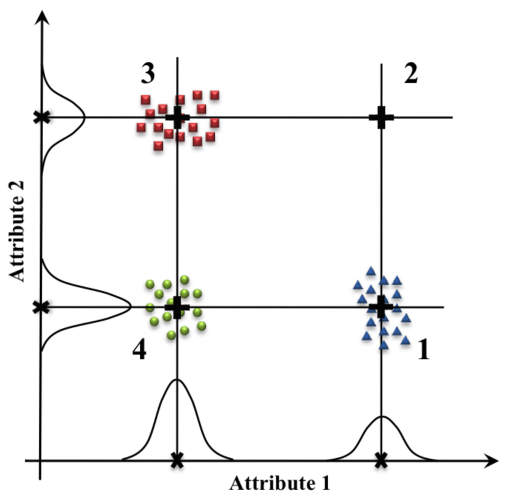
[4]A 2D representation is selected here merely for the sake of visualization simplicity.



(a) Before grid



(b) 1-d density and centroid identification



(c) After grid

Figure 4.2: Component 1 – D-dimensional grid generation. 4.2(a) data set with no grid; 4.2(b) identification of attribute-specific centroids/dense regions; 4.2(c) the enumeration of all possible candidate cluster locations through the D-dimensional grid.

of attributes supporting the subspace within which the coordinate is embedded. S can take any value between 2 and D (data set dimensionality), although it is always possible to modify both tails of this range to get simple or more complex solutions, or to modify them based on any domain knowledge about the data set.

On the other hand, a clustering solution attempts to compose a solution by sampling from the set of currently available cluster centroids. No assumptions need be made regarding the number of cluster centroids necessary to build a cluster solution. A nearest neighbour assignment procedure assigns each data point to the closest cluster centroid while considering only the attribute support of each cluster centroid when calculating the distance between a data point and a cluster centroid.

Symbiosis is the close and often long-term mutual interaction between different species living together and benefiting from each other [75]. Symbiosis implies that a number of lower-level symbionts from one species undergoes an aggregation. The resulting (higher-level) host organism represents a new species. Such a process also introduces the concept of 'levels of selection' [86] in which fitness is evaluated either at the level of the symbiont or the host. The utility of symbiosis is assumed explicitly for this work, as opposed to attempting to model the transition between symbiont and host; hence fitness will be evaluated only at the 'level' of the host.

The above interaction between cluster solutions and cluster centroids is interpreted as forming a symbiotic relationship. Using the symbiotic vocabulary, the lower-level cluster centroids in S-ESC correspond to symbionts, and higher-level clustering solutions correspond to hosts. Hereafter, the terms symbiont, cluster centroid (CC) and lower-level individual are used interchangeably, as are host, clustering solution (CS) and higher-level individual.

S-ESC takes advantage of two separate populations, Figure 4.1: the host (or CS) population is of a fixed size whereas the symbiont (or CC) population size is variable. Although the symbiont population has a dynamic size, there are minimum and maximum bounds. The minimum size of this population is equal to the size of the host population whereas the maximum bound for it is equivalent to 5 times the size of the host population. These limits are arbitrary, with the minimum limit set to establish a starting point for population initialization and the upper which defines a (weak) computational limit.[5]

The first step in S-ESC after making the grid is to initialize both populations (Steps 2 and 3, Algorithm 1). Initialization begins by generating a population of symbionts (CS) randomly as the initial genetic material to be indexed by the host (CC) individuals.

[5]The worst case computational scenario would be for each host to be of some 'maximum' genome length. This does not necessarily imply a particularly large symbiont population, but in practice there is a correlation.

Symbionts index at most one (attribute-specific) 1-d cluster centroid from a subset of attributes. The corresponding representation assumes a series of integer pairs, $\langle a, c\rangle$, where a determines the attribute index, and c identifies the (attribute-specific) 1-d centroid, or:

$$Symbiont_i \equiv CC_i \equiv \{\langle a_1, c_1\rangle, \langle a_2, c_2\rangle, \cdots, \langle a_p, c_p\rangle\} \tag{4.1}$$

where $a_i \neq a_j$ is required for $i \neq j$. The number of 1-d centroids, p, comprising a symbiont (CS) is potentially unique, and assumes a value between 2 and the data set dimensionality D ($2 \leq p < D$), i.e. for subspace identification to take place $p < D$ whereas $p > 2$ avoids the trivial case of a one dimensional cluster.

To initialize a symbiont the length (dimensionally) of the symbiont is initially selected randomly. Call this length L for a sample symbiont. Then for each pair of integers a random attribute index is selected – a in the $\langle a, c\rangle$ pair – followed by a random number between 1 and the number of 1-d centroids within the selected attribute – c in the $\langle a, c\rangle$ pair. This process is repeated L times until the symbiont is initialized fully. Such a representation enables a cluster centroid (symbiont) to be expressed independently of data set dimensionality or cardinality. As noted above – Step 5a, Algorithm 1 – S-ESC populates the symbiont population with H symbionts initially with H being the host population size and a pre-defined parameter.

The encoding/decoding of a symbiont is summarized in Figure 4.3 using two symbionts each defined in terms of two attributes for the sake of brevity. The two integer pairs used in cluster centroid 1 ($CC1$) are $\langle 2, 1\rangle$ and $\langle 1, 2\rangle$. The first pair encodes the 1st 1-d centroid from attribute 2, and the latter pair encodes the 2nd 1-d centroid from attribute 1. Extending these two 1-d centroids along the axes returns the point in 2D space marked by a plus sign in the center of blue triangles and numbered '1'. This potential cluster centroid is clearly a 'good' cluster centroid since it is the center of a dense region. Once this potential cluster centroid is indexed in a clustering solution it attracts all the points around it and makes a cluster which is distinctly separate from other clusters. Similarly, the other cluster centroid, $CC2$, encodes the point marked by a plus sign and numbered '2' with no instances close to it. It would be expected that the utility of this potential cluster centroid is 'null.'

Naturally, in higher dimensions the number of integer pairs per symbiont is free to

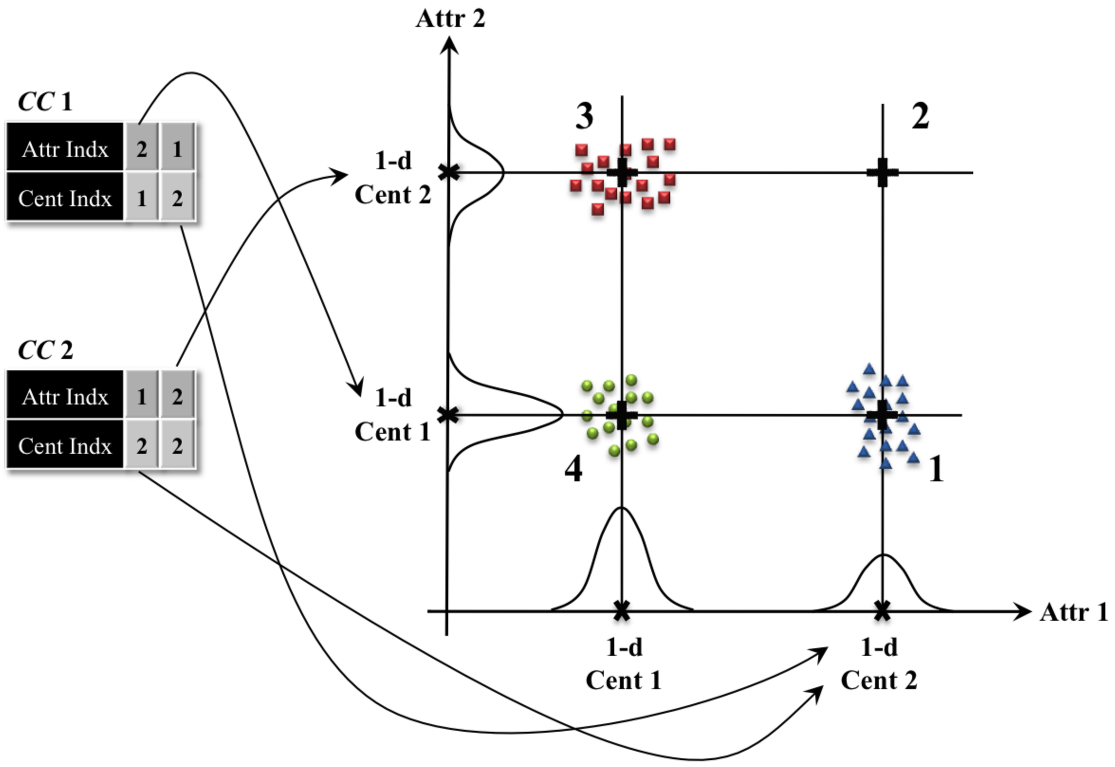


Figure 4.3: Representation for symbiont or CC individuals relative to a 2-d grid. Two symbionts $CC1$ and $CC2$ are (in this example) expressed in terms of a common pair of attributes, Attr1 and Attr2. $CC1$ consists of two integer pairs: $\langle 2, 1\rangle$ denoting attribute 2, 1-d density 1, and, $\langle 1, 2\rangle$ 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'.

change, implying that symbiont dimensionally is potentially unique. In practice, the search for candidate cluster centroids is limited to the interval $[2, 20]$ in order to avoid declaring trivially degenerate 1-d cluster centroids or complex and incomprehensible cluster centroids with more than 20 dimensions. Naturally, if prior knowledge indicated that symbionts of 20 dimensions or greater would be expected, the range could be modified appropriately. Obviously if the data set dimensionality is smaller than 20, the upper bound of this range would be set to match the data set dimensionality.

Hosts index groups of symbionts under a variable length representation. Each host is encoded as an index vector, each integer indexing (or linking to) a symbiont. Thus:

$$Host_i \equiv CS_i \equiv \{CC_1, CC_2, \cdots, CC_q\} \tag{4.2}$$

where $CC_i \neq CC_j$ is required for $i \neq j$. The number of cluster centroids in a clustering solution, q, is potentially unique to a host and can vary between 2 and the symbiont population size ($2 \leq q \leq |Symb\ Pop|$).

Hosts conduct a search for 'good' symbiont combinations and it is possible for the same symbiont to appear in multiple hosts. Two sample hosts, along with a population of four symbionts, are illustrated in Figure 4.4. Each host defines a partitioning of the data relative to its combination of candidate cluster centroids, care of the nearest neighbour allocation of exemplars to the host membership.

The first clustering solution ($CS1$) is linked to three symbionts: $CC3$, $CC1$ and $CC4$. The order of the symbionts is not important. This clustering solution partitions the given data set into three clusters with cluster centroids encoded by the three indexed CCs. Assigning data points to their nearest cluster centroids gives the three clusters shown in the top-left corner of the figure, illustrated with different colours for each cluster. The three cluster centroids used here are the three promising cluster centroids which were the centres of the dense regions of the data set.

The other clustering solution ($CS2$) uses $CC2$ and $CC4$ to partition the given data set into two clusters. $CC4$ is a potential cluster centroid being in the center of a cluster of data items. $CC2$, on the other hand, is not a good candidate cluster centroid. Partitioning the given data set using the two mentioned cluster centroids gives the partitioning shown in the left-bottom corner of the figure. It is a linear partitioning, care of the two cluster centroids, assigning almost half of the data items from the two undetected clusters to $CC4$ and the rest to $CC2$.

Similar to the representation of symbionts, a minimum and maximum bound is also placed on the number of symbionts to which a host can link, i.e. the minimum and maximum clusters expected in a data set. These parameters are also adjustable based on any prior domain knowledge a user might have.

To initialize a clustering solution, the number of cluster centroids within the clustering solution is defined first with uniform probability: call this k. A vector of size k is allocated to this clustering solution and random links between the clustering solution and the k cluster centroids are established. The initialization process makes sure that there are no duplicate links within a clustering solution. This process repeats H times to populate the host population with H clustering solutions.

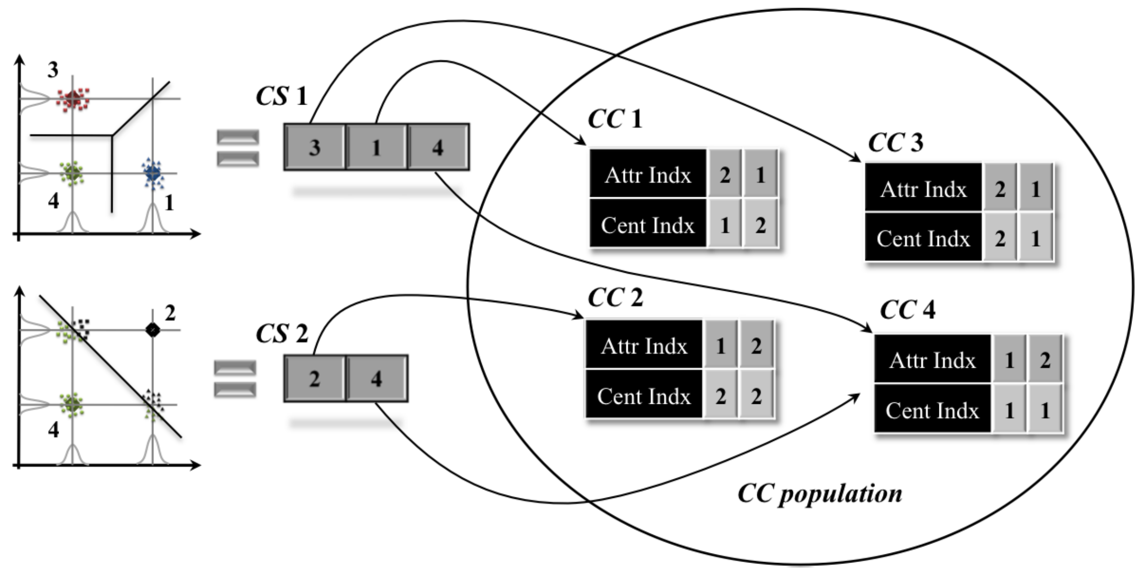


Figure 4.4: Representation of host or CS individuals. Host $CS1$ indexes three symbionts, forming a clustering solution (CS) with three clusters. It makes a reasonable partitioning of the given data set. Host $CS2$ defines a CS from two symbionts. Clearly in this case the partitioning is not optimal, because it misses two clusters and assigns data points to the wrong clusters. (LHS subfigure).

4.3 Component 3: Fitness Evaluation

Evolutionary methods frequently benefit from adopting a multi-objective approach; hence there is widespread use of multi-objective methods for both optimization [126] and machine learning tasks [53]. The principle 'trick' lies in addressing how to return a scalar fitness value from multiple performance measures. Moreover, maintaining (population) diversity against the multiple performance metrics may also require some additional consideration. At present, evolutionary multi-objective optimization (EMO) tends to assume a Pareto dominance methodology, with the bi-objective approach popularized by Handl and Knowles representing the most well-known result in (full-dimensional) clustering [43]. From the perspective of the proposed S-ESC algorithm it is important to assume a Pareto bi-objective approach in order to benefit from the additional generality that pursuing dual objectives provides. Beyond this, any number of generic EMO algorithms for genetic algorithms could be employed.

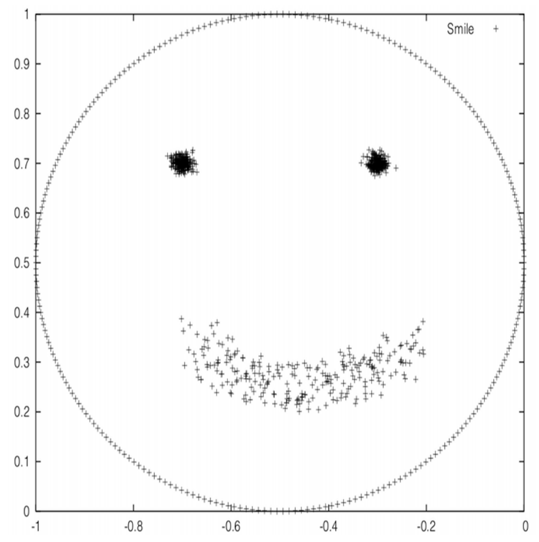
A start is made by establishing which entities are evaluated by the S-ESC fitness functions, i.e. host, symbiont or both (Section 4.3.1). Next, in Section 4.3.2, the Pareto approach to EMO is summarized and the biases associated with the generic design decisions are highlighted. Section 4.3.3 summarizes the two objectives for clustering as popularized by Handl and Knowles and describes the generic properties of the Pareto dominance approach. Also noted are the requirements for champion solution identification (Pareto methods typically return more than one equally good solutions) and fitness evaluation using a subset of the available data. These issues will later be addressed through Sub-components A and B of the S-ESC framework.

4.3.1 Group Selection

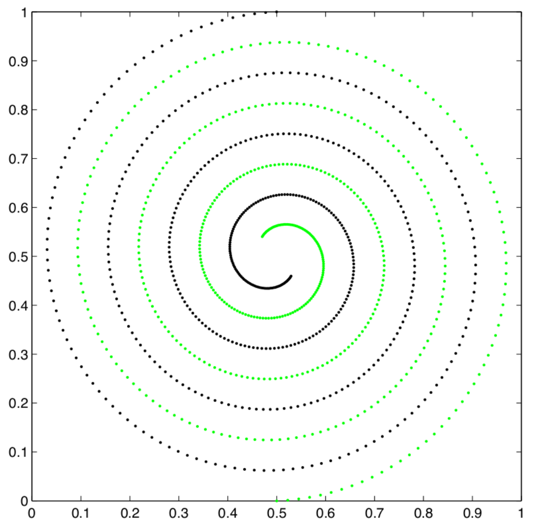
An important property of S-ESC which is worth noting again is the concept of 'group selection' (Section 4.2). That is to say, each host identifies a unique subset of symbionts, hence group selection is involved. The fitness of a group can be evaluated in one of the two ways: MLS1 and MLS2 [86]. In MLS1 the fitness of a host is proportional to the sum of symbionts fitnesses, e.g. the sum or the average of the corresponding symbionts fitnesses. On the other hand, in MLS2 the fitness of a host is the result of the interaction between the symbionts to which it is linked, regardless of how each symbiont performs individually. In other words, here the group/team performance is measured rather than the individual/member performance. MLS2 is assumed with the potential of benefitting from the group fitness exceeding the mere sum of its members. Hence, symbiont 'fitness' is implicit. Should a symbiont *not* receive any host indices 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).

4.3.2 Pareto-dominance and Evolutionary Multi-objective Optimization

Handl and Knowles used the smile and spiral data sets (Figures 4.5(a) and 4.5(b)) to motivate the adoption of a (Pareto) bi-objective approach to clustering in general [43]. There are two dense regions showing the eyes of the face and a rather curved elongated less dense region forming the mouth of the smiley face. These three dense regions are more likely to be detected by algorithms relying on single density-based optimization functions. However the (relatively) sparse *outline* of the face in the same data set or any of the two spirals in Figure 4.5(b) is less likely to be found by such algorithms. These clusters need a different optimization function that is less sensitive



(a) Smile



(b) Spirals

Figure 4.5: Sample data sets with arbitrary cluster shapes; (a) Smile and (b) Spirals.

to data distribution and cluster shape. For these clusters the label of each exemplar can be determined by observing the cluster label of its neighbouring points. The optimization function is penalized if the cluster label of a data point is different from those of its neighbours.

Adopting more than one objective presents the problem of how to rank individuals for reproduction and/or survival (Steps 5(a)i and 5d, Algorithm 1). As of this writing, the most widely employed framework for 'mapping' multiple objectives into a scalar ranking employs the concept of Pareto dominance. Assume F is an n-dimensional vector of objective functions, i.e. $F(x) = (f_1(x), \cdots, f_n(x))$. Individual x dominates individual y if and only if x is better or at least matches y in all but one objective and is strictly better than y in at least one objective:

$$x \prec_o y \;\Leftrightarrow\; \forall i \in \{1, \cdots, n\} :\; f_i(x) \leq_o f_i(y) \;\wedge\; \exists k :\; f_k(x) <_o f_k(y) \tag{4.3}$$

where, assuming f_i comprises all functions for minimization, then $f_i(x) \leq_o f_i(y)$ indicates that x is either better than or equivalent to y in objective i, and $f_k(x) <_o f_k(y)$ indicates that x is strictly better than y for some objective k. In other words, x needs to be strictly better than y in at least one objective function and at least as good as y in the remaining objective functions. In this case x is said to dominate y.

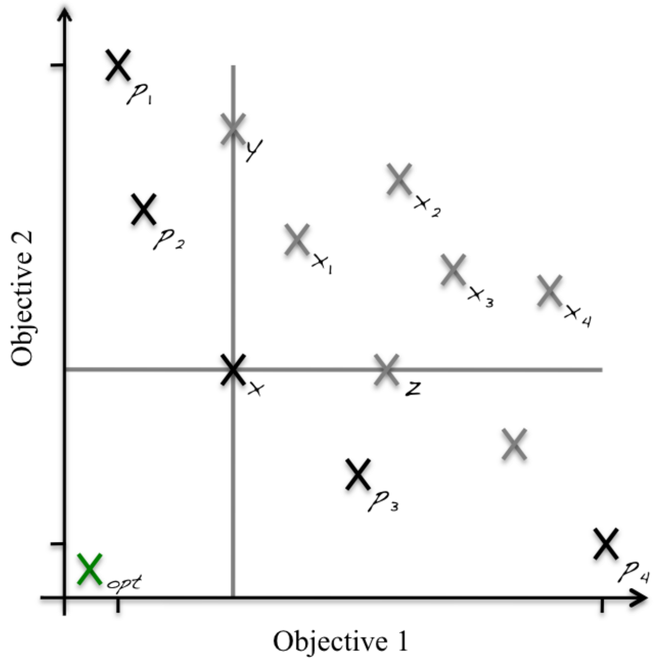


Figure 4.6: Dominance in a bi-objective space. Individual x dominates all the individuals within the box delimited by extending its objective values.

Figure 4.6 illustrates the concept of dominance for a problem, which tries to minimize 2 objective functions. This example uses 2 objective functions. However, the same concept can be generalized to more objective functions. As can be seen in this figure, individual x has the same objective value as individual y in objective 1 ($f_1(x) \leq_o f_1(y)$) and has a smaller value compared to y in objective 2 ($f_2(x) <_o f_2(y)$).

Similarly, individual x dominates individual z because it has a similar value compared to z in objective 2 ($f_2(x) \leq_o f_2(z)$) and a smaller value compared to z in objective 1 ($f_1(x) <_o f_1(z)$). Individual x also dominates four individuals marked as $x_1, \cdots, x_4$ simply because it has a smaller objective value for both objective 1 and 2, i.e. $\forall i \in \{1, 2\} : \;\; f_i(x) <_o f_i(x_1)$. The same holds true for the x_2, x_3 and x_4 individuals. Individual *opt*, which is close to the origin, is a hypothetical individual which would dominate all other individuals if it could be reached.

For all grey individuals in Figure 4.6 there is at least one grey or black individual which dominates them. These individuals are therefore called *dominated* individuals. On the other hand, there is no individual to dominate the black individuals; therefore, these individuals are called *non-dominated* individuals. The curve going through all

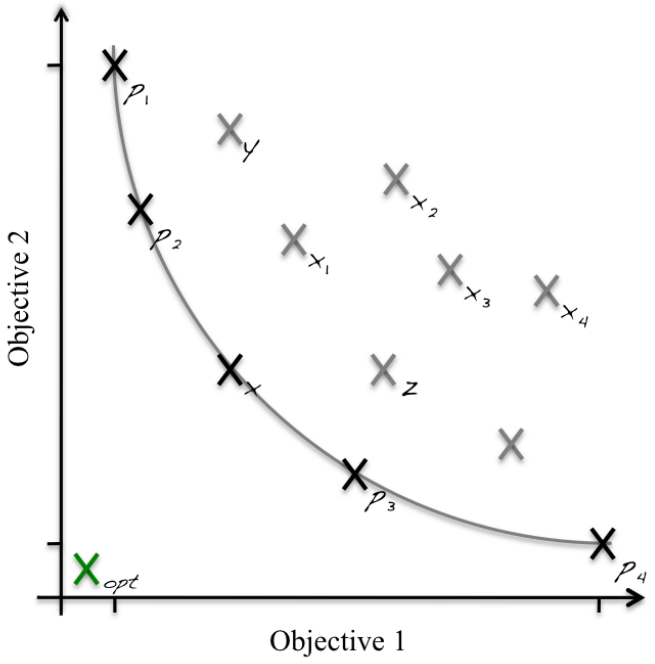


Figure 4.7: The Pareto front of a evolutionary bi-objective optimization problem. The curve going through all non-dominated individuals is called a *Pareto front* and the individuals on this front are called Pareto-optimal individuals.

non-dominated solutions, shown in Figure 4.7, is called the *Pareto front* [98] and individuals lying on it are also called 'Pareto-optimal solutions.' Adopting a Pareto approach addresses the problem of mapping between a vector of objectives to a scalar ranking of individuals by assigning the same rank to individuals with the same 'level' of dominance. Pareto EMO algorithms assume one of the two generic approaches: dominance rank and dominance count, resulting in the introduction of different biases into the search process:

- *Dominance rank* counts the number of individuals which better the candidate solution (visualized as counting any solutions in a quadrant to the bottom left of the candidate solution in the objective space) and should be minimized. Adopting a dominance rank-based scheme is known to reward diversity along the breadth of the front.

- *Dominance count* counts the number of individuals bettered by the candidate solution (visualized as counting the solutions in a quadrant to the top right of the candidate solutions location in the objective space) and should be maximized.

> Adopting a dominance count-based scheme is known to reward most exploration in the midpoint of the Pareto front.

Without loss of generality this work assumes the approach adopted by NSGA-II [29], where this makes use of dominance ranking. This type of decision is made in part due to the availability of optimized code to reduce the run-time complexity of Pareto dominance-based evaluations [51]. By contrast, other EMO algorithms exist which make use of both dominance ranking and dominance counts, e.g. SPEA2 [128]. Although potentially more accurate, there is also an additional computational overhead in adopting multiple mechanisms for measuring dominance. Once a good pool of Pareto-optimal solutions is found the solutions are monitored manually or automatically and the best solution (sometimes called winner or champion solution) is selected. Section 4.5.2 will summarize the process adopted for this (cf., knee detection).

In addition to dominance ranking, NSGA-II also utilizes 'crowding distance' as an additional bias for maintaining diversity. Specifically, diversity maintenance (exploration) versus fitness-based selection (exploitation) needs to be addressed in order to represent an effective search strategy. Crowding as used in NSGA-II is a function of the distance (in the objective space rather than the attribute space) between each individual and its neighbouring individuals on the same front. The larger the crowding distance of an individual, the more distinct from other individuals it is. On a given front NSGA-II prefers individuals with a larger crowding distance over the ones with smaller crowding distances.

NSGA-II sorts all parent and offspring individuals on different fronts in the objective space. Individuals on the Pareto front (front 1) are kept to evolve further in subsequent generations (i.e. elitist archiving). If there is room for more individuals in the population, individuals from fronts 2, 3, ... are also included in the population for the next generation. If there is not enough room to include all the individuals of a given front, the crowding distance heuristic is invoked, i.e. individuals with a higher crowding distance are kept for further evolution. Such a heuristic helps NSGA-II select the most diverse set of individuals by picking individuals which have the least number of neighbours on the same front.

At the end of each generation all (parent and offspring) host individuals make a pool of $2 \times H$ hosts, where H is the host population size. Being an elitist method, NSGA-II keeps only H more fit hosts and reserves the rest only to be replaced with H new offspring in the next generation. Thus, with respect to Algorithm 1, Step 5a adds H hosts and Step 5d removes H hosts.

4.3.3 Bi-objectives for the Clustering Task

For S-ESC a multi-objective formulation will be assumed based on two potentially conflicting objectives; *compactness* (or distortion) and *connectivity* (or connectedness). This was proposed first by Handl and Knowles in a full dimensional clustering context [43]. Compactness tries to find dense clusters by minimizing the distance between data points and their nearest cluster centroids. This objective function creates clusters by putting data points into those clusters whose centroid is closest to them. Ultimately, this objective function creates spherical clusters such as the eyes and mouth of the smile data set in Figure 4.5(a). Minimizing the compactness of a host, in the degenerate case, results in as many 'singleton' clusters as data instances. The compactness value of such a solution would be zero.

Connectivity, on the other hand, tries to put a data item into the cluster which 'owns' most of data items in the neighbourhood of the current data point. Therefore, it can potentially find non-Gaussian elongated clusters as well as Gaussian spherical clusters. This objective function can potentially discover the larger circle of the face in Figure 4.5(a) and the two spirals in Figure 4.5(b). In the degenerate case connectivity tries to put all data instances into one cluster making one 'super' cluster. Similar to the compactness case the connectivity value of such an extreme solution is zero.

Obviously there is a tension between the two objectives. Compactness tries to find a large number of smaller clusters by minimizing an objective function which is based on the distance between data points and cluster centroids, while connectivity penalizes the objective function for putting neighbouring data points into different clusters promoting fewer larger clusters. This tension is a key factor for estimating the optimal number of clusters in a data set [43].

The most significant drawback of utilizing the connectivity objective, however, is that under a subspace setting it is necessary to re-evaluate the metric continuously for each subspace [112], whereas under full-space clustering such a metric can be precomputed [43]. Unless this is addressed, it becomes a constraint on scaling to data sets with larger cardinality. With this in mind, a process for data subsampling will be introduced later in 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 4.4). This nearest neighbour allocation determines the symbiont (within the host) for each data instance. 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) calculated over the attributes supported by the symbiont:

$$Com(CS) = \sum_{i=1}^{k} \sum_{j=1}^{|CC_i|} dis(x_j, CC_i) \tag{4.4}$$

where CC_i is the nearest CC to instance x_j, k is the number of CCs (symbionts) within the CS (host) under evaluation, $|CC_i|$ is the number of instances assigned to CC_i, and $dis(.)$ calculates the distance between instance x_j and the closest cluster centroid CC_i over the attributes supported by CC_i. CC and CS can be referred to as Symb(iont) and Host for more readability:

$$Com(Host) = \sum_{i=1}^{k} \sum_{j=1}^{|Symb_i|} dis(x_j, Symb_i) \tag{4.5}$$

Connectivity tries to give more weight to subspace clusters which assign neighbouring data instances to the same subspace cluster by penalizing adjacent data points in different subspace clusters [43]:

$$Con(CS) = \sum_{i=1}^{N} \sum_{j=1}^{M} \begin{cases} \frac{1}{j}; & \text{if } \nexists CC_s : \; x_i \in CC_s \;\wedge\; x_{i,NN_j} \in CC_s \\ 0; & \text{otherwise} \end{cases} \tag{4.6}$$

where N is the exemplar count of the data set; M is the number of nearest neighbour exemplars to x_i S-ESC considers for calculating connectivity which is defined as $M = max(10,\; 0.01 \times N)$ and x_{i,NN_j} is the j^{th} nearest neighbour exemplar to x_i. The analogous expression substituting Symb(iont) and Host for CC and CS has the form:

$$Con(Host) = \sum_{i=1}^{N} \sum_{j=1}^{M} \begin{cases} \frac{1}{j}; & \text{if } \nexists Symb_s : \; x_i \in Symb_s \; \wedge \; x_{i,NN_j} \in Symb_s \\ 0; & \text{otherwise} \end{cases} \tag{4.7}$$

Penalizing connectivity by $1/j$ for the j^{th} nearest instance to x_i – which has a different cluster membership than x_i – captures this concept, whereas penalizing connectivity by a fixed value of 1 implies that all neighbouring instances contribute equally to the connectivity of an instance, regardless of their distance to x_i. A large value for connectivity indicates that a large number of neighbouring points are allocated to different clusters, whereas a small value of connectivity indicates that only a small number of adjacent 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 for each cluster centroid is potentially different in each generation and therefore the nearest neighbours require rediscovery at each generation. Based on the definition of connectivity, M nearest data instances to each data instance need to be identified which requires N comparisons for each data point, hence a complexity of $O(N^2)$. Later in Section 4.5.1 it will be explained how this will be reduced to $O(P \times N)$ by subsampling $P \ll N$ instances in the beginning of each generation.

It is also worth noting that although the nearest neighbour assignment tries to create spherical clusters; however, it is not the only factor in deciding cluster shapes. Three factors can be involved in deciding a cluster shape; the grid, nearest neighbour distance function and the objective functions. The grid is formed by superimposing different attributes' 1-d centroids, which in turn are controlled by data distribution and dense regions in each attribute. This converts the continuous attribute space into a discrete search space and defines the possible points that can be considered as subspace cluster centroids.

Nearest neighbour can take advantage of different distance functions, Euclidean distance (L_2 norm) being the most popular metric. Other metrics such as Minkowski metric (L_k norm), Manhattan distance (L_1 norm), Tanimoto metric (mostly used in taxonomy) and tangent distance are also less frequently used [32]. In general terms Euclidean distance tries to form dense hyper-spherical clusters; however, in the case of S-ESC this is only one of the three deciding factors to form clusters, and therefore clusters are not hyper-spherical merely because there is a nearest neighbour assignment involved.

The third deciding factor on cluster shapes as mentioned in this chapter is the tension between the primary and secondary objective functions. While the compactness function focuses to detect small and dense hyper-spherical clusters, connectivity tries to punish formation of clusters in which adjacent instances are grouped in different clusters and therefore is does not make any assumption on the shape of clusters.

4.4 Component 4: Variation Operators

Variation operators represent the mechanism for introducing new genetic material to both host and symbiont populations. As such they also have a role to play in the exploration–exploitation trade off. However, given that the same symbiont (CC) might well appear in multiple hosts (CS), care is necessary in order to ensure that (genotypic) modifications are limited to the offspring. To this end, an asexual model is assumed in which a parent is first cloned and variation is introduced only relative to the cloned child. Parent host individuals are selected by a tournament against three other hosts. Such a process is explicitly elitist. Two mutation operators are utilized in the inner loop of S-ESC: Single-Level Mutation (SLM) and Multi-Level Mutation (MLM) or Steps 5(a)iii and 5(a)vi, Algorithm 1.

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. In other words MLM fine-tunes a single symbiont within a host to achieve a better overall clustering solution. The motivation for adopting such a scheme is to have the combinatorial search for good symbiont combinations have a higher turn-over rate (as conducted at the host level) than the rate of introducing new symbiont content. That is to say, for a given set of symbionts it is necessary to look at more combinations at the host level in order to have a better chance of discovering the relevant symbiont specialization.

4.4.1 Single-Level Mutation

As shown in Figure 4.8 the single-level mutation (SLM) operator consists of 4 phases once a host individual is selected after a tournament selection against three other hosts. First, the selected host is cloned in Phase 1. To clone a host, a vector of the size of the parent's gene count is created and the links to symbionts within the parent host are duplicated in the cloned host. Further modification will be applied on the cloned host. In Phase 1 of Figure 4.8 the top-left green arrow shows how the new host is cloned from the parent host by duplicating all symbionts links within the new host.

The recently-cloned host potentially goes through three forms of mutation or atomic SLM operators:

1. Remove one (or more) random link(s) to the currently linked symbionts – relative to Figure 4.8; the last symbiont link is removed (marked by a cross sign);

2. Add one (or more) random new symbiont links – relative to Figure 4.8; a new symbiont link is inserted after the second symbiont link (marked by a plus sign) and;

3. Replace one (or more) current symbiont link(s) with the same number of random links to symbionts from the symbiont population – relative to Figure 4.8; the first symbiont link is replaced by a new symbiont link (marked by a star).

Naturally, the third form of mutation performs in one step what the first two forms might achieve together. Replacing a symbiont within a host as shown in Phase 4 of Figure 4.8 replaces the link to the current symbiont (marked by a cross) with a link to a different symbiont in the symbiont population (marked by a plus sign).

Once a host is selected by the tournament selection, each atomic variant of the SLM operator is applied iteratively with an annealed probability of application. Thus for each atomic variant, they are applied first with a probability of 1.0. Thereafter the probability of application decreases by an order of magnitude, i.e. $0.1^2, 0.1^3, ...$, until the test for applying the operator returns false. Thus, each child is *always* the result of calling each of the three atomic SLM variants at least once, and a variable

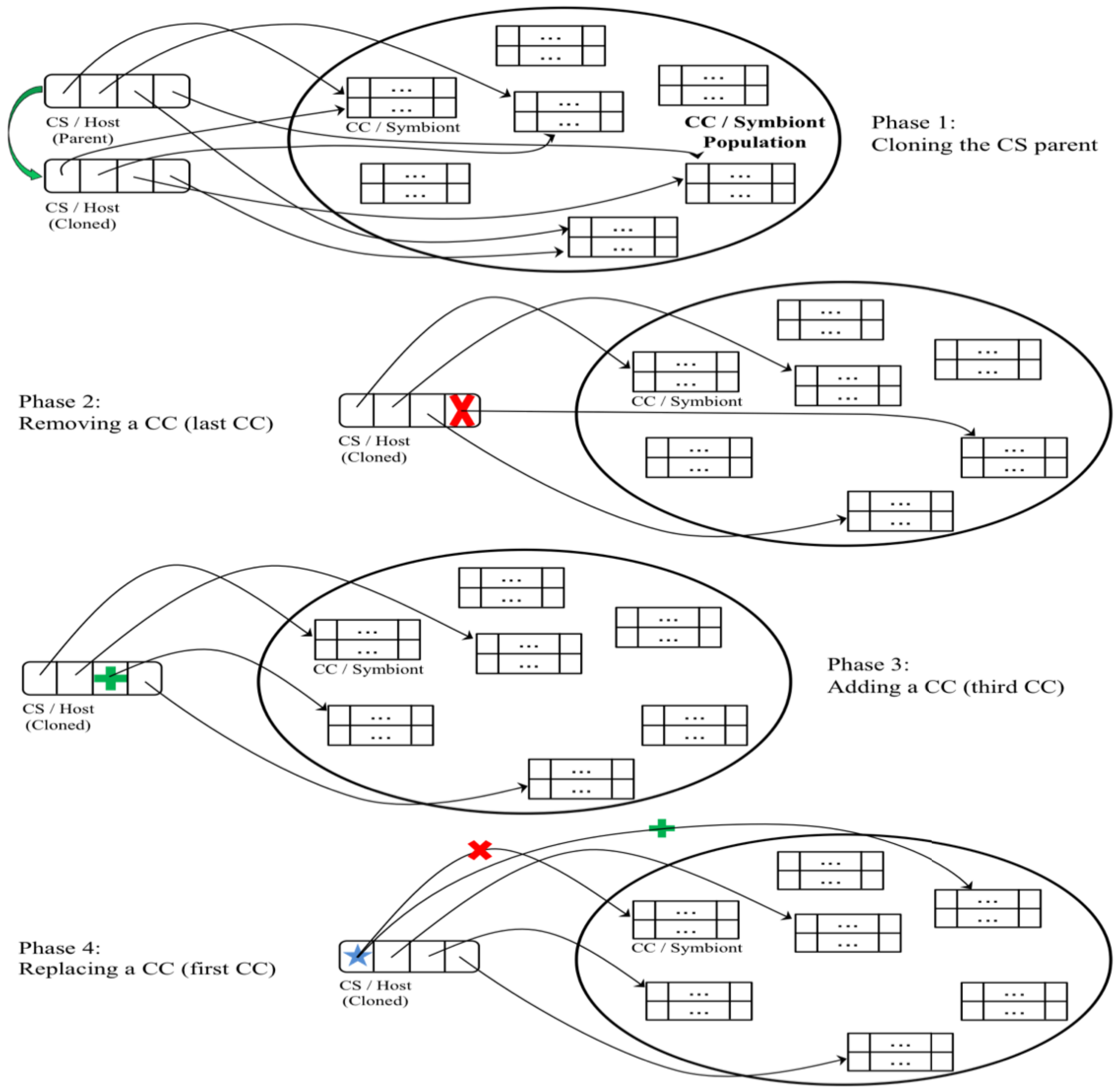

CS / Host (Parent)
CS / Host (Cloned)
CC / Symbiont
CC / Symbiont Population
Phase 1: Cloning the CS parent
Phase 2: Removing a CC (last CC)
Phase 3: Adding a CC (third CC)
Phase 4: Replacing a CC (first CC)

Figure 4.8: The four phases of the Single-Level Mutation (SLM) operator.

number of times thereafter. The process repeats until the test no longer returns true. There are also checks to enforce minimum/maximum sizes for the clusters per host.[6]

4.4.2 Multi-Level Mutation

Figure 4.9 illustrates the multi-level mutation (MLM) operator consisting of six phases. As per SLM, a parent host is selected first among four hosts through a tournament selection and cloned with both parent and clone retained in the host

[6]Values of 2 and 20 are assumed respectively in this work, where less than 2 would be a degenerate solution and the upper bound is arbitrary.

population (Phase 1). Unlike SLM, MLM acts on individuals from both host and symbiont populations, as described below:

1. With uniform probability a symbiont within the cloned host is selected – relative to Figure 4.9 (Phase 2); the second linked symbiont is selected (marked by a star);

2. The selected symbiont is cloned with both parent and cloned child retained in the symbiont population – relative to Figure 4.9 (Phase 3); the green arrow shows how a symbiont is cloned and kept within the symbiont population. While this cloning occurs the original link within the host which points to the parent symbiont also gets replaced by a link to the cloned symbiont. This is shown by removing the original link (marked by a cross) and creating a new link (marked by a plus);

3. The newly cloned symbiont goes through three mutation steps or atomic operators as listed below:

   (a) Remove one (or more) random attribute(s) and 1-*d* centroid pairs from the symbiont – relative to Figure 4.9 (Phase 4); the second attribute and the 1-*d* centroid pair is removed (marked by a cross);

   (b) Add one (or more) random attribute(s) and 1-*d* centroids to the symbiont – relative to Figure 4.9 (Phase 5); a new pair of attribute and a 1-*d* centroid is inserted before the first attribute and the 1-*d* centroid pair (marked by a plus); and

   (c) Replace one (or more) random attribute's 1-*d* centroid – relative to Figure 4.9 (Phase 6); the third 1-*d* centroid is modified (marked by a star). Care should be given since the number of 1-*d* centroids for different attributes are different, hence the random value here should be checked against the number of possible 1-*d* centroids for the attribute in question.

As per SLM, the selected parent host is guaranteed initially to see one application of each phase of variation and thereafter is subject to an order of magnitude decrease in the probability of application. Again, as with the case for SLM, there are checks

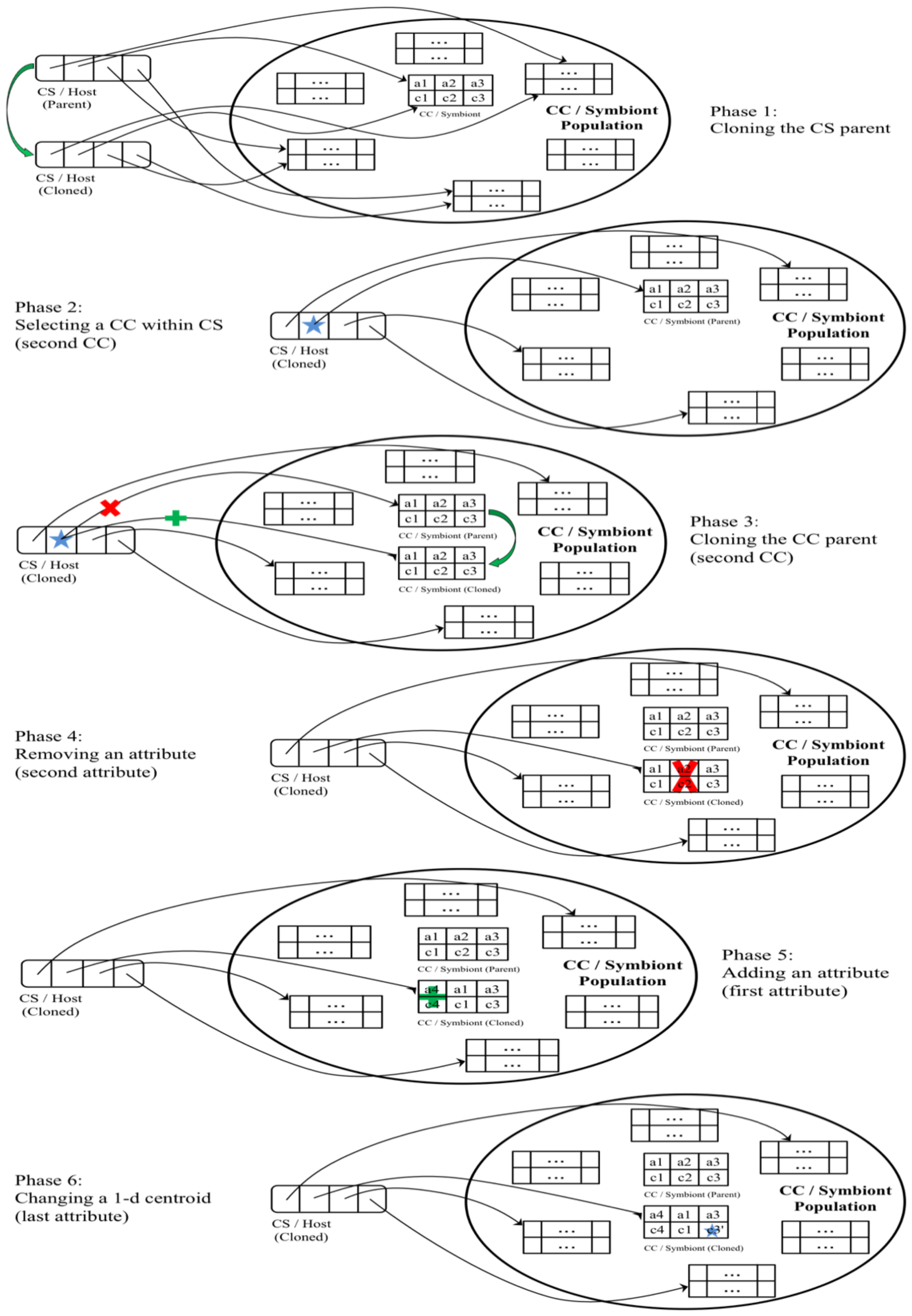

CS / Host (Parent)
CS / Host (Cloned)
a1 a2 a3
c1 c2 c3
CC / Symbiont
CC / Symbiont Population
Phase 1: Cloning the CS parent
Phase 2: Selecting a CC within CS (second CC)
CC / Symbiont (Parent)
Phase 3: Cloning the CC parent (second CC)
CC / Symbiont (Cloned)
Phase 4: Removing an attribute (second attribute)
Phase 5: Adding an attribute (first attribute)
a4 a1 a3
c4 c1 c3
Phase 6: Changing a 1-d centroid (last attribute)
c3'

Figure 4.9: The six phases of the Multiple-Level Mutation operator.

in MLM to make sure that the new cloned and mutated symbionts satisfy any size constraints set by the user.

Note that the decreasing probability of atomic mutation operators enables SLM and MLM to occasionally make big modifications to the cloned host or symbiont, respectively, reproducing more diverse offspring and adding to the diversity of host and symbiont populations. It will also be shown in Section 6 that S-ESC performs at its best when population diversity is maximized. However it should also be mentioned that the proposed mutation operators are not the only possible operators compatible with S-ESC and other forms of variation can be considered given that the same level of diversity is maintained.

One important property of S-ESC which needs to be restated is that S-ESC supports two different populations: a fixed size host population and a variable-sized symbiont population. As explained in this section each selected parent host goes through SLM and MLM after the tournament selection, care of the 100% probability for both operators. This process repeats H times with H being the host population size. Therefore, at the end of the breeding process at each generation, H new hosts are bred which, combined with the already existing H hosts in the population, make a pool of $2 \times H$ hosts. As mentioned earlier, the chosen evolutionary multi-objective optimization algorithm of our choice, NSGA-II, is an elitist algorithm. NSGA-II sorts the pool of all hosts based on their front number and crowding distance and picks the best H hosts to go through the next generation (Step 5d, Algorithm 1).

On the other hand, there is a population of symbionts with a dynamic size and an upper limit of $5 \times H$ symbionts. As with the hosts, H new symbiont are bred in each generation, although there is no elitist process to remove some of the symbionts. There is, however, a counter implemented in each symbiont which determines how many hosts are linking to it. Once a new host links to the symbiont this counter is increased by one and any time a link is removed from the symbiont this counter is decreased by one. Naturally, there is a check at the end of each generation to remove symbionts with zero links to make room for more symbionts in the next generation (Step 5e, Algorithm 1).

4.5 Pragmatics

The four components discussed above are the main building blocks of S-ESC [112]. However, two additional components are necessary for scaling S-ESC with respect to data set cardinality (subsampling) and facilitating the post-training identification of a single 'champion' solution from a pool of equally-fit Pareto non-dominated solutions (knee detection).

4.5.1 Sub-component A: Subsampling

Handl and Knowles [43] use connectivity as the second objective of their bi-objective approach. Being a full-space approach it can find the M nearest neighbours of each data point as a pre-processing step, reusing this information throughout the evolutionary process.

However, this is not an option in a subspace scenario simply because the nearest neighbours to each point might be different in different subspaces. For example, call the point for which neighbours are being sought x_i. First, the attribute support of the cluster to which x_i belongs is called subspace S_i. Then the distances between all the data set points and x_i are calculated over the attributes of S_i, and normalized by the number of attributes in S_i. These distances are sorted and the M nearest instances to x_i are selected. M is defined as $max(0.01 \times N, 10)$. Also note that same process should be repeated anew for each point in each generation because the attribute support of the cluster to which they belong has possibly changed following the SLM and MLM operators.

Based on the definition of connectivity, M neighbours of each data instance should be identified, which requires N comparisons for each data point. The time complexity of this process is $O(N^2)$ which makes it infeasible for large-scale data sets. To address this issue subsampling (or simply sampling) is introduced. The idea is to calculate the objective values using a subset of data set points rather than the whole data set. By reducing the number of active points per objective calculation from N to P the time complexity drops to $O(P \times N)$ where $P \ll N$. These P points are selected with uniform probability, through *Random Subset Selection* (RSS) and the selected subset is hereafter called the *active set* or *point population*, as shown in Figure 4.10. The

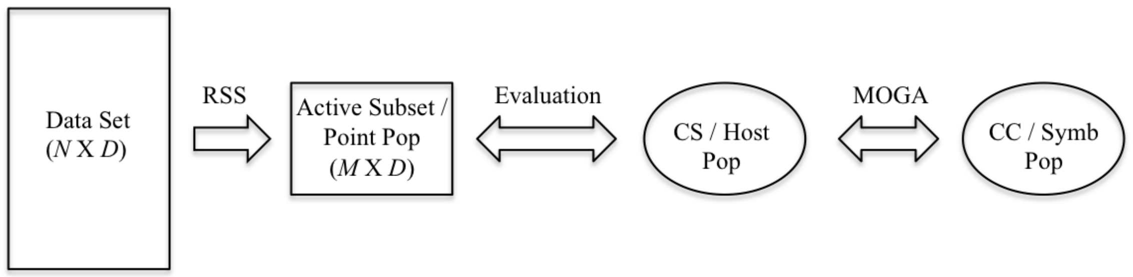


Figure 4.10: Subsampling process used in S-ESC.

active set is refreshed at each generation by resampling all P data points (Step 5f, Algorithm 1). Naturally, the fitness evaluation is now decoupled from the cardinality of the original data set, at the possible expense of the accuracy associated with any single evaluation.

4.5.2 Sub-component B: Knee Detection

EMO methods based on Pareto dominance are expected generally to provide a diverse and distributed set of non-dominated solutions. All these (Pareto-optimal) solutions are essentially equally important. Thus, after reaching some computational limit (typically a generation count) it is necessary to select a single ‘champion’ solution for further post-training quality evaluation (Step 6, Algorithm 1). With this in mind the knee detection approach will be adopted in this work.

Knee detection represents one of the more well-established heuristics for identifying a single representative champion from the non-dominated set [99, 93]. A knee on a Pareto front is where a small change in one objective makes a significant change in the others.

The method proposed by Schütze *et al.* [99] has been adopted here 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 is evaluated; and 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 4.11 illustrates the process relative to a set of Pareto front individuals. The tail individuals p_1 and p_4 establish the diagonal and the individual with the greatest distance from the diagonal, individual x, denotes

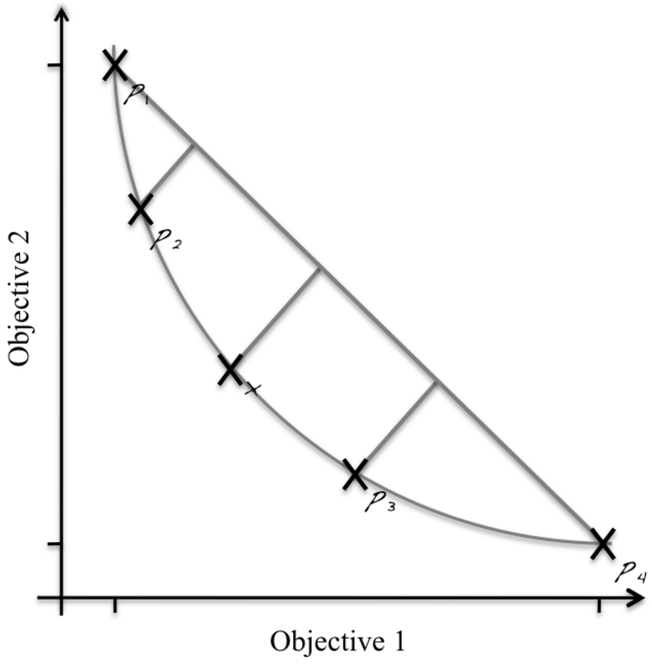


Figure 4.11: A sample Pareto front with champion identification through knee detection. The Pareto set $\{p_1, p_2, p_x, p_3, p_4\}$ denotes the front of candidate solutions from which $\{p_1, p_4\}$ are the tails. The champion solution in this case is p_x.

the champion.

Naturally, other heuristics could be assumed, including robustness and/or sensitivity. Note, however, 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.

Chapter 5

Evaluation Methodology

The S-ESC code along with the benchmarking data sets utilized in this work are publicly available at http://web.cs.dal.ca/˜mheywood/Code/S-ESC/.

5.1 Benchmarking Data

In order to evaluate the effectiveness of subspace clustering algorithms, data sets with a specific structure are required. Generally, this is a problem in the case of 'real-world' data sets. Specifically, it is not possible to define with any certainty how many clusters exist or quantify the nature of the clusters involved in the data set unless the process of creating the data itself is known. As a consequence, researchers frequently design and generate artificial data sets of their own to show the various properties of subspace algorithms [87, 88, 80, 83].

Two different sets of synthetic data sets will be utilized in this thesis. The first to be employed are the data sets provided by Moise *et al.* in their benchmarking survey of more than 10 different (axis-parallel) subspace clustering algorithms on more than 50 data sets [80], hereafter referred to as the *incremental* benchmark or Moise data sets. The underlying theme of the benchmarking approach adopted by Moise *et al.* is to test the sensitivity of subspace clustering algorithms to one property at a time. The goal is to identify which factors different subspace clustering algorithms are most sensitive to. Section 5.1.1 summarizes the properties of the incremental benchmark.

Although very thorough, the incremental benchmark does have two limitations. Firstly, the data dimensionality and cardinality is relatively low in most cases, e.g. dimensionality ≤ 50 and cardinality ≤ 300. Secondly, all the clusters within a data set use the same data distribution for all clusters, and no data set has clusters with different data distributions. With this in mind a second *large-scale* benchmarking data set[1] will be introduced consisting of 5 data sets with larger overall dimensionality,

[1]That is to say, 'large-scale' relative to the tasks defined by the incremental benchmarks of Moise.

cardinality and multiple cluster distribution types within a single data set. As a consequence, the large-scale data sets will be able to investigate the appearance of multiple cluster properties simultaneously. Section 5.1.2 summarizes the properties of the large-scale benchmark.

5.1.1 Incremental Benchmarking Data Sets

A total of 9 properties are varied independently in order to construct the incremental benchmark developed by Moise *et al.*:

1. Type of distribution defining a cluster, using either Gaussian or Uniform distributions;

2. Cluster support, or defining clusters with an equal number versus varying numbers of relevant attributes;

3. Relevant attributes per cluster, or varying the average number of attributes representing a subspace cluster;

4. Data set dimensionality, or varying the overall number of attributes (D). This experiment defines data sets with increasing overall attribute count from 20 to 100 in 5 steps;

5. Data set cardinality or the overall number of instances (N). Under this experiment the overall number of data instances per data set increases from 80 to 1650 in 5 steps.;

6. Cluster count, or varying the overall number of subspace clusters (k). Specifically, the number of embedded clusters in a data set is increased from 2 to 5 in 4 steps;

7. Cluster extent, or varying the range over which subspace cluster attributes vary ($Extent$). This defines data sets in which the range/extent of the uniform distribution from which instances are drawn is increased from 0.1 to 0.4 in 4 steps. Given that attributes *not* associated with a subspace cluster (for any data set) are defined by a uniform distribution sampled from the interval $[0, 1]$, then

increasing the range for attributes associated with subspace attributes makes them more difficult to identify;

8. Cluster overlap, or varying the number of shared attributes between different subspace clusters ($Overlap$). In this experiment, the overlapping range of relevant attributes from different clusters increases from 0 to 0.3 in 4 steps;

9. Instances per cluster, or varying the average number of data instances used to define each cluster ($ClusterSize$). This experiment results in a total of 4 data sets.

Table 5.1 summarizes how the different data sets were designed to assess the above 9 subspace clustering properties, resulting in 10 'experiments' and a total of 54 data sets. Labels are also declared for identifying each set used in the 10 experiments. The first 3 properties are merged into 4 data sets: GE, GD, UE and UD in Table 5.1:

- The GE experiment represents data sets with a 'Gaussian' data distribution and an 'Equal' number of relevant attributes per cluster. There are 7 data sets in this experiment with an average (relevant) cluster dimensionality ranging from 2 to 20;

- The GD experiment represents data sets with a 'Gaussian' distribution in which the number of relevant attributes for each cluster is 'Different.' As with GE there are 7 data sets in this experiment with an average (relevant) cluster dimensionality ranging from 2 to 20;

- The UE and UD represent the corresponding data sets with 'Uniform' distribution with either equal or different attribute counts per cluster.

Experiments GE and UE consist of 7 data sets in which the attribute support of each cluster increases from 2 relevant attributes per cluster (data set 1) to 20 relevant attributes per cluster (data set 7). However, the number of relevant attributes is equal within each subspace cluster from the same data set. By contrast, in the GD and UD experiments, the *average* attribute support per cluster increases from 2 relevant attributes per cluster (data set 1) to 20 relevant attributes per cluster (data set 7), with different subspace clusters in the same data set retaining different attribute

Table 5.1: A summary of the properties of the Incremental data sets. A parenthesis after the experiment identifier denotes the number of variants for the data set within each experiment. GE denotes a Gaussian distribution with an Equal number of attributes per cluster; GD denotes a Gaussian distribution with a Different number of attributes per cluster; the same applies to UE and UD but in terms of the Uniform distribution. 'Irr.' stands for Irrelevant, 'Dim.' for dimensionality and 'Inst.' for Instance.

Data Set	Data Distr.	Cluster Shape	*D* #	*N* #	*k* #	Irr. Attr #	Irr. Attr %	Min Dim #	Max Dim #	Min Inst %	Max Inst %
GE (7)	Gaussian	Spherical	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)	Gaussian	Spherical	50	240	5	40,30 20,10 0,0,0	80,60 40,20 0,0,0	2,2,3 5,6 10,10	2,6,11 12,14 20,24	17	25
UE (7)	Uniform	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)	Uniform	Square	50	240	5	40,30 20,10 0,0,0	80,60 40,20 0,0,0	2,2,3 5,6 10,10	2,6,11 12,14 20,24	17	25
D (5)	Uniform	Square	20,35 50,75 100	240	5	0,15 30,55 80	0,33 60,73 80	4	4	17	25
N (5)	Uniform	Square	50	80,240 400,820 1650	5	30	60	4	4	17	25
K (4)	Uniform	Square	50	250	2,3 4,5	42,38 34,30	84,76 68,60	4	4	20	50
Extent (4)	Uniform	Rectangular	50	240	5	30	60	4	4	17	25
Overlap (4)	Uniform	Square	50	250	2	46	92	4	4	17	50
Cluster Size (4)	Uniform	Square	50	145,200 240,265	5	30	60	4	4	14	25

counts. Note that the overall dimensionality of all data sets in these 4 experiments is always 50 and the subspace cluster count is always 5. Thus, where the attribute support of each cluster is 20, each attribute is relevant in 2 clusters.

Note that the incremental benchmark is used both with and without the inclusion of 'outliers.' Outliers are data instances for which data values represent noise for all attributes. Since S-ESC and all but one of the comparator methods do not filter explicitly for outliers, the outliers in all data sets are removed before commencing the evaluation.[2] Naturally, the resulting outlier-free data sets still retain the noise terms associated with attributes not associated with defining a subspace cluster. A

[2]Outliers are labelled explicitly as an extra cluster in the incremental data sets and are straightforward to remove.

separate set of experiments will be conducted to assess the impact of including the pure outlier data.

5.1.2 Large-scale Benchmarking Data Sets

The Incremental benchmarking data sets popularized by Moise *et al.* used above cover a wide range of properties. However, as noted earlier in this chapter, there are some basic shortcomings, as noted below:

1. The dimensionality and cardinality remains low;
2. The cluster embedding assumes only one data distribution per data set with aspect ratios which are not particularly diverse. In other words, the clusters built into each data set use either a Gaussian distribution or a uniform distribution, but never both;
3. Cluster properties are only introduced independently, but multiple factors might be expected to appear simultaneously in the data. Thus, the large-scale data sets clusters might possess overlapping attributes, a variable attribute support per cluster and a high percentage of noisy attributes simultaneously.

A further 5 data sets are designed here to cover the shortcomings of incremental data sets. These data sets (called Large-Scale data sets) range between 50 to 1000 dimensions, 1000 to 4000 data instances and with up to 10 clusters per data set (Table 5.2). Two data sets from the large-scale set (200D and 800D) have both low- (10D) and high-dimensional (50D in the case of the former and 100D in the case of the latter data set) clusters within the same data set. There is also one data set (800D) in which clusters with a Gaussian data distribution and clusters with a uniform distribution are included. It is also worth noting that different data distributions with different parameterizations are used to generate the different clusters of a data set (in the case of the 800D data set).

Attribute support for different clusters within a data set ranges between 10 and 100. In the case of the '1000D' data set 90% of the attributes are irrelevant, leaving only 10% of attributes actually relevant to the clustering task. Data sets are highly unbalanced, meaning that data instances are not split equally between clusters. The

Table 5.2: A summary of the properties of the large-scale data sets.

Data Set	Data Distr.	Clus. Shape	D #	N #	k #	Irr. Attr #	Irr. Attr %	Min Dim #	Max Dim #	Min Inst %	Max Inst %
50D4C	Gaus.	Sph.	50	1289	4	10	20%	10	10	8%	43%
200D5C	Uni.	Rect.	200	2000	5	50	25%	10	50	10%	25%
500D4C	Gaus.	Ellip.	500	1286	4	100	20%	100	100	13%	32%
800D10C	Mixed	Mixed	800	3814	10	240	30%	10	100	4%	16%
100D10C	Gaus.	Sph.	1000	2729	10	900	90%	10	10	3%	20%

largest cluster of the '50D' data set covers 43% of the data set while the smallest cluster covers only 8% of the instances. Cluster attribute support is not balanced either, meaning that different clusters in a data set have different attribute support counts. Under the '800D' data set the largest cluster support is 100 attributes while the smallest cluster support is only 10 attributes. These data sets are available for research purposes.[3]

The process used to create the large-scale data set with subspace clusters with an axis-parallel structure assumes the following form. Figure 5.1 shows a sample data set with 3 subspace clusters embedded in it. The cluster points in relevant subspaces (or the values inside the coloured boxes) are sampled from different distributions with generated randomly 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 data set: [0,1] in this case. Data points with relevant attributes and those with irrelevant attributes are within the same range [0,1] and therefore overlap non-trivially.

5.1.3 Data Standardization

Clustering algorithms generally assume some form of 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 [9]. Here a

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

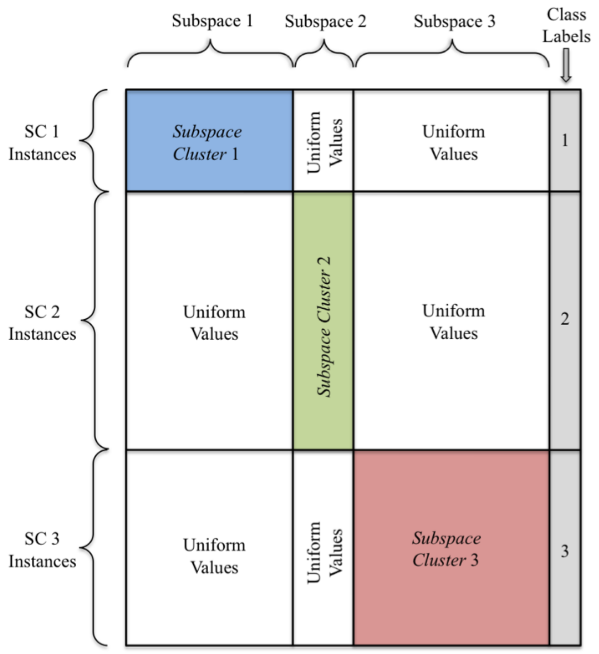


Figure 5.1: Sample visualization of a data set with subspaces. The ‘Class Labels’ indicates cluster membership and is used only for post-training evaluation along with classification-style performance metrics such as F-measure.

linear standardization of data sets to the unit interval $[0, 1]$ per attribute is assumed, where this is common to all the algorithms benchmarked.

5.1.4 Post-training Performance Measures

Several post-training performance measures can be employed for any given clustering task [88, 49]. Each measure has its bias and therefore comes with its own advantages and drawbacks. Therefore, each performance measure may be more or less suitable to a given analysis or application scenario. For example some measures are more suitable and intuitive for gene clustering in bioinformatics [52], while other measures are more appropriate and insightful for text clustering and document categorization [54].

S-ESC and its comparators are evaluated with respect to three performance measures: *accuracy* as measured through micro F-measure, *simplicity* as measured by the number of unique attributes and *CPU time* where this is the wall-clock time on a common computing platform to complete training.

Accuracy. Given that it is known which data point is a member of which cluster, each data instance has a 'label,' albeit one is not used for training. Following the practice of Moise *et al.*, let the labels partition the data into the *input* clusters or ground truth (a.k.a. *classes* in a classification task). The clusters identified by the algorithm denote the *output* clusters. For each output cluster i, the input cluster j_i with which it shares the largest number of data instances is identified. This will enable the construction of a confusion matrix characterizing exactly how many of the 'input' clusters have been identified. Given that multiple runs will be necessary, a way is needed for summarizing the content of the confusion matrix, preferably leading to a simple visual interpretation of the results against which suitable statistical tests can be applied. In this respect, an approach similar to Moise *et al.* will be used once again. Specifically, the balanced F-measure metric will be assumed. The F-measure combines precision and recall under a weighted harmonic mean; in this case an equal weighting between precision and recall. Precision and recall measure the count of correctly labelled data instances from the confusion matrix rows and columns. In the case of a binary task:

$$Prec(i) = \frac{TP}{TP + FP} \tag{5.1}$$

$$Rec(i) = \frac{TP}{TP + FN} \tag{5.2}$$

where *TP*, *FP* and *FN* are respectively the true positives, false positives and false negatives respectively.

Under a multi-class setting precision and recall are estimated for each column and row and averaged, as, for example, in the case of precision: $Prec = \frac{1}{C}\sum Prec(i)$. The resulting definition for the balanced F-measure is:

$$\text{F-measure} = \frac{2 \times Prec \times Rec}{Prec + Rec} \tag{5.3}$$

The above formulation for balanced F-measure has one underlying assumption, sometimes distinguished among the concept of micro and macro (balanced) F-measures. A 'macro' F-measure rewards cluster purity alone. Thus, as long as an output cluster is associated with data instances from the same input cluster, it will return a perfect score. This does not preclude the 'splitting' of an input cluster between *multiple* output clusters, as long as the output clusters do not mix data instances from multiple input clusters. In effect one can have more output clusters than input clusters as long as they are pure. Conversely, a 'micro' F-measure also requires the number of output clusters to match the number of input clusters. The essential difference lies in how the original confusion matrix is constructed.

Tables 5.3 and 5.4 illustrate the difference between micro versus macro F-measures for a sample clustering solution. Table 5.3 shows the Micro Confusion Matrix of an 8-cluster solution (i.e. 8 output clusters) for a 5-class data (i.e. 5 input clusters). *Cluster A* through *Cluster H* identify the 8 clusters found in the solution. The 'C2C Map' column determines the class label in the majority of instances in each cluster and maps each output cluster to an input cluster. For example, *Cluster A* has 1 instance from *Class 2*, 400 instances from *Class 4* and 1 instance from *Class 5*; therefore, it is assigned the class label for the majority of its instances, i.e. *Class 4*.

Table 5.3: Micro Confusion Matrix

Confusion Matrix	**Class 1**	**Class 2**	**Class 3**	**Class 4**	**Class 5**	**C2C Map**	**F**
Cluster A	0	1	0	400	1	4	0.9975
Cluster B	0	172	0	0	0	2	0.9247
Cluster C	0	0	273	0	0	3	0.9529
Cluster D	0	0	0	0	381	5	0.8649
Cluster E	600	0	1	0	0	1	0.9992
Cluster F	0	27	0	0	0	2	0.2379
Cluster G	0	0	26	0	0	3	0.1595
Cluster H	0	0	0	0	118	5	0.3819
F-measure							**0.6898**

Figure 5.2 is a circos plot[4] [65] visualizing the mapping between output clusters and input clusters. Starting from 12 o'clock and counter-clockwise to 6 o'clock all 5 classes (input clusters) are shown as 5 segments and labelled $Class1$ to $Class5$. Similarly from 6 o'clock to 12 o'clock counter-clockwise all 8 output clusters are shown as 8 segments and labelled $ClusterA$ to $ClusterH$. Ribbons connecting each pair of output clusters and input clusters define what percentage (inner ring values) and what number (outer ring values) of instances from each input cluster is mapped to an output cluster and vice versa. For example all 600 instances of $Class1$ are

[4]Circos is publicly available at: http://circos.ca/

mapped to $ClusterE$ whereas $Class2$ is split mainly between $ClusterB$ (172 out of 200, i.e. 86%) and $ClusterF$ (27 out of 200, i.e. 13.5%). Also, there is one instance from $Class2$ mapped to $ClusterA$ which is shown as a very narrow ribbon between $Class2$ and $ClusterA$.

Once all output clusters are mapped into their corresponding input cluster (class), clusters with similar class labels (C2C Map) are merged and form 'Macro Confusion Matrix' (shown in Table 5.4). The C2C Map column of Table 5.3 shows that there are 3 pairs of output clusters which share the same class label. $ClusterB$ and $ClusterF$ share class label $Class2$, $ClusterC$ and $ClusterG$ share $Class3$ and $ClusterD$ and $ClusterH$ share $Class5$. Note the changes of cluster names.

Table 5.4: Macro Confusion Matrix

Confusion Matrix	**Class 1**	**Class 2**	**Class 3**	**Class 4**	**Class 5**	**C2C Map**	**F**
Cluster 1(A)	0	1	0	400	1	4	0.9975
Cluster 2(B,F)	0	199	0	0	0	2	0.9975
Cluster 3(C,G)	0	0	299	0	0	3	09983
Cluster 4(D,H)	0	0	0	0	499	5	0.9990
Cluster 5(E)	600	0	1	0	0	1	0.9992
F-measure							**0.9983**

Figure 5.3 shows the circos plot of the same sample clustering solution once the output clusters sharing the same class instances are merged. The number of output clusters has been reduced from 8 to 5 (and renamed from $ClusterA-H$ to $Cluster1-5$). Three smaller output clusters are now merged with larger output clusters and an obvious mapping between input and output clusters is formed now. This is also reflected in Table 5.4.

F-measures for all 8 micro clusters as well as all 5 macro cluster are shown in Tables 5.3 and 5.4 respectively. The average micro F-measure – before merging – is only 0.6898, whereas the average macro F-measure – after merging – is 0.9983, a much higher value compared with the micro F-measure. This example shows that regardless of the optimal number of clusters in a data set, as long as the output clusters are pure the macro F-measure has a high (even close to unity) value. Conversely, the micro F-measure is penalized by under- or over-estimating the number of optimal clusters.

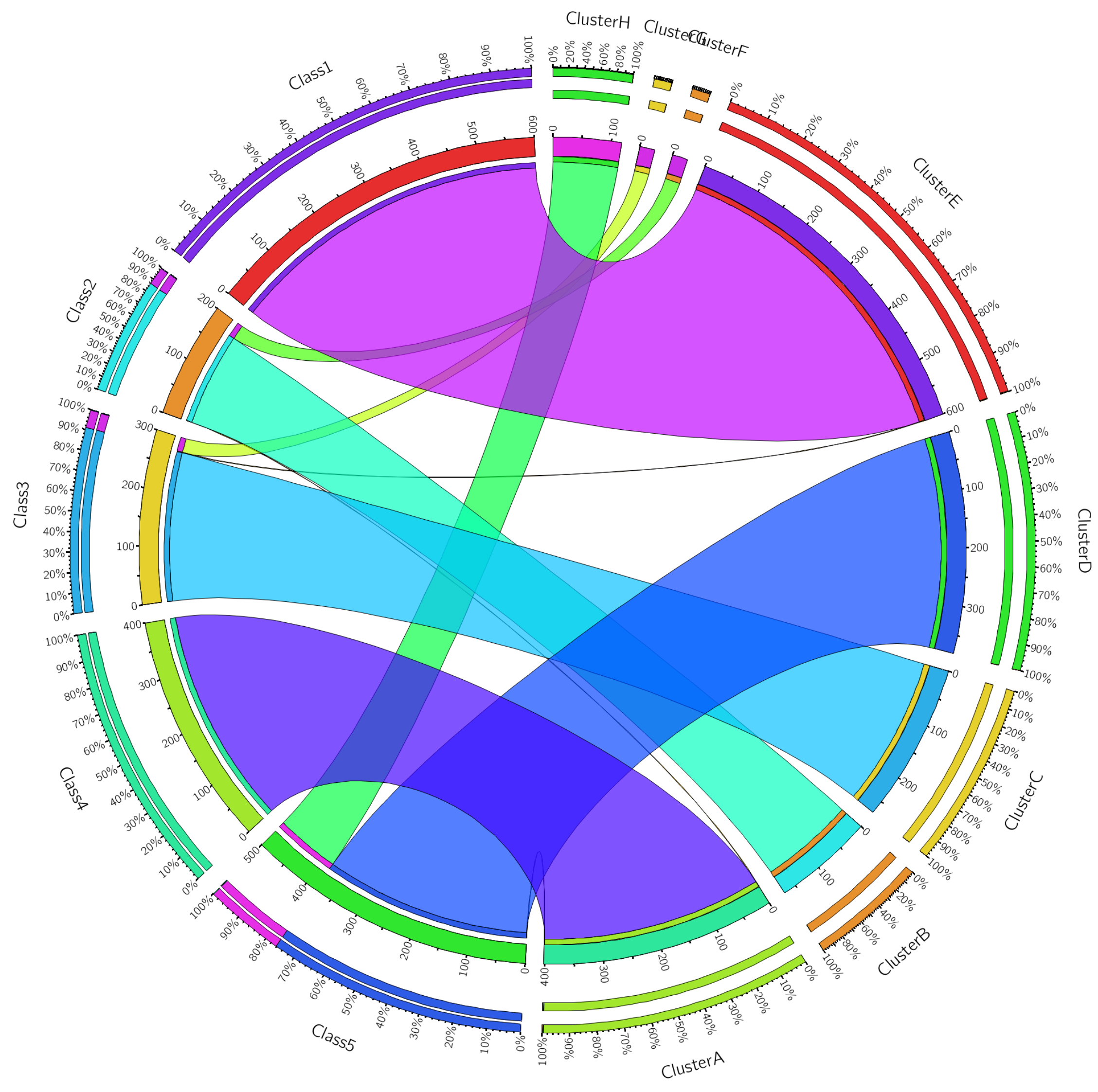


Figure 5.2: A sample visualization of an 8-cluster S-ESC solution to a 5-class data set 'before' merging output clusters.

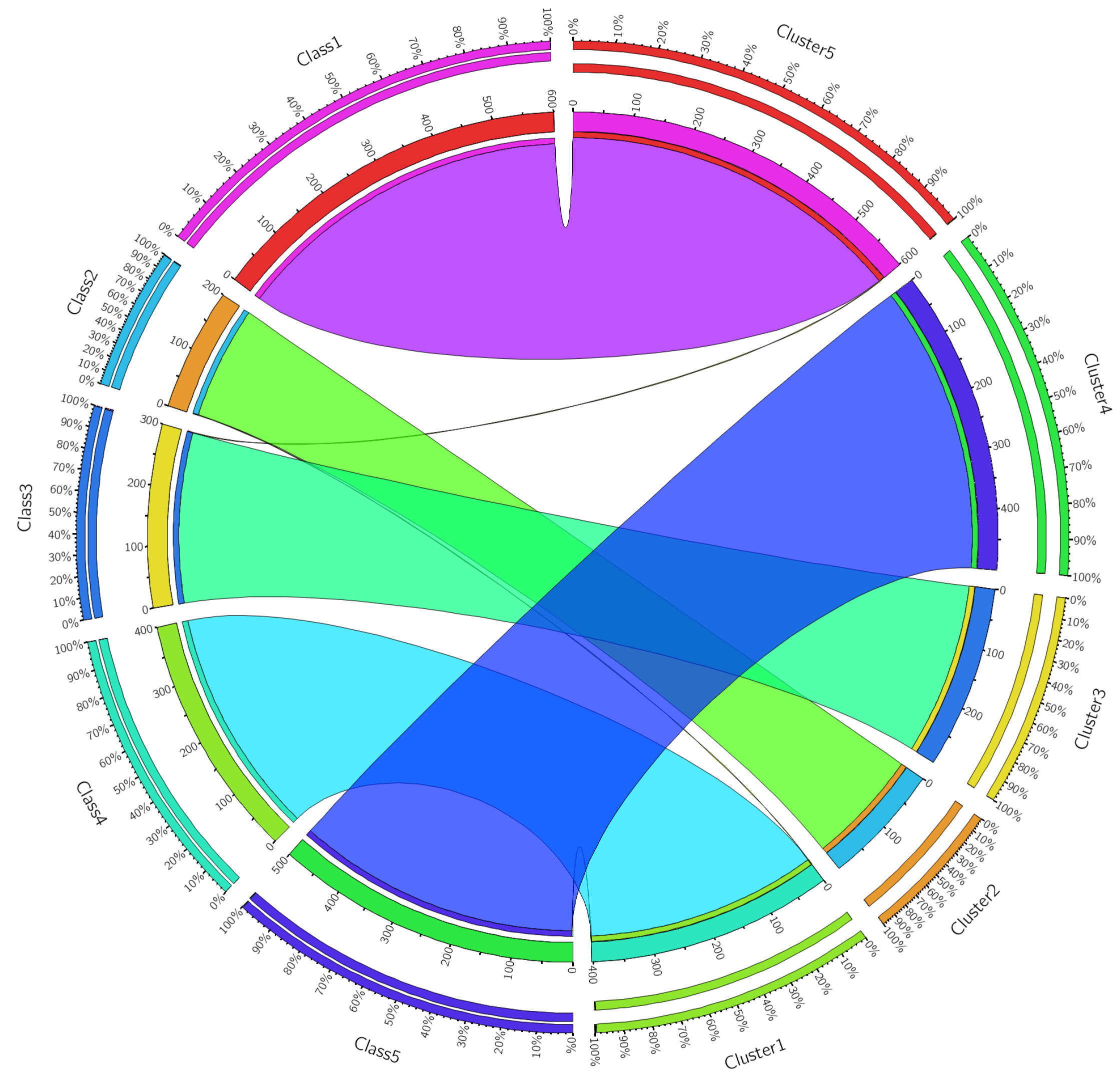


Figure 5.3: A visualization of the same S-ESC solution ‘after’ merging output clusters.

The micro F-measures for S-ESC and its comparators across all experiments are documented in this work.

Rather than misclassification (accuracy), which requires some form of ground truth (e.g. in form of class labels), a clustering method can be evaluated through visualization. This is particularly applicable to S-ESC because of the parsimonious property of S-ESC. However since different clusters in S-ESC support different attribute subsets, they need to be visualized in different subspaces and therefore clusters are observed individually (unless two or more clusters share the same subspace). Another method of evaluating a subspace clustering algorithm is to investigate the attribute support of each cluster. Similar to visualization, this method is also a subjective method and relies on domain expertise.

Simplicity. The second performance metric – we evaluate our solutions and compare them against comparator methods – is simplicity. Simplicity of a solution is determined by the total number of attributes indexed by it, counting attributes with multiple occurrences only once. Obviously the fewer attributes utilized, the simpler the solution is, where this is assumed to facilitate greater user understandability.

CPU time. CPU time reflects the computational cost in identifying solutions. Such a metric is likely to be a function of the quality of an implementation. Benchmarking will assume comparator algorithms taken from the WEKA open source library. As such the computational assessment reflects the quality of a widely available 'toolbox' of subspace clustering algorithms which are likely to be used in practice. Basing computational overhead on the wall-clock time on a common computing platform implies that both time and memory efficiency are being measured implicitly. In each case, care is taken to ensure that only single runs are performed, thus there is no inter process competition for cache resources.

5.2 Comparator Methods

To compare S-ESC against state-of-the-art algorithms using common benchmarking conditions comparator algorithms are chosen from popular full-dimensional and subspace clustering algorithms. The EM algorithm is assumed as the representative of the full-dimensional clustering algorithms [30, 77]. The properties which make it particularly appealing include the wide utility of the algorithm, its robustness and the relatively low number of parameters – (e.g. a user need not define the number of clusters required a priori. Moreover, as noted by the survey of Kriegel *et al.* [63], soft projected clustering algorithms do not identify subspace clusters, thus any 'subspace' clustering algorithm taking the soft projected clustering approach is not returning 'true' subspace clusters. The EM algorithm is taken to represent the principle instance of such soft projected clustering algorithms.

Moise *et al.* introduce STATPC [78] as the preferred subspace clustering method following a comparison with a large number of other subspace clustering algorithms [80]; whereas Muller *et al.* recommend MINECLUS [122, 123] and PROCLUS [1] as the most consistent methods among subspace clustering algorithms [83]. Therefore, the benchmarking study reported here compares S-ESC against all 4 methods on 59 data sets (54 data sets from the 'incremental' data set (Section 5.1.1) and the additional 5 'large-scale' data sets of Section 5.2) and evaluates them with respect to cluster accuracy, solution simplicity and CPU time. Refer to Section 3 for a summary of the algorithmic properties of the three comparator algorithms – PROCLUS, MINECLUS and STATPC.

5.3 Parameterization

5.3.1 S-ESC

The parameterization of S-ESC remains fixed and intact for all experiments on all data sets. The host population size (H_{size}) is set to 100. The symbiont population is populated initially with the same number of symbionts as hosts; however, this population is allowed to expand up to 5 times its initial size. RSS picks P_{size} points randomly from data set at each generation. This is set to 100 for all data sets unless the data set size is smaller than 100, in which case RSS is deactivated and all points are considered for evaluating host individuals. Some experiments were conducted using different values for the RSS point population size to understand the effect of this parameter on S-ESC performance. The results are presented in Appendix A.

The minimum and maximum number of allowed attributes in a symbiont is set to 2 and 20 respectively ($minAttributes$ and $maxAttributes$) with the same values defined for the minimum and maximum number of clusters in a host ($minSymbionts$

and *maxSymbionts*). Probabilities of atomic actions within single-level and multi-level mutation operators start with 1.0, and decrease to 0.1, 0.01, ... for the second, third, and further rounds of application (*psd*, *psa* and *pss*). The evolutionary loop is repeated for 1000 generations (G_{max}) at each run.

The same parameter setting is used to run S-ESC 50 times on each data set, using different seed values for the stochastic initialization of host/symbiont populations and RSS content. The knee individual on Pareto front represents the champion from each run. The results are reported from 50 knee solutions, each chosen from a run. S-ESC takes advantage of EM [30, 77] for 1-d density estimations (grid construction) where this takes the form of the implementation from WEKA [115].

5.3.2 Comparator Algorithm Parameterization

In all cases, the WEKA-compatible *OpenSubspace*[5] implementations for MINECLUS, PROCLUS and STATPC as developed by Muller *et al.* will be assumed [83]. 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.

The parameter settings established by the studies of Moise *et al.* [80, 78] will be assumed throughout for the incremental data sets (Table 5.1). In the case of MINECLUS and PROCLUS these settings appeared to be relatively robust, whereas for STATPC parameter sweeps were necessary. The specific parameterization procedure adopted is summarized below:

- MINECLUS. $\alpha = 0.1$, $\beta = 0.25$ and $\omega = 0.3$, whereas k was enumerated over the range [2, 20] – the same range as specified for 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 the 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

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

the study. The algorithm is run 10 times with different seed numbers. Variance in result distribution should be significantly lower than the other algorithms since much more information is given a priori.

- STATPC. As recommended by the authors, the following parameterization was used first: $\alpha_0 = 10^{-10}$, $\alpha_K = \alpha_H = 10^{-3}$ [78]. However, considerable sensitivity was observed. Hence it was 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.[6] All STATPC distributions make use of such a process, thus biasing results in favour of STATPC.

- EM. As implemented in WEKA, EM conducts cross validation thus resolving the cluster count k automatically. Per data set tuning is still necessary for target standard deviation and the iteration limit. Having established specific values for these two parameters the EM algorithm is run with 10 different seed numbers and the distribution of 10 results is presented.

In the case of the large-scale data sets (Table 5.2), parameter sweeps were utilized for MINECLUS. Specifically, the α, β and ω parameters of MINECLUS were determined first and then 19 runs were performed with $k \in [2, 20]$ using the predetermined parameter setting. Therefore, results with unknown k is reported for the case of MINECLUS. PROCLUS 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 data sets were assumed. This was necessary as the larger data sets precluded the use of parameter sweeps for computational reasons. Similarly, for EM the best parameter setting from the incremental data sets was selected and EM is run 10 times with different seed numbers.

[6]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. Only the top 20 results are reported (according to F-measure).

5.4 Flat Evolutionary Subspace Clustering (F-ESC)

For the sake of completeness it is necessary to compare S-ESC against other evolutionary computation-based subspace clustering algorithms, however none of the comparator approaches utilized in this work uses evolutionary computation. Moreover, the code for the few evolutionary subspace clustering algorithms described in Section 2 is not available publicly. Taking the basic ideas from the S-ESC algorithm, a genetic algorithm was implemented to perform the subspace clustering task. The proposed method is a simplified version of the S-ESC algorithm where the symbiotic (dual-population) representation is replaced by a single 'flat' (single population) representation of subspace clusters. In other words, each individual is responsible for encoding all aspects of a clustering solution (including all cluster centroids within the clustering solution) while assuming the same 1-d attribute-specific pre-processing activity (Component 1 of the S-ESC framework, Section 4) and the evolutionary bi-objective formulation of fitness (cf. MOCK [43]). Hereafter the resulting algorithm will be referred to as 'flat' Evolutionary Subspace Clustering (F-ESC.)

The main difference between F-ESC and S-ESC is that F-ESC replaces the two-level hierarchical representation with a single-level flat representation, thereby eliminating the symbiotic relationship. Also, since the two-level representation is removed the single-level mutation operator is no longer practical and is replaced by a crossover operator performing essentially the same task. Apart from the symbiotic process being removed and the single level mutation operator being replaced with a crossover operator, everything else is shared between the two variants of the ESC algorithm. The other components of the algorithm (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 of 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 the crossover operator. The results of applying F-ESC to both of the synthetic data sets of Tables 5.1 and 5.2 are compared with those of S-ESC in Section 6.

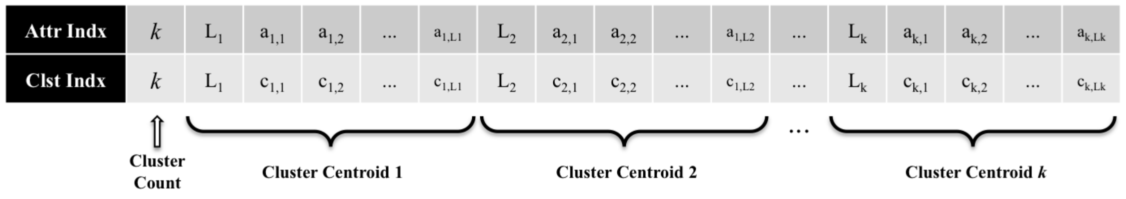

Attr Indx
k L_1 $a_{1,1}$ $a_{1,2}$ ... $a_{1,L1}$ L_2 $a_{2,1}$ $a_{2,2}$ ... $a_{1,L2}$ ... L_k $a_{k,1}$ $a_{k,2}$ ... $a_{k,Lk}$
Clst Indx
k L_1 $c_{1,1}$ $c_{1,2}$ ... $c_{1,L1}$ L_2 $c_{2,1}$ $c_{2,2}$ ... $c_{1,L2}$ ... L_k $c_{k,1}$ $c_{k,2}$ ... $c_{k,Lk}$
Cluster Count
Cluster Centroid 1
Cluster Centroid 2
...
Cluster Centroid k

Figure 5.4: 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. AS with the S-ESC symbiont, an F-ESC individual is composed of two integer strings, one for attributes, and the one for 1-d centroids.

5.4.1 Representation

The representation and encoding in F-ESC is much simpler and more straightforward than S-ESC. The complexity of the representation is alleviated by compressing the two-level representation of S-ESC into a flat single-level representation as shown in Figure 5.4. Here, each individual encodes a clustering solution including the cluster count (k) as well as 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, the F-ESC chromosome has two interconnected 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 (or cluster count k). Then k cluster centroids are encoded, each similar to a S-ESC symbiont representation, with a minor change. The first integer of each cluster centroid encodes the number of attributes the cluster centroid supports (or attribute count) in both attribute and 1-d centroid strings. In other words L_1, L_2 and L_k in Figure 5.4 represent the attribute count for cluster centroids 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 the minimum and maximum number of clusters in a data set and the attributes per cluster centroid. 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.

5.4.2 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 from/to/between hosts. The crossover operator in F-ESC 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. This performs multiple SLM atomic operations (removing a symbiont from a host and/or adding a symbiont to a host) in one step. There is a repair mechanism to make sure the offspring meets the cluster limit constraints. The parent selection process in F-ESC is a tournament selection of size 4 with the dominant individual returned as a parent. The tiebreaker in cases in which none of the individuals dominate the other one is simplicity, the preference being given to the shorter individual. The tournament selection is applied twice to select two parents for each crossover operator.

5.4.3 Similarities

The remaining components and sub-components of S-ESC are copied intact into F-ESC. Grid generation is the pre-processing component with its output being the genetic material used by the evolutionary process. F-ESC employs the same EMO algorithm – NSGA-II – utilizing both compactness and connectivity objectives. The selection operator is a tournament process among four individuals, hence, 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.

To account for robustness against data set cardinality, the RSS subsampling process is used in F-ESC. M points are selected randomly anew at each generation and the individuals’ objectives are evaluated against this set instead of the whole data set. This set (called the 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 [99] identifies the knee solution as the champion solution. All the components and sub-components mentioned are detailed in Section 4.

Chapter 6

Results

The results are presented in separate subsections for the incremental and large-scale data sets (as defined in Section 5). All results are in terms of the quartile statistic and are 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, or the 25 and 75 percentiles respectively. A missing bar for any $\langle algorithm, dataset\rangle$ pair represents the inability of the algorithm to produce results for the specified data set within a 24 hour time period, although some algorithms are allowed 96 hours on large-scale data sets. S-ESC and the comparator algorithms are compared against incremental and large-scale data sets in Sections 6.1 and 6.2 respectively. A comparison between the Symbiotic and Flat versions of ESC is given in Section 6.3. The effects of introducing outliers to the modified incremental benchmarks are investigated in Section 6.4, and the algorithm runtimes are reported in Section 6.5.

6.1 Incremental Data Sets

Table 5.1 defines a total of 10 different categories of incremental benchmarks. The y-axis in Figures 6.1 to 6.10 is the micro F-measure which ranges between 0 and 1 where larger values are preferred. The x-axis reflects the design variable (Table 5.1). For example, in the GE, GD, UE and UD data sets, the x-axis represents the average dimensionality of clusters embedded within an experiment which is 2, 4, 6, 8, 10, 15 and 20 (7 data sets in each case). 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 B.

Cluster embedding/data distribution/average cluster dimensionality experiments. The GE versus GD task pertains to a Gaussian process with an equal

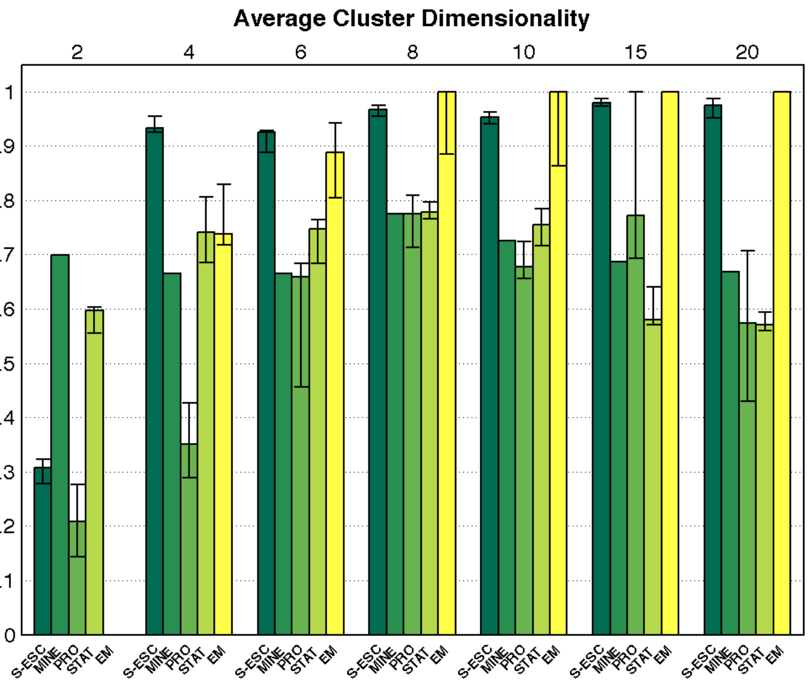


Figure 6.1: Micro F-measure for the incremental *GE* experiment. Bar order: S-ESC, MINECLUS, PROCLUS, STATPC, EM.

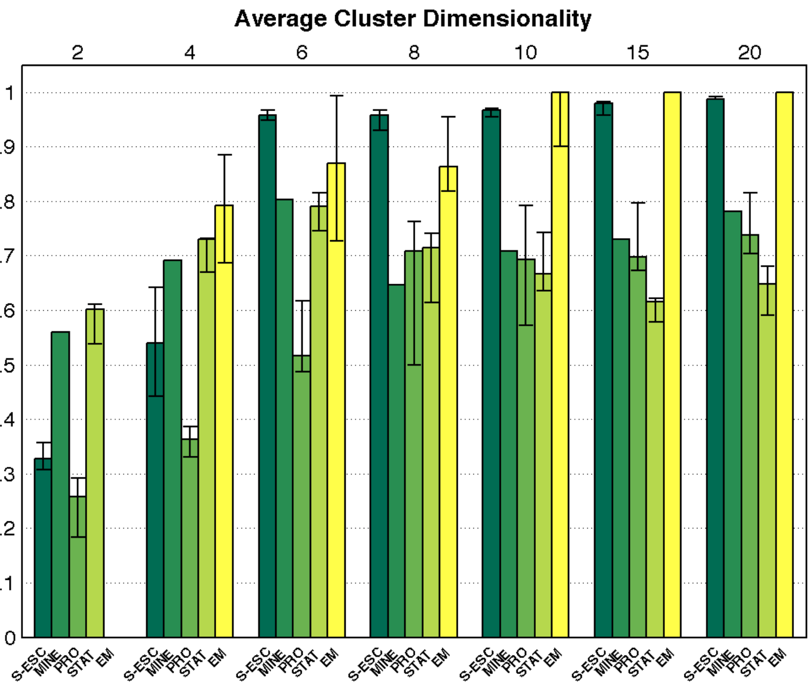


Figure 6.2: Micro F-measure for the incremental *GD* experiment. Bar order: S-ESC, MINECLUS, PROCLUS, STATPC, EM.

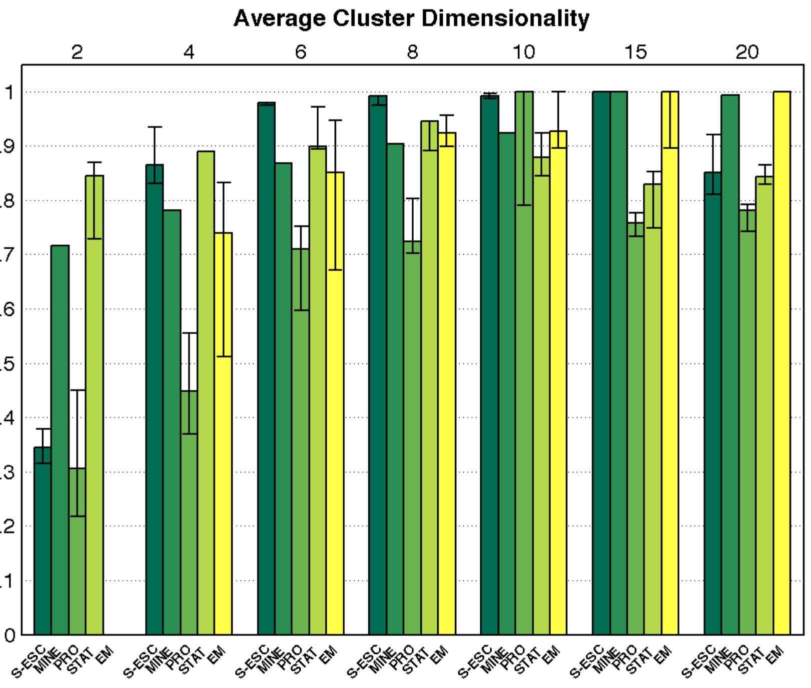


Figure 6.3: Micro F-measure for the incremental UE experiment. Bar order: S-ESC, MINECLUS, PROCLUS, STATPC, EM.

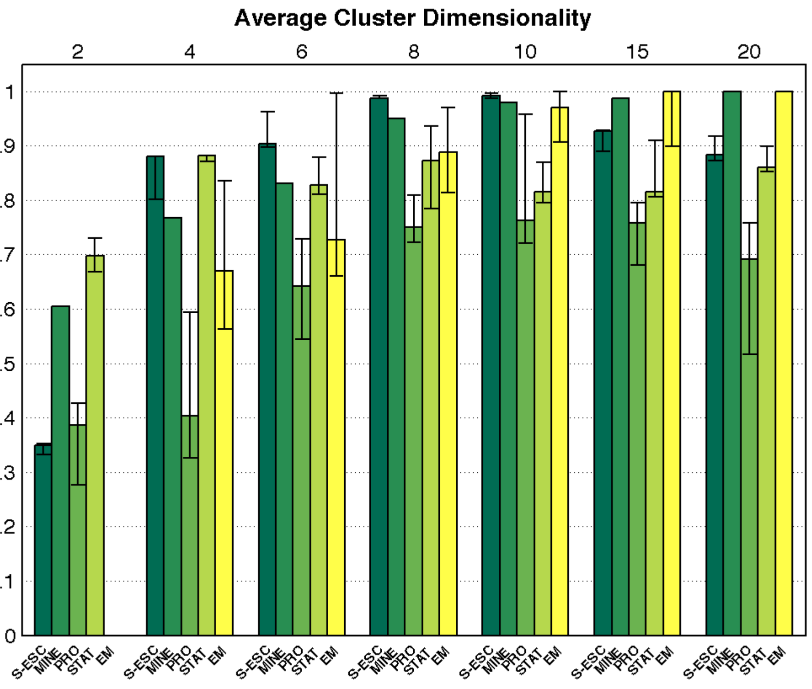


Figure 6.4: Micro F-measure for the incremental UD experiment. Bar order: S-ESC, MINECLUS, PROCLUS, STATPC, EM.

versus differing attribute support per cluster. It is the same for the UE/UD pair, but for clusters defined by a uniform probability distribution function. S-ESC and EM tend to dominate other methods with respect to the F-measure for most data sets (Figures 6.1 to 6.4). However, EM fails to return results when the average dimensionality of the clusters is 2 in all experiments. MINECLUS had a distinct preference for data with a uniform distribution (Figure 6.3 and 6.4). PROCLUS is the least effective method throughout all data sets, although it is given more a-priori information compared with other methods. As with MINECLUS, STATPC prefers clusters described by a uniform distribution, particularly as the cluster dimensionality of the embedded clusters increases.

Data set dimensionality (D) experiment. This experiment consists of 5 data sets with 20, 35, 50, 75 and 100 attributes, respectively. They all embed 5 clusters with 4 relevant attributes per cluster. Both S-ESC and STATPC provide the most consistent results as seen in Figure 6.5. In contrast, EM starts as one of the top three methods on the 20D data set, but the accuracy of EM drops as the dimensionality increases, and it fails even to return results when the data set dimensionality reaches 100. This is to be expected given that EM is the only full-space clustering algorithm. MINECLUS maintains a constant level of mediocre accuracy and does not improve or diminish with increasing dimensionality. PROCLUS appears to perform best on data set with fewest dimensionally, but even its best performance is not comparable with the other methods.

Data set cardinality (N) experiment. This experiment evaluates the effect of data set cardinality or instance count. There are 5 data sets with 80, 240, 400, 820 and 1650 instances, respectively. They all embed 5 clusters with 4 relevant attributes per cluster and 50 attributes in total. Here EM provides the stronger performance after failing initially on the smallest data set as seen in Figure 6.6 (note the missing bar for the EM for data set with 80 instances). S-ESC and STATPC represent the most consistent subspace clustering algorithms. However note the bias in favour of STATPC as a result of the parameter sweep and keeping the top 10% results. As with the previous experiments, MINECLUS and PROCLUS fail to produce quality results compared with the other methods, although they both perform slightly better than they did in the cluster embedding (GE, ..., UD) and dimensionality (D) experiments.

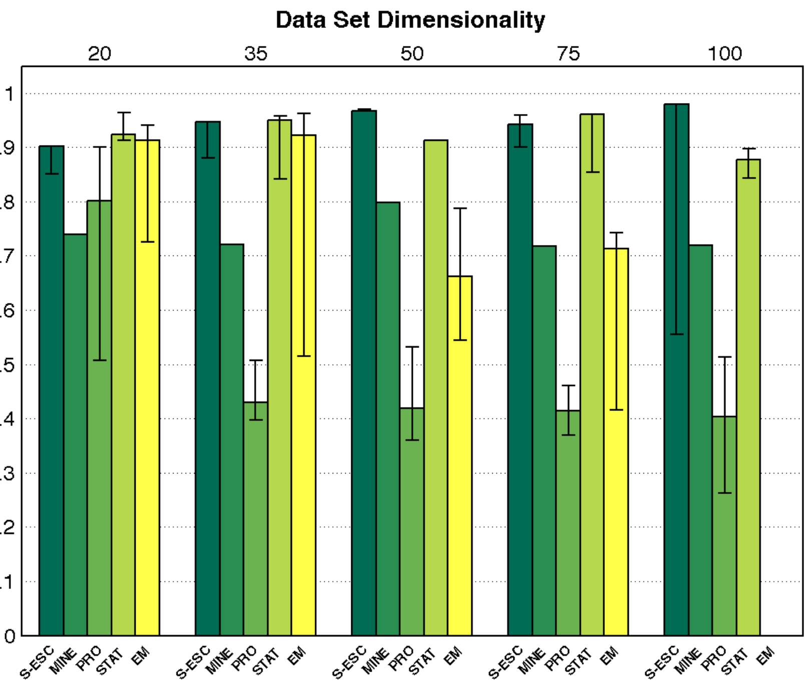


Figure 6.5: Micro F-measure for the incremental dimensionality (D) experiment. Bar order: S-ESC, MINECLUS, PROCLUS, STATPC and EM.

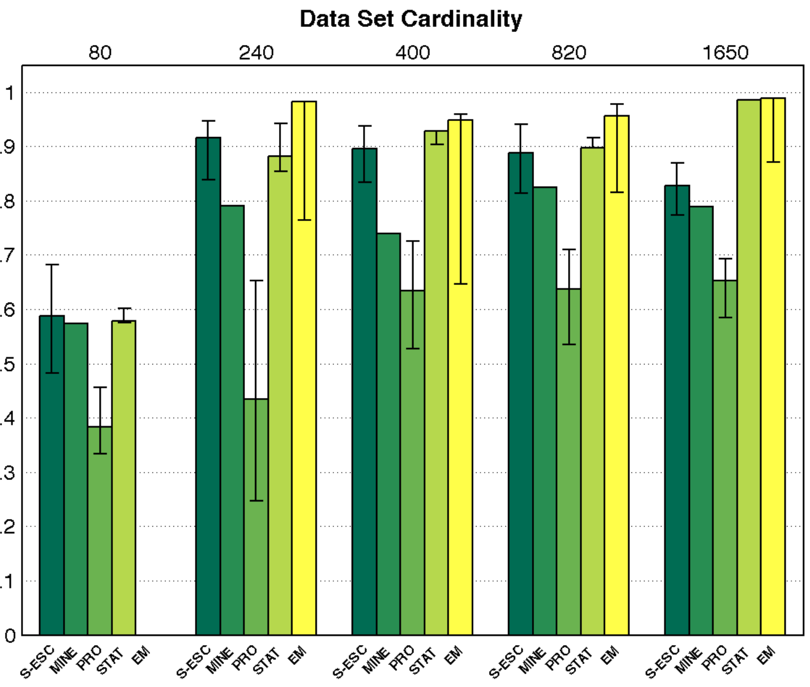


Figure 6.6: Micro F-measure for the incremental cardinality (N) experiment. Bar order: S-ESC, MINECLUS, PROCLUS, STATPC and EM.

Cluster count (k) experiment. In this experiments, the cluster count increases from 2 in data set 1 to 5 in data set 4. There are 50 attributes in each data set but only 4 of them are relevant attributes in each cluster. Therefore, as with the N experiment, the majority of attributes are not relevant for the clustering task. S-ESC, STATPC and EM perform equally well and there is no obvious winner among these three (Figure 6.7). The S-ESC and STATPC results are dominated by EM on the 2-cluster data set, whereas EM is dominated by the other two on the 4- and 5-cluster data sets. PROCLUS provides comparable results for the first two data sets for the first time, but deteriorates as the number of clusters increases beyond 3, an interesting result given that PROCLUS is given the correct cluster count a priori. Contrary to PROCLUS, MINECLUS returns the poorest results for the first two data sets, but improves as the cluster count increases.

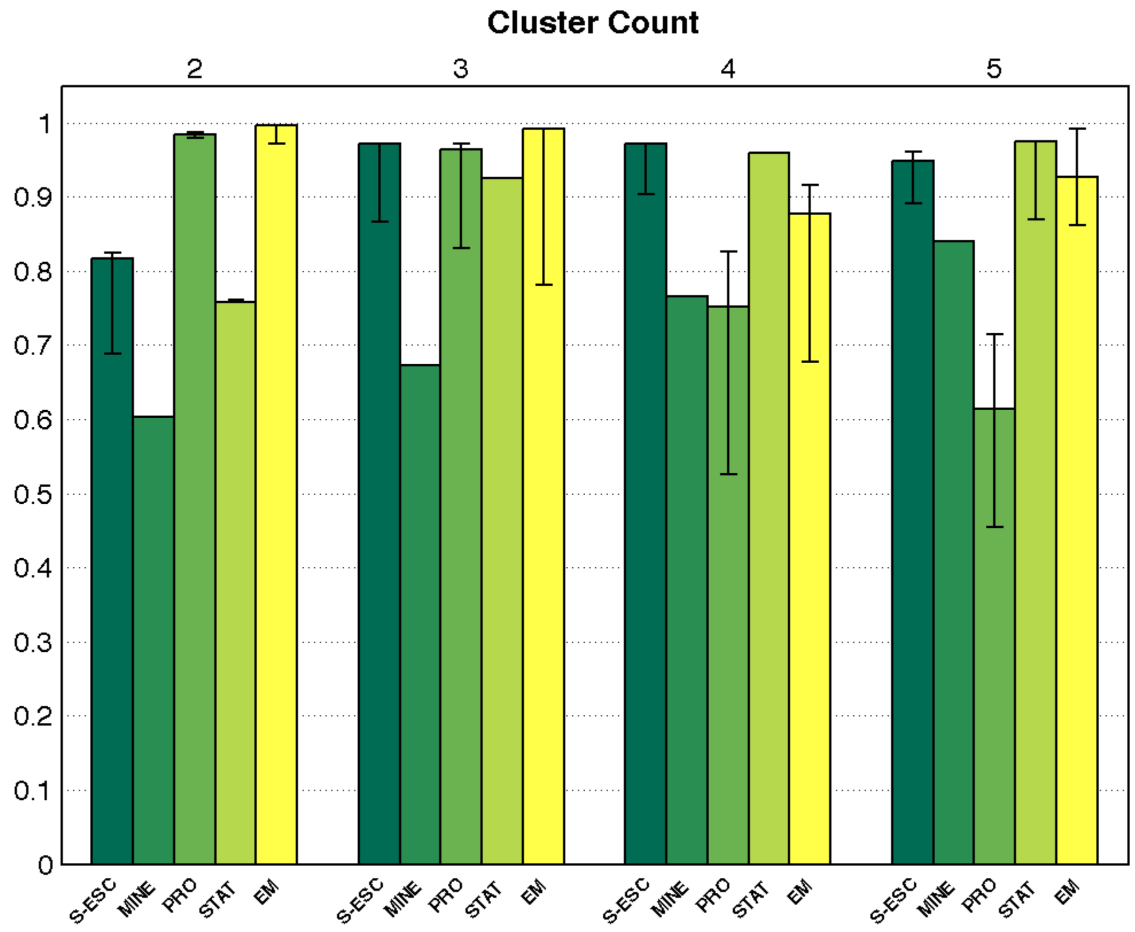


Figure 6.7: Micro F-measure for the incremental cluster count (k) experiment. Bar order: S-ESC, MINECLUS, PROCLUS, STATPC and EM.

Attribute spread (*Extent*) experiment. The next Moise experiment varies the attribute spread of the relevant attributes. The spread of the uniform distribution from which the relevant instances are drawn increases from 0.1 to 0.4 in 4 steps. In other words as the distribution gets wider in each relevant attribute, the cluster densities decrease, which makes identifying the clusters more difficult. All 4 data sets in this experiment have 5 clusters with 4 relevant attributes each, and 30 irrelevant attributes. As can be seen in Figure 6.8, all the methods deteriorate rapidly as distributions on the relevant attributes get wider and more diverse, becoming more difficult to distinguish from the noise associated with the non-relevant attributes. However, STATPC degrades more gracefully, albeit the with F-measure used to select the appropriate STATPC parameterizations. EM fails to return results for the last data set where the relevant attributes spread upto 0.4 out of the possible [0,1] range.

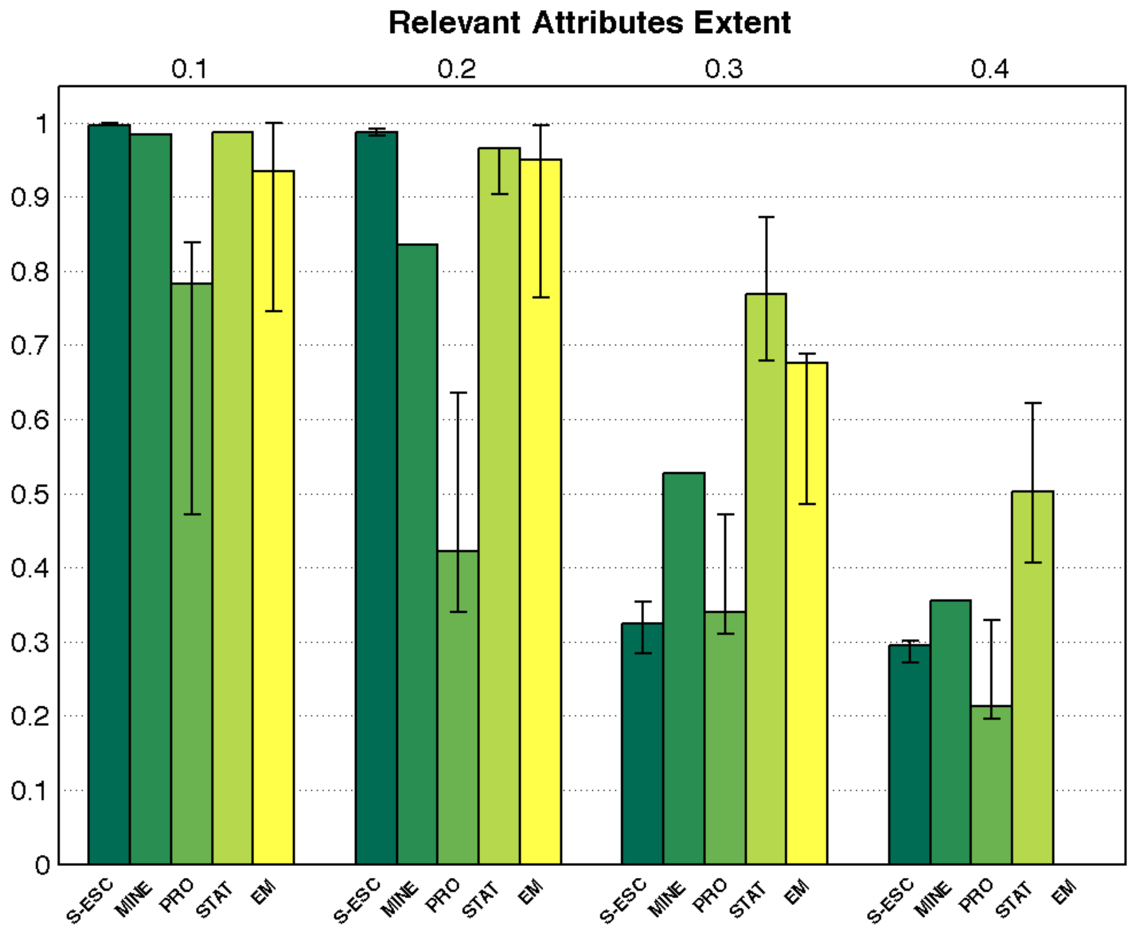


Figure 6.8: Micro F-measure for the incremental attribute spread (*Extent*) experiment. Bar order: S-ESC, MINECLUS, PROCLUS, STATPC and EM.

Attribute overlap (*Overlap*) experiment. Varying the overlap between the relevant attributes of the different clusters has a strong effect on all of the methods. Each data set in this experiment has 2 clusters with the first 4 attributes being relevant to both clusters, leaving a total of 46 attributes irrelevant. The overlap between the attributes from the different clusters increases from 0 to 0.3 in 4 steps. As shown in Figure 6.9, EM performs more consistently in all but the last data set in which it returns no results, indicating a possible need for re-parameterization. However, EM algorithm will naturally not be able to select the correct attribute space. PROCLUS appears to be the strongest of the subspace methods in this experiment, but it drops drastically to the poorest method on the last experiment. S-ESC appears to be the most consistent subspace method and also provides the best results for the last experiment where the relevant attributes of different clusters overlap by 30%.

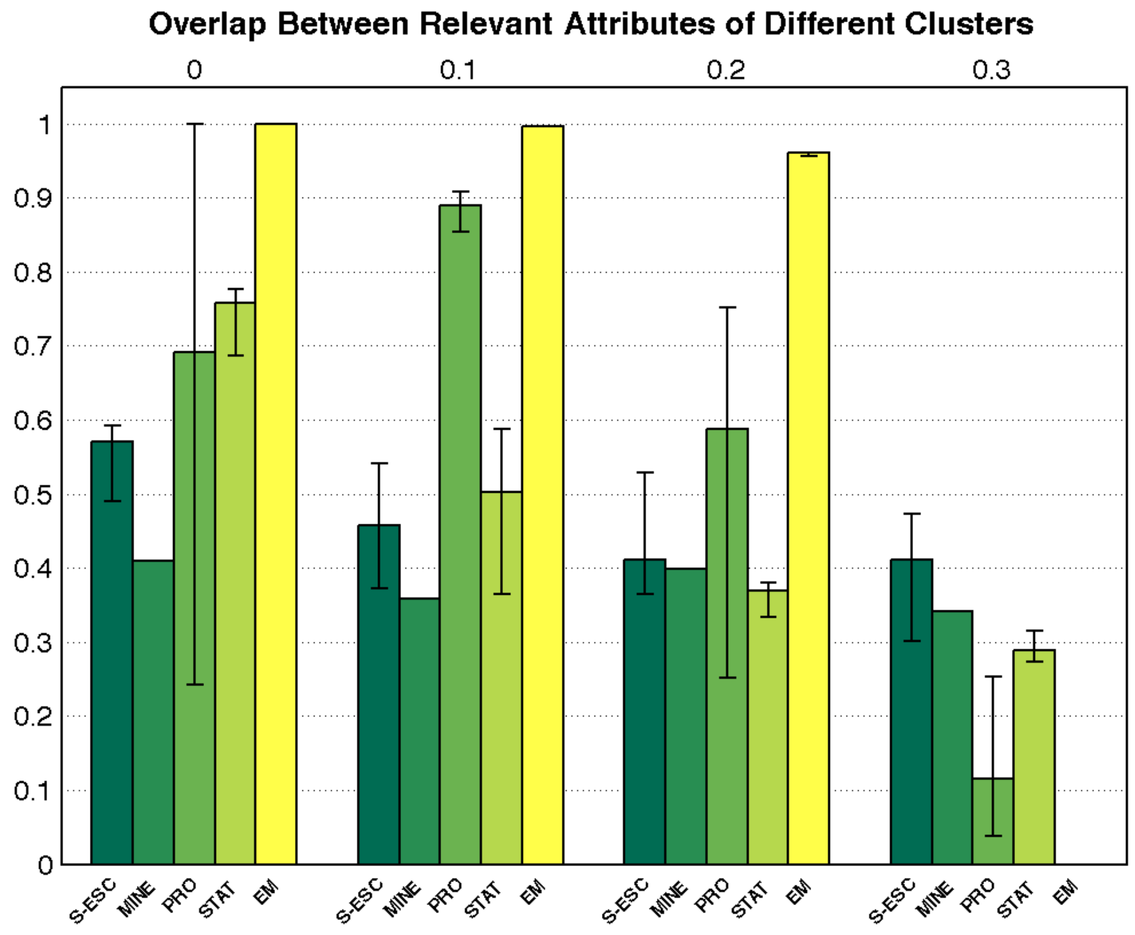


Figure 6.9: Micro F-measure for the incremental attribute overlap (*Overlap*) experiment. Bar order: S-ESC, MINECLUS, PROCLUS, STATPC and EM.

Instances per cluster (*ClusterSize*) experiment. The final experiment evaluates the effect of varying the number of data instances per cluster within a data set. The 4 data sets in this experiment all have 5 clusters using 4 (different) relevant attributes each, and 30 irrelevant attributes. The average instance per cluster (cluster size) is 30, 40, 50 and 55 respectively. EM fails to provide results for the smallest data set, while S-ESC and STATPC dominate all other methods in all but the largest data set. The variance is very minimal between the last two data sets: S-ESC and STATPC produce almost identical results for the two data sets, but surprisingly, EM, which is dominated by S-ESC and STATPC on third data set, dominates the other two methods on the last data set (Figure 6.10). PROCLUS returns the poorest results, as expected, with MINECLUS improving accuracy with increasing cluster size, but not able to reach the S-ESC and STATPC level.

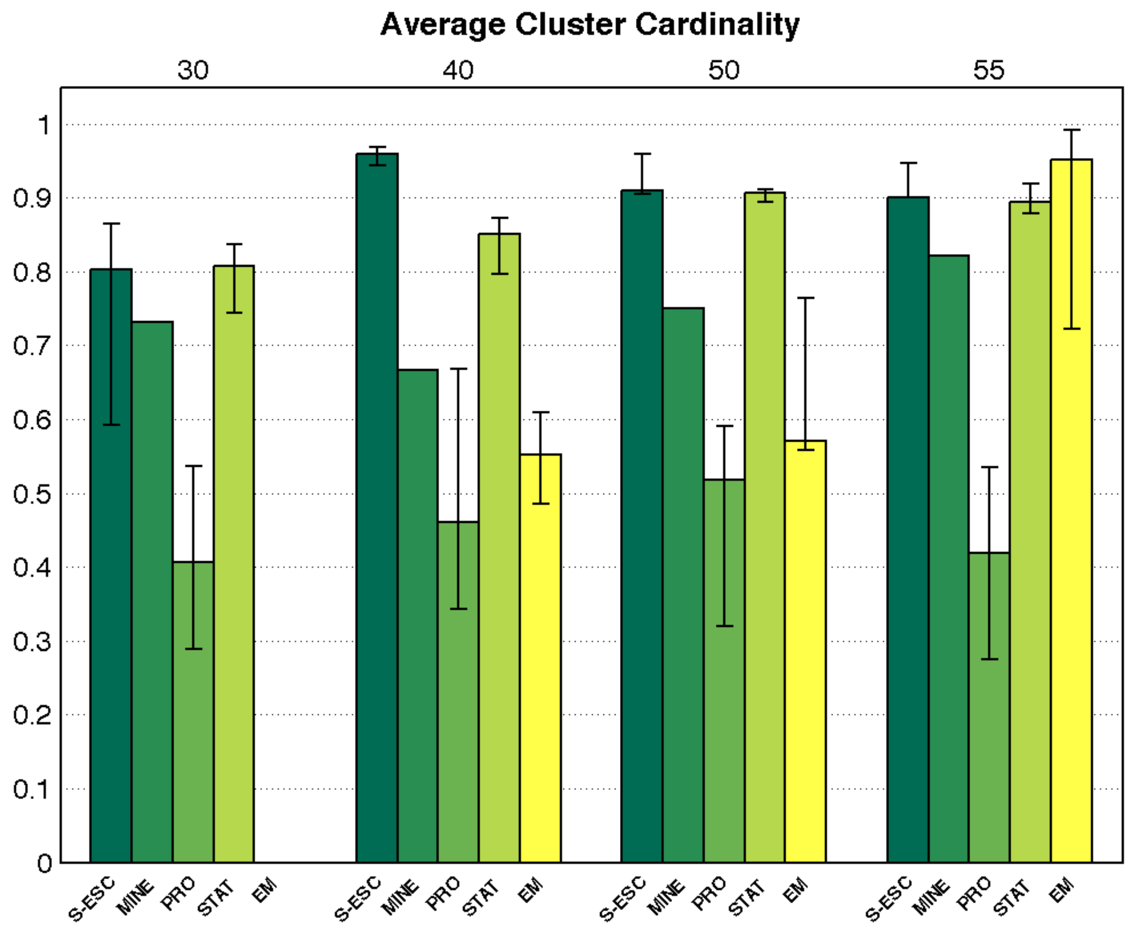


Figure 6.10: Micro F-measure for the incremental cluster size (*ClusterSize*) experiment. Bar order: S-ESC, MINECLUS, PROCLUS, STATPC and EM.

The above summary evaluates the outcome of the Moise experiments in terms of accuracy alone, the motivation being that the purity of the clusters (as determined by the F-measure) represents the primary objective. However, as a secondary objective it is assumed that the resulting clusters should be simple as well. 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*. In other words, each attribute is counted only once for a clustering solution regardless of how many cluster centroids in which it appears. Figures 6.11 to 6.17 show the average attribute count per solution for S-ESC and its comparators on 7 experiment data sets of Table 5.1. For sake of brevity, we report only results under incremental benchmark with significant trends.

Starting with the specific case of the **cluster embedding** experiment (Figures 6.11 to 6.14) S-ESC falls behind all subspace methods on the smallest number of attributes per cluster. However, once the average cluster dimensionality grows beyond 2, S-ESC appears to be among the two most simple solutions. MINECLUS seems to compete with S-ESC on Gaussian-distributed data sets (GE and GD), however, it starts to index a larger percentage of attributes once the average cluster dimensionality grows beyond 10. In the uniformly-distributed data sets (UE and UD) the situation is even more critical for MINECLUS since it indexes almost all the attributes for the data sets with an average cluster dimensionality greater than 10. PROCLUS shows the poorest performance with respect to solution simplicity compared with all the other methods on almost all the data sets in this experiment. STATPC does not have a consistent trend, but it always ends up among the top two methods (along with S-ESC) on the data sets with the highest average cluster dimensionality. EM is only plotted as the baseline here to show the maximum number of attributes per data set since it will always include all attributes.

In the **data set dimensionality** (D) experiments all subspace clustering methods perform equally well and index between 13 and 20 (unique) attributes per data set (Figure 6.15). Almost the same trend holds true for the **data set cardinality** (N) and **cluster count variation** (k) experiments, Figures 6.16 and 6.17 respectively. MINECLUS and S-ESC utilize the least number of attributes consistently.

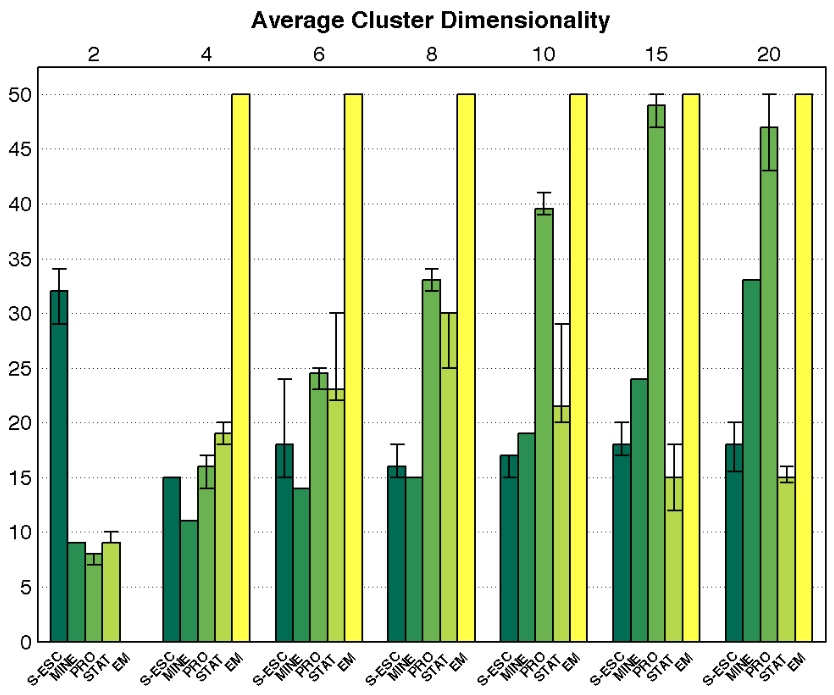


Figure 6.11: Attribute count per solution for the incremental *GE* experiment. Bar order: S-ESC, MINECLUS, PROCLUS, STATPC and EM.

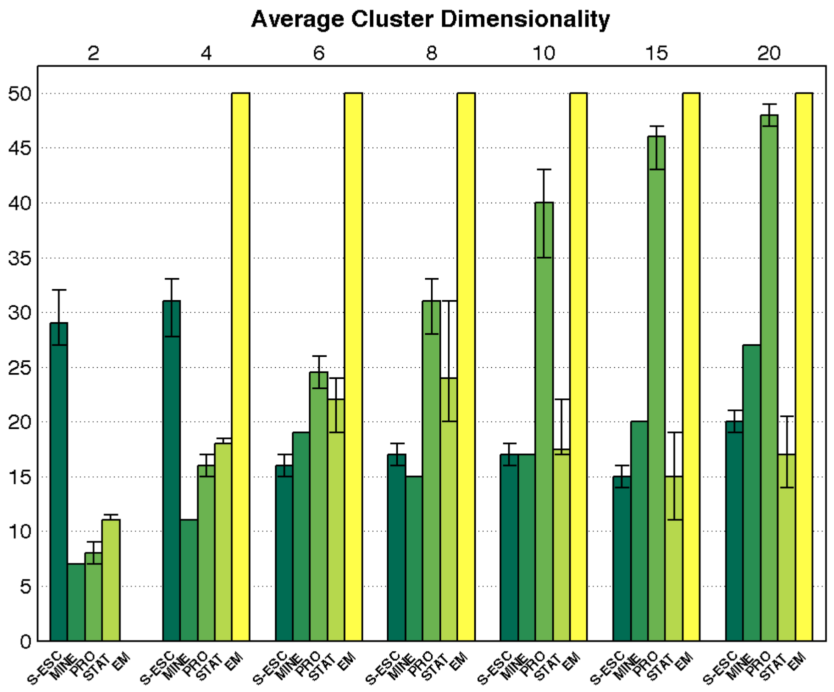


Figure 6.12: Attribute count per solution for the incremental *GD* experiment. Bar order: S-ESC, MINECLUS, PROCLUS, STATPC and EM.

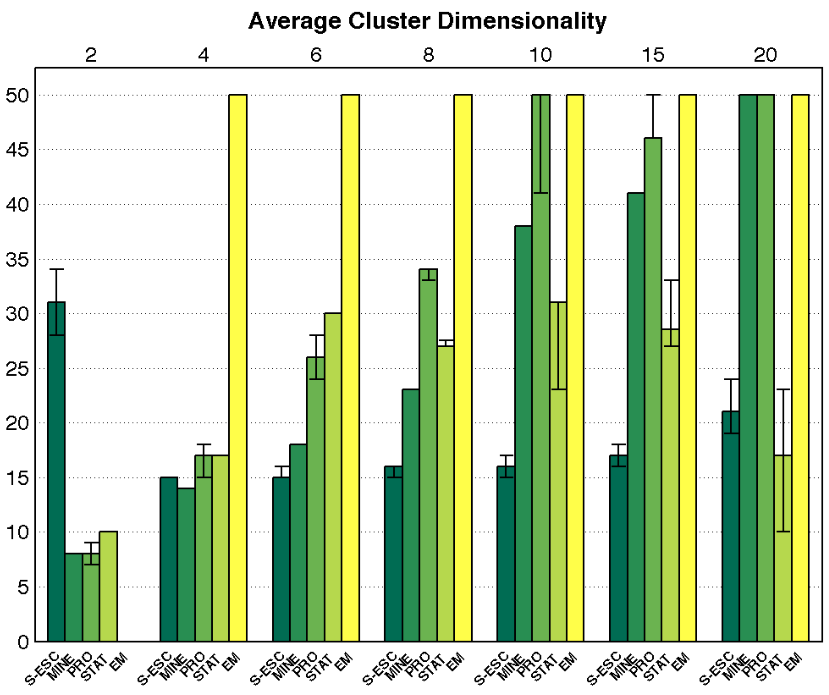


Figure 6.13: Attribute count per solution for the incremental UE experiment. Bar order: S-ESC, MINECLUS, PROCLUS, STATPC and EM.

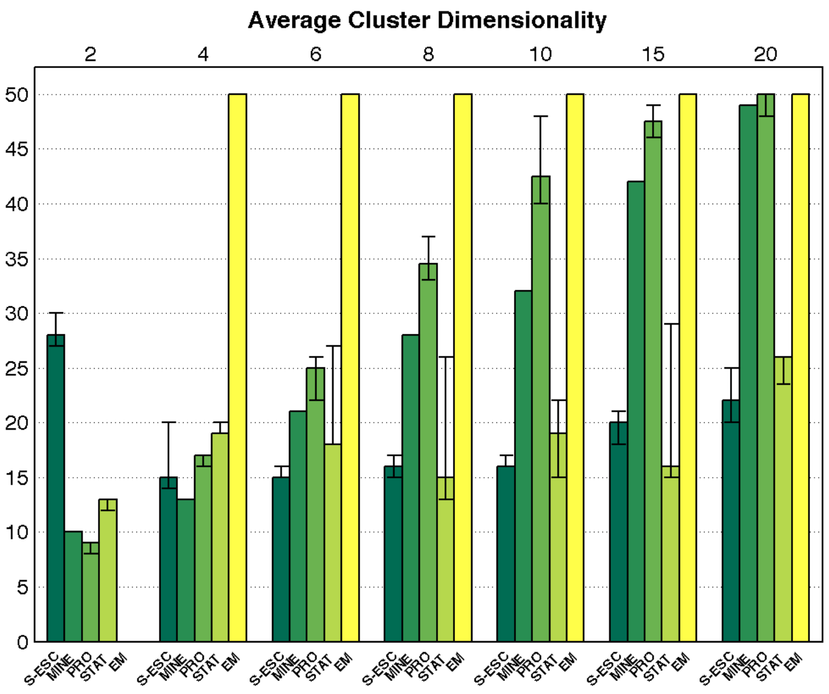


Figure 6.14: Attribute count per solution for the incremental UD experiment. Bar order: S-ESC, MINECLUS, PROCLUS, STATPC and EM.

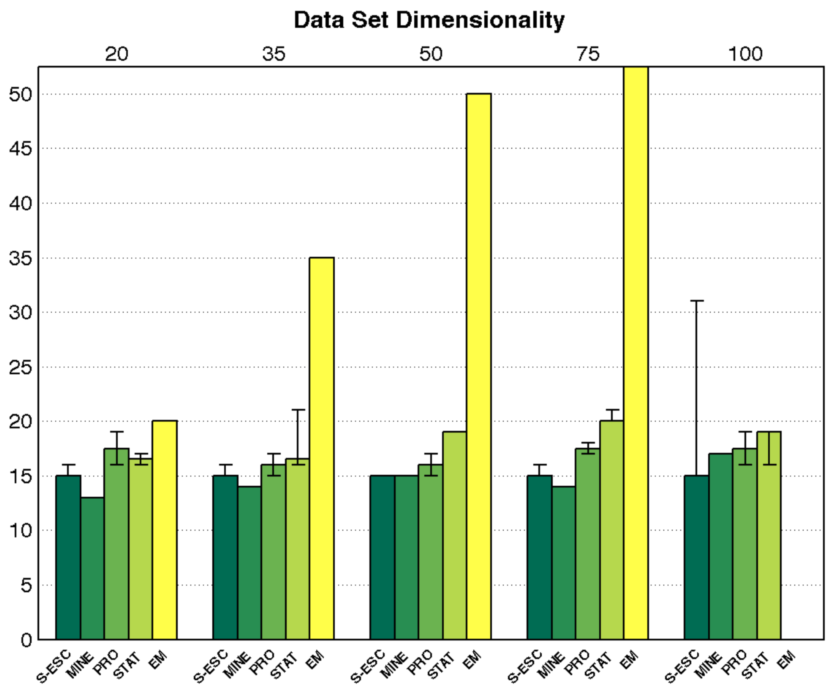


Figure 6.15: Attribute count per solution for the incremental dimensionality (D) experiment. Bar order: S-ESC, MINECLUS, PROCLUS, STATPC and EM.

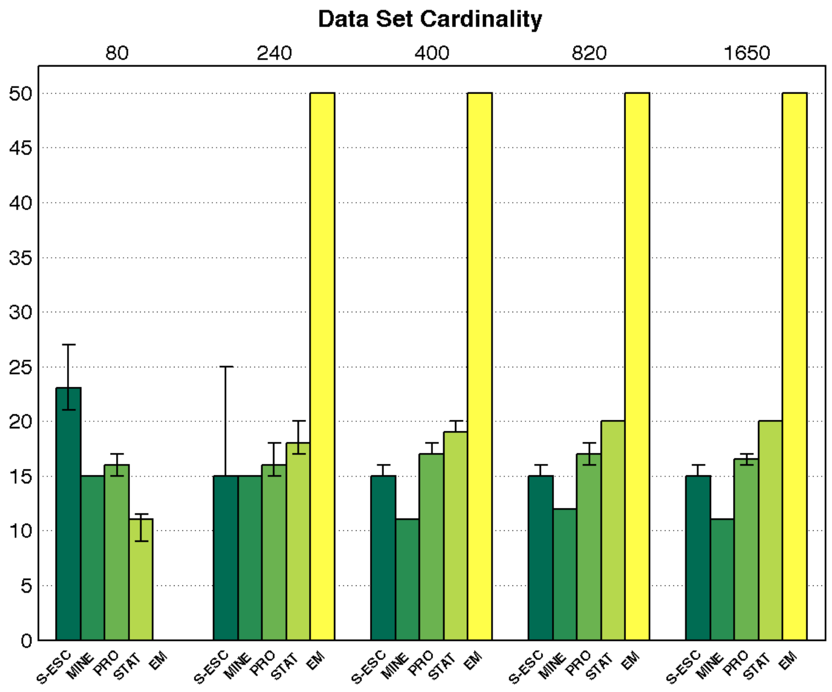


Figure 6.16: Attribute count per solution for the incremental cardinality (N) experiment. Bar order: S-ESC, MINECLUS, PROCLUS, STATPC and EM.

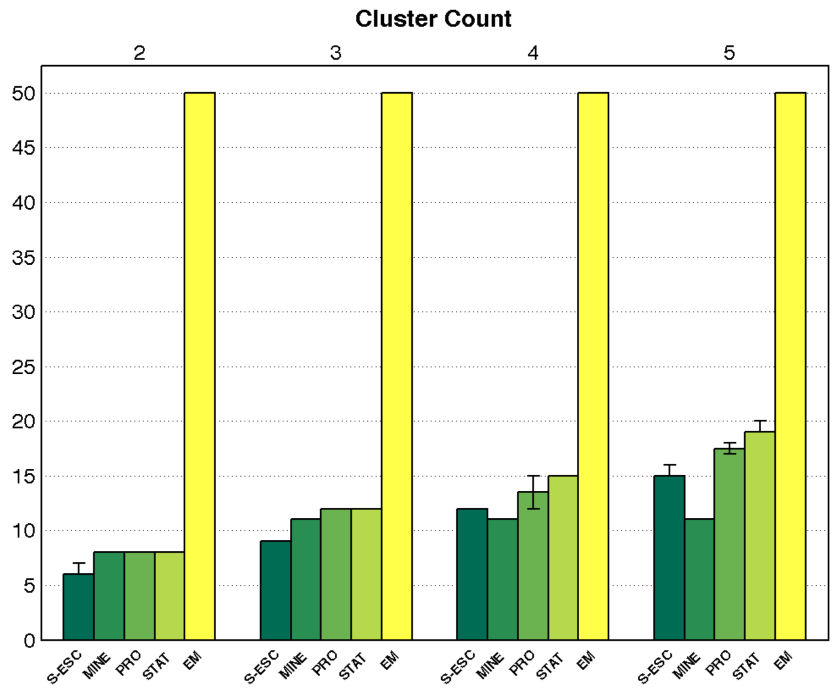


Figure 6.17: Attribute count per solution for the incremental cluster count (k) experiment. Bar order: S-ESC, MINECLUS, PROCLUS, STATPC and EM

6.2 Large-scale Data Sets

The Moise *et al.* benchmarking data sets of Table 5.1 explained in the previous section cover a wide range of properties (the distribution relevant instances are drawn from, the average cluster dimensionality, the overlap between relevant attributes of different clusters within a data set, etc). However, as noted in Section 5, there are three basic shortcomings: 1. the dimensionality and cardinality remains low; 2. all clusters within a single data set use the same distribution; and 3. the cluster embedding assumes aspect ratios which are not particularly diverse.

The last point emphasizes 'compactness' style objectives. In particular, the 800D data set (Table 5.2) has both low-dimensional (10D) and high-dimensional (100D) clusters within the same data set. Moreover, the implanted clusters use different distributions (both Gaussian and uniform) at different aspect ratios. This makes the 800D data set very challenging. The 1000D data set also embeds ten 10D clusters, thus only 10% of the attributes are relevant and the rest are irrelevant. Benchmark results are presented for the 5 additional 'large-scale' data sets of Table 5.2. Given the larger size of these data sets it was necessary to enforce a computational limit of 24 or 96 hours per run for MINECLUS and STATPC respectively. If results were not returned within this time then the implementation/algorithm was deemed inappropriate for the task dimension/cardinality.

Figure 6.18 summarizes how the F-measure varies as a function of each data set (identified in terms of data dimensionality). S-ESC betters all other methods on the first three data sets, and is the runner up method for the 800D and 1000D data sets (with regards to full-space EM clustering), thus bettering all other subspace methods consistently.

MINECLUS fails for data sets with 500 dimensions or more after 24 hours. It is the second best subspace method (after S-ESC) with respect to the accuracy of solutions for the 200D data set. PROCLUS matches S-ESC in the specific case of the 500D data

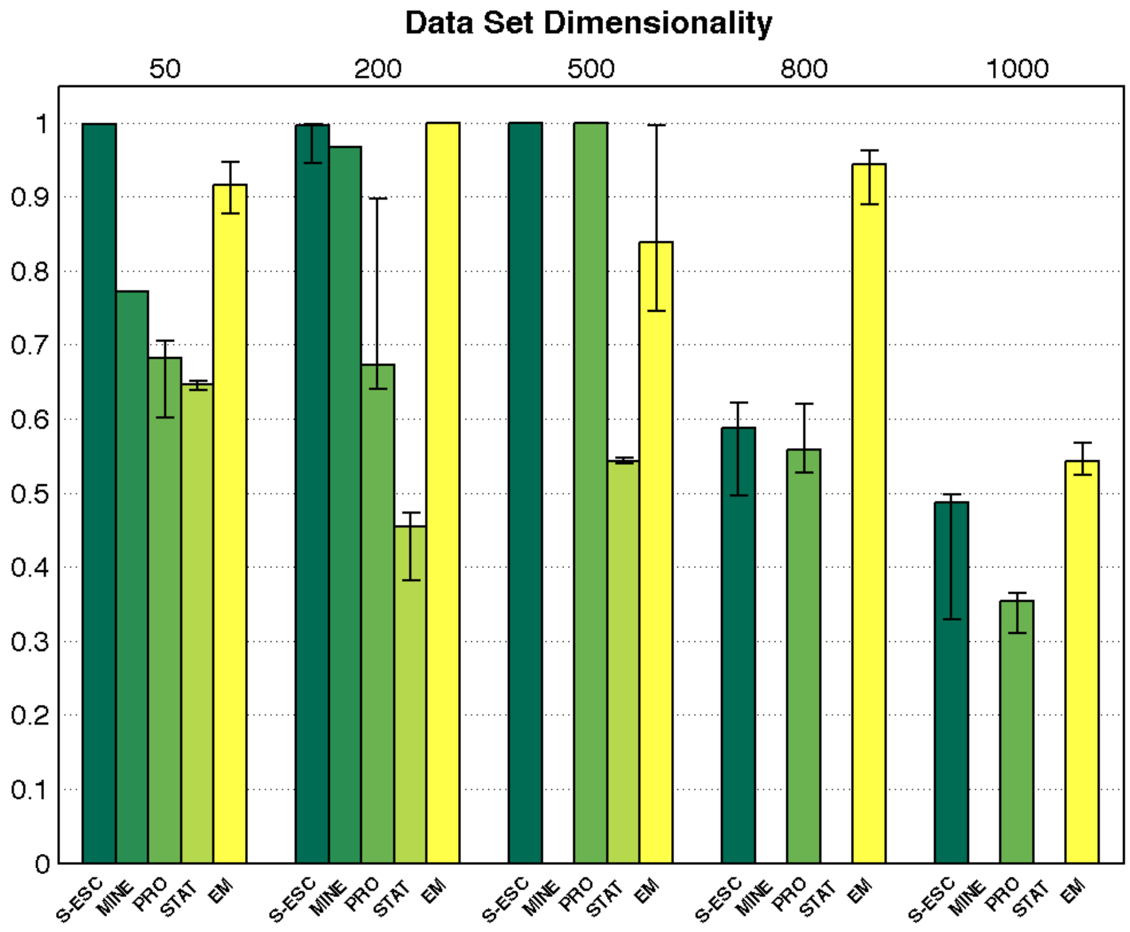


Figure 6.18: Micro F-measure for the large-scale data sets. Bar order: ESC, MINECLUS, PROCLUS, STATPC and EM.

set. It is also the only implementation of a subspace clustering algorithm, besides S-ESC, which returns solutions for the 800D and 1000D data sets. The STATPC implementation produces the least accurate results for the first three data sets, and fails altogether for data sets beyond 500 dimensions, i.e. results are not produced within 96 hours. Part of this might be due to the sensitivity of parameterization. However, as the scale of a task increases, the cost of the parameter sweeps increases significantly.

Bar plots for the number of unique attributes included (solution complexity measure) illustrates clearly the efficiency of S-ESC with respect to parsimony (Figure 6.19). S-ESC uses fewer than 50 attributes for all data sets. Medians are 12, 16, 13, 37 and 28 for the five data sets respectively, using the most attributes for the 800D

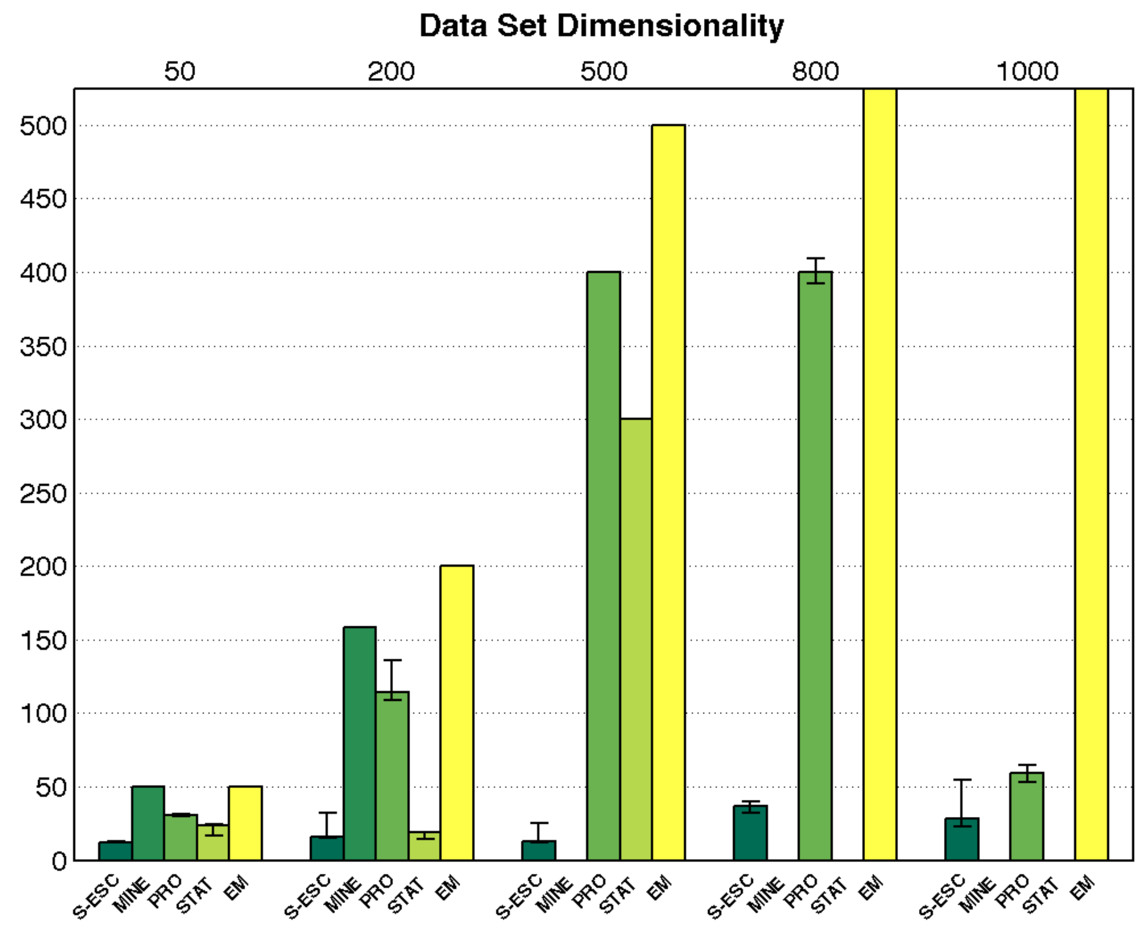


Figure 6.19: Attribute count per solution for the large-scale data sets. For visualization EM attribute count under 800 and 1000 dimensional data sets is cropped at 550. Bar order: ESC, MINECLUS, PROCLUS, STATPC and EM.

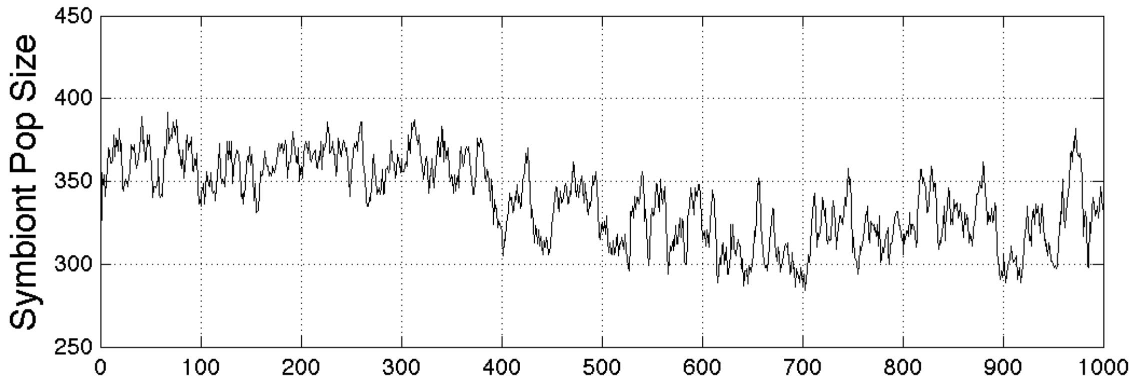
and 1000D data sets. MINECLUS indexes all 50 attributes for the 50D data set and 158 for the 200D data set and fails on the larger data sets. PROCLUS indexes 36 and 112 for the first two data sets, and then jumps to use almost 400 attributes for the 500D and 800D data sets, but manages to index only over 60 attributes for the 1000D data set. STATPC employs 24, 19 and 300 attributes for the first three data sets and stops for the data sets with higher dimensionality.

6.3 Symbiotic vs. Flat ESC

What follows is a comparison of the results of performing the search for subspace clusters using a 'flat' version of ESC (a.k.a. F-ESC). F-ESC is compared with S-ESC with respect to the large-scale data sets from Table 5.2 as well as selected incremental data sets of Table 5.1. Rather than just assuming S-ESC parameter values, preliminary experiments are performed to optimize the 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 are achieved when both crossover and mutation probabilities are set to 100%, but note that this is discounted by the annealing schedule of Section 4.4. For the sake of brevity no plots for this experiment will be presented.

The situation is somewhat different with regards to population size. S-ESC uses two distinct populations in which the host population is of a fixed size (100), and the symbiont population size varies as a natural effect of group selection (Section 1.4.3) and the variation operators (Section 4.4). However, 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 is recorded. Figure 6.20 shows the population profile for two data sets from the large-scale data sets of Table 5.2. Symbiont population size in the case of the 200D data set (top profile) fluctuates between 284 and 392 individuals, while the more complex



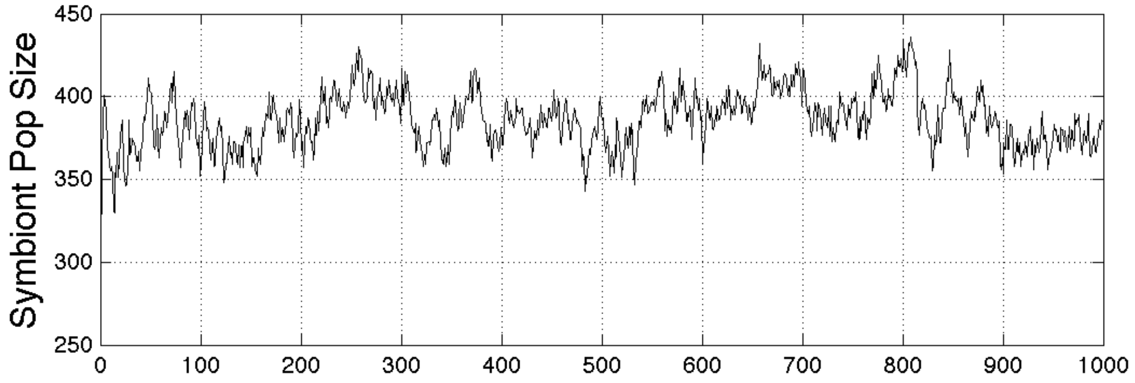


Figure 6.20: The S-ESC symbiont population size profile for the 200D and 800D data sets from Table 5.2 over 1000 generations.

800D data set (bottom profile) utilized between 297 and 436. Further experiments show that there is some variation between the profiles of the different data sets of Table 5.2; however, the minimum and maximum bounds are very close. Similar trends with minor changes can be observed for the other three data sets (50D, 500D and 1000D). In general the dynamic 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. Figure 6.21 illustrates the F-measures of F-ESC with different population sizes with regards to the large-scale data sets (Table 5.2). For comparison purposes the earlier S-ESC results are included as well. In all cases 50 runs are performed on each data set and one knee solution is selected as the champion solution for each run. As with results from previous sections, the top of each bar defines the median of a distribution with first and third quartiles shown in

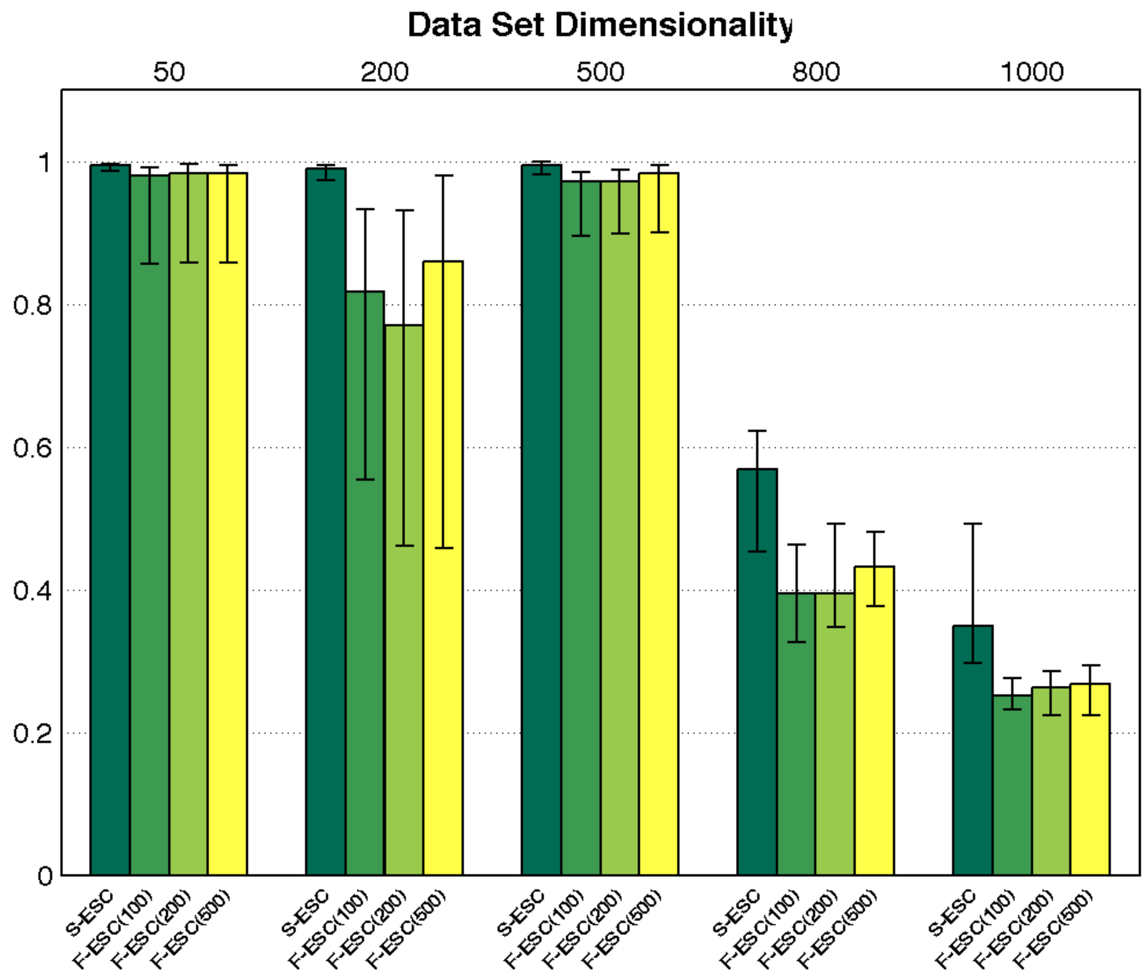


Figure 6.21: Micro F-measure comparison between S-ESC and F-ESC with different population sizes for the large-scale data sets. Bar order: S-ESC, F-ESC (100), F-ESC (200) and F-ESC (500).

the form of error bars.

It appears that the overall performance of F-ESC improves as the population size increases, but 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, for the incremental data sets of Table 5.1, F-ESC will assume a fixed population size of 100. Figures 6.22 and 6.23 compare S-ESC and F-ESC in terms of F-measures for the GD and UD experiments with 7 data sets each.

It is apparent that there is only one data set on which F-ESC outperforms S-ESC: a 5-class, 50-dimensional data set with Uniform distribution in which the average

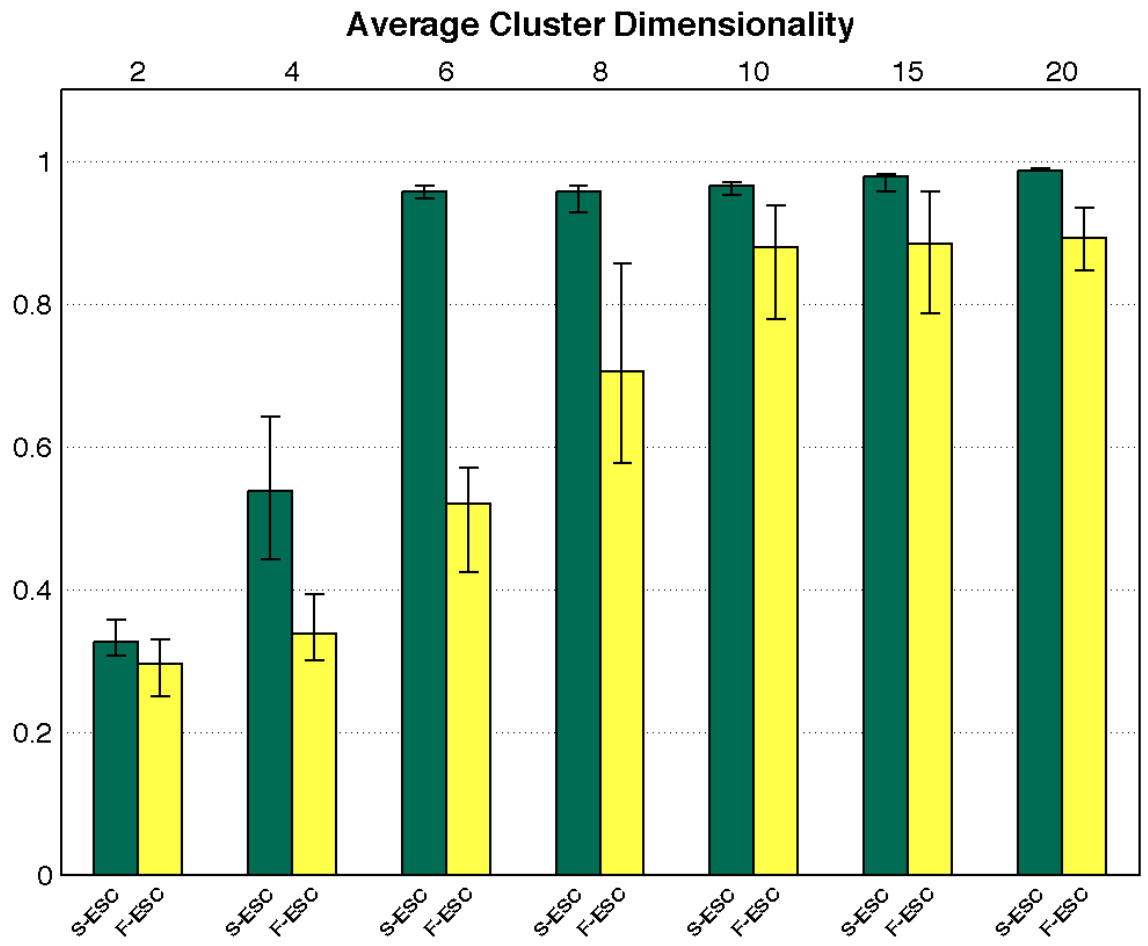


Figure 6.22: Micro F-measure comparison between S-ESC and F-ESC for the incremental *GD* experiment. Bar order: S-ESC and F-ESC.

dimensionality of clusters is 20 (last column of Figure 6.23). In all other cases S-ESC outperforms F-ESC. In most cases S-ESC dominates F-ESC by a significant margin; moreover, these correspond to the more difficult scenarios in which the number of supporting attributes per cluster subspace is very low. Results produced by the other 38 cases of the incremental data sets (not shown for the sake of brevity) support this general trend as well.

6.4 Outlier Effects

S-ESC has no integrated outlier detection process, where outliers represent data instances for which all attributes represent noise. As noted in Section 3.3, only STATPC explicitly includes an outlier detection capability. In the following the incremental

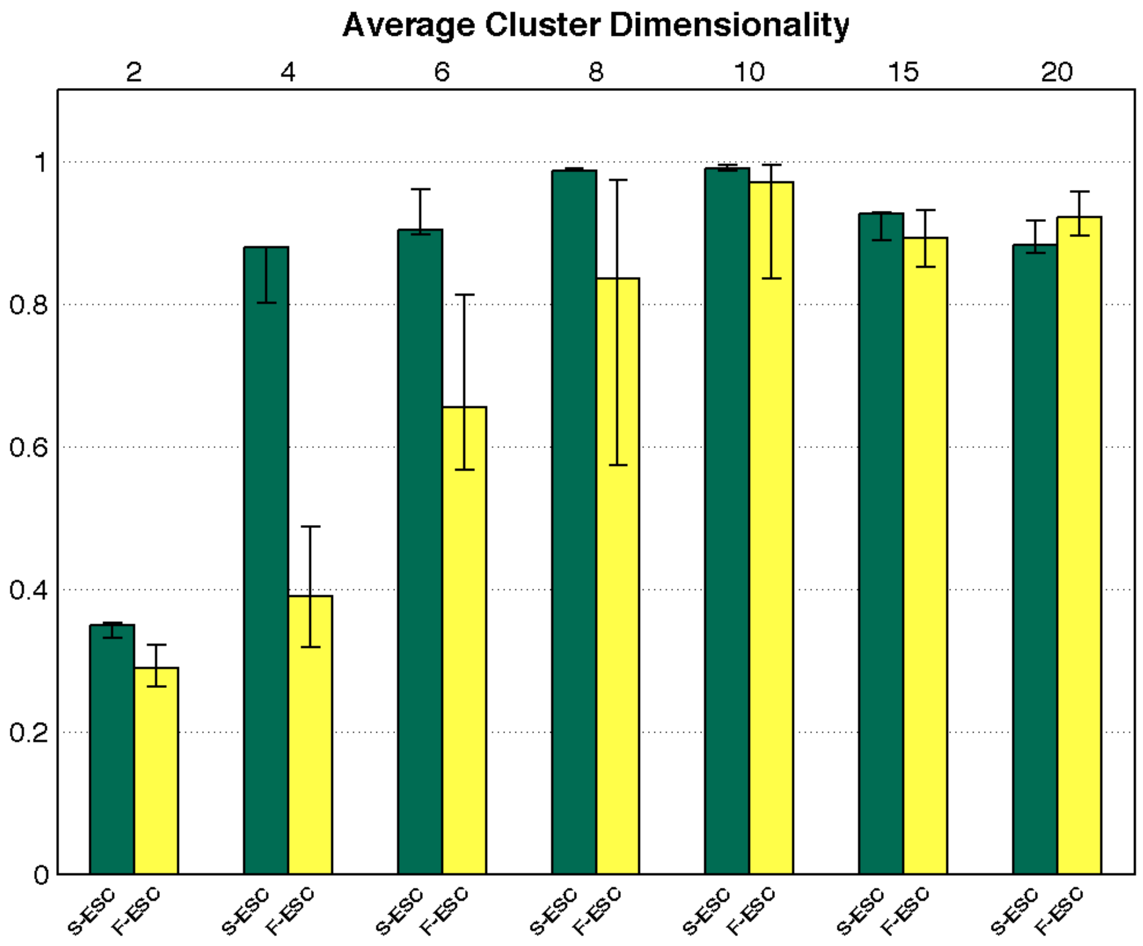


Figure 6.23: Micro F-measure comparison between S-ESC and F-ESC for the incremental UD experiment. Bar order: S-ESC and F-ESC.

data sets of Table 5.1 are used in their original form with the outliers included. From the S-ESC perspective this can be used as an opportunity to assess the impact of outliers on different components of the S-ESC framework. This prompts the following questions: 1. What impact do outliers have on the initial grid construction alone? 2. What impact do outliers have on the identification of the S-ESC subspace clusters alone? and 3. What is the combined impact of the outliers on the entire S-ESC?

Two data sets with different data distributions [Gaussian (GD) vs. Uniform (UD)] with 5 embedded clusters and 8 attributes per cluster will be utilized for the outlier experiments. Figure 6.24 summarizes the performance using a combined violin–box plot quantifying the distribution of the F-measure across 50 runs per experiment. A violin plot establishes the nature of the distribution – to what extent a bimodal

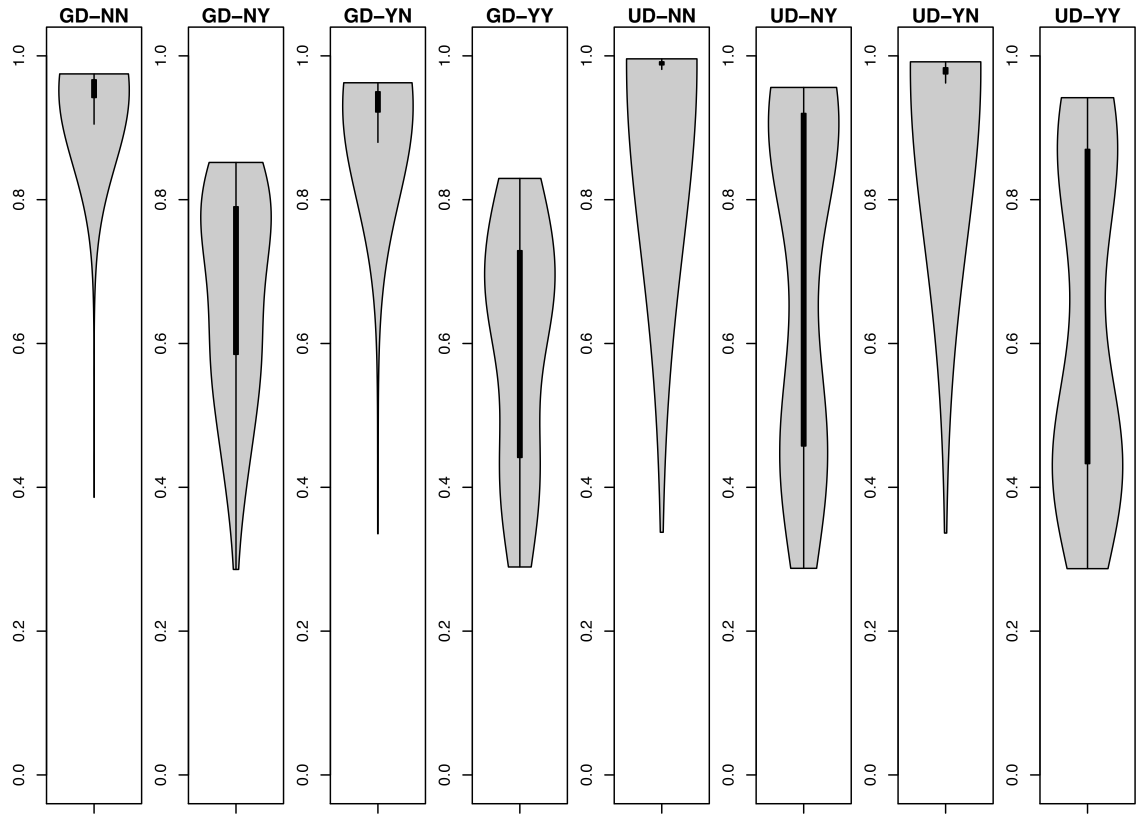


Figure 6.24: The effect of introducing outliers to data sets before and after the grid generation process. GD–XX denote the experiments on the Gaussian data set and UD–XX the Uniform data set; XX–NN are the base case with no outliers; XX–NY imply no outliers during the pre-processing step of grid generation, but are included during the S-ESC evolutionary identification of subspaces; XX–YN imply that outliers are present only during grid generation; and XX–YY imply the outliers are present throughout.

distribution is present – while the box plot defines the quartile statistics (1st, 3rd quartile and median). For reference purposes results with no outliers are included (labelled GD–NN and UD–NN).

It is apparent that the pre-processing step of grid generation affected the least by the introduction of outliers (compare GD–NN with GD–YN and UD–NN with UD–YN), whereas a 20% reduction in the 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 bi-modal distribution results in half of the solutions returned with an F-measure of 80% or higher when noise is introduced to the S-ESC cluster identification process

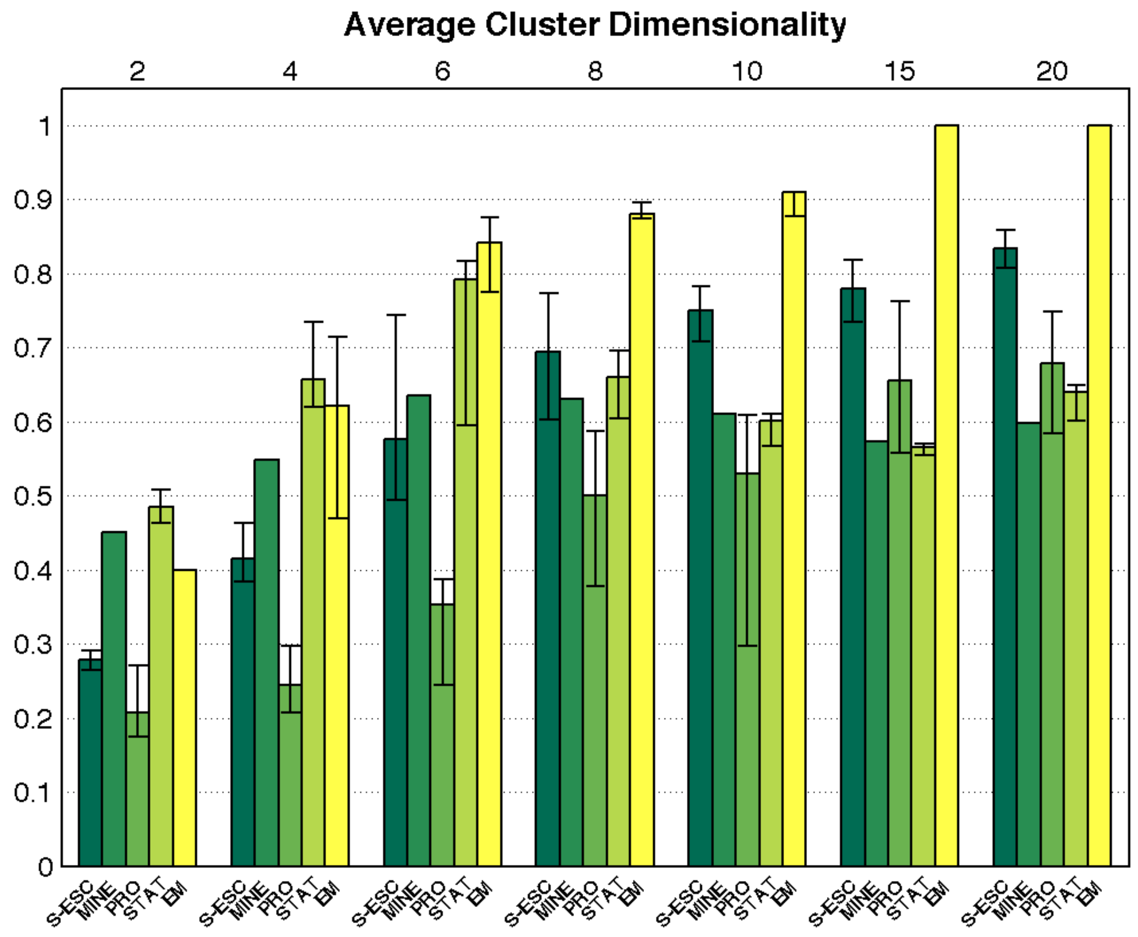


Figure 6.25: Micro F-measure comparison between S-ESC and comparator methods for the incremental GD experiment including outliers. Bar order: S-ESC, MINECLUS, PROCLUS, STATPC and EM.

(UD–NY and UD–YY).

Figures 6.25 and 6.26 show the effect of introducing outliers into the S-ESC and comparator methods on 14 data sets from the GD and UD experiments of the incremental data sets (first two rows of Table 5.1). As these figures illustrate (compared with the results from the same methods on the data sets without outliers, Figures 6.2 and 6.4) the accuracy of all methods drops. Results from the EM algorithm remain strong as long as there is a sufficient number of attributes per cluster. However, once cluster attribute support drops below eight then the performance of one or more subspace algorithms is significantly better. Under the Gaussian distribution (Figure 6.25), S-ESC remains the stronger algorithm for data sets with 6 or more attributes per cluster. At lower dimensions STATPC is preferred, but must undergo extensive

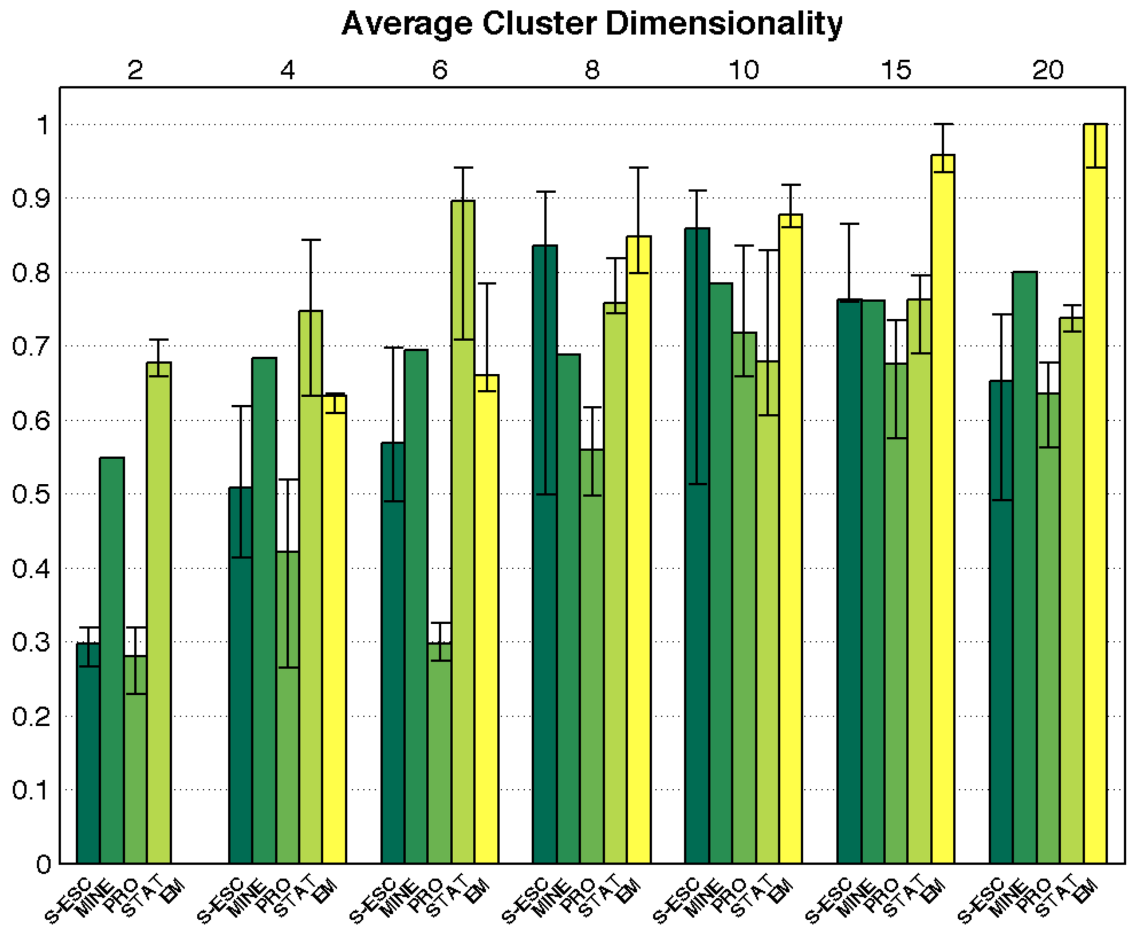


Figure 6.26: Micro F-measure comparison between S-ESC and comparator methods for the incremental UD experiment including outliers. Bar order: S-ESC, MINECLUS, PROCLUS, STATPC and EM.

parameter sweeps to do so. In the case of experiments with regards to the Uniform distribution (Figure 6.26) MINECLUS was the most consistent subspace algorithm, bettering S-ESC in 4 out of 7 data sets. Despite having an integrated outlier detector, STATPC appeared to be still sensitive to parameterization, with the lowest ranked performance in 4 of the 7 experiments. The fact 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.

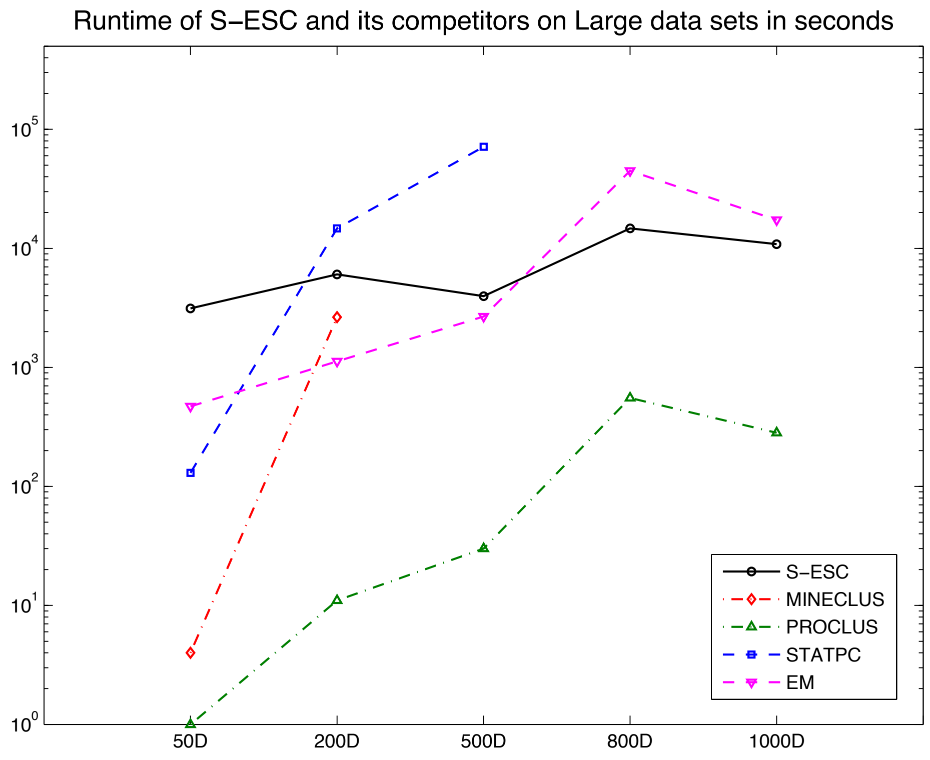


Figure 6.27: CPU runtime in seconds on Large-scale data sets. Lines between data points are for illustrative purposes alone.

6.5 Algorithm Runtime

Runtime is assessed by running each algorithm on a common computing platform.[1] Given that we are interested only in the execution time to complete tasks of increasing size, a single run is performed on each algorithm using each of the large-scale data sets (Table 5.2). Naturally, some variation can occur between different runs, but even in the case of S-ESC the principal algorithmic factors are constant across all runs, i.e. host population size (100), RSS sample size (100) and generation count (1,000). The major sources of variation come from the cardinality and dimensionality of the benchmarks. Figure 6.27 summarizes the respective runtimes for each of the clustering algorithms.

The profiles for MINECLUS and STATPC are not complete in the sense that neither algorithm returns results after 24 and 96 hours of CPU time, respectively, for

[1]16 core Intel Xeon 2.67Ghz server, 48GB RAM, Linux CentOS 5.5.

the larger data sets. Taken at face value PROCLUS is always the fastest algorithm by one or two orders of magnitude. However, PROCLUS does require that the correct number of clusters and the average attribute support 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 reduces the impact of the curse of dimensionality. 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 approximately an order of magnitude shorter runtime. As the data sets increase, however, this advantage is lost, with EM taking longer on the larger data sets. 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 rather than the whole data set. Thus, between the smallest and largest tasks, S-ESC undergoes the least increase in runtime by less than an order of magnitude, whereas all the 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 [51], thus for the case of a bi-objective EMO the cost is $O(GH \log H)$; where G is the number of generations and H is the host population size. In the case of the distance estimation, the cost of compactness objective is linear, whereas the cost of the connectivity objective is $O(N^2)$ without subsampling and $O(P \times N)$ with subsampling, where N is the total number of exemplars and P is the size of the RSS point population [43]. Further simplifications have been proposed, but these are again implementation specific[2].

6.6 Discussion

Each clustering method requires some form of prior information: MINECLUS requires knowledge of the relevant number of clusters; PROCLUS requires both the relevant cluster count and the (average) cluster dimensionality; EM and STATPC benefit from parameter tuning, STATPC in particular. By contrast, S-ESC was used with

[2]For example time complexity of connectivity (without subsampling) can be alleviated to $O(MN \log N)$ based on [113], where M is the number nearest neighbours needed for each instance.

a common parameterization throughout all of the experiments, and prior knowledge was limited to assuming that the cluster count and attribute support per cluster lay somewhere between 2 and 20. Various previous benchmarking studies have introduced specific data sets for evaluating subspace clustering algorithms, e.g. [88, 80, 83]. A start was made here by taking the data sets from one of the more 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 since, 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 top 10% solutions, thus creating a bias in favour of STATPC. PROCLUS has a stochastic component and 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 the most consistently, albeit with EM unable to provide any information regarding cluster attribute support. Moreover, it was necessary to sweep EM parameters to achieve this. STATPC, when provided with the necessary information regarding cluster purity, was effective on the incremental benchmarks, but was not effective with the 'large-scale' data sets. Indeed, neither STATPC nor MINECLUS scaled to all of the larger data sets. 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, the results are either weak and sit below the 1st quartile due to improper choices for k or are all equally good and sit *between* the 1st and 3rd quartiles[3] . PROCLUS benefitted from the most prior information, but the only real benefit this appears to have had was in terms of not failing to return results in the large-scale experiments. What was certainly surprising was how well the EM algorithm performed. Part of this was made possible

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

by conducting parameter sweeps to optimize the algorithm on the incremental data sets. However, the size of the large-scale data sets precluded a similar treatment. Thus, the EM algorithm failed to return results most frequently in the cases of clusters (data sets) with small support (dimension), e.g. Figures 6.1 to 6.4, 6.6 and 6.10.

Chapter 7

Conclusion

A bi-objective evolutionary subspace clustering algorithm has been introduced in which symbiosis provides the metaphor for the coevolution of both cluster centroids and clustering solutions. In a symbiotic relationship these entities are called symbionts and hosts respectively. Assuming such a symbiotic relationship appears to provide a more effective model for the sharing of promising centroids among multiple clustering solutions while promoting specialization. Moreover, variation operators can be designed such that they focus on distinct aspects of the combinatorial search in each population. Adopting a bi-objective approach enables the S-ESC algorithm to model clusters with a wide range of distributions – a property which 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.

An extensive benchmarking study built on the approach established in previous research – in particular the study of Moise *et al.* was pursued (54 data sets in 10 different experiments evaluating the effect of 9 different parameters in designing a data set). Additional experiments were designed to assess the impact of higher dimensions/cardinality and clusters introduced to include multiple factors simultaneously into the subspace clustering task. Relative to the set of comparator algorithms (EM, MINECLUS, PROCLUS, STATPC) and the goals established in the Introduction for a subspace clustering algorithm (Section 1.1) the following general trends were demonstrated:

1. Scalability. Individual representation and subsampling were effective at increasing the scalability to higher dimensionality and larger cardinality data sets respectively. This is increasingly important when the dimensionality of the data set or the vector length of training instances increases;

2. Simplify results. It is clear that S-ESC retains the ability to locate accurate

clusters of low dimensionality under multiple variations of the subspace clustering task;

3. Self determination of cluster counts: This is a key property of the S-ESC algorithm and was explicitly quantified by the use of the micro F-measure as the post training performance objective. Conversely, MINECLUS and PROCLUS require appropriate prior information and EM uses k-fold cross validation to answer this question;

4. Minimal prior knowledge. S-ESC can be deployed without a lot of prior trials to optimize parameters. In this respect S-ESC was unique;

5. Detect arbitrary cluster shapes. This is facilitated through the use of a bi-objective Pareto formulation for fitness. Conversely, other subspace clustering algorithms tended to have a preference towards the detection of Gaussian style clusters;

6. Detection of noise attributes and outliers. In this respect STATPC performed better, albeit but only through reference to label information to select the relevant parameterization;

7. Insensitivity to instance order. This was satisfied through adopting the stochastic sampling procedure. However, no specific tests were performed to determine the sensitivity of the other algorithms;

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

1. Streaming capabilities are becoming increasingly important. Although stochastic subsampling was effective at decoupling the cost of fitness evaluation from the ultimate cardinality of the data, issues regarding cluster drift/tracking are not currently addressed by S-ESC.

2. The initial grid is based on the ‘axis-parallel’ constraint whereas other projections might be investigated for the purpose of relaxing this constraint.

3. Tools for visualizing the resulting subspace clusters would aid in the interpretation of the resulting clusters. However, this is not as straightforward as full-space clustering as clusters do not share a common plane of reference cf., attribute space.

Appendices

Appendix A

RSS Parameterization

To understand the effect of RSS point population size (a.k.a. active set size) an experiment was conducted using two of the Moise sets as well as the large-scale data sets. Figures A.1 and A.2 show the micro F-measure for 7 data sets from GD and UD experiments respectively (Table 5.1). Each data set is evaluated by the S-ESC algorithm using 3 different settings, hence there are three bar for each data set. The left-most bar gives the performance distribution for knee solutions when 50 data points are used as the active set for evaluating each solution. These data points are refreshed anew at the beginning of each generation. The middle bar in each batch of bars gives the performance distribution when the RSS point population size is set to 100 points and the right-most bar is the results when RSS is deactivated and therefore all the data set points (less than 300 instances) are used to evaluate each solution's fitness values.

As can be seen in Figure A.1 the micro F-measure in the three experiments (using 50, 100 and all instances) is not always consistent. For the first data set with 2 relevant attributes per cluster (the first batch of bars) the best results are obtained when the RSS size is the smallest (50) whereas the second data set with 4 relevant attributes per cluster gives the best results when RSS is deactivated (i.e. when the RSS size is equal to the data set size). Also, there is a case in which 100 is the preferred size of the RSS point population size (i.e. the fifth data set with 10 relevant attributes per cluster).

Figure A.2 is similar to the previous figure in that there does not seem to be a consistent increasing or decreasing trend and one cannot make a clear and obvious decision about the RSS population size. However there does not seem to be a big difference between the different choices of RSS population size and the performance is affected minimally by the RSS sample size.

One reason why increasing the RSS sample size does not result in an expected

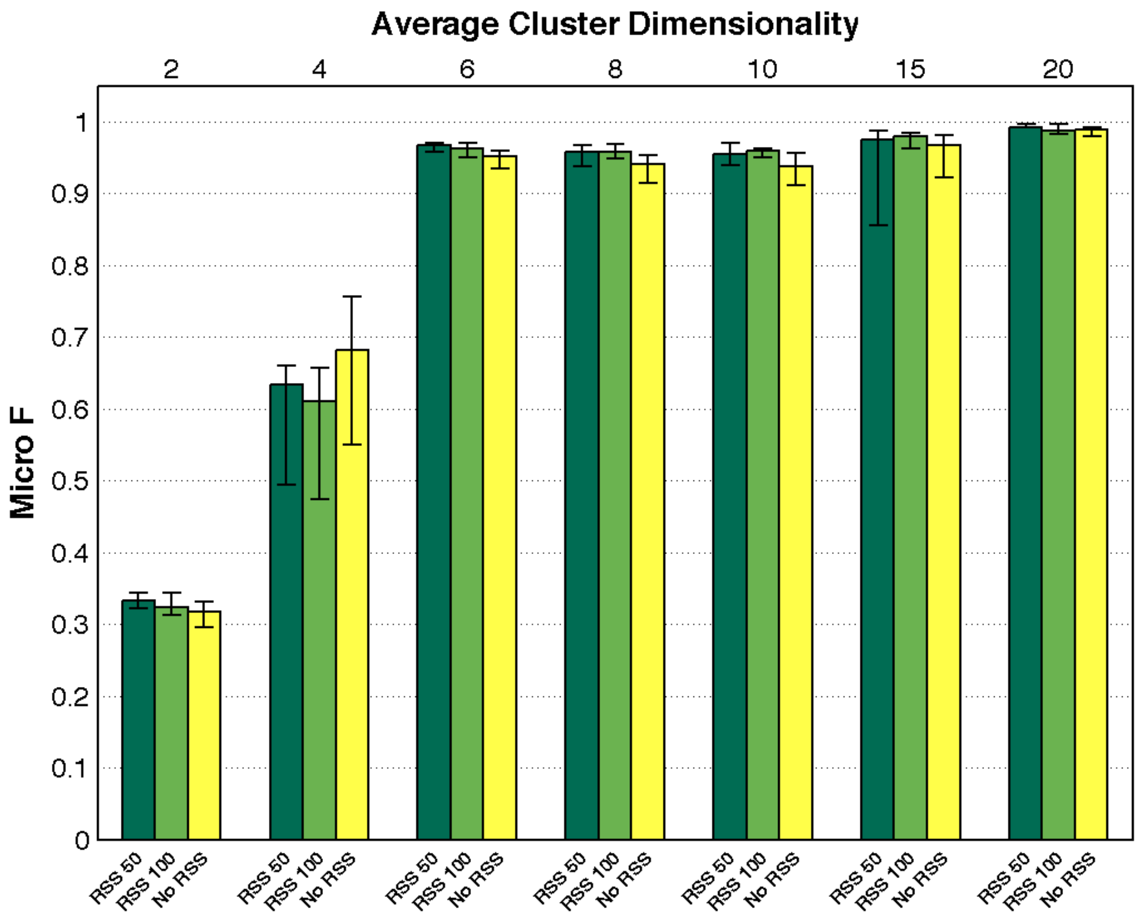


Figure A.1: Micro F-measure of S-ESC with different RSS sample sizes for the incremental *GD* experiment. Bar order: RSS = 50, RSS = 100 and No RSS.

increase of performance might be the fact that Moise the data sets are very small in size (up to 300 instances per data set). Consequently, the experiment is continued with the large-scale data sets. However, for the third part of the experiment, instead of running the algorithm with RSS deactivated, the S-ESC algorithm is run with a RSS sample size of 500. The size of the large-scale data sets makes S-ESC prohibitive to be run with RSS deactivated.

Figure A.3 is more consistent with the anticipated increase in performance measures as an effect of increasing the RSS sample size due to a better representation of the data sets for each individual evaluation. Although the F-measure distribution for the first three data sets (50D, 200D and 500D) does not change by much (very close to unity for all three different sample sizes) the performance improvement is more prominent for the last two data sets (800D and 1000D). Note that the 800D

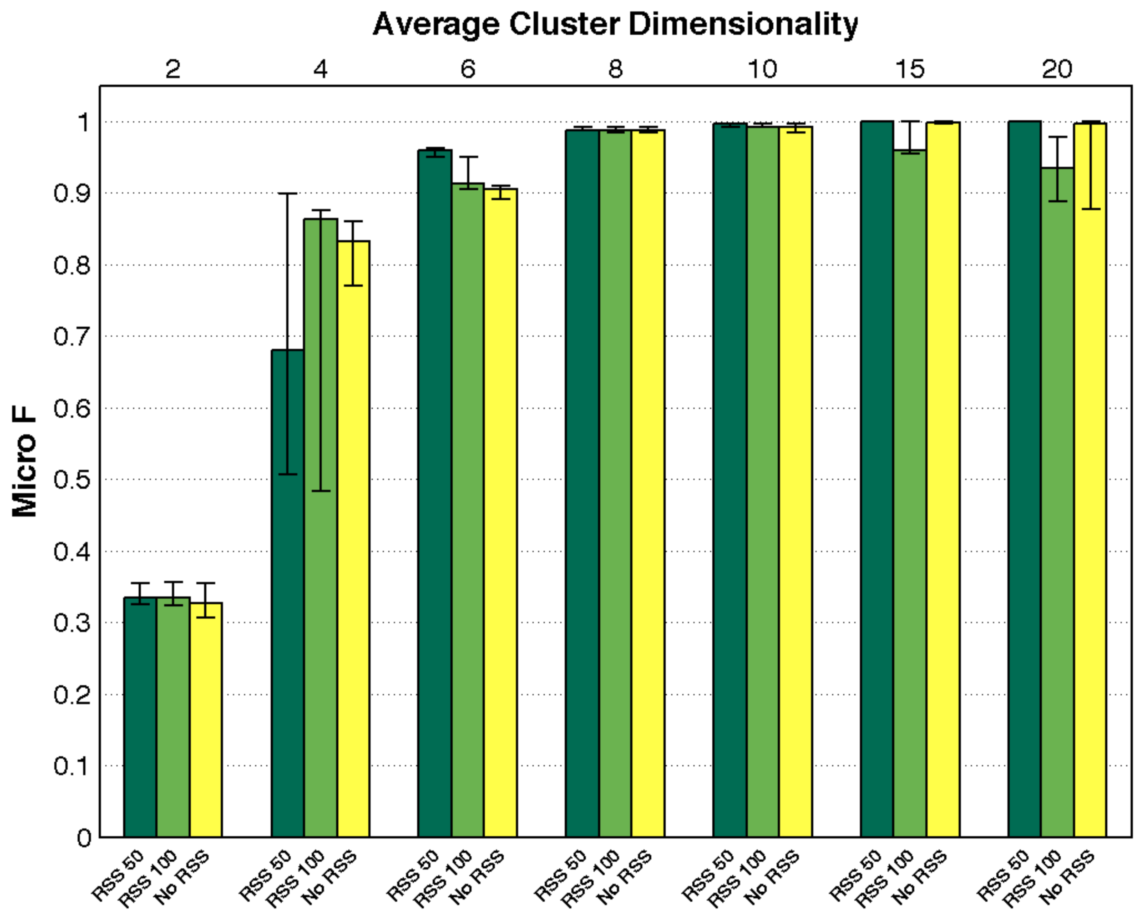


Figure A.2: Micro F-measure of S-ESC with different RSS sample sizes for the incremental UD experiment. Bar order: RSS = 50, RSS = 100 and No RSS.

and 1000D data sets are the most difficult data sets to cluster with respect to multiple features including: the largest data set size (800D), different attribute support counts for different clusters (800D), multiple data distributions within the same data set (800D), low attribute support relative to data set dimensionality (1000D) and a high noise to relevant attribute ratio (1000D).

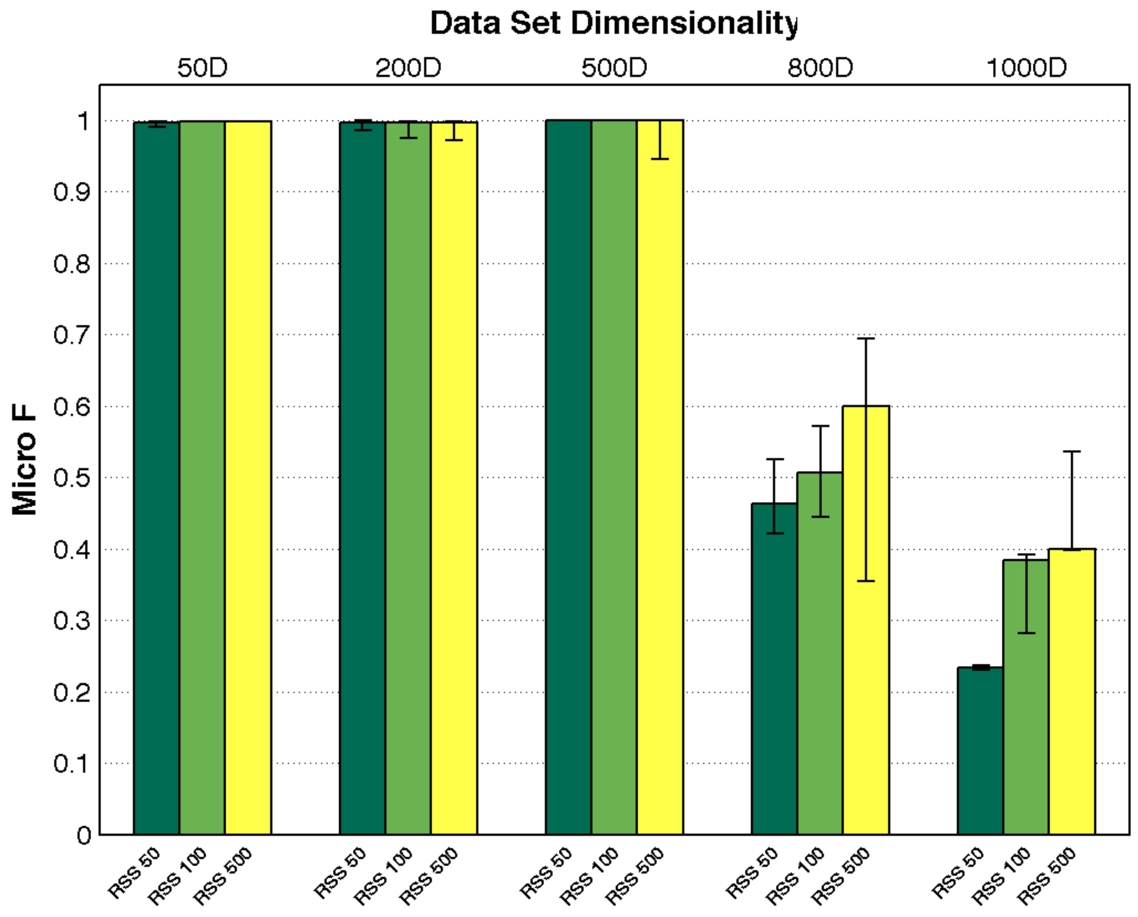


Figure A.3: Micro F-measure of S-ESC with different RSS sample sizes for the large-scale data sets. Bar order: RSS = 50, RSS = 100 and RSS = 500.

Appendix B

Hypothesis 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 data sets 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 methods 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 B.1 to B.2) 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 B.1 to B.2). 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.

Out of the $5 \times 4 = 20$ tests on the large-scale data sets of Table 5.2 there are

5 cases in which comparator methods (MINECLUS and STATPC) fail to produce a result (NaN's in Table B.3) and 5 cases in which the results are not significantly different (the bold cases in Table B.3). 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.

Hypothesis tests for comparing F-ESC (with a different population sizes) and S-ESC (with population size equal to 100) are presented in Table B.4. Only in 3 cases (shown in bold) are the results not significantly different. S-ESC outperforms F-ESC in the other 12 cases.

Table B.5 shows the t-test p values for comparison between S-ESC and F-ESC on the GD and UD experiments by Moise *et al.* The distribution of the results in all 14 cases is significantly different between S-ESC and F-ESC and in only 1 case is S-ESC outperformed by F-ESC.

The last table (Table B.6) shows the hypothesis test p values for comparing S-ESC against competitors in the GD and UD incremental experiments with outliers included. This hypothesis test is however a Kruskal-Wallis non-parametric test instead of a student t-test because the distributions are not necessarily Normal. In 16 out of 56 (14×4) cases the results are not significantly different and in 1 case EM is not able to return results.

Table B.1: The t-test p values for F-measure significance of the incremental benchmark data sets (Section 6.1, Table 5.1). The numbers in parentheses define the specific data set to be tested. For the case of the GE, GD, UE and UD experiments, it is the average dimensionality of the data set. For the D, N and k experiments, it is the dimensionality, cardinality and cluster count of the data set, 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 $ClusterSize$ experiment, it is the average instance count of the clusters.

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

Table B.2: continued from Table B.1

Data set	**MINECLUS vs. S-ESC**	**PROCLUS vs. S-ESC**	**STATPC vs. S-ESC**	**EM vs. S-ESC**
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**
Extent (0.1)	6.55E-02	5.07E-03	2.45E-19	1.23E-12
Extent (0.2)	6.00E-09	7.70E-06	9.64E-53	1.18E-07
Extent (0.3)	3.22E-52	**0.15**	9.25E-28	2.61E-19
Extent (0.4)	7.24E-30	**0.13**	2.05E-16	NaN
Overlap (0.0)	1.04E-16	**0.49**	1.27E-32	5.43E-53
Overlap (0.1)	2.57E-09	3.93E-06	0	3.08E-56
Overlap (0.2)	0	**0.35**	0	1.11E-70
Overlap (0.3)	1.67E-05	**0.05**	9.47E-07	NaN
ClustSz (30)	**0.86**	8.45E-05	1.69E-11	NaN
ClustSz (40)	1.26E-13	3.54E-05	1.83E-13	9.85E-25
ClustSz (50)	7.92E-10	0	1.29E-05	4.40E-40
ClustSz (55)	1.96E-07	1.13E-06	0	**0.02**

Table B.3: The t-test p values for F-measure significance in the Large-scale benchmark data sets (Section 6.2, Table 5.2).

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

Table B.4: The t-test p values for comparison between S-ESC vs. F-ESC with different population sizes in the Large-scale data sets (Section 6.3)

Data set	**F-ESC (100) vs. S-ESC**	**F-ESC (200) vs. S-ESC**	**F-ESC (500) vs. S-ESC**
50D	3.27E-25	**0.09**	**0.01**
200D	2.78E-14	3.84E-08	1.17E-05
500D	4.87E-27	1.46E-06	**0.42**
800D	1.80E-31	1.12E-13	1.10E-19
1000D	2.66E-20	2.36E-23	1.95E-09

Table B.5: The t-test p values for comparison between S-ESC vs. F-ESC in the Moise GD and UD data sets (Section 6.3)

Data set	**F-ESC vs. S-ESC**	**Data set**	**F-ESC vs. S-ESC**
GD (2)	0	**UD (1)**	2.50E-10
GD (4)	4.92E-19	**UD (2)**	6.53E-64
GD (6)	3.42E-30	**UD (3)**	6.04E-10
GD (8)	1.39E-59	**UD (4)**	5.44E-08
GD (10)	1.04E-08	**UD (5)**	0
GD (15)	3.10E-14	**UD (6)**	1.18E-13
GD (20)	7.49E-25	**UD (7)**	1.38E-75

Table B.6: The Kruskal-Wallis non-parametric test for comparison between S-ESC and the comparator methods on the data sets with outliers (Section 6.4)

Data set	**MINECLUS vs. S-ESC**	**PROCLUS vs. S-ESC**	**STATPC vs. S-ESC**	**EM vs. S-ESC**
GD (2)	6.63E-11	0	6.52E-12	0
GD (4)	1.04E-09	9.43E-07	2.85E-12	5.90E-14
GD (6)	**0.26**	1.12E-05	9.71E-09	1.34E-13
GD (8)	**0.19**	0	**0.05**	2.19E-20
GD (10)	7.41E-08	5.12E-06	**0.28**	2.10E-21
GD (15)	1.21E-12	**0.03**	7.32E-12	1.66E-22
GD (20)	1.57E-12	1.39E-05	4.48E-11	6.33E-23
UD (2)	1.72E-11	**0.47**	5.70E-12	NaN
UD (4)	0	**0.02**	7.29E-12	0
UD (6)	**0.02**	1.25E-06	9.19E-12	1.89E-06
UD (8)	**0.22**	**0.07**	**0.18**	0.009
UD (10)	**0.11**	**0.25**	**0.74**	0
UD (15)	0.007	0	**0.09**	7.13E-22
UD (20)	0	**0.55**	5.73E-08	1.54E-24

Bibliography

[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] Charu C. Aggarwal and Philip S. Yu. Finding generalized projected clusters in high dimensional spaces. In *ACM SIGMOD International Conference on Management of Data*, SIGMOD, pages 70–81, New York, NY, USA, 2000. ACM.

[3] Rakesh Agrawal, Johannes Gehrke, Dimitrios Gunopulos, and Prabhakar Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. *ACM SIGMOD International Conference on Management of Data*, 27:94–105, Jun. 1998.

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

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

[6] Ira Assent, Ralph Krieger, Emmanuel Muller, and Thomas Seidl. INSCY: Indexing subspace clusters with in-process-removal of redundancy. *IEEE International Conference on Data Mining*, 0:719–724, 2008.

[7] 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.

[8] Phanendra G. Babu and Narasimha M. Murty. A near-optimal initial seed value selection in k-means means algorithm using a genetic algorithm. *Pattern Recognition Letters*, 14(10):763–769, 1993.

[9] 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.

[10] Sanghamitra Bandyopadhyay, Ujjwal Maulik, and Anirban Mukhopadhyay. Multiobjective genetic clustering for pixel classification in remote sensing imagery. *IEEE Transactions on Geoscience and Remote Sensing*, 45(5):1506–1511, 2007.

[11] Slim Bechikh, Lamjed Ben Said, and Khaled Ghédira. Searching for knee regions in multi-objective optimization using mobile reference points. In *ACM Symposium on Applied Computing*, pages 1118–1125. ACM, 2010.

[12] Pavel Berkhin. Survey of clustering data mining techniques. Technical report, Accrue Software, Inc., San Jose, CA, USA, 2002.

[13] Pavel Berkhin. A survey of clustering data mining techniques. In *Grouping Multidimensional Data*, pages 25–71. Springer, 2006.

[14] James C. Bezdek. *Pattern recognition with fuzzy objective function algorithms.* Kluwer Academic Publishers, 1981.

[15] James C. Bezdek, Srinivas Boggavarapu, Lawrence O. Hall, and Amine Bensaid. Genetic algorithm guided clustering. In *IEEE World Congress on Computational Intelligence*, volume 1, pages 34–39, Jun. 1994.

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

[17] Jürgen Branke, Kalyanmoy Deb, Henning Dierolf, and Matthias Osswald. Finding knees in multi-objective optimization. In *Parallel Problem Solving from Nature (PPSN VIII)*, pages 722–731. Springer, 2004.

[18] Brett Calcott, Kim Sterelny, and Eörs Szathmáry. *The Major Transitions in Evolution revisited.* The Vienna Series in Theoretical Biology. MIT Press, 2011.

[19] Arantza Casillas, MT González De Lena, and R. Martínez. Document clustering into an unknown number of clusters using a genetic algorithm. In *Text, Speech and Dialogue*, pages 43–49. Springer, 2003.

[20] Jae-Woo Chang and Du-Seok Jin. A new cell-based clustering method for large, high-dimensional data in data mining applications. In *ACM Symposium on Applied computing*, SAC, pages 503–507, New York, NY, USA, 2002. ACM.

[21] Chun-Hung Cheng, Ada Waichee Fu, and Yi Zhang. Entropy-based subspace clustering for mining numerical data. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD, pages 84–93, New York, NY, USA, 1999. ACM.

[22] Stephen L. Chiu. Fuzzy model identification based on cluster estimation. *Journal of Intelligent and Fuzzy Systems*, 2(3):267–278, 1994.

[23] 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*, pages 114–125, 2004.

[24] André L.V. Coelho, Everlândio Fernandes, and Katti Faceli. Inducing multi-objective clustering ensembles with genetic programming. *Neurocomputing*, 74(1):494–498, 2010.

[25] Carlos A. Coello Coello, Gary B. Lamont, and David A. Van Veldhuizen. *Evolutionary algorithms for solving multi-objective problems.* Springer, 2007.

[26] Jason M. Daida, Catherine S. Grasso, Stephen A. Stanhope, and Steven J. Ross. Symbionticism and complex adaptive systems I: Implications of having symbiosis occur in nature. In *Evolutionary Programming*, pages 177–186, 1996.

[27] César S. De Oliveira, Aruanda S.G. Meiguins, Bianchi S. Meiguins, P.I. Godinho, and Alex A. Freitas. An evolutionary density and grid-based clustering algorithm. In *XXIII Brazilian Symposium on Databases (SBBD)*, pages 175–189, 2007.

[28] Kalyanmoy Deb. Multi-objective optimization. *Multi-Objective Optimization Using Evolutionary Algorithms*, pages 13–46, 2001.

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

[30] 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)*, 39:1–38, 1977.

[31] John Doucette and Malcolm I. Heywood. Revisiting the acrobot heighttask: An example of efficient evolutionary policy search under an episodic goal seeking task. In *IEEE Congress on Evolutionary Computation (CEC)*, pages 468–475. IEEE, 2011.

[32] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification.* John Wiley & Sons, 2001.

[33] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *ACM SIGKDD International Conference on Knowledge and Data Discovery*, pages 226–231. KDD, 1996.

[34] Vladimir Estivill-Castro and Alan T. Murray. *Spatial clustering for data mining with genetic algorithms.* Queensland University of Technology Australia, 1997.

[35] Katti Faceli, André C.P.L.F. de Carvalho, and Marcílio C.P. de Souto. Multi-objective clustering ensemble. *International Journal of Hybrid Intelligent Systems*, 4(3):145–156, 2007.

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

[37] Xiaoli Zhang Fern and Carla E. Brodley. Solving cluster ensemble problems by bipartite graph partitioning. In *21st International Conference on Machine Learning*, pages 36–43. ACM, 2004.

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

[39] Chris Fraley and Adrian E. Raftery. How many clusters? which clustering method? answers via model-based cluster analysis. *The Computer Journal*, 41(8):578–588, 1998.

[40] Jerome H. Friedman and Jacqueline J. Meulman. Clustering objects on subsets of attributes (with discussion). *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 66(4):815–849, 2004.

[41] Sanjay Goil, Harsha Nagesh, and Alok Choudhary. MAFIA: Efficient and scalable subspace clustering for very large data sets. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 443–452, 1999.

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

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

[44] John A. Hartigan. *Clustering Algorithms*. John Wiley & Sons, Inc., 1975.

[45] John A. Hartigan and Manchek A. Wong. Algorithm AS 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):100–108, 1979.

[46] Jeffrey Horn, Nicholas Nafpliotis, and David E. Goldberg. A niched pareto genetic algorithm for multiobjective optimization. In *IEEE World Congress on Computational Intelligence*, pages 82–87. IEEE, 1994.

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

[48] Eduardo R. Hruschka and Nelson F.F. Ebecken. A genetic algorithm for cluster analysis. *Intelligent Data Analysis*, 7(1):15–25, 2003.

[49] Anil K. Jain and Richard C. Dubes. *Algorithms for clustering data*. Prentice-Hall, Inc., 1988.

[50] Anil K. Jain, M. Narasimha Murty, and Patrick J. Flynn. Data clustering: A review. *ACM Computing Surveys (CSUR)*, 31(3):264–323, 1999.

[51] 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.

[52] Daxin Jiang, Chun Tang, and Aidong Zhang. Cluster analysis for gene expression data: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 16(11):1370–1386, 2004.

[53] Yaochu Jin, editor. *Multi-objective Machine Learning*, volume 16 of *Studies in Computational Intelligence*. Springer, 2006.

[54] Liping Jing, Michael K. Ng, Jun Xu, and Joshua Zhexue Huang. Subspace clustering of text documents with feature weighting k-means algorithm. In *Advances in Knowledge Discovery and Data Mining*, pages 802–812. Springer, 2005.

[55] Karin Kailing, Hans-Peter Kriegel, and Peer Kröger. Density-connected subspace clustering for high-dimensional data. In *SIAM Conference on Discrete Mathematics*, pages 246–257, 2004.

[56] Yeong Seog Kim, W. Nick Street, and Filippo Menczer. Feature selection for unsupervised learning via evolutionary search. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 365–369, 2000.

[57] Juha Kivijärvi, Pasi Fränti, and Olli Nevalainen. Self-adaptive genetic algorithm for clustering. *Journal of Heuristics*, 9(2):113–129, 2003.

[58] 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.

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

[60] Emin Erkan Korkmaz, Jun Du, Reda Alhajj, and Ken Barker. Combining advantages of new chromosome representation scheme and multi-objective genetic algorithms for better clustering. *Intelligent Data Analysis*, 10(2):163–182, 2006.

[61] Hans-Peter Kriegel, Peer Kröger, Matthias Renz, and Sebastian Wurst. A generic framework for efficient subspace clustering of high-dimensional data. In *IEEE International Conference on Data Mining*, pages 250–257, Nov. 2005.

[62] 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.

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

[64] K. Krishna and M. Narasimha Murty. Genetic k-means algorithm. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 29(3):433–439, 1999.

[65] Martin Krzywinski, Jacqueline Schein, İnanç Birol, Joseph Connors, Randy Gascoyne, Doug Horsman, Steven J. Jones, and Marco A. Marra. Circos: an information aesthetic for comparative genomics. *Genome Research*, 19(9):1639–1645, 2009.

[66] Ludmila I. Kuncheva and James C. Bezdek. Selection of cluster prototypes from data by a genetic algorithm. In *European Congress on Intelligent Techniques and Soft Computing*, pages 1683–1688, 1997.

[67] William B. Langdon. Large scale bioinformatics data mining with parallel genetic programming on graphics processing units. In *Parallel and Distributed Computational Intelligence*, pages 113–141. Springer, 2010.

[68] Bing Liu, Yiyuan Xia, and Philip S. Yu. Clustering through decision tree construction. In *International Conference on Information and Knowledge Management (CIKM)*, pages 20–29, New York, NY, USA, 2000. ACM.

[69] Huan Liu and Hiroshi Motoda. Feature transformation and subset selection. *IEEE Intelligent Systems and Their Applications*, 13(2):26–28, 1998.

[70] José Antonio Lozano and Pedro Larrañaga. Applying genetic algorithms to search for the best hierarchical clustering of a dataset. *Pattern Recognition Letters*, 20(9):911–918, 1999.

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

[72] Yi Lu, Shiyong Lu, Farshad Fotouhi, Youping Deng, and Susan J. Brown. FGKA: A fast genetic k-means clustering algorithm. In *ACM Symposium on Applied Computing*, pages 622–623. ACM, 2004.

[73] Carlos B. Lucasius, Adrie D. Dane, and Gerrit Kateman. On k-medoid clustering of large data sets with the aid of a genetic algorithm: background, feasiblity and comparison. *Analytica Chimica Acta*, 282(3):647–669, 1993.

[74] James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *5th Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297. California, USA, 1967.

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

[76] Ujjwal Maulik and Sanghamitra Bandyopadhyay. Genetic algorithm-based clustering technique. *Pattern recognition*, 33(9):1455–1465, 2000.

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

[78] 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 SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 533–541. ACM, 2008.

[79] Gabriela Moise, Jörg Sander, and Martin Ester. P3C: A robust projected clustering algorithm. *IEEE International Conference on Data Mining*, 0:414–425, 2006.

[80] 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.

[81] Luis Carlos Molina, Lluís Belanche, and Àngela Nebot. Feature selection algorithms: A survey and experimental evaluation. In *IEEE International Conference on Data Mining (ICDM)*, pages 306–313. IEEE, 2002.

[82] Anirban Mukhopadhyay, Ujjwal Maulik, and Sanghamitra Bandyopadhyay. Multiobjective genetic fuzzy clustering of categorical attributes. In *International Conference on Information Technology (ICIT)*, pages 74–79. IEEE, 2007.

[83] 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, Aug. 2009.

[84] Raymond T. Ng and Jiawei Han. Efficient and effective clustering methods for spatial data mining. In *International Conference on Very Large Data Bases*, pages 487–499, 1994.

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

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

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

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

[89] 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.

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

[91] 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.

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

[93] 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.

[94] Kazi Shah Nawaz Ripon, Chi-Ho Tsang, Sam Kwong, and Man-Ki Ip. Multi-objective evolutionary clustering using variable-length real jumping genes genetic algorithm. In *International Conference on Pattern Recognition (ICPR)*, volume 1, pages 1200–1203. IEEE, 2006.

[95] G. Sangeetha and Sornamaheswari. Density conscious subspace clustering for high dimensional data using genetic algorithms. *International Journal of Computer Applications*, 10(4), 2010.

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

[97] Ioannis A. Sarafis, Phil W. Trinder, and Ali Zalzala. NOCEA: a rule-based evolutionary algorithm for efficient and effective clustering of massive high-dimensional databases. *Applied Soft Computing*, 7(3):668–710, 2007.

[98] James David Schaffer. Some experiments in machine learning using vector evaluated genetic algorithms. Technical report, Vanderbilt University, Nashville, TN (USA), 1985.

[99] 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.

[100] Karlton Sequeira and Mohammed Zaki. SCHISM: A new approach for interesting subspace mining. *IEEE International Conference on Data Mining*, 0:186–193, 2004.

[101] Weiguo Sheng and Xiaohui Liu. A hybrid algorithm for k-medoid clustering of large data sets. In *IEEE Congress on Evolutionary Computation*, volume 1, pages 77–82. IEEE, 2004.

[102] Weiguo Sheng, Stephen Swift, Leishi Zhang, and Xiaohui Liu. A weighted sum validity function for clustering with a hybrid niching genetic algorithm. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 35(6):1156–1167, 2005.

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

[104] J. R. Slagle, C. L. Chang, and S. R. Heller. A clustering and data-reorganizing algorithm. *IEEE Transactions on Systems, Man and Cybernetics*, SMC-5(1):125–128, 1975.

[105] Dong Song, Malcolm I. Heywood, and A. Nur Zincir-Heywood. Training genetic programming on half a million patterns: an example from anomaly detection. *IEEE Transactions on Evolutionary Computation*, 9(3):225–239, Jun. 2005.

[106] E. Stanley Lee and Rong Jun Li. Fuzzy multiple objective programming and compromise programming with pareto optimum. *Fuzzy Sets and Systems*, 53(3):275–288, 1993.

[107] Hao-jun Sun and Lang-huan Xiong. Genetic algorithm-based high-dimensional data clustering technique. In *International Conference on Fuzzy Systems and Knowledge Discovery (FSKD)*, volume 1, pages 485–489. IEEE, 2009.

[108] Anil Kumar Tiwari, Lokesh Kumar Sharma, and G. Rama Krishna. Entropy weighting genetic k-means algorithm for subspace clustering. *International Journal of Computer Applications*, 7(7):90–105, 2010.

[109] Alexander Topchy, Anil K. Jain, and William Punch. A mixture model of clustering ensembles. In *SIAM International Conference on Data Mining*, 2004.

[110] Lin Yu Tseng and Shiueng Bien Yang. A genetic approach to the automatic clustering problem. *Pattern Recognition*, 34(2):415–424, 2001.

[111] Ali Vahdat, Malcolm I. Heywood, and A. Nur Zincir-Heywood. Bottom-up evolutionary subspace clustering. In *IEEE Congress on Evolutionary Computation*, pages 1–8, Jul. 2010.

[112] Ali Vahdat, Malcolm I. Heywood, and A. Nur Zincir-Heywood. Symbiotic evolutionary subspace clustering. In *IEEE Congress on Evolutionary Computation*, pages 1–8, Jun. 2012.

[113] Pravin M. Vaidya. Ano (n logn) algorithm for the all-nearest-neighbors problem. *Discrete & Computational Geometry*, 4(1):101–115, 1989.

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

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

[116] Kyoung-Gu Woo, Jeong-Hoon Lee, Myoung-Ho Kim, and Yoon-Joon Lee. FINDIT: a fast and intelligent subspace clustering algorithm using dimension voting. *Information and Software Technology*, 46(4):255–271, 2004.

[117] Xuanli Lisa Xie and Gerardo Beni. A validity measure for fuzzy clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(8):841–847, 1991.

[118] Rui Xu and Donald Wunsch. Survey of clustering algorithms. *IEEE Transactions on Neural Networks*, 16(3):645–678, 2005.

[119] Rui Xu and Donald Wunsch. *Clustering*. IEEE Press Series on Computational Intelligence. Wiley, 2009.

[120] Jiong Yang, Wei Wang, Haixun Wang, and Philip Yu. δ-clusters: capturing subspace correlation in a large data set. In *International Conference on Data Engineering*, pages 517–528, 2002.

[121] Kevin Y. Yip, David W. Cheung, and Michael K. Ng. Harp: A practical projected clustering algorithm. *IEEE Transactions on Knowledge and Data Engineering*, 16(11):1387–1397, 2004.

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

[123] Man Lung Yiu and Nikos Mamoulis. Iterative projected clustering by subspace mining. *IEEE Transactions on Knowledge and Data Engineering*, 17(2):176–189, 2005.

[124] Hye-Sung Yoon, Sun-Young Ahn, Sang-Ho Lee, Sung-Bum Cho, and Ju Han Kim. Heterogeneous clustering ensemble method for combining different cluster results. In *Data Mining for Biomedical Applications*, pages 82–92. Springer, 2006.

[125] Bo Yuan, George J. Klir, and John F. Swan-Stone. Evolutionary fuzzy c-means clustering algorithm. In *IEEE International Conference on Fuzzy Systems*, volume 4, pages 2221–2226. IEEE, 1995.

[126] A Zhou, B.Y. Qu, H. Li, S.Z. Zhao, P.N. Suganthan, and Q. Zhang. Multiobjective evolutionary algorithms: A survey of the state of the art. *Swarm and Evolutionary Computation*, 1:32–49, 2011.

[127] Lin Zhu, Longbing Cao, and Jie Yang. Multiobjective evolutionary algorithm-based soft subspace clustering. In *IEEE Congress on Evolutionary Computation*, pages 1–8, Jun. 2012.

[128] Eckart Zitzler, Marco Laumanns, and Lothar Thiele. SPEA2: Improving the strength pareto evolutionary algorithm for multiobjective optimization. In *Evolutionary Methods for Design, Optimisation, and Control*, pages 95–100.