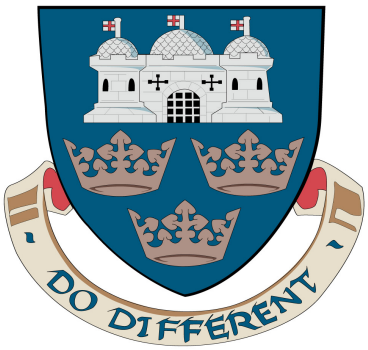
Multi-Objective Evolutionary Algorithms for Data Clustering

Oliver Andrew Kirkland

A thesis submitted for the Degree of
Doctor of Philosophy

University of East Anglia
School of Computing Sciences



October 2014

Abstract

In this work we investigate the use of Multi-Objective metaheuristics for the data mining task of clustering. We first investigate methods of evaluating the quality of clustering solutions, we then propose a new Multi-Objective clustering algorithm driven by multiple measures of cluster quality and then perform investigations into the performance of different Multi-Objective clustering algorithms.

In the context of clustering, a robust measure for evaluating clustering solutions is an important component of an algorithm. These Cluster Quality Measures (CQMs) should rely solely on the structure of the clustering solution. A robust CQM should have three properties: it should be able to reward a "good" clustering solution; it should decrease in value monotonically as the solution quality deteriorates and, it should be able to evaluate clustering solutions with varying numbers of clusters. We review existing CQMs and present an experimental evaluation of their robustness. We find that measures based on connectivity are more robust than other measures for cluster evaluation.

We then introduce a new Multi-Objective Clustering algorithm (MOCA). The use of Multi-Objective optimisation in clustering is desirable because it permits the incorporation of multiple measures of cluster quality. Since the definition of what constitutes a good clustering is far from clear, it is beneficial to develop algorithms that allow for multiple CQMs to be accommodated. The selection of the clustering quality measures to use as objectives for MOCA is informed by our previous work with internal evaluation measures. We explain the implementation details and perform experimental work to establish its worth. We compare MOCA with k-means

and find some promising results. We find that MOCA can generate a pool of clustering solutions that is more likely to contain the optimal clustering solution than the pool of solutions generated by k-means.

We also perform an investigation into the performance of different implementations of MOEA algorithms for clustering. We find that representations of clustering based around centroids and medoids produce more desirable clustering solutions and Pareto fronts. We also find that mutation operators that greatly disrupt the clustering solutions lead to better exploration of the Pareto front whereas mutation operators that modify the clustering solutions in a more moderate way lead to higher quality clustering solutions.

We then perform more specific investigations into the performance of mutation operators focussing on operators that promote clustering solution quality, exploration of the Pareto front and a hybrid combination. We use a number of techniques to assess the performance of the mutation operators as the algorithms execute. We confirm that a disruptive mutation operator leads to better exploration of the Pareto front and mutation operators that modify the clustering solutions lead to the discovery of higher quality clustering solutions. We find that our implementation of a hybrid mutation operator does not lead to a good improvement with respect to the other mutation operators but does show promise for future work.

Acknowlegments

I would like to thank my supervisor Beatriz De La Iglesia for her supervision. Her advice and never ending patience has made the last five years possible. Thanks to my viva examiners, Wenjia Wang and Alex Freitas and to all the lecturers who provided me with additional advice, support or opportunities at sometime or another including: Piere Chardaire, Tony Bagnall, Gavin Cawley and Barry Theobold and Stephen Laycock.

I would like to acknowledge Craig Howard and the staff at Evoke Systems for providing me the opportunity to work for them during and after my studies. I don't know where I would be without their support and exceptional flexibility.

Thanks to all the people I shared working space with and socialised with within the department. In no particular order: Sarah Taylor, Philip Harding, Jason Lines, Luke Davis, John Taylor, Jacob Newman, Osama Dorgham, Dominic Howell, Chris Watkins, Faheem Khan, Chris Asque, Andrea DeMarco, Chris Bocking, Felix Shaw, Helen Bear, Chris Applegate, Stuart Lynch, Mike Harris, Geoffrey French, Jao Bettencourt Da Silva and George Kettleburgh. I would also like to acknowledge all the people from UEA I played football with, I don't know how you survived my ineptitude. Thanks to all of my friends outside of CMP who have provided support, even though they had no idea what I was talking about, including: Michelle Sturman, Alex Morter, Mark Davidson, Donald Ashworth, Mark Beeby, Lauren Gollop, Alex Swain, Oliver Henderson, Andy Cadley, Andy Courtenay, Justin Breeze, Daniel Norton, Julian Polzella, Alex Jenkins, Steve Watson and Taya Allen.[1]

Finally, I would like to thank my parents, Monica and Ivan, as without them none of this would have been possible.

[1]If I have forgotten anyone I'l buy you a pint to make up for it... There are a lot of you!

Contents

List of Abbreviations

Abbreviation	Meaning
CBRE	Centroid Based Real Encoding
CLARA	Clustering LARge Applications
CLARANS	Clustering Large Applications based upon RANdomized Search
CQM	Cluster Quality Measures
DB	Davies-Bouldin index
DBSCAN	Density Based Spatial Clustering of Applications with Noise
GA	Genetic Algorithms
GD	Generational Distance
HM	Hybrid Mutation
HypE	Hypervolume E Optimisation
IGD	Inverted Generational Distance
KMA	k-Means Algorithm
KMLM	k-Means Like Mutation
LBIE	Label Based Integer Encoding
MBBE	Medoid Based Binary Encoding
MCDM	Multiple Criteria Decision Making
MO	Multi-Objective
MOCK	Multi-Objective Clustering with automatic determination of k
MOEA	Multi-Objective Evolutionary Algorithms
MOGA	Multiple Objective Genetic Algorithm
NPGA	Niched Pareto Genetic Algorithm
NSGA-II	Non-Dominated Sorting Genetic Algorithm II
OPTICS	Ordering Points to Identify the Clustering Structure
PAM	Partitioning Around Medoids
PESA- II	Pareto Evolutionary Strength Algorithm
RM	Randomness Mutation
RMSSTD	Root-Mean-Square Standard Total Deviation
RS	R-Squared
SMS-EMOA	S Metric Selection Evolutionary Multi-Objective Algorithm
SPEA	Strength Pareto Evolutionary Algorithm
SWC	Silhouette Width Criterion
VEGA	Vector Evaluated Genetic Algorithm
VIENNA	Voronoi Initialised Evolutionary Nearest-Neighbour Algorithm
VRC	Variance Ratio Criterion

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

Clustering is the process of dividing a set of observations into subsets in an unsupervised manner. It has a wide variety of applications within Knowledge Discovery and Data Mining and in other areas such as image processing [21, 131] and psychology [4]. Clustering is now a well developed field.

A large number of techniques have been investigated with an aim to solving the clustering problem [76]. This has led to the development of a large number of clustering algorithms. An inherent problem with clustering algorithms developed to date is the definition of what is a good solution to a clustering problem. There are a large number of conflicting definitions and opinions on what makes a good clustering. Clustering is also a very subjective problem and may be task dependant. It may be helpful in this context to find a range of potentially good clustering solutions.

Multi-Objective Evolutionary Algorithms (MOEA) [46, 72] are a class of algorithm that attempts to solve problems where there are conflicting objectives. That is to say, there is no single solution to the problem, instead they find a set of trade-off solutions. These algorithms are an interesting area of study as they apply the concepts of biological evolution to problem solving with good results in real world problems. They will suit our goal of trying to produce a range of solutions to clustering problems according to different objectives.

MOEAs have been previously applied to clustering problems successfully. We believe that there is room for further work on applying MOEAs to the clustering problem. We will investigate various methods of assessing the quality of clustering solutions to see if combining these methods together will lead to the discovery of a good range of clustering solutions. We will also investigate various ways of representing and manipulating clustering solutions within the context of Multi-Objective Evolutionary Algorithms.

1.1 Research Methodology

In broad terms a MOEA consists of: a representation of solutions to a problem, a method of initialising solutions, method(s) of assessing the fitness of solutions, a mutation operator to modify solutions, a crossover operator to combine solutions, a method for selecting solutions and a strategy to manage the Pareto front. In out work we will experiment with some of those, in particular representation, initialisation, fitness functions, mutation and crossover operators to create an effective MO clustering algorithm.

We begin by reviewing a number of Cluster Quality Measures (CQMs) as CQMs will be suitable fitness measures of an MOEA. We also review methods for comparing clustering solutions.We perform our experiment by degrading clustering solutions in a steady fashion on the assumption that we should also see a steady degradation of CQMs in a similar fashion. A good CQM should show a gradual improvement as a clustering solution approaches an optimal clustering solution.

We then review a number of MOEAs and MO tools to inform our decision of which MOEA to use. We are focussing our studies on applying existing MOEAs to the clustering problem and we do not propose any new strategies for maintaing a population of solutions or selecting solutions as part of a MOEA. We review a number of ways of representing a clustering solution that are compatible with an MOEA and ways of modifying these solutions using crossover and mutation

operators.

We also review and discuss a number of methods of assessing the quality of Pareto fronts and individual clustering solutions generated by an MOEA. We need to assess the quality of sets of solutions generated by an MOEA so we can determine if different ways of representing solutions, assessing the fitness of solutions with CQMs and manipulating solutions lead to higher quality sets of solutions.

We propose an MOEA for clustering with novel mutation and crossover operators. We base this upon an existing MOEA, representation of a clustering solution informed by our previous review and CQMs as the fitness functions based upon our previous experimentation. We then review the performance of this algorithm by assessing the quality of the clustering solutions generated.

We perform a more in depth study of representations and their manipulation within a MOCA. Finally, we perform a further experimental evaluation that focusses on the effects on performance of mutation operators.

1.2 Contributions

In this thesis we provide four main contributions in Chapters 3, 5, 6 and 7.

- In Chapter 3 proposes an experimental method for assessing CQMs based upon degrading solutions and assess a number of CQMs. This was originally published in [83].
- Chapter 5 proposes a MOCA that uses novel mutation and crossover operators. This was originally described in [85].
- Chapter 6 provides an experimental comparison of representations of clustering solutions and methods for manipulating these representations.
- Chapter 7 is a further experimental comparison that focusses upon the performance of mutation operators measured by the quality of the clustering

solutions produced and the quality of the Pareto fronts discovered. This was originally published in [84].

We also provide an extensive review of CQMs in Chapter 2 and a review of representations of clustering solutions and methods of manipulating them in Chapter 4.

1.3 Thesis Structure

Here we describe the content and arrangement of the other chapters of this thesis.

Chapter 2 serves as an introduction and background to the clustering problem. It provides some definitions that are used throughout the thesis. We introduce a number of algorithms used to solve clustering problems. We then introduce methods of assessing the quality of clustering solutions and the concept of Cluster Quality Measures (CQM). These are used throughout the thesis.

Chapter 3 defines an experiment to determine the performance of CQMs. This is important for our later work. The results of this experiment were published in [83].

Multi-Objective Evolutionary Algorithms (MOEA) are introduced in Chapter 4. We provide some overview of MOEAs and introduce methods of assessing the quality of the Pareto fronts produced by MOEAs.

Chapter 5 presents a MOEA we devised to solve clustering problems. We perform a brief evaluation of the algorithm to test its validity. Our results for this were presented in [85].

In Chapter 6, we define an experiment to assess combinations of the various MOEA operators and problem representations we defined in Chapters 4 and 5. This study focusses on the problem representation.

In Chapter 7, we present a further study focussing on the performance of various mutation operators. We also define enhancements to the mutation operator we defined in Chapter 5. The experiment and its conclusions were presented in [84].

Chapter 8 concludes the thesis. We present a discussion of our findings, and the problems and limitations we encountered. Finally, we present suggestions for future work.

Chapter 2

The Clustering Problem

Clustering algorithms divide a data set of observations (or objects) into a number of partitions or clusters. This is an important task in the Knowledge Discovery and Data mining (KDD) process, in visualisation, and in other contexts. Many clustering algorithms have been proposed. These are tailored to produce different results for specific types of data sets.

Here we introduce the problem of clustering, some of the relevant algorithms and the many, valid, clustering evaluation measures [25, 140] that may encapsulate the essential properties of clustering.

2.1 Problem Definitions

We define an object, $\vec{x}_i$, as a d dimensional feature vector, $\vec{x}_i = (x_{i1}, \ldots, x_{id})$, where $x_{ie} \in \mathbb{R}, e = 1...d$. A data set, $\mathcal{D}$, is a set of n of these objects $\mathcal{D} = \{\vec{x}_1, \ldots, \vec{x}_n\}$.

For any two objects, $\vec{x}_i$ and $\vec{x}_j$, we can measure the distance between them using a distance function, $\delta(\vec{x}_i, \vec{x}_j)$. This is used to give an indication of the similarity of any two given objects.

2.1.1 Distance Measure Definitions

A distance measure that is used for clustering must satisfy the following properties:

- All distances should be non-negative: $\delta(\vec{x}_i, \vec{x}_j) \in [0, \infty) \, \forall \vec{x}_i, \vec{x}_j \in \mathcal{D}$;
- The distance between two objects is 0 only if and only if the objects are equal: $\delta(\vec{x}_i, \vec{x}_j) = 0$ if and only if $\vec{x}_i = \vec{x}_j$, i.e. identity;
- The distance between two objects must be consistent; or symmetric: $\delta(\vec{x}_i, \vec{x}_j) = \delta(\vec{x}_j, \vec{x}_i) \, \forall \vec{x}_i, \vec{x}_j \in \mathcal{D}$; and,
- The distance function must adhere to the triangle inequality: $\delta(\vec{x}_i, \vec{x}_j) \leq \delta(\vec{x}_i, \vec{x}_k) + \delta(\vec{x}_k, \vec{x}_j) \, \forall \vec{x}_i, \vec{x}_j, \vec{x}_k \in \mathcal{D}$.

Numerous distance measures that satisfy these properties have been defined to date. Here we detail several popular distance measures for the clustering problem.

The Minkowski distance measure [76] is the generalised form of three popular distance measures. It is given as:

$$\delta_{Minkowski}(\vec{x}_i, \vec{x}_j) = \left(\sum_{e=1}^{d} (x_{ie} - x_{je})^p \right)^{\frac{1}{p}} \tag{2.1}$$

The Manhattan distance measure [76], also known as the city block distance measure, is the sum of the absolute distances of the objects. It is a special case of the Minkowski distance where $p = 1$. It is defined formally as:

$$\delta_{Manhattan}(\vec{x}_i, \vec{x}_j) = \sum_{e=1}^{d} |x_{ie} - x_{je}| \tag{2.2}$$

The Euclidean distance measure [76] is another special case of the Minkowski distance measure where $p = 2$. It is formally defined as:

$$\delta_{Euclidean}(\vec{x}_i, \vec{x}_j) = \sqrt{\sum_{e=1}^{d} (x_{ie} - x_{je})^2} \tag{2.3}$$

The Chebyshev distance measure [76], also known as the chessboard distance measure, is another special case of the Minkowski distance measure where p is infinite. This is equivalent to the greatest distance between the objects in a single dimension.

$$\delta_{Chebyshev}\left(\vec{x}_i, \vec{x}_j\right) = \lim_{p \to \inf} \left(\sum_{e=1}^{d} \left(x_{ie} - x_{je}\right)^p \right)^{\frac{1}{p}} = \max_{e=1}^{d} \left|x_{ie} - x_{je}\right| \tag{2.4}$$

The Canberra distance measure [93] is similar to the Manhattan distance measure. It is more sensitive to distances between objects that are close to the origin than the Manhattan distance measure, but is less sensitive to high values in a given dimension than the Manhattan distance measure. This measure is useful for detecting differences between objects in high dimensional spaces.

$$\delta_{Canberra}\left(\vec{x}_i, \vec{x}_j\right) = \begin{cases} 0 & \text{for } \vec{x}_i = \vec{x}_j = 0 \\ \sum_{e=1}^{d} \frac{|x_{ie} - x_{je}|}{|x_{ie}| + |x_{je}|} & \text{for } \vec{x}_i \neq 0 \text{ or } \vec{x}_j \neq 0 \end{cases} \tag{2.5}$$

The Pearson correlation coefficient [128] may be used as a measure of dissimilarity between two objects which gives us a distance between any two objects. The Pearson correlation coefficient of two objects, $\vec{x}_i$ and $\vec{x}_j$, is given as $\phi\left(\vec{x}_i, \vec{x}_j\right)$ where $-1 \leq \phi\left(\vec{x}_i, \vec{x}_j\right) \leq 1$. By transforming the value of the Pearson correlation coefficient into the interval $[0, 1]$ we can obtain a distance measure:

$$\delta_{Pearson}\left(\vec{x}_i, \vec{x}_j\right) = 1 - \phi\left(\vec{x}_i, \vec{x}_j\right) \tag{2.6}$$

where

$$\phi\left(\vec{x}_i, \vec{x}_j\right) = \frac{\sum_{e=1}^{d} \left(x_{ie} - \mu_{\vec{x}_i}\right)\left(x_{je} - \mu_{\vec{x}_j}\right)}{\sqrt{\sum_{e=1}^{d} \left(x_{ie} - \mu_{\vec{x}_i}\right)^2} \sqrt{\sum_{e=1}^{d} \left(x_{je} - \mu_{\vec{x}_j}\right)^2}} \tag{2.7}$$

and

$$\mu_{\vec{x}_i} = \frac{\sum_{e=1}^{d} x_{ie}}{d} \tag{2.8}$$

Unfortunately this equation has the drawback of computing the mean across all of the variables. Each of the variables has different meanings and may be on a different scale which means that this measure of dissimilarity is not very meaningful in some cases.

Similarly we can use the cross product index of two objects, $\vec{x}_i$ and $\vec{x}_j$, to obtain the angular separation of the objects. Again by transforming the result into the interval $[0, 1]$ we obtain another distance measure:

$$\delta\left(\vec{x}_i, \vec{x}_j\right) = 1 - \phi\left(\vec{x}_i, \vec{x}_j\right) \tag{2.9}$$

where

$$\phi\left(\vec{x}_i, \vec{x}_j\right) \frac{\sum_{e=1}^{d} x_{ie} x_{je}}{\sqrt{\sum_{e=1}^{d} x_{ie}^2 \sum_{e=1}^{d} x_{je}^2}} \tag{2.10}$$

In this work we use the Euclidean distance metric to measure the distance between any given pair of objects, hence $\delta\left(\vec{x}_i, \vec{x}_j\right)$ refers to the Euclidean distance between $\vec{x}_i$ and $\vec{x}_j$.

2.1.2 Clustering Solution Definitions

$\mathcal{D}$ can be partitioned to form a set of k subsets representing a clustering, $\mathcal{P} = \{\mathcal{P}_1, \ldots, \mathcal{P}_k\}$. Each cluster $\mathcal{P}_g$, where g runs from 1 to k, is a set of n_g objects from $\mathcal{D}$, $\mathcal{P}_g = \left\{\vec{x}_1^g, \ldots, \vec{x}_{n_g}^g\right\}$, so each object is a d dimensional feature vector, $\vec{x}_i^g = (x_{i1}^g, \ldots, x_{id}^g)$.

We are concerned with complete, non-overlapping, clustering solutions. That is to say,

- all objects must belong to at most one cluster: $\mathcal{P}_g \cap \mathcal{P}_h = \emptyset, \forall \mathcal{P}_g, \mathcal{P}_h \in \mathcal{P}$ where $g \neq h$;
- all objects must belong to one cluster so no objects are classified as outliers or noise: $\mathcal{P}_1 \cup \cdots \cup \mathcal{P}_k = \mathcal{D}$; and,

- no cluster is allowed to be empty: $\mathcal{P}_g \neq \emptyset$.

Assigning an object to a cluster may be a complete or partial assignment, known as crisp or fuzzy clustering respectively. In this work we use crisp clustering as each object in the data set i assigned to one cluster only. Fuzzy clustering is an alternative method of assigning each object from the data set to more than one cluster. Each object is assigned proportionately to each cluster in the solution with a strength of membership in the range $[0, 1]$. In traditional crisp clustering the membership of each cluster is either 0 or 1 for a given object. In fuzzy clustering an object could have, for instance, a membership to one cluster of 0.6 and a membership to another cluster of 0.4. This allows us to say that an object belongs to more than one cluster. This is useful where the boundaries of clusters overlap or the area around the objects is noisy. Fuzzy clustering is a more complex form of clustering than crisp clustering. The calculation of centroids and medoids is more complicated, and the final result may be required in a crisp form to be useful. We have chosen to use crisp clustering in this work as it simplifies the assessment of clustering solutions and the techniques for assessing the quality of clustering solutions that we will introduce in section 2.3. Fuzzy clustering is of relevance as there are algorithms that have been developed for the task of fuzzy clustering such as fuzzy c-means [10]. There have also been developments in the areas of measuring cluster quality that we use in this work and developments in multi-objective optimisation algorithms for the task of fuzzy clustering that are similar to the goals of this work [105].

The centroid of $\mathcal{P}_g$, $\vec{c}_g$, is a d dimensional feature vector representing the centre point of $\mathcal{P}_g$. $\vec{c}_g$ may, or may not, correspond to a member of $\mathcal{P}_g$. To compute $\vec{c}_g$ the value of each dimension must first be calculated. The e-th dimension is calculated as: $c_{ge} = \left(\sum_{i=1}^{n_g} x_{ie}^g\right)/n_g$. Thus, $\vec{c}_g = (c_{g1}, \ldots, c_{gd})$. The centroid of the data set $\mathcal{D}$, $\vec{v} = (v_1, \ldots, v_d)$, is calculated in a similar fashion.

For a given data set, $\mathcal{D}$, a set of N clustering solutions may be generated $\mathfrak{P} = \left\{\mathcal{P}^1, \ldots, \mathcal{P}^N\right\}$. Many sets of of clustering solutions may be produced for a given data set by using different clustering algorithms or from different executions

of the same algorithm. Depending on the clustering algorithm used, this may be as part of the process or because execution is repeated with different parameters (e.g. hierarchical clustering, partitional clustering) or as the result of a multi-objective genetic algorithm. The set of clustering solutions often contains solutions constructed with a varied number of clusters, k. Selecting the correct k is an important task within cluster analysis [109, 138]. Even for a fixed k, many solutions may be constructed which divide a data set differently into k clusters and selecting the "best" solution from this set is still a challenging task [145]. More detail on how to evaluate and compare individual solutions within a set of solutions is presented in Chapter 4.

2.1.3 Clustering Solution Properties

First we will establish some essential properties of the structure of a clustering solution that can be measured and often form part of different quality measures.

The total variation, $\mathcal{D}^T$, of a given data set, $\mathcal{D}$, is the sum of the squared Euclidean the centroid of the data set, $\vec{v}$, and each element in the data set, $\vec{x}_i$:

$$\mathcal{D}^T = \sum_{i=1}^{n} \delta\left(\vec{x}_i, \vec{v}\right)^2 \tag{2.11}$$

For a given cluster, $\mathcal{P}_g$, the total variation, $\mathcal{P}_{\mathcal{P}_g}^T$, a measure of heterogeneity, is calculated as:

$$\mathcal{P}_{\mathcal{P}_g}^T = \sum_{i=1}^{n_g} \delta\left(\vec{x}_i^g, \vec{c}_g\right)^2 \tag{2.12}$$

The within-cluster variation, $\mathcal{P}^W$, is the sum of the total variation for all clusters:

$$\mathcal{P}^W = \sum_{g=1}^{k} \mathcal{P}^T_{\mathcal{P}_g} = \sum_{g=1}^{k} \sum_{i=1}^{n_g} \delta\left(\vec{x}_i^g, \vec{c}_g\right)^2 \tag{2.13}$$

A good solution is expected to group objects that are similar to each other into the same cluster. A good solution will therefore lead to homogeneous clusters with low within-cluster variation.

A good solution should also ensure that clusters are different from each other, and therefore, well separated. In other words the clusters should be heterogeneous with respect to one another. The between-cluster variation, $\mathcal{P}^B$, is a measure of heterogeneity calculated as the weighted sum of squared distances between the clusters centroids and the data centroid. The weight for each term in the sum is the number of members of a given cluster. For a good clustering solution $\mathcal{P}^B$ should have a high value.

$$\mathcal{P}^B = \sum_{g=1}^{k} n_g \delta\left(\vec{c}_g, \vec{v}\right)^2 \tag{2.14}$$

It is worth noting that the between-cluster variation and within-cluster variation sum to become the total variation of the data set, $\mathcal{D}^T = \mathcal{P}^W + \mathcal{P}^B$. These measures work in parallel, and therefore minimisation of one should result in maximisation of the other.

As well as homogeneity of objects within a cluster and heterogeneity (or separation) of the clustering solution, a third property often considered in clustering is that of connectivity between the objects within a cluster and disconnectivity between objects of different clusters. The concepts of k-nearest neighbour(kNN) and k-mutual nearest neighbour (kMN) consistent clustering have been proposed [32] in the context of connected clusters. Note that k in this context does not refer to the number of clusters.

The nearest neighbour of an object, $\vec{x}_i$, is the object, $\vec{x}_i^1$, where

$$\delta\left(\vec{x}_i, \vec{x}_i^1\right) \leq \delta\left(\vec{x}_i, \vec{x}_j\right) \quad \text{for } \forall j \in \mathcal{D}, \quad \text{where } \vec{x}_i \neq \vec{x}_i^1 \tag{2.15}$$

Following from this, we can construct a ranked list of nearest neighbours of object $\vec{x}_i$ where $\vec{x}_i^1$ is the nearest neighbour, $\vec{x}_i^2$ is the second nearest neighbour, and so on. The m^{th} nearest neighbour of $\vec{x}_i$ is the object $\vec{x}_i^m$ as ranked by its distance to object $\vec{x}_i$. If $\vec{x}_i$ is the k^{th} nearest neighbour of $\vec{x}_j$ and $\vec{x}_j$ is the l^{th} nearest neighbour of $\vec{x}_i$ then $\vec{x}_i$ and $\vec{x}_j$ are the p^{th} mutual nearest-neighbour of each other, where $p = \max(k, l)$. A kNN consistent cluster is a group of objects where the k nearest-neighbours of each object are also within the cluster and a kMN consistent cluster is a group of objects where the k mutual nearest-neighbours are members of that cluster.

2.2 Existing Clustering Techniques

A clustering algorithm is an algorithm that partitions a set of objects into a set of subsets according to some measure of similarity so that similar objects are placed in the same subset.

In this work we are only considering hard, or crisp, partitions of the objects into mutually disjoint sets. It is worth noting that clustering techniques exist for partitioning the data set into soft, or fuzzy, sets of clusters [121].

There are a multitude of clustering algorithms presented in the literature [8, 65, 76, 81, 94, 150]. These can be grouped into many categories such as: partitional techniques, hierarchical techniques and density based techniques, among others.

The clustering techniques most relevant to this work are partitional techniques and hierarchical clustering techniques. A brief summary of these are given:

2.2.1 Partitional Techniques

Partitional clustering techniques produce a single clustering solution for the data. These techniques start with a random or user-defined clustering solution. This solution is then optimised according to some objective by changing the cluster membership of the objects until a stopping criterion is met, such as no change in the membership of any objects or no change in a clustering-quality measure.

Partitional algorithms are in general very efficient and easy to implement. However, parameters must be defined such as the number of clusters and the initial clustering solution. The final clustering solution generated may vary depending upon these initial parameters, so it is desirable to run the algorithm several times with different parameters and select the best of these solutions.

2.2.1.1 k-Means

The k-means algorithm is the classic example of a clustering algorithm. The main goal of the algorithm is to discover a given number of clusters by minimising the distance between each object in a cluster and the centroid of the cluster [64, 69, 101]. Given a dataset, $\mathcal{D}$, the algorithm will create a clustering solution, $\mathcal{P}$, that consists of k clusters. The value of k is typically specified by the end user of the algorithm.

First, k objects must be defined as the initial cluster centroids. The k objects may be selected at random from the data set or selected using prior knowledge. At the second stage, the clusters within the clustering solution are instantiated. Each object in the dataset is assigned to the cluster with the closest centroid. Once each object has been assigned to a cluster, the cluster centroids are recomputed. At this stage a set of initial clusters exist.

The next stage of the algorithm is the improvement stage, where the clusters are rearranged to find an optimum clustering solution. For each object in the dataset the closest cluster centroid is calculated. If the object is not assigned to that cluster then the object is reassigned and the cluster centroids are recalculated. This sequence

of steps is repeated until a stopping criterion is met. This may be a given number of iterations or when no change in cluster membership has occurred. The h-means algorithm [137] differs from k-means at this stage. h-means is a clustering algorithm commonly used in place of k-means as they are nearly identical. When using h-Means the cluster centroids are recomputed once for every iteration through the dataset. Given the same initial cluster centroids the same clustering solution is usually calculated. However, the amount of time needed to compute the solution is less as the cluster centroids are recomputed less often. h-means also lends itself to parallel processing allowing a further reduction in the computing time required.

Algorithm 2.1 k-means

```
Designate k initial centroids
Associate each object in the dataset to its nearest centroid
Recalculate the centroids
repeat
  for each object in the dataset do
    Calculate the nearest centroid to the object
    Associate the object to that centroid
    Recalculate the affected centroids
  end for
until no more improvement is possible
```

2.2.1.2 k-Medoids

k-Medoids algorithms are conceptually similar to k-means, the main difference is the use of medoids in the place of centroids. The medoid of a cluster is an object that is a typical representative of the cluster. It is usually the most centrally located object within the cluster. To determine the medoid of a cluster we identify the object that is a member of the cluster and has the minimal average dissimilarity to all of the other objects within the cluster. There may be one or more objects that have a minimal dissimilarity to other objects in the cluster. Where there are more than one candidates for the medoid of a cluster only one candidate should be chosen. Formally we define a medoid as:

$$\vec{m}_g = \underset{\vec{x}_i^g \in \mathcal{P}_g}{\arg\min}\ cost\left(\vec{x}_i^g\right) \tag{2.16}$$

$$cost\left(\vec{x}_i^g\right) = \sum_j \delta\left(\vec{x}_i^g, \vec{x}_j^g\right) \forall \vec{x}_j^g \in \mathcal{D} \text{ where } j \neq i \tag{2.17}$$

Determining the medoid of a cluster is computationally inefficient as each object in the cluster must be tested. For large datasets the calculation of medoids becomes time consuming as the number of objects in each cluster normally increases in line with the size of the dataset.

Partitioning Around Medoids (PAM) [79] is a common implementation of the k-Medoids algorithm. The implementation of a basic k-Medoids algorithm is very similar to the k-Means algorithm. The main difference is the computation of medoids in place of centroids.

Algorithm 2.2 k-Medoids

```
Designate k initial medoids
Associate each object in the dataset to its nearest medoid
Recalculate the medoid
repeat
  for each object in the dataset do
    Calculate the nearest medoid to the object
    Associate the object to that medoid
    Recalculate the affected medoid
  end for
until no more improvement is possible
```

The centroid of a cluster is sensitive to outliers within the data set. This can lead to partitions of the data that are influenced by these outliers. The use of medoids reduces the effects of these outliers on the final clustering solution. Computing the medoids of the cluster is more complex than calculating the centroid of a cluster. However, in some cases [144] it has been found that k-medoids converges to a solutions faster than using k-means, which can negate the effects of the increased time of finding the medoids of the clusters.

CLARA Clustering LARge Applications (CLARA) [80] is a clustering algorithm designed for use with large data sets. It does not use all of the data to calculate partitions. Instead, a sample of the data set is used to calculate a set of medoids using the PAM algorithm previously described in Section 2.2.1.2. This is performed several times to produce a range of solutions. This approach leads to a known number of calculations regardless of the size of the data sets. Therefore, the computational complexity is linear instead of quadratic.

The set of medoids that is judged best on the whole data set is accepted as the final set of medoids. For each set of medoids that has been produced the average dissimilarity between the objects and their medoid is calculated. The solution that has the lowest dissimilarity is accepted as the final solution.

Clustering Large Applications based upon RANdomized Search (CLARANS) [117] is a further extension of CLARA that also produces sets of medoids using a more complex method of drawing subsets from the overall data set. CLARANS differs from CLARA by resampling the original data set at each step of the algorithm by drawing a new subset of objects from the neighbours of the current subset. This allows the algorithm to find better clusterings than the original CLARA algorithm at the cost of greater computational complexity.

2.2.1.3 Selecting k

The main challenge within partitional clustering is selecting the correct value of k [109]. For this, we can use many runs of k-means to generate a set of clustering solutions for a range of possible values of k. From this set of solutions, we can then select the clustering solution that we consider to be "best" by using internal clustering quality measures (see Section 2.3) to compare their relative quality and therefore attempt to determine the correct value of k.

Partitional algorithms can also be affected by initial values. In the case of k-means, this is the selection of the initial centroids. Initial centroids could be selected at random or by using some prior analysis of the data set. k-means will produce the

same clustering solution for a given set of initial centroids but may produce different clustering solutions for different initial centroids. It may be appropriate to execute k-means several times with different initial values to produce a pool of clustering solutions with the same value of k and then select the clustering solution that we consider "best" from the pool. An internal clustering quality measure can be used to select the "best" solution from this pool.

2.2.2 Hierarchical Techniques

Hierarchical techniques are subdivided into agglomerative methods and divisive methods [77, 114, 91]. Agglomerative techniques may be thought of as "bottom up" techniques as they merge clusters together to create new clustering solutions that have less clusters than the clustering solutions that were used to produce them. Divisive techniques may be thought of as "top down" processes as they divide a cluster to produce a new clustering solution with more clusters than the previous clustering solution.

Agglomerative methods [28] begin with each object in the data set being treated as a cluster that consists of only one object. At each step in the algorithm two clusters are merged together to form a new cluster and this is repeated until only one cluster remains. There are a number of methods for determining which clusters should be merged together detailed further in this section.

Divisive methods begin with one cluster containing every object within the data set, at each step in the algorithm a cluster is divided into new smaller clusters until the clustering solution consists of only singleton clusters. Divisive clustering algorithms require a method of selecting which cluster should be divided and a method of dividing the cluster into sub-clusters.

At each step in these methods the clustering solutions are usually stored. This leads to the production of a set of clustering solutions, defined as $\mathcal{S}_a = \{\vec{s}_{a1}, \ldots, \vec{s}_{aN}\}$. The set of clustering solutions may be represented visually as a dendrogram. This

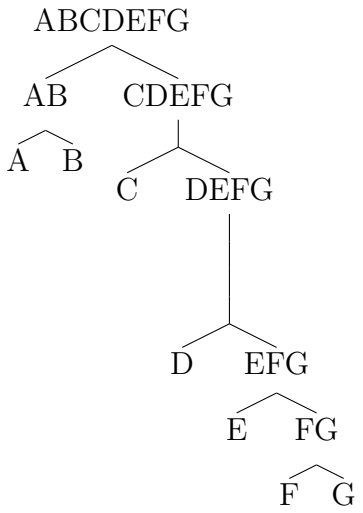


Figure 2.1: An example Dendrogram

is a tree structure that shows which clusters are merged together at each step of the algorithm. The length of each edge between each pair of clusters that have been merged on the dendrogram may represent the distance between the two clusters that have been merged at the previous level to form the new cluster. We elaborate on ways of measuring the distance between two clusters later in this section. A dendrogram may be used to select the number of clusters and therefore the final clustering solution from the set of clustering solutions. There may be a significant change in the distance between the merged clusters at some point during the process that is usually obvious when represented with a dendrogram. Where the distance between the merged clusters is unusually large or small this is often the bases to identifying a good clustering solution. An example of a dendrogram is shown in Figure 2.1.

2.2.2.1 Ward's Method

Ward's method [147] is an agglomerative technique that attempts to minimise the within-cluster variation $\mathcal{P}^W$. For each possible new clustering solution that can be formed by merging any pair of clusters from the current clustering solution, the value of $\mathcal{P}^W$ is computed. The clustering solution that is found to lead to the

smallest increase in the value of $\mathcal{P}^W$ is accepted as the next clustering solution. The process is repeated until all of the objects have been merged into one cluster. Clustering solutions and the value of $\mathcal{P}^W$ are recorded at each stage to produce a set of clustering solutions. The increase in the value of $\mathcal{P}^W$ may alternatively be calculated as:

$$\text{Ward}\left(\mathcal{P}_g, \mathcal{P}_h\right) = \frac{n_g n_h}{n_g + n_h} \delta\left(\vec{c}_g, \vec{c}_h\right) \tag{2.18}$$

The final clustering solution may be selected in a number of ways. The user may wish for a predetermined number of clusters and can therefore select best clustering solution that has that number of clusters. The user may use the change in the value of $\mathcal{P}^W$ to inform their decision of which clustering solution to pick as the size of the change may indicate a natural stopping point, or the user may use a cluster quality measure to select a clustering solution from the candidate solutions.

2.2.2.2 Other Methods

In addition to Ward's method, a number of other methods may be used to determine which clusters should be merged together to form new clusters. The process is in general the same as Ward's method: a clustering solution is initialised so that each object within the data set is a member of a singleton cluster. Clusters are selected for merging where they minimise some property. They are merged together and the process is repeated until all of the clusters have been merged together into one cluster that encompasses the whole data set. Here, we detail a number of methods to determine the distance between clusters. The clusters that minimise the property are considered closest and will be chosen to be merged together. The methods are as follows:

Single Linkage The first method of measuring the distance between two given clusters, $\mathcal{P}_g$ and $\mathcal{P}_h$, is the single linkage distance, also known as the nearest-neighbour distance. In this technique, the distance between two clusters is defined as the distance between the closest pair of individuals where one object is in cluster $\mathcal{P}_g$ and the other is in $\mathcal{P}_h$. This is given formally as:

$$\text{Single Linkage}\left(\mathcal{P}_g, \mathcal{P}_h\right) = \min_{\substack{\vec{x}_i \in \mathcal{P}_g, \\ \vec{x}_j \in \mathcal{P}_h}} \delta\left(\vec{x}_i, \vec{x}_j\right) \tag{2.19}$$

This method of measuring the distance between two clusters leads to clusters that are continuous and near each other. However, it may cause some clusters to be merged together because they both have outliers near each other, which may lead to undesirable clusterings.

Complete Linkage The complete linkage distance, also known as the furthest neighbour distance, defines the distance between two clusters, $\mathcal{P}_g$ and $\mathcal{P}_h$, as the maximum distance between a pair of objects, where only pairs of objects consisting of an object from $\mathcal{P}_g$ and an object from $\mathcal{P}_h$ are considered. This is given as:

$$\text{Complete Linkage}\left(\mathcal{P}_g, \mathcal{P}_h\right) = \max_{\substack{\vec{x}_i \in \mathcal{P}_g, \\ \vec{x}_j \in \mathcal{P}_h}} \delta\left(\vec{x}_i, \vec{x}_j\right) \tag{2.20}$$

Unlike in the single linkage method, clusters will not be merged because they have outliers that are near each other. The use of complete linkage often leads to compact and dense clusters. However, outliers may also interfere with the calculation.

Average Linkage Alternatively, the average linkage method may be used to measure the distance between two clusters, $\mathcal{P}_g$ and $\mathcal{P}_h$. The distance is given as the average of the distance between every pair of objects, where each pair of objects

comprises of one object from $\mathcal{P}_g$ and one from $\mathcal{P}_h$.

$$\text{Average Linkage}\left(\mathcal{P}_g, \mathcal{P}_h\right) = \frac{1}{n_g \cdot n_h} \sum_{i=1}^{n_g} \sum_{j=1}^{n_h} \delta\left(\vec{x}_i, \vec{x}_j\right) \tag{2.21}$$

Average linkage is less sensitive to outliers than single linkage and complete linkage.

Centroid Linkage The centroid linkage method defines the distance between two clusters as the distance between the centroids of the clusters:

$$\text{Centroid Linkage}\left(\mathcal{P}_g, \mathcal{P}_h\right) = \delta\left(\vec{c}_{\mathcal{P}_g}, \vec{c}_{\mathcal{P}_h}\right) \tag{2.22}$$

When using centroid linkage the clusters that are created are often dominated by the characteristics of the largest of the two clusters that was merged. This can lead to some clusters becoming insignificant during the merging process.

Median Linkage The median linkage method requires the definition of a weighted centroid. This is an object defined recursively based upon the clusters that were merged to create the cluster it is associated with. If cluster $\mathcal{P}_g$ was formed from the merging of clusters $\mathcal{P}_{g'}$ and $\mathcal{P}_{g''}$ then the weighted centroid is recursively defined as:

$$\overline{\vec{c}_g} = \{\overline{c_{g1}}, \overline{c_{g2}}, \ldots, \overline{c_{gd-1}}, \overline{c_{gd}}\} \tag{2.23}$$

$$\text{where } \overline{c_{ge}} = \frac{c_{g'e} + c_{g''e}}{2}, e = 1...d \tag{2.24}$$

For the base case, where the cluster was not formed from merging two clusters, the value of the cluster centroid is used instead, $\overline{\vec{c}_g} = \vec{c}_g$. This is known as a

weighted centroid as it is the midpoint of the two clusters that were merged to form the clusters instead of the centroid of the current cluster.

The median linkage method of determining the distance between two clusters is defined as:

$$\text{Median Linkage}\left(\mathcal{P}_g, \mathcal{P}_h\right) = \delta\left(\overline{\vec{c}_{\mathcal{P}_g}}, \overline{\vec{c}_{\mathcal{P}_h}}\right) \tag{2.25}$$

Median linkage avoids the problem with centroid linkage by locating the new weighted centroid in-between the weighted centroids of the merged clusters. This causes new clusters to not be dominated by the largest of the clusters that were merged to form it.

2.2.2.3 Selecting the Number of Clusters

Selecting the best clustering solution within a set of solutions generated by a hierarchical clustering algorithm is a difficult task. Clustering quality measures are used to select which clustering solution should be returned as the final clustering solution. There have been many studies [42, 109, 138] about how to do this with no one conclusive technique identified to date.

2.2.3 Density Based Techniques

In section 2.2.2.2 we discussed the linkage between objects and how this can be used with hierarchical clustering to merge clusters together to form larger clusters. Density based clustering aims to identify regions of the data set where there are many objects close together and regions where there are not: these are dense and sparse areas of the data set respectively.

Dense regions of the data set are then treated as clusters and sparse regions are treated as the space between clusters. Any data points in these areas are then treated

as outliers or anomalies and may not be included as members of a cluster. Density Based Spatial Clustering of Applications with Noise (DBSCAN) [35] and Ordering Points to Identify the Clustering Structure (OPTICS) [1] are popular density based clustering algorithms.

Density based techniques rely upon the idea that there will be a significant change in the distance between points that will allow the edges of each cluster to be identified. This is advantageous as it allows for the identification of arbitrarily shaped clusters whereas a method such as k-Means may only discover clusters that are hyper-spherical in nature. These techniques generally only need a single linear scan of the data set to identify the borders between clusters, which can reduce the run time of density based techniques compared to other techniques. Density based techniques rely on there being a significant difference between sparse and dense regions to define the edge of a cluster. Some clusters that are close together may be merged together if there is not a large enough sparse region between them. Clusters that do not have a homogeneous density may also not be correctly identified by density based techniques as there may not be a clear border between the clusters or they may be divided into many clusters as the density within them varies.

2.3 Internal Cluster Quality Measures

Internal cluster quality measures are methods of evaluating the quality of clustering solutions using only the internal structures of the clusters to make these judgements. They are often used to select the "best" solutions from a set of potential clustering solutions. This would be useful after a range of solutions have been generated from multiple executions of the k-means algorithm to select the best solution or as a stopping criterion for a hierarchical clustering technique.

There are numerous measures of internal cluster quality, and algorithms have been developed to optimise some of the measures of cluster quality. Many clustering algorithms have similar goals, even if they are optimising different measures of cluster

quality, so the results of the algorithms are often similar to each other. However, it is possible that a clustering solution that is considered good by one measure of internal cluster quality may be considered bad by another.

We review a number of measures based upon the concepts of density and separation, a measure that is not explicitly related to either and two measures which are based upon connectivity. We have redefined the measures to ensure a consistent notation.

2.3.1 Variance Ratio Criterion

The Variance Ratio Criterion (VRC) [16] is a cluster validity measure dependent on homogeneity and heterogeneity. The value of VRC for a given clustering solution is defined as the ratio between the between-cluster variation and the within-cluster variation. High values of this ratio suggest a better clustering. The ratio must be normalised to stop it increasing monotonically as the number of clusters increases.

$$VRC\left(\mathcal{P}\right) = \frac{\mathcal{P}^B / \left(k-1\right)}{\mathcal{P}^W / \left(n-k\right)} \tag{2.26}$$

The clustering solution that generates the maximum value is the optimal clustering solution according to VRC.

2.3.2 Dunn and Dunn like Indices

Dunn's index, $\mathcal{DN}\left(\mathcal{P}\right)$, [33] is a cluster quality measure based upon cluster compactness and cluster separation. The measure requires the definition of a measure of cluster diameter,$\Delta\left(\mathcal{P}_l\right)$, and a measure of cluster distance, $d\left(\mathcal{P}_g, \mathcal{P}_h\right)$. There are many possible measures of cluster diameter and cluster separation, some of which will be defined in the following section. The general form of Dunn's index is defined as:

$$\mathcal{DN}(\mathcal{P}) = \min_{\substack{\mathcal{P}_g, \mathcal{P}_h \in \mathcal{P} \\ g \neq h}} \left\{ \frac{d(\mathcal{P}_g, \mathcal{P}_h)}{\max_{\mathcal{P}_l \in \mathcal{P}} \Delta(\mathcal{P}_l)} \right\} \tag{2.27}$$

Possible measures of the distance between two given clusters, $\mathcal{P}_g$ and $\mathcal{P}_h$ that are used for computing $\mathcal{DN}(\mathcal{P})$ are introduced below [146].

Previously in Section 2.2.2.2 we described several measures of similarity between two clusters. Some of these may be used as the measure of the distance between two clusters for use with Dunn's Index, we also introduce additional ways of measuring the distance between two clusters here.

The single linkage distance measure is given as:

$$d_a(\mathcal{P}_g, \mathcal{P}_h) = \text{Single Linkage}(\mathcal{P}_g, \mathcal{P}_h) = \min_{\substack{\vec{x}_i \in \mathcal{P}_g, \\ j \in \mathcal{P}_h}} \delta(\vec{x}_i, \vec{x}_j) \tag{2.28}$$

The complete linkage distance measure is given as:

$$d_b(\mathcal{P}_g, \mathcal{P}_h) = \text{Complete Linkage}(\mathcal{P}_g, \mathcal{P}_h) = \max_{\substack{\vec{x}_i \in \mathcal{P}_g, \\ j \in \mathcal{P}_h}} \delta(\vec{x}_i, \vec{x}_j) \tag{2.29}$$

The average linkage distance measure is given as:

$$d_c(\mathcal{P}_g, \mathcal{P}_h) = \text{Average Linkage}(\mathcal{P}_g, \mathcal{P}_h) = \frac{1}{n_g \cdot n_h} \sum_{i=1}^{n_g} \sum_{j=1}^{n_h} \delta(\vec{x}_i, \vec{x}_j) \tag{2.30}$$

The centroid linkage distance measure is given as:

$$d_d(\mathcal{P}_g, \mathcal{P}_h) = \text{Centroid Linkage}(\mathcal{P}_g, \mathcal{P}_h) = \delta\left(\vec{c}_{\mathcal{P}_g}, \vec{c}_{\mathcal{P}_h}\right) \tag{2.31}$$

The inter group distance measure is given as:

$$d_e\left(\mathcal{P}_g, \mathcal{P}_h\right) = \frac{1}{n_g + n_h}\left(\sum_{i=1}^{n_g} \delta\left(\vec{x}_i^g, \vec{c}_h\right) + \sum_{j=1}^{n_h} \delta\left(\vec{x}_j^h, \vec{c}_g\right)\right) \tag{2.32}$$

The Hausdorff distance metric is calculated by using the Supremum and Infimum, the least upper bound of a set and greatest lower bound of a set, of each dimension of the objects belonging to a pair of clusters. The Hausdorff distance metric is given as:

$$d_f\left(\mathcal{P}_g, \mathcal{P}_h\right) = \max\left\{\underset{\vec{x}_i^g}{\text{supinf}}\underset{\vec{x}_j^h}{\delta}\left(\vec{x}_i^g, \vec{x}_j^h\right), \underset{\vec{x}_j^h}{\text{supinf}}\underset{\vec{x}_i^g}{\delta}\left(\vec{x}_i^g, \vec{x}_j^h\right)\right\} \tag{2.33}$$

Possible measures of cluster diameter are given below and include the maximum distance between any pair of objects in the cluster, defined as:

$$\Delta_a\left(\mathcal{P}_l\right) = \max_{i \neq j} \delta\left(\vec{x}_i^l, \vec{x}_j^l\right) \tag{2.34}$$

The average distance among all pairs of objects is defined as:

$$\Delta_b\left(\mathcal{P}_l\right) = \frac{2}{n_l\left(n_l - 1\right)} \sum_{i=1}^{n_l} \sum_{j=1}^{i} \delta\left(\vec{x}_i^l, \vec{x}_j^l\right) \tag{2.35}$$

The average distance between each object in the cluster and its centroid is defined as:

$$\Delta_c\left(\mathcal{P}_l\right) = \frac{2}{n_l} \sum_{i=1}^{n_l} \delta\left(\vec{x}_i^l, \vec{c}_l\right) \tag{2.36}$$

where 2 is used to convert the radius of the cluster into a diameter [12].

Several of the seventeen possible variations on $\mathcal{DN}$ are included in an experimental comparative study by Vendramin [146]. It was found that using d_b and Δ_c to calculate $\mathcal{DN}$ was more effective at finding the correct clustering solution than other combinations of d and Δ. For this reason, in our experimental work we will use d_b and Δ_c to calculate $\mathcal{DN}$.

2.3.3 Davies-Bouldin Index

The Davies-Bouldin index (DB) [89] is based on the ratio of a measure of the between-cluster and within-cluster distances so we define those first.

We start by defining the average of the within-cluster distance of a cluster:

$$s\Big(\mathcal{P}_g\Big) = \frac{1}{n_g}\sum_{i=1}^{n_g}\delta\left(\vec{x}_i^g, \vec{c}_g\right) \tag{2.37}$$

We also define a measure of the between-cluster distance for two given clusters, $\mathcal{P}_g$ and $\mathcal{P}_h$. This can be the centroid linkage between the clusters, given as $\delta\left(\vec{c}_g, \vec{c}_h\right)$

Usually, a similarity measure, $r\left(\mathcal{P}_g, \mathcal{P}_h\right)$, is then defined based on the between-cluster and within-cluster distances:

$$r\left(\mathcal{P}_g, \mathcal{P}_h\right) = \frac{s\Big(\mathcal{P}_g\Big) + s\Big(\mathcal{P}_h\Big)}{\delta\left(\vec{c}_g, \vec{c}_h\right)} \tag{2.38}$$

However, other similarity measures $r\left(\mathcal{P}_g, \mathcal{P}_h\right)$ could be defined freely, providing they meet the following conditions:

- $r\left(\mathcal{P}_g, \mathcal{P}_h\right) \geq 0$
- $r\left(\mathcal{P}_g, \mathcal{P}_h\right) = r\left(\mathcal{P}_h, \mathcal{P}_g\right)$
- if $s\Big(\mathcal{P}_g\Big) = 0$ and $s\Big(\mathcal{P}_h\Big) = 0$ then $r\left(\mathcal{P}_g, \mathcal{P}_h\right) = 0$
- if $s\Big(\mathcal{P}_g\Big) > s\Big(\mathcal{P}_m\Big)$ and $\delta\left(\vec{c}_g, \vec{c}_h\right) = \delta\left(\vec{c}_g, \vec{c}_m\right)$ then $r\left(\mathcal{P}_g, \mathcal{P}_h\right) > r\left(\mathcal{P}_g, \mathcal{P}_m\right)$

- if $s\left(\mathcal{P}_g\right) = s\left(\mathcal{P}_m\right)$ and $\delta\left(\vec{c}_g, \vec{c}_h\right) < \delta\left(\vec{c}_g, \vec{c}_m\right)$ then $r\left(\mathcal{P}_g, \mathcal{P}_h\right) > r\left(\mathcal{P}_g, \mathcal{P}_m\right)$

The DB Index determines for each cluster which other cluster it is most similar to and measures this similarity, using a measure of similarity such as the one just presented. The index provides an average of these maximum cluster similarities:

$$DB\left(\mathcal{P}\right) = \frac{1}{k}\sum_{g=1}^{k} \max_{h=1\ldots k, h\neq g} r\left(\mathcal{P}_g, \mathcal{P}_h\right) \tag{2.39}$$

If DB is low then the clusters are very dissimilar to each other. This indicates that the clusters have similar objects (homogeneous) and are well separated (heterogeneous).

2.3.4 Halkidi Indexes

A series of cluster quality measures have been developed incrementally by Halkidi and colleagues. These are SD [55], SDbw [53] and CDbw [54]. The SD validity index is based upon a cluster quality measure developed for the fuzzy clustering algorithm Fuzzy-C-Means [121].

2.3.4.1 SD Validity Index

Again, to define this measure, a number of other measures must be presented first. The average scattering of the objects within the clusters is an indication of homogeneity. It is defined as the average of the ratio between the variation of each cluster and the variation of the data set.

$$scatt\left(\mathcal{P}\right) = \frac{1}{k}\sum_{g=1}^{k} \frac{\mathcal{P}^T_{\mathcal{P}_g}}{\mathcal{D}^T} \tag{2.40}$$

The maximum and minimum centroid linkages can be defined as
$\delta_{\max} = \max\left(\delta\left(\vec{c}_g, \vec{c}_h\right)\right)$ and $\delta_{\min} \min\left(\delta\left(\vec{c}_g, \vec{c}_h\right)\right)$ for $\forall \mathcal{P}_g, \mathcal{P}_h \in \mathcal{P}$ where $\mathcal{P}_g \neq \mathcal{P}_h$.

The total separation of the clusters, $dis\,(\mathcal{P})$, based on $\delta_{\max}$ and δ_{min} measures the between-cluster distance of the clustering solution.

$$dis\,(\mathcal{P}) = \frac{\delta_{\max}}{\delta_{\min}} \sum_{g=1}^{k} \left(\sum_{h=1, h \neq g}^{k} \delta\,(\vec{c}_g, \vec{c}_h) \right)^{-1} \tag{2.41}$$

The total separation of clusters is influenced by k. To mitigate this a weighting factor, $dis\,(\mathcal{P}')$, is used where $\mathcal{P}'$ is the solution in the set of solutions to be evaluated by SD for which the value of k is the highest.

Now we are in a position to define SD as, $SD\,(\mathcal{P}) = dis\,(\mathcal{P}') \cdot scatt\,(\mathcal{P}) + dis\,(\mathcal{P})$. A low value of SD corresponds to a good clustering solution.

2.3.4.2 SDbw Validity Index

The SDbw Validity Index [53] is an enhancement of the SD index. The term $scatt\,(\mathcal{P})$ is retained. A weighting factor is no longer required as the term $dis\,(\mathcal{P})$ is replaced with the term $densBW\,(\mathcal{P})$, defined below.

First, let us define for an object, $\vec{x}_i$, its neighbourhood as a hyper-sphere where the radius is equal to the standard deviation of the clusters:

$$stdev = \frac{1}{k} \sqrt{\sum_{g=1}^{k} ||\sigma(\mathcal{P}_g)||} \tag{2.42}$$

$$||x|| = (x^T x)^{\frac{1}{2}} \qquad \text{where } x \text{ is a vector.} \tag{2.43}$$

and $\sigma(\mathcal{P}_g)$ is the variance of cluster for each dimension.

The density of $\vec{x}_i$ within two given clusters, $\mathcal{P}_g$ and $\mathcal{P}_h$, is defined as the number of objects belonging to either $\mathcal{P}_g$ or $\mathcal{P}_h$ that occur within the neighbourhood of $\vec{x}_i$:

$$den\left(\vec{x}_i, \mathcal{P}_g, \mathcal{P}_h\right) = \sum_{\vec{x}_j \in \mathcal{P}_g \cup \mathcal{P}_h} f\left(\vec{x}_i, \vec{x}_j\right) \tag{2.44}$$

$$\text{where } f\left(\vec{x}_i, \vec{x}_j\right) = \begin{cases} 0 & \text{if } \delta\left(\vec{x}_i, \vec{x}_j\right) > \sigma_{\mathcal{D}}, \\ 1 & \text{otherwise} \end{cases} \tag{2.45}$$

The midpoint of two clusters, $\mathcal{P}_g$ and $\mathcal{P}_h$, is the midpoint of a line running from $\vec{c}_g$ to $\vec{c}_h$, defined as the feature vector $m\left(\mathcal{P}_g, \mathcal{P}_h\right) = \{\left(c_{g1} + c_{h1}\right)/2, \ldots, \left(c_{gd} + c_{hd}\right)/2\}$.

$densBW\left(\mathcal{P}\right)$ represents the inter-cluster density of a clustering solution:

$$densBW\left(\mathcal{P}\right) = \frac{1}{k\left(k-1\right)} \sum_{g=1}^{k} \sum_{h=1, h \neq g}^{k} \frac{den\left(m\left(\mathcal{P}_g, \mathcal{P}_h\right), \mathcal{P}_g, \mathcal{P}_h\right)}{\max\left(den\left(\vec{c}_g, \mathcal{P}_g, \mathcal{P}_h\right), den\left(\vec{c}_h, \mathcal{P}_g, \mathcal{P}_h\right)\right)} \tag{2.46}$$

It evaluates the average density of the region between each pair of clusters in relation to the density of the clusters in the clustering solution. It is desirable for the density of the regions between clusters to be low compared to the density of the clusters.

SDbw is defined as the sum of the inter-cluster density and the separation of clusters, so $SDbw\left(\mathcal{P}\right) = scatt\left(\mathcal{P}\right) + densBW\left(\mathcal{P}\right)$. The value of SDbw should be minimised to obtain dense and well separated clusters.

2.3.4.3 CDbw Validity Index

The CDbw index [54] consists of two new terms: $densBW\left(\mathcal{P}\right)$ is replaced by $interDens\left(\mathcal{P}\right)$ as a measure of the inter-cluster density, and the term $scatt\left(\mathcal{P}\right)$ is replaced by $sep\left(\mathcal{P}\right)$ as a measure of cluster separation.

We now look at some building blocks for the CDbw definition. The closest rep-

resentatives of a pair of clusters, $\mathcal{P}_g$ and $\mathcal{P}_h$, are the pair of objects $\vec{x}_i$ and $\vec{x}_j$, where $\vec{x}_i \in \mathcal{P}_g$ and $\vec{x}_j \in \mathcal{P}_h$, that minimise $\delta(\vec{x}_i, \vec{x}_j)$. The midpoint of two clusters is now redefined as the midpoint of their closest representatives. This is represented by the vector $m(\mathcal{P}_g, \mathcal{P}_h) = \{(x_{i1} + x_{j1})/2, \ldots, (x_{id} + x_{jd})/2\}$.

The neighbourhood of an object is redefined as a hyper-sphere where the radius is equal to the average of the standard deviations, $\sigma_{\mathcal{P}_g}$ and $\sigma_{\mathcal{P}_h}$, of the clusters $\mathcal{P}_g$ and $\mathcal{P}_h$. The density between two clusters, $\mathcal{P}_g$ and $\mathcal{P}_h$, is calculated as the sum of the ratio between the number of objects belonging to the clusters within the hyper-sphere and the combined number of objects in the clusters.

$$den(\mathcal{P}_g, \mathcal{P}_h) = \sum_{\vec{x}_i \in \mathcal{P}_g \cup \mathcal{P}_h} \frac{f(\vec{x}_i)}{n_g + n_h} \tag{2.47}$$

$$\text{where } f(\vec{x}_i) = \begin{cases} 0 & \text{if } \delta(\vec{x}_i, m(\mathcal{P}_g, \mathcal{P}_h)) > \frac{\sigma_{\mathcal{P}_g} + \sigma_{\mathcal{P}_h}}{2}, \\ 1 & \text{otherwise} \end{cases} \tag{2.48}$$

The between-cluster density of the clustering solution is defined as $interDens(\mathcal{P})$. This is the sum, for each possible pairing of clusters $\mathcal{P}_g$ and $\mathcal{P}_h$ from $\mathcal{P}$, of the density of the clusters multiplied by the ratio of the distance between the closest representatives of the clusters and the summed standard deviations of the clusters.

$$interDens(\mathcal{P}) = \sum_{g=1}^{k} \sum_{h=1, g \neq h}^{k} den(\mathcal{P}_g, \mathcal{P}_h) \cdot \frac{\min_{\vec{x}_i \in \mathcal{P}_g, j \in \mathcal{P}_h} \delta(\vec{x}_i, \vec{x}_j)}{\sigma_{\mathcal{P}_g} + \sigma_{\mathcal{P}_h}} \tag{2.49}$$

The cluster separation, $sep(\mathcal{P})$, is defined here as the ratio between the summed distances of the closest representatives of every pair of clusters and the between-cluster density. In well separated clusters the area between them has low density, so this term should be maximised.

$$sep\left(\mathcal{P}\right) = \frac{\sum_{g=1}^{k}\sum_{h=1,g\neq h}^{k}\min_{\vec{x}_i\in\mathcal{P}_g,\vec{x}_j\in\mathcal{P}_h}\delta\left(\vec{x}_i,\vec{x}_j\right)}{1+interDens\left(\mathcal{P}\right)} \tag{2.50}$$

The neighbourhood of an object, in the context of the within-cluster density, is defined as a hyper-sphere where the radius is the standard deviation of the data set. The density of an object, $\vec{x}_i$, is the number of objects in $\mathcal{P}_g$ that are within its neighbourhood.

$$den\left(\vec{x}_i\right) = \sum_{i=1}^{n_g} f\left(\vec{x}_i,\vec{x}_i^g\right) \tag{2.51}$$

where $f\left(\vec{x}_i,\vec{x}_i^g\right)$ was previously defined in equation 2.45. The average density of a cluster is the average of the ratio between the density of each object and the standard deviation of the data set, given as:

$$intraDens\left(\mathcal{P}\right) = \frac{1}{k}\sum_{g=1}^{k}\frac{1}{n_g}\sum_{i=1}^{n_g}\frac{den\left(\vec{x}_i^g\right)}{\sigma_{\mathcal{D}}} \tag{2.52}$$

CDbw should be maximised and will not be influenced by k. The cluster quality index, CDbw, is defined as the product of the between-cluster density and the separation of the clusters, $CDbw\left(\mathcal{P}\right) = intraDens\left(\mathcal{P}\right)\cdot sep\left(\mathcal{P}\right)$.

2.3.5 RMSSDT & RS

The Root-Mean-Square Standard Total Deviation (RMSSTD) [132] of a cluster is a measure of its heterogeneity. A lower RMSSTD indicates that the cluster is heterogeneous, however, there is no guidance as to what a "high" or "low" value is as it is a characteristic of the data set. Calculating the RMSSTD of a cluster formed at an iteration of a hierarchical algorithm can help to establish if creating the cluster

was beneficial. The RMSSTD of a cluster is the value of total variation of the cluster divided by the number of objects within. For other techniques the sum of the RMSSTD of each cluster can be used as a measure of quality based upon density and not on separation. This technique is not sensitive to k.

$$RMSSTD\left(\mathcal{P}\right) = \sum_{g=1}^{k} \frac{\mathcal{P}_{\mathcal{P}_g}^{T}}{n_g - 1} \tag{2.53}$$

R-Squared (RS) [132] is a cluster quality measure used as part of the same ensemble of cluster quality measures. RS is the ratio of the between-group variation of the clustering solution and the total variation of the data set: $RS\left(\mathcal{P}\right) = \mathcal{P}^{B}/\mathcal{D}^{T}$.

RS ranges from zero to one. A value of one indicates that the clusters are well separated and a value of zero indicates the inverse.

2.3.6 Silhouette Width Criterion

The Silhouette Width Criterion (SWC) [129] is a cluster quality measure that assesses how well objects fit into the clusters they are members of. Firstly the average distance between an object, $\vec{x}_i$, and the other members in its cluster, $\mathcal{P}_g$, must be calculated.

$$a\left(\vec{x}_i\right) = \frac{\sum_{j \in \mathcal{P}_g, j \neq \vec{x}_i} \delta\left(\vec{x}_i, \vec{x}_j\right)}{n_g - 1} \quad \vec{x}_i \in \mathcal{P}_g \tag{2.54}$$

Secondly, the average distance between $\vec{x}_i$ and the objects in its nearest neighbour cluster must be calculated as $b\left(\vec{x}_i\right)$:

$$b\left(\vec{x}_i\right) = \min_{\mathcal{P}_h \neq \mathcal{P}_g} \frac{\sum_{\vec{x}_j \in \mathcal{P}_h} \delta\left(\vec{x}_i, \vec{x}_j\right)}{n_h} \quad \vec{x}_i \in \mathcal{P}_g \tag{2.55}$$

where $\mathcal{P}_h$ is the nearest neighbour cluster of $\mathcal{P}_g$.

The silhouette of an object, $s(\vec{x}_i)$, measuring how well $\vec{x}_i$ fits into the cluster it is currently a member of, rather than its nearest neighbour cluster is given as:

$$s(\vec{x}_i) = \frac{b(\vec{x}_i) - a(\vec{x}_i)}{\max(a(\vec{x}_i), b(\vec{x}_i))} \tag{2.56}$$

If $s(\vec{x}_i)$ is close to one then $\vec{x}_i$ is well classified. If it is close to negative one then $\vec{x}_i$ is misclassified and should belong to the neighbour cluster. If it is close to zero then it is unclear if $\vec{x}_i$ should belong to the neighbour cluster or to the cluster it is currently a member of. The average silhouette width for a clustering solution is defined as:

$$SWC(\mathcal{P}) = \frac{\sum_{i=1}^{n} s(\vec{x}_i)}{n} \tag{2.57}$$

The average silhouette width can be used as a measure of the quality of a clustering solution. Higher values of this measure are desirable. The Silhouette width is insensitive to k. Silhouette width does not explicitly relate to either density or separation.

2.3.7 Connectivity & Disconnectivity

Chen and Wang [20] introduced a new clustering method using a multi-objective algorithm to determine a solution for a data set. Two quality measures are used as the objectives: Overall Deviation, equivalent to $\mathcal{P}^W$ defined in equation 2.13, and connectivity. The algorithm optimises the clustering solution by minimising both measures.

First we define the nearest neighbour of an object, $\vec{x}_i$, as, $\vec{x}_i^1$, where

$$\delta\left(\vec{x}_i, \vec{x}_i^{1}\right) \leq \delta\left(\vec{x}_i, \vec{x}_j\right) \quad \text{for } \forall \vec{x}_j \in \mathcal{D}, \quad \text{where } \vec{x}_i \neq \vec{x}_i^{1} \tag{2.58}$$

A solution is connected if objects are in a cluster with their l nearest neighbours. Connectivity measures the degree to which neighbouring objects have been placed in the same cluster by calculating penalties for each object. An enhancement to connectivity has been proposed by Handl and Knowles [62]. The penalty for an object in relation to its u^{th} nearest neighbour is 0 if they are in the same cluster and $\frac{1}{u}$ otherwise. The decreasing penalties emphasise the nearest neighbours and allow for clusters smaller than l.

$$Conn = \sum_{i=1}^{n} \sum_{u=1}^{l} p\left(\vec{x}_i, \vec{x}_i^{u}\right) \tag{2.59}$$

$$\text{where} \qquad p\left(\vec{x}_i, \vec{x}_i^{u}\right) = \begin{cases} \frac{1}{u} & \text{if } \nexists\, \mathcal{P}_g : \vec{x}_i \in \mathcal{P}_g \wedge \vec{x}_i^{u} \in \mathcal{P}_g, \\ 0 & \text{otherwise} \end{cases} \tag{2.60}$$

We have not performed a significant investigation into selecting the correct value of l. The original works suggested that 6 would be a sensible value to use.

Disconnectivity [97] measures the violation of kNN and kMN consistency in a solution, it is therefore conceptually similar to connectivity. If $\vec{x}_i$ is the k^{th} nearest neighbour of $\vec{x}_j$ and $\vec{x}_j$ is the l^{th} nearest neighbour of $\vec{x}_i$ then $\vec{x}_i$ and $\vec{x}_j$ are the p^{th} mutual nearest-neighbour of each other, where $p = \max(k, l)$. A kNN consistent cluster is a group of objects where the k nearest-neighbours of each object are also within the cluster to which that object belongs, and a kMN consistent cluster is a group of objects where the k mutual nearest-neighbours are members of the same cluster.

$$disconn\left(\mathcal{P}\right) = \sum_{\mathcal{P}_g \in \mathcal{P}} \sum_{\vec{x}_i \in \mathcal{P}_g} \sum_{\vec{x}_j \notin \mathcal{P}_g} \left(n\left(\vec{x}_i, \vec{x}_j\right) + n\left(\vec{x}_j, \vec{x}_i\right)\right) \frac{1}{\delta\left(\vec{x}_i, \vec{x}_j\right)} \tag{2.61}$$

$$\text{where} \qquad n\left(\vec{x}_i, \vec{x}_j\right) = \begin{cases} 1 & \text{if } \vec{x}_i \in \left\{\vec{x}_j^1, \ldots, \vec{x}_j^l\right\} \\ 0 & \text{otherwise} \end{cases} \tag{2.62}$$

A low value of $disconn(p)$ indicates a good clustering.

2.4 External Clustering Quality Measures

An external cluster quality measure is a method that evaluates the quality of a clustering solution, $\mathcal{P}$, against a known optimal clustering solution, $\mathcal{P}'$. The optimal solution is designated as such because it has been labelled by a human or has been specifically generated for the purpose. External cluster quality measures compare two given clustering solutions and determine how similar they are to each other where one solution is the intended solution and the other is the solution to be tested. Here we have chosen to use the Rand Statistic, the Jaccard Coefficient and The Fowlkes and Mallows index which measures the similarity of any two clustering solutions, $\mathcal{P}_g$ and $\mathcal{P}_h$.

A pair of objects, $\vec{x}_i$ and $\vec{x}_j$, are classified as follows:

SS If $\vec{x}_i \in \mathcal{P}_g$, $\vec{x}_j \in \mathcal{P}_g$, $\vec{x}_i \in \mathcal{P}_h$ and $\vec{x}_j \in \mathcal{P}_h$

SD If $\vec{x}_i \in \mathcal{P}_g$, $\vec{x}_j \in \mathcal{P}_g$, $\vec{x}_i \in \mathcal{P}_h$ and $\vec{x}_j \notin \mathcal{P}_h$

DS If $\vec{x}_i \in \mathcal{P}_g$, $\vec{x}_j \notin \mathcal{P}_g$, $\vec{x}_i \in \mathcal{P}_h$ and $\vec{x}_j \in \mathcal{P}_h$

DD If $\vec{x}_i \in \mathcal{P}_g$, $\vec{x}_j \notin \mathcal{P}_g$, $\vec{x}_i \in \mathcal{P}_h$ and $\vec{x}_j \notin \mathcal{P}_h$

where $\mathcal{P}_g \in \mathcal{P}$ and $\mathcal{P}_h \in \mathcal{P}'$ and **SD** stands for "same" and "different".

The values of a, b, c and d are the numbers of pairs of objects classified as **SS**, **SD**, **DS** and **DD** respectively. From this the Rand Statistic[122] is defined as:

$$\mathcal{RI}\left(\mathcal{P}_g, \mathcal{P}_h\right) = \frac{a+d}{a+b+c+d} \tag{2.63}$$

The value of R is between 0 and 1. A value of 0 indicates that the solutions are totally dissimilar and a value of 1 indicates that they are identical.

The Jaccard Coefficient [74] is calculated in a similar manner:

$$\mathcal{J} = \frac{a}{a+b+c} \tag{2.64}$$

and also gives values between 0 and 1. The result is usually similar to the value of R.

The Fowlkes and Mallows index [41] is defined as:

$$\mathcal{FM} = \sqrt{\frac{a}{a+b}.\frac{a}{a+c}} \tag{2.65}$$

Higher values of $\mathcal{FM}$ indicate that the two clustering solutions are of greater similarity.

2.5 Summary

In this chapter we have introduced the clustering problem, including distance measures, types of clustering and characteristics of clustering solutions. Additionally we reviewed the main classic methods of partitional and hierarchical clustering techniques, including methods to select the number of clusters, k.

Finally, we introduced the concept of Clustering Quality Measures (CQM). CQMs are used to determine the quality of a clustering solution so they can be ranked in an order of preference. They are essential to our later work on optimisation algorithms for clustering. We also introduced a number of external measures for comparing how similar clustering solutions are.

Overall we have found that the clustering quality problem is hard. Correct clustering depends on the context of the data, the types of clusters that the end user wishes to find within the data and the method of measuring the similarity. The preferred solution to a clustering problem is often subjectively chosen.

Chapter 3

Cluster Quality Measures Experimentation

3.1 Introduction

In Chapter 2, we reviewed a number of techniques to determine the best clustering solution from a set of clustering solutions using cluster quality measures. Our overall goal is to create a multi-objective clustering algorithm, hence understanding which Cluster Quality Measures to use as objectives is important. We believe that a Cluster Quality Measure should exhibit certain behaviour. A Cluster Quality Measure that robustly increases in value as the clustering solution improves is more useful than one that shows us low quality for all solutions other than the perfect solution.

In this chapter we propose and use a method of assessing Cluster Quality Measures to test their behaviour. We have published our results in [83]. Our study will focus on the following measures: the Variance Ratio Criterion (VRC), the Davies-Bouldin index (DB), the SD validity index (SD), the SDbw validity index (SDbw), the CDbw validity index (CDbw), Root-Mean-Square Standard Total Deviation (RMSSTD), R-Squared(RS), Silhouette Width Criterion (SWC), Overall Deviation (Dev), Connectivity and Disconnectivity.

We anticipate that some measures may fare better in certain situations. For example, a measure may be poor at dealing with clustering solutions where clusters are of varying sizes or where clusters are very sparse. However, to test different scenarios we propose a method for generating problems which includes a number of parameters to produce diverse test beds.

In general, we expect the value of a clustering quality measure to be at its highest for what we determine to be the "perfect" or intended clustering solution. Since datasets are generated synthetically, the correct clustering solution is assumed to be the intended clustering generated. To corrupt a perfect clustering solution we propose to change the cluster membership of some objects in the data set in a random incremental fashion, thereby introducing noise in the clusters. We believe this should lead to a stepwise deterioration in the quality of the clustering solution which should be detectable using the internal quality measures.

An external quality measure should be used to measure the similarity of a given solution with the intended solution. Here we use the Rand Statistic defined in Equation 2.63 as our external quality measure. As solutions are successively corrupted, the similarity between the corrupted solution and the intended solution, and therefore the value of the external quality measure, should decrease. We can then measure the correlation between the internal clustering quality measure and the external clustering quality measure. High correlation over a set of incrementally degraded clustering solutions indicates robust performance of the internal quality measure.

3.2 Methodology

For this study, we constructed a large number of synthetic data sets following a technique used by Milligan [107]. For the study, one hundred and eight synthetic data sets were produced by identifying three parameters in the data generation process and combining them to produce data sets. We introduce an additional parameter

that allows us to introduce outliers, so we explore the following: datasets with different number of clusters; different dimensionality of the data; different sizes of clusters; and different number of outliers. We also extend the data construction method by using larger ranges of possible values for each parameter. Milligan has explained in detail the method for generating the data sets [108]. It is briefly summarised here and then expanded upon.

To generate data objects we must first identify the boundaries of each cluster. Points are generated within these boundaries. The boundaries of the clusters may not overlap in the first dimension. The length of the boundaries is selected from a uniform distribution running from ten to forty. The centroid of each cluster is then determined. The value of the centroid for a given dimension is the midpoint of its boundary for that dimension. The standard deviation of a cluster for a given dimension is defined as a third of the length of its boundary for that dimension. Points are generated using a multivariate normal distribution with the centroid of the distribution defined as the centroid of the cluster to be generated. The diagonal entries of the variance-covariance matrix are set to the standard deviations of each dimension of the cluster. Each point that is generated must be within 1.5 standard deviations of the centroid. The process is repeated for each cluster that is to be generated.

In our experimentation, first we consider the number of clusters in a data set: values between two and forty are used. The second parameter is the number of dimensions: values used range from two to twenty dimensions within Euclidean space so that no one dimension dominates the other dimensions. The third parameter is the proportion of objects that are members of each cluster. For this, we use three possible designs: in the first design the objects are evenly distributed between all of the clusters; in the second design a cluster consists of 10% of the objects and the rest are as evenly distributed as possible; in the third design, a cluster consists of 60% of the objects and the rest are as evenly distributed as possible. Finally, the fourth parameter is the proportion of objects that are generated as outliers.

Outliers are defined as within 9 standard deviations of the centroid of each cluster. The proportion of outliers is either: 0%, 20% or 40% of the objects generated.

The variation of these factors produces six thousand six hundred and sixty nine different data set designs. Each design is then generated three times resulting in a final set of twenty thousand and seven data sets. Each data set consists of five hundred objects.

For each data set, twenty clustering solutions are generated where the first represents the optimal clustering solution and those that follow are copies where a proportion of the objects have been randomly misclassified in 1% steps. That is, in the first solution all objects are correctly assigned to the clusters; in the second solution 1% of objects are misclassified, then 2%, etc. The quality of the clustering solutions should decrease as more objects are misclassified. For each solution, the value of each internal quality measure is calculated. This process produces a set of results for each internal quality measure. Each set of results for an internal quality measure associated with a data set are normalised to the range 0 to 1. Finally minimisation measures are then inverted so they become maximisation measures to allow for easier comprehension of the results.

For each solution, the Rand Index [107] is calculated in relation to the optimal solution. It is expected that as the clusters are misclassified the Rand Index should deteriorate in value. The correlation between each set of average results for a measure and a set of average results of the Rand Index for a given data set is calculated using the Pearson's Correlation Coefficient [128] across the 20 clustering solutions. Measures that are robust with respect to the deterioration of a solution should correlate well to the Rand Index and the correlation should be consistent. In Figure 3.1 we show a simple example of the decrease in the value of the Rand Index as we use the misclassification process on the classic Iris data set [2]. We start with the standard three cluster solution and then degrade this solution. This is included for illustration only as we do not use this data set in this work. In this example we have misclassified the objects in 1% steps as it is a very small data set.

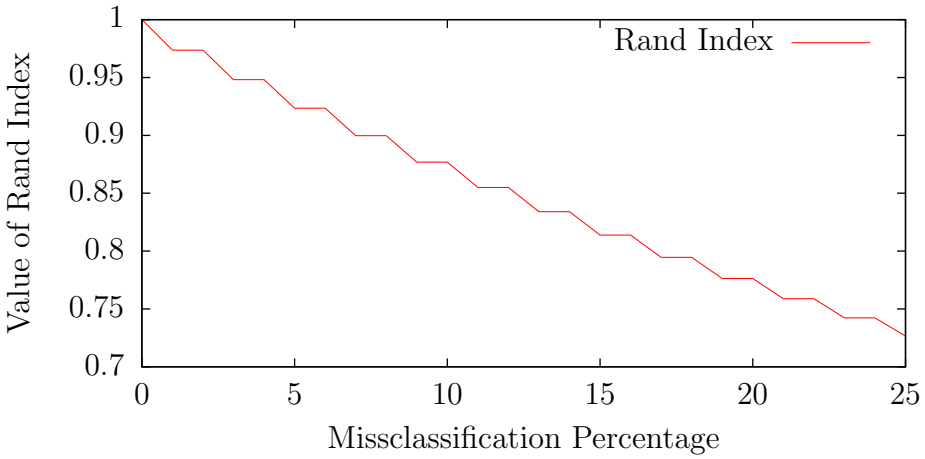


Figure 3.1: Example of the change of the Rand Index on the Iris data set as it is misclassified.

As each data set design was generated three times we averaged the correlations from each of the data set generations, to assess each given criterion (outliers, dimensions, density factor and number of clusters). This allows us to examine the results and isolate the behaviour that is the result of one of these factors.

3.3 Results

We examined problems with different numbers of dimensions, numbers of clusters, numbers of outliers and relative sizes of clusters. We have presented our results as a series of plots or tables. A key that is common to all plots is given in figure 3.2. Our findings are as follows:

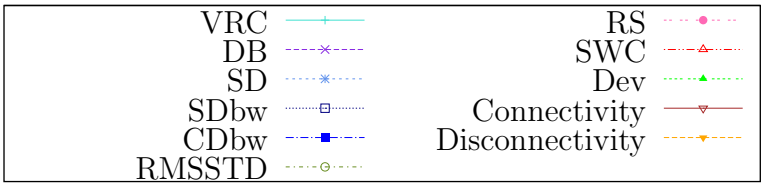


Figure 3.2: Key to plots

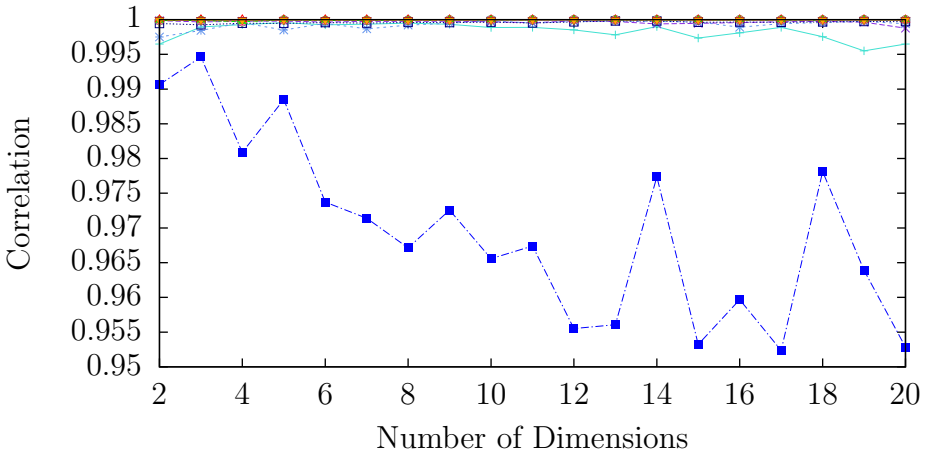


Figure 3.3: Change in maximum correlation value between external and internal cluster quality measures as the number of dimensions is varied.

3.3.1 Varying the Number of Dimensions

We first test the effect of generating data in a lower or higher dimensional space. The maximum, minimum and mean values of correlation are shown in figures 3.3, 3.4 and 3.5 respectively. Measures tend to perform better as the number of dimensions is increased initially and this may be due to a larger and more sparse space allowing for better clustering definition.

In terms of performance, we observe that CDbw is very erratic as we vary the number of dimensions whereas all the other cluster quality measures have steady behaviour. We see that at low dimensionality the minimum correlation shown by SD, SWC and DB is low and it increases with the number of dimensions until there are 5 dimensions when it levels out. The other measures remain constant both in terms of minimum and maximum correlation. The best values, showing the highest minimum and maximum correlation, are obtained by Connectivity, Disconnectivity, RS and Dev.

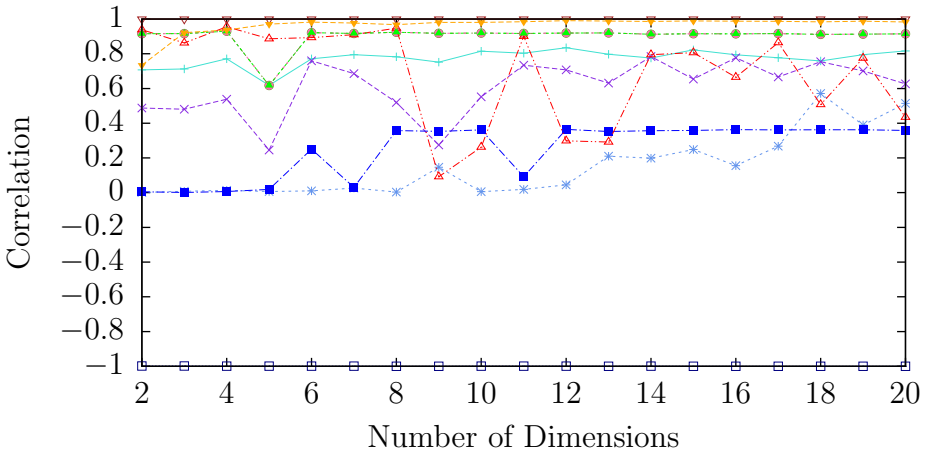


Figure 3.4: Change in minimum correlation value as the number of dimensions is varied.

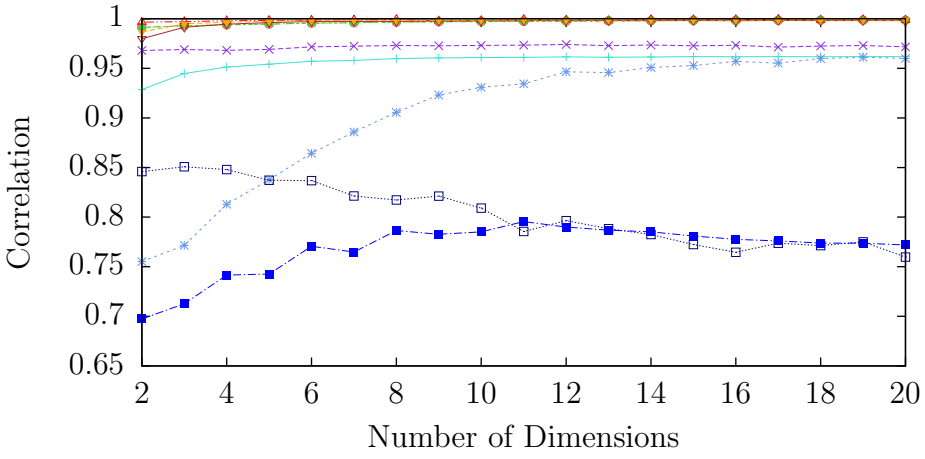


Figure 3.5: Change in mean correlation value between external and internal cluster quality measures as the number of dimensions is varied.

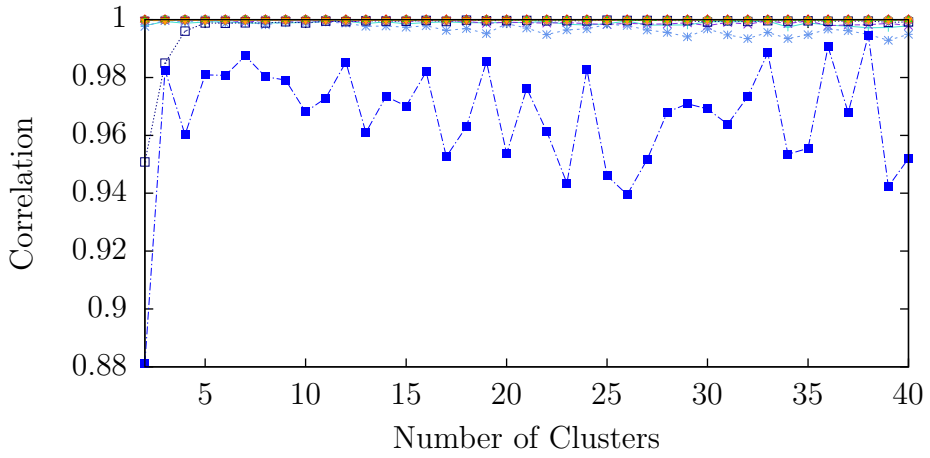


Figure 3.6: Change in maximum correlation value as the number of clusters is varied.

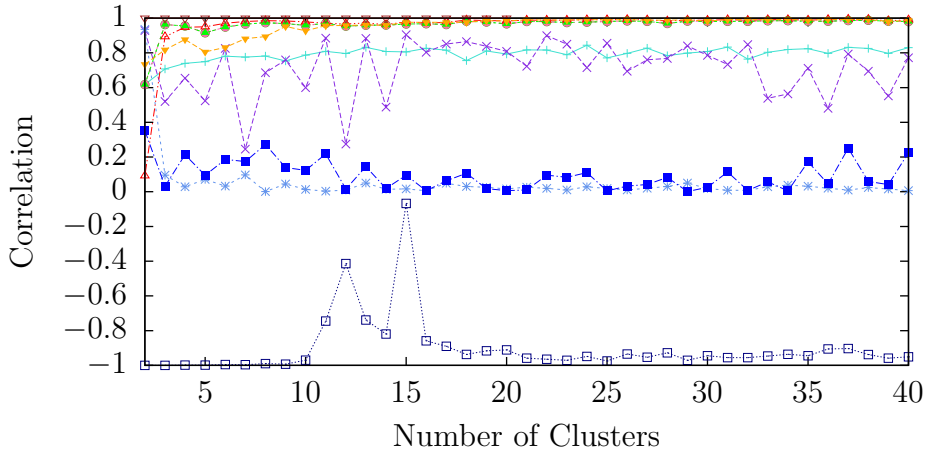


Figure 3.7: Change in minimum correlation value as the number of clusters is varied.

3.3.2 Varying the Number of Clusters

The results for this are presented in figures 3.6, 3.7 and 3.8. As the number of clusters increases the correlation for some measures deteriorates. The performance of the CDbw measure appears to be less than the other measures. SDbw improves as the number of clusters increases. Also maximum and minimum observed correlations get closer as the number of clusters increases, showing that performance of the cluster quality measures tends to converge as the number of clusters is increased. We can also see that the connectivity and disconnectivity do not appear to be affected by the change in the value of k.

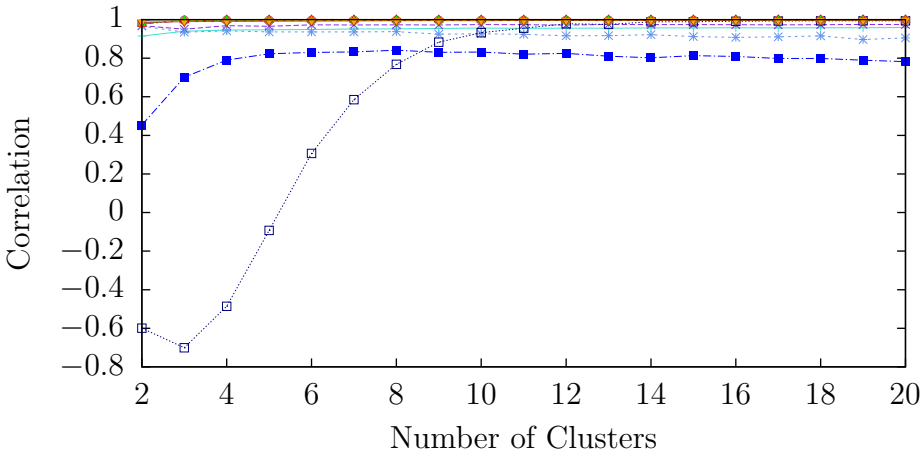


Figure 3.8: Change in mean correlation value as the number of clusters is varied.

Table 3.1: Change in maximum correlation value as the cluster size is varied.

CQM	a	b	c
VRC	0.9996	0.9994	0.9995
DB	0.9998	0.9998	0.9998
SD	0.9999	0.9998	0.9989
SDbw	0.9997	0.9998	0.9997
CDbw	0.9906	0.9946	0.9814
RMSSTD	1	1	1
RS	1	1	1
SWC	1	1	1
Dev	1	1	1
Connectivity	0.9999	0.9999	0.9999
Disconnectivity	1	0.9999	1

3.3.3 Varying the Cluster Size

For this, we present data in tables, as with only 3 different parameter values to test, the results are easier to appreciate in a table. The maximum minimum and mean correlation values are show in Tables 3.1, 3.2 and 3.3 respectively. In the tables, we have denoted the case where all of the clusters are the same size as a, the case where 10% of objects belong to one cluster and the rest are distributed among equal size clusters as b and the final case where 60% of objects belong to one cluster and the rest are distributed among equal size clusters as c. Connectivity, Disconnectivity, RS, RMSSTD and Dev show the best maximum and minimum correlations.

Table 3.2: Change in minimum correlation value as the cluster size is varied.

CQM	a	b	c
VRC	0.7349	0.7070	0.6182
DB	0.5384	0.5522	0.2457
SD	0.0066	0.0080	0.0011
SDbw	-0.9998	-0.9999	-0.9995
CDbw	0.0011	0.0050	0.0130
RMSSTD	0.9158	0.9455	0.6171
RS	0.9158	0.9455	0.6171
SWC	0.9497	0.9664	0.0917
Dev	0.9158	0.9455	0.6171
Connectivity	0.8051	0.6993	0.7206
Disconnectivity	0.7757	0.8078	0.7344

Table 3.3: Change in mean correlation value as the cluster size is varied.

CQM	a	b	c
VRC	0.9569	0.9579	0.9574
DB	0.9732	0.9726	0.9697
SD	0.9091	0.9101	0.8982
SDbw	0.8210	0.8106	0.7774
CDbw	0.7673	0.7691	0.7681
RMSSTD	0.9971	0.9969	0.9959
RS	0.9971	0.9969	0.9959
SWC	0.9992	0.9989	0.9971
Dev	0.9971	0.9969	0.9959
Connectivity	0.9963	0.9963	0.9964
Disconnectivity	0.9976	0.9975	0.9975

Table 3.4: Change in maximum correlation value as the number of outliers is varied.

CQM	a	b	c
VRC	0.9739	0.9980	0.9996
DB	0.9998	0.9998	0.9997
SD	0.9999	0.9998	0.9997
SDbw	0.9998	0.9997	0.9997
CDbw	0.9656	0.9854	0.9946
RMSSTD	1	1	1
RS	1	1	1
SWC	1	1	1
Dev	1	1	1
Connectivity	0.9999	0.9999	0.9999
Disconnectivity	1	1	1

3.3.4 Varying the Number of Outliers

Again with only 3 different parameter values, results are presented as Tables 3.4, 3.5 and 3.6 for maximum, minimum and mean correlation values respectively. We find that changing the number of outliers generated for each cluster does not have a significant effect on the correlation between the clustering quality measures and the external clustering quality measures. The best measures in terms of both minimum and maximum correlation are Connectivity and Disconnectivity followed by Dev, RS and RMSSTD.

The results show that in the case of CDbw, Connectivity and Disconnectivity the mean correlations increase as the number of outliers increase. The maximum correlations increase for all of the measures. It may be that more distributed clusters are treated as many clusters by the validity measures. The Rand Index measures whether objects are in the same cluster in a pairwise fashion so if a cluster is divided into more than one part it will not degrade the value as much as dividing and merging two clusters.

Table 3.5: Change in minimum correlation value as the number of outliers is varied.

CQM	a	b	c
VRC	0.7070	0.8591	0.6182
DB	0.6313	0.2457	0.4804
SD	0.0011	0.0038	0.0061
SDbw	-0.9999	-0.9996	-0.9992
CDbw	0.0161	0.0011	0.0073
RMSSTD	0.9116	0.9145	0.6171
RS	0.9116	0.9145	0.6171
SWC	0.8930	0.0917	0.2625
Dev	0.9116	0.9145	0.6171
Connectivity	0.7206	0.6993	0.8396
Disconnectivity	0.7344	0.7757	0.8814

Table 3.6: Change in mean correlation value as the number of outliers is varied.

CQM	a	b	c
VRC	0.9078	0.9731	0.9914
DB	0.9776	0.9725	0.9654
SD	0.9160	0.9054	0.8961
SDbw	0.8078	0.8085	0.7927
CDbw	0.7266	0.7726	0.8054
RMSSTD	0.9982	0.9969	0.9948
RS	0.9982	0.9969	0.9948
SWC	0.9995	0.9986	0.9972
Dev	0.9982	0.9969	0.9948
Connectivity	0.9951	0.9968	0.9971
Disconnectivity	0.9970	0.9979	0.9977

Table 3.7: Minimum, Maximum, Mean, S.D. of correlation values for each cluster quality measure.

CQM	Min	Max	Mean	S.D.
VRC	0.6182	0.9996	0.9574	0.0410
DB	0.2457	0.9998	0.9718	0.0271
SD	0.0011	0.9999	0.9058	0.1517
SDbw	-0.9999	0.9998	0.8030	0.5425
CDbw	0.0011	0.9946	0.7682	0.1344
RMSSTD	0.6171	1	0.9966	0.0069
RS	0.6171	1	0.9966	0.0069
SWC	0.0917	1	0.9984	0.0151
Dev	0.6171	1	0.9966	0.0069
Connectivity	0.6993	0.9999	0.9963	0.0093
Disconnectivity	0.7344	1	0.9975	0.0071

3.3.5 Overall Results

Our results show that Connectivity and Disconnectivity show near identical performance. Of these clustering quality measures we would recommend the use of Connectivity over Disconnectivity as it has the highest performance based on our experiment, and the most simple definition. Further experimentation with various values of l may show improvements to the performance of Connectivity or Disconnectivity.

Of the measures proposed by Halkidi et al., previously defined in section 2.3.4, we find the first iteration of their proposed measures, SD, to be the most effective on average in this experiment; so we recommend the use of SD instead of CDbw and SDbw. SD has the potential to have a lower performance than CDbw but on the whole seems to perform better.

Our results also show that the measures RS, RMSSTD and Dev provide the same information. Hence, only one of these measures should be used when trying to find optimal clustering solutions. The performance of these measures is good; there was a strong correlation between the deterioration of the clustering solution and the value of the measures so any one of them is a good candidate for a clustering quality measure.

The other measures of clustering quality we examined: VRC, DB and SWC, showed mixed performance. The VRC measure was highly correlated only in some cases. On average DB was better than VRC but still showed poor performance. SWC is the best performer of this group.

3.4 Summary & Conclusions

In this chapter we discussed cluster quality measures and we discussed what is a desirable cluster quality measure. We determined that cluster quality measures can be grouped together according to what they measure; for example: separation,

density or connectivity. For discovering interesting clusters it may be desirable to use a mix of clustering quality measures assessing different qualities of the clustering solutions. We have also discussed a search strategy. We decided that cluster quality measures that show a steady change in value as the clustering solutions improve or degrade in quality are more useful than those that can only determine the "best" solution to measure this steady behaviour. We degraded clustering solutions in a step wise manner by randomly reassigning some members of the data set from one cluster to another and assessed how cluster quality measures behaved. We proposed the use of the correlation between an external quality measure, the Rand Statistic, and the various internal quality measures as our indicator of robust performance.

Our results have shown that the measures based on the concept of connectivity have the highest correlation and hence the most robust performance in this study. Some of the measures presented have very similar performance so may not work well together in the context of MO algorithms, where we aim to select measures that are not completely correlated to each other (i.e. from different performance subgroups). Connectivity in an MO context may be complemented by another measure such as RS or Dev as these measures also performed well but attempt to investigate different concepts of cluster quality.

Future work could include a study to identify redundant cluster quantity measures. Cluster quality measures that perfectly correlate with each other could be considered redundant as there would be no point in using them together. If we group cluster quality measures that correlated with each other we could then select the best cluster quality measures based upon other factors such as runtime performance. Future studies could also consider different types of data sets. The data sets that we used in this study were all generated with a multivariate normal distribution and results of this study may not be applicable to other types of data set. The hyper spherical clusters that we generated should be identifiable by algorithms such as k-means but they may behave very differently if the clusters took the form of different shapes. We leave this as an open question for future research.

As part of our overall goals we now have a method for identifying useful Cluster Quality Measures and identified some suitable Cluster Quality Measures. The findings have been published in [83] and the identified measures will be useful for our work with Multi-Objective Evolutionary Algorithms.

Chapter 4

Solving Problems with Multiple Objectives

In an optimisation problem, a large number of valid solutions exist [22, 88, 152]. These solutions can be judged using a function that assesses their quality to determine if one solution is better than another. A solution to a problem is expressed as a set of variables that may all be continuous variables, discrete variables or a mixture. Problems where all of the variables are discrete are known as combinatorial optimisation problems as there is a finite number of possible solutions. In this work we are attempting to optimise the assignment of objects in a data set to a number of clusters. These assignments are of a binary nature, as we are not considering the fuzzy clustering problem. Therefore, the problem of crisp clustering can be considered as a combinatorial optimisation problem.

To find a solution to a combinatorial optimisation problem it may be possible to enumerate all of the possible solutions to the problem. However, for anything but a relatively small problem it is not computationally possible to enumerate all of these solutions. This has led to the development of a large number of heuristic algorithms that search the solution space in an attempt to find an optimal or near optimal solution without testing all of the solutions. In the simplest case we may have a problem where we must find an integer in a large range that minimises a

function. A simple search strategy may involve reducing or increasing the input value and continuing in the direction that gives the desired result until a minimum value of the function is found. Such local search techniques are likely to become trapped in a local minimum and miss the global minimum. Several algorithms have been developed to overcome the problems of local search, for example Simulated Annealing [19, 86] and Tabu search [47, 48]. There are also other methods such as Evolutionary Algorithms inspired by evolution in nature, and particularly genetic algorithms which are relevant to our research.

4.1 Multiple Criteria Decision Making

Multiple Criteria Decision Making (MCDM) is the study of how to find solutions to problems where there are multiple conflicting criteria. MCDM is often performed on problems where each objective is said to be a "black box". General MCDM algorithms are not tailored to specific objective functions.

The process of making a decision where there is only one criterion is relatively trivial. For example if we wish to purchase a piece of equipment we may choose to purchase the piece of equipment with the minimum cost. This choice is relatively trivial as there is only a single criterion. We may also decide that we should purchase the piece of equipment that is of the highest quality. This is also a trivial single criterion problem. However, we may decide that we should purchase a piece of equipment that is of high quality and is also of minimum cost to get the best "value for money". These objectives are in conflict with each other as expensive equipment if often of higher quality. We would be required to make a decision where there are multiple conflicting criteria.

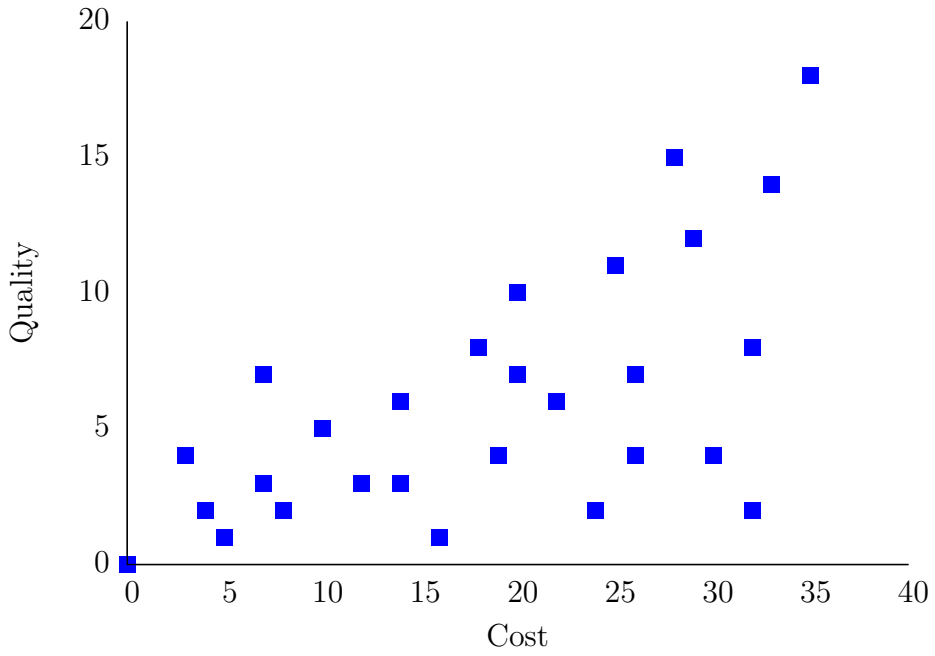


Figure 4.1: Example of a set of solutions (squares) to a problem where the goals are to maximise quality and minimise cost.

In figure 4.1 we show an example where there are a number of solutions to the problem of buying a piece of equipment of the highest quality and lowest cost. That is to say we aim to maximise quality and minimise cost. We have represented each solution as a square where the highest quality solutions are towards the top and the lowest cost solutions are on the left so the best solutions should be in the top left. We cannot say that one solution is better than all of the other solutions based on both criteria but we can say there are solutions that we would consider best if we were only making the decision using a single criterion.

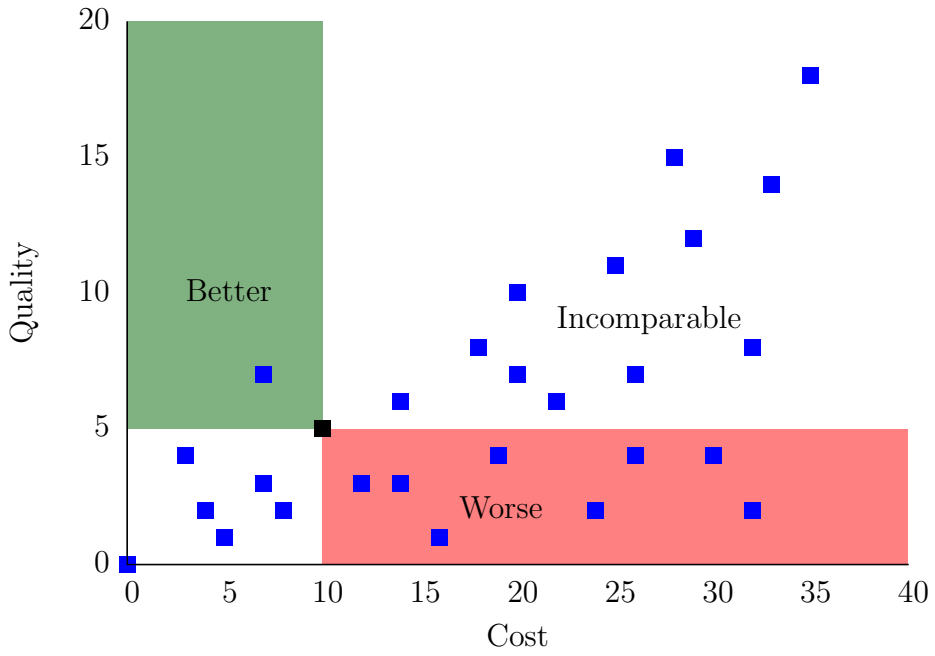


Figure 4.2: Example of which solutions (squares) can be considered: better (in the green area), worse (in the red area) or incomparable (in the white area) when compared to a specific solution (the black square).

If we identify a single solution, we may define certain properties of it. In figure 4.2 we have plotted a number of solutions as squares and highlighted an individual solution by colouring it black. We may say that some solutions are worse than the solution we have highlighted. These solutions are of greater cost and of lower quality, this region has been coloured red. We may also say that some solutions are better that the highlighted solution as they cost less and they are of higher quality, this region has been coloured green. The other solutions are said to be incomparable to the highlighted solution as they are neither better or worse than the highlighted solution. They are either more costly and of higher quality or less costly and of lower quality than the highlighted solution. The incomparable solutions are not better or worse than the highlighted solutions for both criteria.

We will formally define a solution to a given problem as:

$$\vec{s} = (s_1, \ldots, s_o) \tag{4.1}$$

where $\vec{s}$ is an o dimensional vector and each s_e represents the value of one objective function. There are o objective functions. The value of each s_e may be any real value, $s_e \in \mathbb{R}$, and is the value of the e^{th} objective function used to assess a solution to a problem. The exact range of values is dependent on the objective function that is used to assess each solution to the given problem.

Sets of solutions are often generated in an attempt to solve a given problem. In this work we will denote a set of solutions as:

$$\mathcal{S} = \{\vec{s}_1, \ldots, \vec{s}_N\} \tag{4.2}$$

where each $\vec{s}_i$ is a vector containing the value of the objective functions for a solution and N is the number of solutions in the set.

4.1.1 Pareto Dominance

Previously we stated that when we are comparing a solution to another solution we can define the solutions as: better, worse or incomparable. Formally we will call these dominating, dominated or incomparable solutions respectively. In Figure 4.2 we highlighted a specific solution as a black square. The solutions in the green area dominate the highlighted solution, the solutions in the red region are dominated by the highlighted solution and the other solutions are incomparable.

In figure 4.3 we have highlighted several solutions and labeled them A, B, C and D. Solution A has a more desirable cost than solution B. However, solution B has a more desirable quality than solution A demonstrating two trade off solutions. Solutions C and D are dominated by solutions A and B as the values of the objectives of solutions C and D are less desirable or equal to the values of the objectives of solutions A and B. Solution D is strictly dominated by solutions A, B and C. However, solution C is not strictly dominated by solutions A and B as the value of the quality objective of solutions A and C is equal and the value of the cost objective is equal in solutions B and C. We now formally define these concepts with notation:

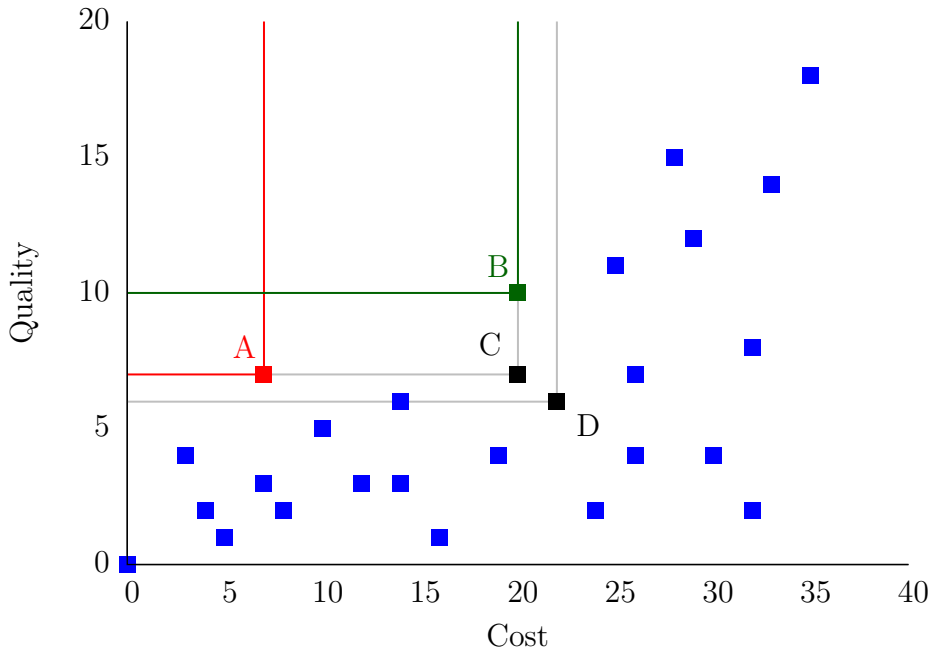


Figure 4.3: Example where solutions that dominate and strictly dominate other solutions have been highlighted for discussion.

Given two solutions, $\vec{s}_i$ and $\vec{s}_j$, we may say: $\vec{s}_i$ is preferred to $\vec{s}_j$; $\vec{s}_j$ is preferred to $\vec{s}_i$; $\vec{s}_i$ is equal to $\vec{s}_j$; or neither $\vec{s}_i$ or $\vec{s}_j$ is the preferred solution. A solution is preferred to another solution if it dominates the other solution. $\vec{s}_i$ dominates $\vec{s}_j$, within a problem where all of the objectives are to be minimised, if all of the values of the objectives in $\vec{s}_i$ are less than or equal to the corresponding values in $\vec{s}_j$ and there is at least one objective in $\vec{s}_i$ with value less than the corresponding value in $\vec{s}_j$. Formally we define this as:

$$\vec{s}_i \succeq \vec{s}_j \text{ if } s_{ie} \leq s_{je} \forall s_{ie} \in \vec{s}_i, s_{je} \in \vec{s}_j \wedge \exists s_{ie} < s_{je}, s_{ie} \in \vec{s}_i, s_{je} \in \vec{s}_j \tag{4.3}$$

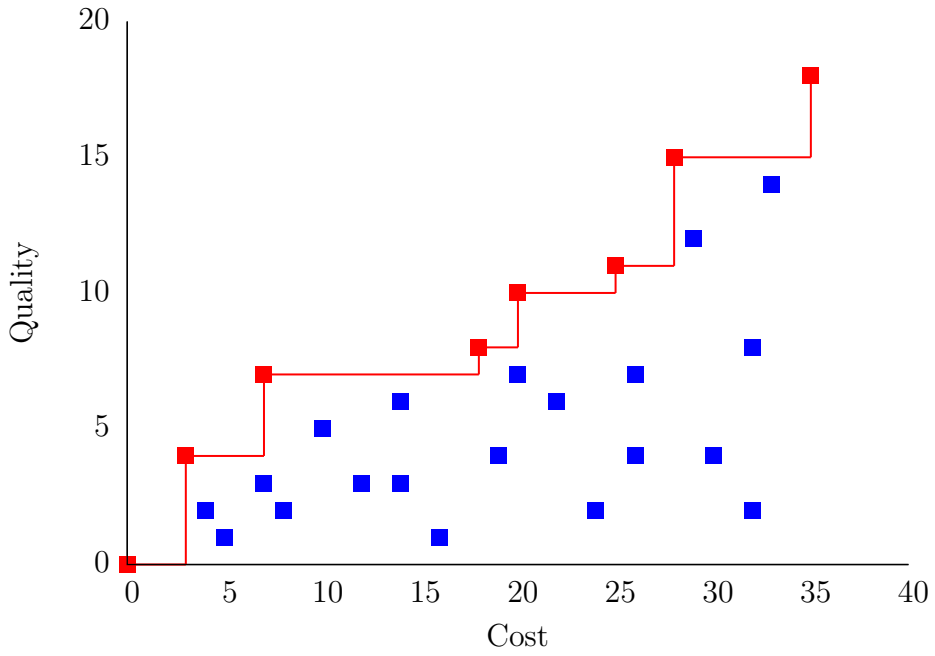


Figure 4.4: Example of which solutions (red) form the Pareto front from a given set of solutions (squares) to a problem.

Solutions that are not dominated by any other solution are said to be non-dominated. When a solution is non-dominated we can say that no other solution is better. From a set of solutions we may identify a subset of solutions that are non-dominated, this set is known as the Pareto front. In Figure 4.4 we have highlighted the solutions that form the Pareto front in our example problem in red. MCDM often results in discovering the Pareto front or a set of solutions that is the best obtainable approximation of the Pareto front.

Alternatively, we may say a solution, s_{ie}, strictly dominates another solution, s_{je}, if all of the values of the objectives in $\vec{s}_i$ are less than the equivalent values in $\vec{s}_j$. This differs from the previous definition of dominance as equal values may not be dominated. This is formally defined as:

$$\vec{s}_i \succ \vec{s}_j \text{ if } s_{ie} < s_{je} \forall s_{ie} \in \vec{s}_i, s_{je} \in \vec{s}_j \tag{4.4}$$

The definition of non-strict dominance that we have given earlier is also known as preference or weak dominance. If $\vec{s}_i \succ \vec{s}_j$ then $\vec{s}_i \succeq \vec{s}_j$, however the reverse is not true. Differing methods of defining dominance lead a different interpretation of the Pareto front [36, 118].

Alternatively more relaxed definitions of dominance may be used such as cone dominance [13, 71]. This is a more relaxed form of dominance that may encourage more evenly spread Pareto fronts. We show a conceptual example of cone dominace in Figure 4.5.

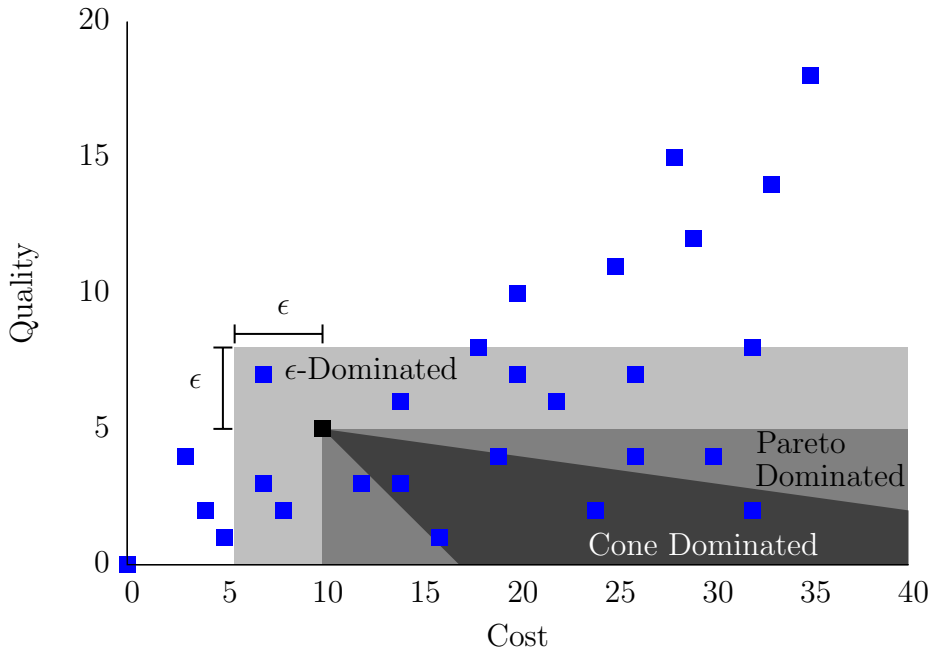


Figure 4.5: Example of ϵ-dominance and cone dominance.

The concept of ϵ-dominance was proposed by Laumanns [95]. ϵ-dominance is also a more relaxed form of dominance that uses a weight, ϵ, to allow a solution to dominate solutions around it. An example of ϵ-dominance is shown in Figure 4.5. Where ϵ-dominance is used in place of the standard definition of dominance, solutions will be at least ϵ apart from each other in all dimensions [7], therefore the value of ϵ must be tuned so a useful number of non-dominated solutions are ϵ-dominated. Finding solutions to problems with a small number of potential solutions may be hampered by the use of ϵ-dominance [68].

4.2 Solving MCDM Problems

There are a number of approaches to solving MCDM problems. A simple way to solve the problem would be to attempt to enumerate all of the possible solutions. This would be unfeasible for all but relatively small problems regardless of the number of objectives. It is also possible to transform multi-objective problems into single objective problems through objective ranking or other techniques. However, this would not produce a Pareto set of solutions. It would instead produce a single solution therefore missing all of the possible trade-off solutions in a Pareto front [44].

Before we embark upon solving MCDM problems we must first define what the goal of the optimisation problem is. We may wish to identify all of the Pareto optimal solutions, but as we said previously this may be impossible if the search space is large. We may therefore try to find a representative subset or an approximation of the Pareto front. We must then try to decide how good the quality of these approximations are. We can assess the quality of approximations by using assessments of Pareto quality. In Section 4.3 we discuss this further.

In our studies we plan to use Multi-Objective Evolutionary Algorithms (MOEA) to solve MCDM problems. Before we introduce MEOAs we first introduce Genetic Algorithms (GA) as they are the single objective precursors of MOEAs.

4.2.1 Genetic Algorithms

A Genetic Algorithm (GA) [40, 52, 66] is a subclass of Evolutionary Algorithms (EA). A number of reviews of the topic have been published [17, 50, 113]. These are algorithms that utilise techniques inspired by natural evolution. Evolution is the combination of two processes: natural selection and sexual reproduction. Natural selection is a process where a population of individuals is reduced in size, individuals that are not fit enough to survive until mating are eliminated from the population and do not pass their traits to the next generation. Fitter individuals are more

likely to find food, survive attacks by predators and find suitable mates. Sexual reproduction is a process where chromosomes from two individuals are mixed to create two new individuals with characteristics from both parents. During sexual reproduction sometimes mutation occurs, this is where some chromosomes of the individual are randomly changed, which may lead to new characteristics emerging in the population.

Algorithm 4.1 Outline of a Simple Genetic Algorithm

Require: N *the population size*
Require: m *the maximum number of generations*
Require: x *the crossover rate*
Require: μ *the mutation rate*
  $P_0 \leftarrow$ population of N randomly generated solutions
  evaluate(P_0) *evaluate solutions*
  $g \leftarrow 1$ *the current generation*
  repeat
    select $(1 - x) \times N$ solutions from P_{g-1} and copy into P_g
    select $x \times N$ solutions from P_{g-1} in pairs for crossover
    crossover each pair of selected solutions and insert into P_g
    randomly mutate $\mu \times N$ solutions from P_g
    evaluate(P_g) *evaluate new solutions*
    $g \leftarrow g + 1$
    create P_g
  until fittest solution in P_g is fit enough **or** $g \geq m$
  return fittest solution in P_g

A GA mimics the process of evolution to solve problems. In Algorithm 4.1 we show an outline of a simple Genetic Algorithm. In that simple algorithm we first create a population of randomly generated solutions to our problem. Each solution in our population is then evaluated to determine its fitness (quality). The following tasks are then repeated until a solution that meets a quality threshold is found or a maximum number of generations is reached.

The first task in our simple GA is to select solutions from the current population to copy into the next population unchanged. A number of techniques are available to select which solutions should be chosen [14, 51]. Three classic techniques for selecting solutions are: rank selection, Roulette wheel selection and tournament

selection. In rank selection the solutions are sorted by fitness and the probability a solution is chosen is based upon the solution's position in the sorted list. In roulette wheel selection the probability that a solution is chosen is proportional to the fitness of the solution, this is often likened to a roulette wheel where the size of each segment is controlled by the fitness of each solution. Tournament selection is a process where a number of solutions are randomly chosen from the population, the fittest solution is retained and the other solutions are discarded. This process is repeated until the desired number solutions are chosen.

The second task in our simple GA is to crossover solutions from the current generation to produce new solutions to join the next generation. Solutions are selected for crossover in the same manner as the copy step. Crossing over solutions mimics the process of sexual reproduction. In a simple example where each solution is a list of boolean values a crossover may just cut each list in two at a random point and then exchange the values either side of the point. We elaborate on some of the simple crossover operators later in Section 4.4.1.2 in the context of modifying representations of clustering solutions.

The third task is mutation. A mutation operator randomly modifies solutions in the new population. If each solution was a list of booleans the mutation operator could randomly flip a bit in the list to mutate a solution. In more complex problems where the solutions are represented as list of real numbers, a random value could be added or subtracted to change a solution. In Section 4.4.1.1 we detail some simple mutation operators.

The final task is to evaluate the new population of solutions. The fittest solution in the population must first be identified. If the fittest solution is fitter than a desired threshold then the algorithm may finish executing and return the solution as its result. If the fittest solution does not exceed the desired fitness the algorithm should return to the first task and repeat the process. A maximum number of generations should also be specified as an alternative stopping criterion.

4.2.2 Multi-Objective Evolutionary Algorithms

Multi-Objective Evolutionary Algorithms follow broadly the same procedure as the Genetic Algorithms we described in Algorithm 4.1 [46]. The main difference here is how the fitness of the solutions is calculated and the method used to select those to be passed from one generation to the next. In the single objective case the solutions can simply be ranked by the value of the objective function but we cannot do this in the Multi-Objective case as we must take all of the objectives into consideration. A MOEA must also attempt to guide the whole population of solutions towards the Pareto front to find a wide range of high quality solutions.

Here we review several well known algorithms from the literature by dividing them into groups based on their method of determining the fitness of solutions.

4.2.2.1 Aggregation Based Algorithms

Aggregation based techniques were among the first to be used for determining the fitness of a solution in a multi-objective environment. These are techniques that combine all of the objectives into a single objective and then evaluate the fitness by computing the highest (or lowest in a minimisation problem) values of this new objective. A classic example of this is weighted-sum aggregation [73].

The weighted-sum algorithm uses a fitness function that combines all of the objective functions for the given problem. A random weight is applied to each of the objective functions and the weighted objectives are then summed. At each generation the weights are randomly regenerated to encourage exploration in different areas of the Pareto front.

A number of the non-dominated solutions are randomly added to the new population that is produced. The new population then passes through the process of crossover and mutation that is common to other evolutionary algorithms and then repeats until some termination criterion is met.

The weighted-sum algorithm also maintains a separate external archive of solu-

tions that contains all of the non-dominated solutions observed during the execution of the algorithm. This improves the performance of the algorithm as it ensures that all of the non-dominated solutions that are found during execution can be reported when the algorithm finishes.

Aggregation based methods can be advantageous as they are computationally efficient. Aggregation methods are not useful when we are attempting to identify a Pareto front that is non-convex or has non-convex regions [106] as these algorithms may not effectively find the solutions in these regions of the Pareto front [78]. These algorithms can be made more effective if more information is known about the problem that is being solved. For example, the weights can be set so that they scale the objectives appropriately. Poorly scaled weights may lead to one objective dominating the Pareto front.

4.2.2.2 Criterion Based Algorithms

Criterion based selection techniques differ from other selection techniques by using each of the objective functions in turn to select the solutions that pass to the next generation [46]. For a population of size N with o objective functions, o sub-populations of size $\frac{N}{o}$ are generated by assessing each sub-population with each objective function in isolation. Solutions are selected from the solutions selected from the sub-populations by using a single objective selection procedure such as roulette wheel selection that we previously described in Section 4.2.1. The sub-populations are then combined together and the solutions are shuffled into a random order to produce the next population of solutions for the algorithm.

Vector Evaluated Genetic Algorithm (VEGA) [130] is the classic example of a criterion based algorithm. The algorithm is based on a simple genetic algorithm. The algorithm has been modified to use the criterion based selection technique described above.

Criterion based algorithms are very simple to implement, which is advantageous. The algorithms are poor at finding tradeoff solutions that represent a compromise

between several objectives. They instead find a set of solutions where each excels at only one of the objectives.

4.2.2.3 Dominance Based Algorithms

An alternative method of ranking the fitness of individuals within a population is to use dominance-based methods. These were first proposed by Goldberg [51]. Dominance-based methods are divided into three main methods: dominance rank, dominance count and dominance depth.

Dominance Rank The dominance rank of a solution is based on the count of the number of other solutions that dominate it. The basic method of calculating the dominance rank of the solutions within a population begins by identifying the non-dominated solutions within the population. These solutions are then assigned a rank of 1. The remaining solutions are then ranked by the number of other solutions that dominate them. For a solution, $\vec{s}$, the rank is calculated as 1 + the number of solutions that dominate $\vec{s}$.

Using the dominance rank leads to the emergence of dominance classes, i.e. subsets of solutions that have the same dominance rank and therefore cannot be ordered. The fitness of the solutions within the same dominance class may be determined within the dominance rank in a number of ways. The density of the solutions may be used as an indicator of the fitness of solutions within a dominance rank. Density may be calculated in a number of ways. For example, a kernel method may be used that is a function of the distances to the other solutions within the same dominance rank. Alternatively a measure of the distance to the k-th nearest neighbour distance of the solution or a histogram method that measures the number of solutions within a bounding area around the solution being evaluated can also be used.

Multiple Objective Genetic Algorithm (MOGA) [38] determines the fitness of the solutions using the dominance rank of the solutions. The final fitness values of the solutions within the current population are determined from the dominance ranks

by using a Niche-formation method so that all of the solutions selected to form the next population are evenly distributed.

Niched Pareto Genetic Algorithm (NPGA) [67] uses the dominance rank of solutions as part of a tournament based selection method. A tournament based selection method is one that randomly chooses two or more solutions from the population; the fittest of those two solutions is added to the new population. This process is completed when enough solutions are selected to form a new population. In NPGA the solution that has the lowest dominance rank is considered the fittest solution and wins the tournament contest. In the case that two solutions have the same dominance rank, the tie must be broken using a niche method. This involves the calculation of the number of other solutions that are contained within a niche around the solutions. The solution that has the smallest number of other solutions within the niche that surrounds it is chosen as the winner of the tournament and is added to the next population.

Dominance Count The dominance count of a solution is the count of the number of individuals that solution dominates. The dominance count can be used in conjunction with the dominance rank. This technique is used by the algorithms Strength Pareto Evolutionary Algorithm (SPEA) [154] and SPEA2 [153]. The strength of a solution is its dominance count. A solution is stronger if it dominates more solutions. The dominance rank of the solutions is calculated in the normal way but each solution is weighted by its strength. A solution that is dominated by a solution that has a low strength is penalised less than a solution that is dominated by a solution that has a high strength.

4.2.2.4 Dominance Depth Algorithms

Dominance depth is a technique for determining fitness that divides the population into a series of non dominated fronts.

In Figure 4.6 we provide an example of how to calculate the dominance depth of

solutions. The solutions that are part of the Pareto front, i.e, the non-dominated solutions, have a dominance depth of 1. In the diagram these solutions are highlighted red. To determine which solutions have a dominance depth of 2 we eliminate the solutions that have a dominance depth of 1 from the population. We then recalculate which solutions form the Pareto front and define the solutions that form the new Pareto front as having a dominance depth of 3. In the diagram these have been highlighted green. We repeat this process until no solutions remain. In the diagram the blue solutions have a dominance depth of 3, the black solutions have a dominance depth of 4 and the grey solution has a dominance depth of 5.

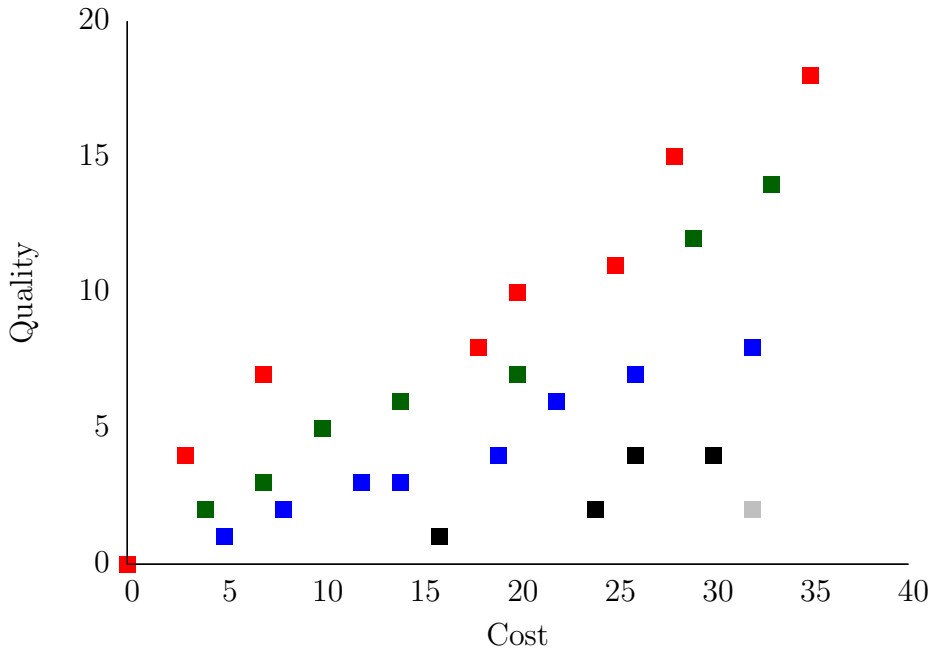


Figure 4.6: Example of Dominance Depth.

The process we have just described for calculating the dominance depth of solutions is at best a naïve and inefficient method. In Algorithm 4.2 we describe an algorithm for determining the dominance depth of solutions within a set of solutions, $\mathcal{S}$. The algorithm generates a set of fronts, $\{\mathcal{F}_1, \ldots, \mathcal{F}_f\}$, where $\mathcal{F}_c$ is the c^{th} front and a set of ranks, $\{r_1, \ldots, r_N\}$, where r_i is the dominance depth of $\vec{s}_i$.

Dominance depth is the technique used by Non-Dominated Sorting Genetic Algorithm (NSGA) [136] and NSGA-II [29] to determine the fitness of solutions within

Algorithm 4.2 Calculate Dominance Depth

- $\forall \vec{s}_i \in \mathcal{S}$
  - Initialise $\mathcal{X}_i = \emptyset$. $\mathcal{X}_i$ will contain all the solutions that are dominated by $\vec{s}_i$.
  - Initialise $n_i = 0$. n_i will be the number of solutions that dominate $\vec{s}_i$.
  - $\forall \vec{s}_j \in \mathcal{S}$
    - if $\vec{s}_i \succ \vec{s}_j$
      - $\mathcal{X}_i = \mathcal{X}_i \cup \{\vec{s}_j\}$
    - else if $\vec{s}_j \succ \vec{s}_i$
      - $n_i = n_i + 1$
  - if $n_i = 0$
    - $r_i = 1$. Rank of $\vec{s}_i$ is 1.
    - $\mathcal{F}_1 = \mathcal{F}_1 \cup \{\vec{s}_i\}$. Add $\vec{s}_i$ to the first front.
- $\mathcal{F}_2 = \emptyset$. Initialise second front.
- $c = 1$. Initialise front counter.
- while $\mathcal{F}_c \neq \emptyset$
  - $\mathcal{Q} = \emptyset$
  - $\forall \vec{s}_i \in \mathcal{F}_c$. For each solution in the current front.
    - $\forall \vec{s}_j \in \mathcal{X}_i$. For each solution dominated by $\vec{s}_i$.
      - $n_j = n_j - 1$. Reduce the number of solutions $\vec{s}_j$ is dominated by.
      - if $n_j = 0$ then $r_j = c + 1$ and $\mathcal{Q} = \mathcal{Q} \cup \{\vec{s}_j\}$. If $\vec{s}_j$ is no longer dominated add it to the current front.
  - $c = c + 1$
  - $\mathcal{F}_c = \mathcal{Q}$

a population. NSGA-II also uses a parameter called the crowding distance to determine the fitness of solutions that have same dominance depth. The crowding distance is a measure of how close a solution is to its neighbouring solutions. Populations of solutions with a large average crowding distance lead to increased diversity in the population of solutions. The crowding distance of the i^{th} solution in the c^{th} front is $\mathcal{F}_c(d_i)$. The crowding distance of a solution is a measure of its distance to its neighbours. The crowding distance of the tails of a front are set to infinity so they are always selected. The calculation of the crowding distance is given in Algorithm 4.3.

Algorithm 4.3 Calculate Crowding Distance

- $\forall \mathcal{F}_c \in \mathcal{F}$
  - $\forall \vec{s}_i \in \mathcal{F}$. For each solution in the current front.
    - $\mathcal{F}_c(d_i) = 0$. Set crowding distance to 0.
  - $e = 1$. Initialise objective counter.
  - while $e \leq o$. For each objective function.
    - $sort(\mathcal{F}_c, e)$. Sort the individuals in the front based on the value of the objective function.
    - $\mathcal{F}_c(d_1) = \infty$. Assign infinite distance to boundary.
    - $\mathcal{F}_c(d_f) = \infty$. Assign infinite distance to boundary.
    - $\forall i \in \{2, \ldots, f-1\}$
      - $\mathcal{F}_c(d_i) = \mathcal{F}_c(d_i) + \frac{s_{(i+1)e} - s_{(i-1)e}}{\mathcal{F}_e^{max} - \mathcal{F}_e^{max}}$, where $s_{(i)e}$ is the value of the e^{th} objective function of the i^{th} solution in $\mathcal{F}$.

The fitness of solutions in the population maintained by NSGA-II is calculated using the crowded comparison operator, $\prec_n$, which sorts solutions by their dominance depth and then by their crowding distance. $\vec{s}_i \prec_n \vec{s}_j$ if $r_i < r_j$ or $r_i = r_j$ and $\mathcal{F}_c(d_i) < \mathcal{F}_c(d_j)$. That is to say, $\vec{s}_i$ has a lower dominance depth than $\vec{s}_j$ or they belong to the same front and the crowding distance of $\vec{s}_i$ is lower than the crowding distance of $\vec{s}_j$.

To execute NSGA-II it must be supplied with a suitable random start population,

this may be randomly generated or generated in a method that is suitable for the problem at hand. The algorithm then selects solutions for mutation and crossover using a standard binary tournament selection with the crowded comparison operator, $\prec_n$. The selected solutions are then mutated and crossed over using supplied mutation and crossover operators. The solutions are added to the current population which is then sorted again using the crowded comparison operator. The next population is then filled with the fittest solutions until it is full, this ensures that all the current and previous best solutions are retained which ensures convergence to the optimal Pareto front. The process is then repeated until a maximum number of generations is met.

In our experiments that we detail in subsequent chapters of this thesis we use NSGA-II as the main framework of our research. We provide start populations, mutation operators and crossover operators to an implementation of this algorithm.

4.2.2.5 Recent Algorithms

Some relatively recent developments in MOEAs now use the hyper volume indicator to guide the selection of solutions. We elaborate on the hyper volume indicator in Section 4.3.1 in the context of using it as a method of assessing and comparing the quality of Pareto fronts. Briefly, the hyper volume measure calculates the volume of the objective space that is covered by a Pareto front. As a Pareto front gets closer to an optimal Pareto front, the volume of the objective space covered should increase. Several algorithms have been developed that use this technique.

S Metric Selection Evolutionary Multi Objective Algorithm (SMS-EMOA) [9, 34] is a similar algorithm to NSGA-II but it uses hyper volume to select a single solution to be replaced at each generation. Hypervolume E Optimisation (HypE) [3] is another algorithm that uses the hyper volume indicator but also adds further improvements that allow for a larger number of objectives to evaluated in a feasible timescale. These more recent developments may be better MOEAs than the ones we use in this work, but we leave this for future work.

4.3 Evaluation of Pareto Fronts

In this section we will introduce some techniques to evaluate Pareto fronts. Previously in Equation 4.2 we defined a set of solutions to a problem as $\mathcal{S}$. A Pareto front was defined as a set of solutions where each solutions is not dominated by any other solution (included or not in that set). When we come to evaluate the performance of an MOEA, an important task is assessing the quality of the set of solutions that is produced. We could also assess algorithms based on other criteria such as execution time or number of generations to converge.

To evaluate a Pareto front we must have other Pareto fronts to compare it to. When we execute a MOEA many times or execute many MOEAs we will generate many Pareto fronts. Because of this, we will first introduce some additional notation. We will define a family of Pareto fronts as:

$$\mathfrak{S} = \{\mathcal{S}_1, \ldots, \mathcal{S}_{\mathfrak{s}}\} \tag{4.5}$$

where $\mathfrak{s}$ is the number of Pareto fronts within a family. $\mathfrak{S}$ will represent all of the Pareto fronts discovered while attempting to solve a problem.

Some techniques for evaluating Pareto fronts require an optimal, or idealised, Pareto front to compare the Pareto front generated by the algorithm against. This will be defined as:

$$\mathcal{S}^* = \{\vec{s}_1^*, \ldots, \vec{s}_N^*\} \tag{4.6}$$

where each solution within the optimal front is defined as: $\vec{s}_i^* = (s_{i1}^*, \ldots, s_{io}^*)$. $\mathcal{S}^*$ is in practice equivalent to the definition of $\mathcal{S}_a$ given in Equation 4.2.

Where an optimal Pareto front is not known one may be approximated from $\mathfrak{S}$ by combining all $\vec{s}_{ai}$ that are not dominated by any solutions in other Pareto fronts.

To assess the solutions we normalise the values of each objective so one objective

is not dominant [36]. $\mathcal{Y}$ is the set of all $\mathcal{S}_a$ and $H_{\mathcal{Y}}$ is the smallest hypercube that contains all of $\mathcal{Y}$:

$$H_{\mathcal{Y}} = \left\{ \vec{z} \in \mathbb{R}^o : a_i \leq z_i \leq b_i; \vec{a}, \vec{b} \in \mathcal{Y}; i = 1, \ldots, o \right\} \quad (4.7)$$
$$\text{where } \mathcal{Y} = \bigcup_{\mathcal{S}_a \in \mathfrak{S}} \mathcal{S}_a$$

The function $h_{\mathcal{Y}}(\vec{z}) : H_{\mathcal{Y}} \longmapsto [0,1]^o$ normalises the solution space. $h_{\mathcal{Y}}(\vec{s}_{ai})$ maps $\vec{s}_{ai}$ into the normalised objective space. For the rest of this work $\vec{s}_{ai}$ shall be taken as the value of $h_{\mathcal{Y}}(\vec{s}_{ai})$ so $s_{aij} \in [0,1]$.

Now, we introduce some of the measures available in the literature to compare Pareto fronts. All of these measures require that the fronts have been normalised first.

4.3.1 Volume of Dominated Space

The volume of the objective space that has been covered by a Pareto front, $\lambda(\mathcal{S}_a)$, may be used as an indication of the quality of a Pareto front. Pareto fronts that dominate a higher volume of the objective space are assumed to be better [125, 98]. However, Pareto fronts that dominate similar areas may be very different to each other, as seen in figure 4.7, where $\mathcal{S}_a$ and $\mathcal{S}_b$ have similar volumes of dominated space. The method used to determine the volume of the dominated objective space varies based upon the number of objectives [45]. For example, in a problem with two objectives it would be possible to calculate the area of a polygon formed from the Pareto front, whereas in a problem with a greater number of objectives an approximation of the Lebesgue measure [148] could be used. Algorithms such as the the Dimension-Sweep Algorithm [39] can also be used to give an indication of the volume of the dominated objective space.

In this work, to evaluate the area of $H_{\mathcal{Y}}$ that is dominated by a solution that

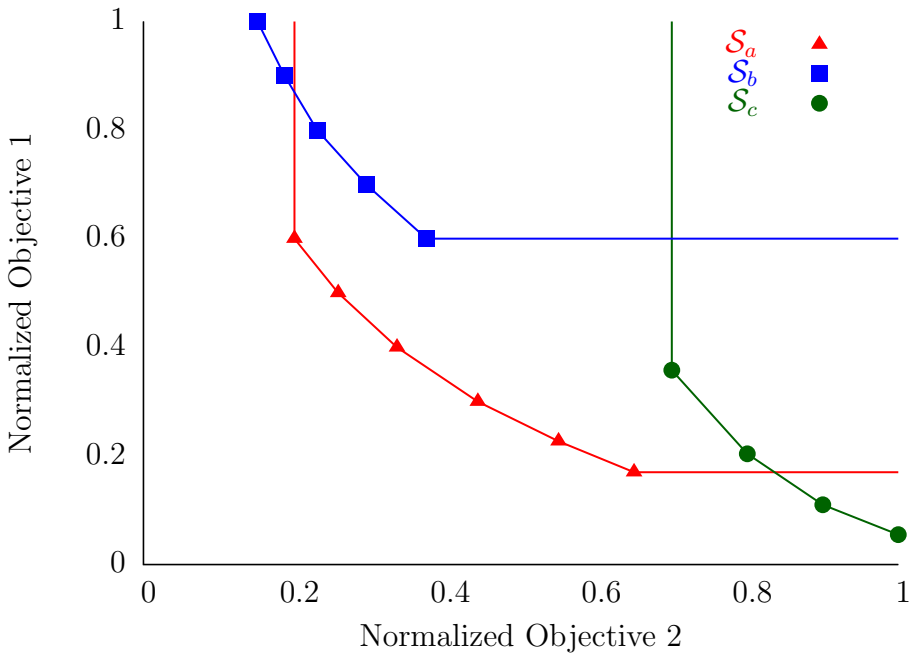


Figure 4.7: Example of the area dominated by different sets of solutions. $\lambda(\mathcal{S}_a) = 0.58805$ $\lambda(\mathcal{S}_b) = 0.3025$ $\lambda(\mathcal{S}_c) = 0.24795$.

has been previously normalised, $\mathcal{S}_a$, we form an o dimensional polytope by adding vectors on each edge of the hypercube at the minimum value of each dimension and in the corner:

$$\mathcal{S}_a \bigcup \{(\min s_{ai1}, \ldots, 1), \ldots, (1, \ldots, \min s_{aie}, \ldots, 1), \ldots (1, \ldots, \min s_{aio}), (1, \ldots, 1)\} \tag{4.8}$$

We then calculate the area of the polytope to give a measure of the area of the objective space. In the case of two objectives this method results in a simple polygon where the area can be calculated trivially. Calculating the volume where the number of objectives is higher than in this work is more complex but is out of the scope of this work.

4.3.2 Coverage

The C-Measure, proposed by [151], provides an indication of the coverage of a Pareto front, $\mathcal{S}_b$, by another Pareto front, $\mathcal{S}_a$. To calculate this, first the number of solutions in $\mathcal{S}_a$ that dominate or equal solutions in $\mathcal{S}_b$, $\vec{s}_{ai} \succeq \vec{s}_{bi}$, must be determined. A solution is said to dominate another solution if the value of each objective function is better than or equal to the corresponding value in the other solution and at least one objective value of the former solution is better than the corresponding value in the latter solution. This is then normalised by the size of the Pareto front. The C-Measure is defined formally as:

$$C\left(\mathcal{S}_a, \mathcal{S}_b\right) = \frac{\left|\{\vec{s}_{bi} \in \mathcal{S}_b : \exists \vec{s}_{ai} \in \mathcal{S}_a : \vec{s}_{ai} \succeq \vec{s}_{bi}\}\right|}{N} \tag{4.9}$$

where N is the number of solutions in Pareto front $\mathcal{S}_b$.

The value of $C\left(\mathcal{S}_a, \mathcal{S}_b\right)$ is in the interval $[0, 1]$ and gives the fraction of $\mathcal{S}_b$ dominated by $\mathcal{S}_a$. If $C\left(\mathcal{S}_a, \mathcal{S}_b\right) = 1$ then all solutions in $\mathcal{S}_b$ are dominated by $\mathcal{S}_a$, if $C\left(\mathcal{S}, \mathcal{S}_b\right) = 0$ then no solutions in $\mathcal{S}_b$ are strictly dominated by $\mathcal{S}_a$ and if $C\left(\mathcal{S}_a, \mathcal{S}_b\right) > C\left(\mathcal{S}_b, \mathcal{S}_a\right)$ then $\mathcal{S}_a$ has better solutions than $\mathcal{S}_b$.

A modified version of the C-Measure has also been proposed by [36, 125], this is defined as:

$$\tilde{C}\left(\mathcal{S}_a, \mathcal{S}_b\right) = \frac{\left|\{\vec{s}_{bi} \in \mathcal{S}_b : \exists \vec{s}_{ai} \in \mathcal{S}_a : \vec{s}_{ai} \succ \vec{s}_{bi}\}\right|}{N} \tag{4.10}$$

The $\tilde{C}$-Measure is the same as the C-Measure except the number of solutions in $\mathcal{S}_a$ that 'covered' in $\mathcal{S}_b$, $\vec{s}_{ai} \succ \vec{s}_{bi}$, is determined instead of dominated or equal solutions. The C-Measure also has the property that if $\mathcal{W}$ is a bi-dominated set, one where two Pareto fronts overlap, where $\mathcal{S}_a \subseteq \mathcal{W}$ and $\mathcal{S}_b \subseteq \mathcal{W}$ then $C\left(\mathcal{S}_a, \mathcal{S}_b\right)$ may take any value in $[0, 1]$. The $\tilde{C}$-Measure removes this property.

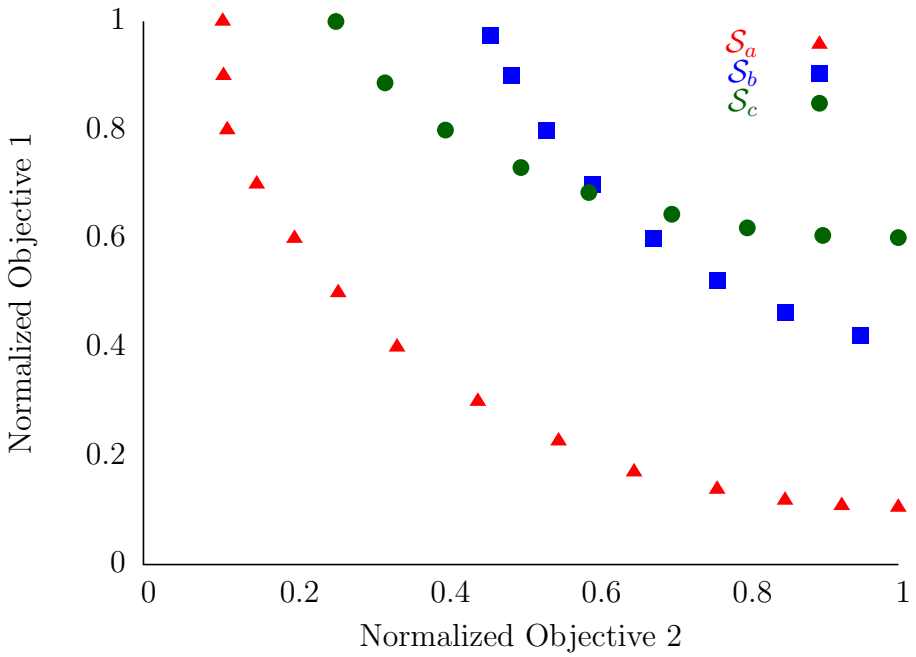


Figure 4.8: Examples of $\tilde{C}$-Measure; $\tilde{C}(\mathcal{S}_a, \mathcal{S}_b) = 1$, $\tilde{C}(\mathcal{S}_a, \mathcal{S}_c) = 1$, $\tilde{C}(\mathcal{S}_b, \mathcal{S}_a) = 0$, $\tilde{C}(\mathcal{S}_b, \mathcal{S}_c) = \frac{4}{9}$, $\tilde{C}(\mathcal{S}_c, \mathcal{S}_a) = 0$ and $\tilde{C}(\mathcal{S}_c, \mathcal{S}_b) = \frac{4}{8}$.

In this work we use the $\tilde{C}$-Measure to compute how many solutions dominate other solutions. For each set of solution, $\mathcal{S}_a$, in the family of solutions, $\mathfrak{S}$, we calculate the $\tilde{C}$-Measure against every other set of solutions, $\mathcal{S}_b$, in $\mathfrak{S}$. We then compute the number of cases where $\tilde{C}(\mathcal{S}_a, \mathcal{S}_b) > \tilde{C}(\mathcal{S}_b, \mathcal{S}_a)$ as $\tilde{C}_{\mathcal{S}_a}$. Higher values of $\tilde{C}_{\mathcal{S}_a}$ indicate that the set of solutions is better than other sets of solutions within the family of sets of solutions. An example of the $\tilde{C}$-Measure is shown in figure 4.8.

4.3.3 Spread

A Pareto front can be said to be good if the solutions upon it are evenly spread [98, 29]. This means that the search space has been well explored as the solutions are not clumped around local optima. A method of assessing this is the Spread measure [116], also known as the diversity metric [29].

The spread of a given Pareto front, $\mathcal{S}_a$, denoted $S(\mathcal{S}_a)$ measures the distance between consecutive solutions in the Pareto front. It also takes into account the

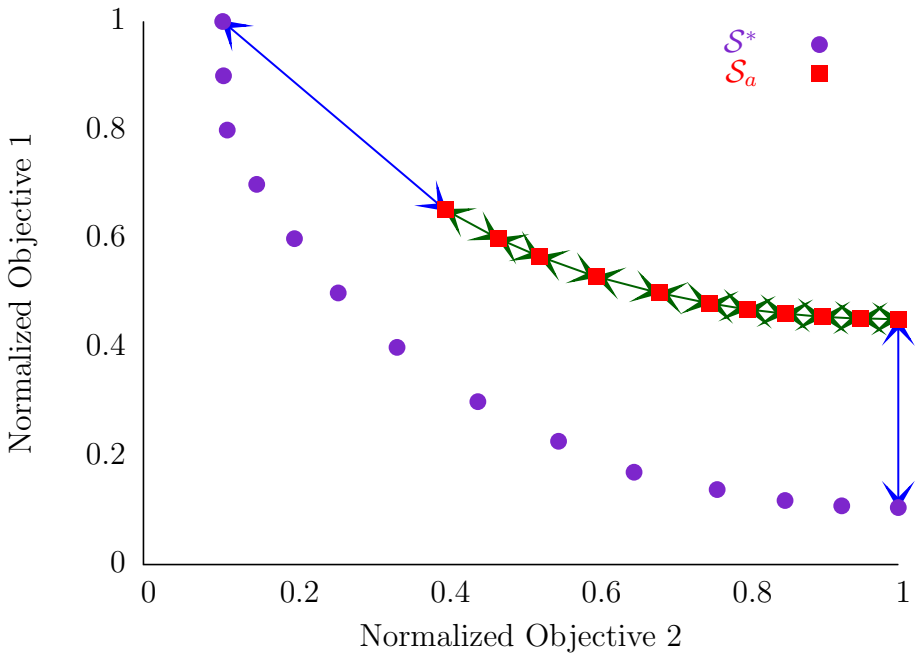


Figure 4.9: Example of the distances used to calculated $\mathcal{S}_a$ for a constructed optimal Pareto front and a constructed Pareto front.

extreme solutions in the optimal Pareto front and the Pareto front that is being assessed. There are o extreme solutions in each Pareto front.

$$S(\mathcal{S}_a) = \frac{\left(\sum_{e=1}^{o} \delta_e\right) + \sum_{i=1}^{N-1} \left(\delta\left(\vec{s}_{ai}, \vec{s}_{ai+1}\right) - \bar{\delta}\right)}{\left(\sum_{e=1}^{o} \delta_e\right) + (N-1)\,\bar{\delta}} \tag{4.11}$$

where $$\bar{\delta} = \left(\sum_{i=1}^{N-1} \delta\left(\vec{s}_{ai}, \vec{s}_{ai+1}\right)\right) / N - 1$$

and $$\delta_e = \delta\left(\vec{s}_{ai}, \vec{s}_i^*\right), \min s_{aie} \in \mathcal{S}_a, \min s_{ie}^* \in \mathcal{S}^*$$

If the Pareto front includes the extreme solutions and they are evenly spread then the value of the measure is zero. Higher values of the measure indicate that the Pareto front is not well spread. A visual example of the distances used to calculate the spread of a Pareto front is shown in figure 4.9.

4.3.4 Generational Distance & Inverted Generational Distance

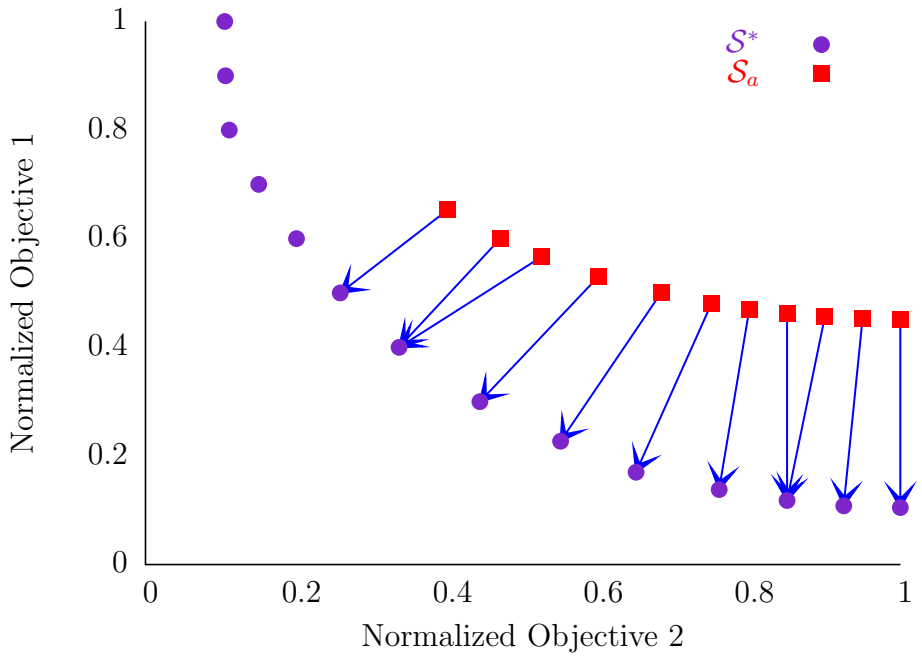
A Pareto front, $\mathcal{S}_a$, can be said to be good if the solutions within it are among the optimal Pareto front, $\mathcal{S}^*$. The Generational Distance, GD, [142] is a measure that determines if all of the solutions are also within the optimal Pareto front, in which case $GD(\mathcal{S}_a) = 0$, and if they are not it gives an indication of how far the set of solutions is from the optimal set of solutions $GD(\mathcal{S}_a) > 0$. It has been used in previous experimental studies as an indication of how good a Pareto front is [116, 30]. More precisely the generational distance measures the distance between each solution in the Pareto front and its nearest neighbour in the optimal Pareto front, and it is defined formally as follows:

$$GD(\mathcal{S}_a) = \frac{\sqrt{\sum_{i=1}^{N} \min \delta\left(\vec{s}_i, \vec{s}_j^*\right)}}{N}, \ \vec{s}_j^* \in \mathcal{S}^* \tag{4.12}$$

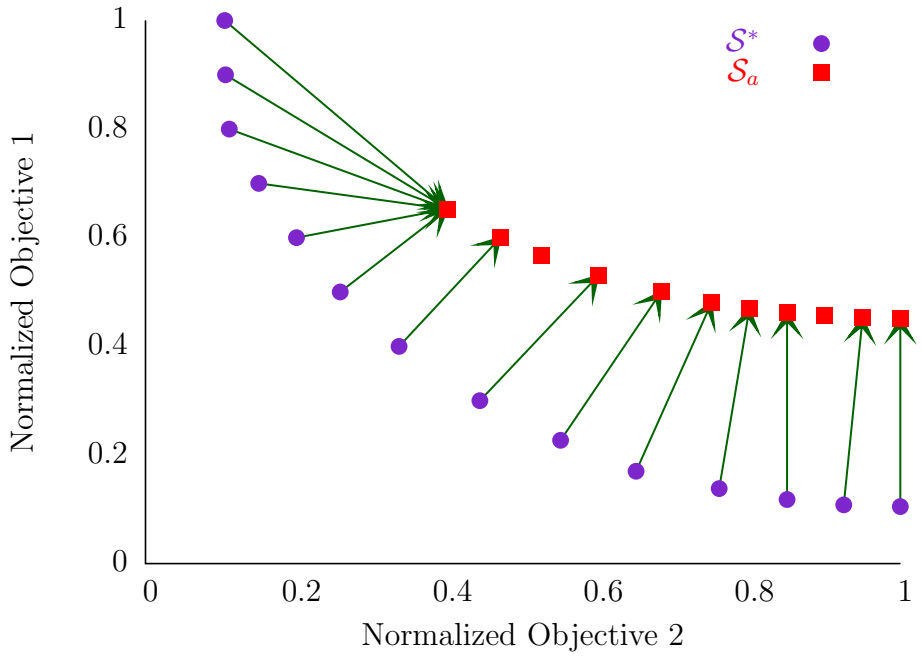
where N is the number of solutions in $\mathcal{S}_a$.

The generational distance gives an indication of how close a Pareto front is to the optimal Pareto front. However, the Pareto front being evaluated may only cover a small area of the optimal Pareto front, so a set of solutions may be indicated to be good by GD but may not cover the majority of the optimal Pareto front. This can be seen in figure 4.10a.

The Inverted Generational Distance, IGD, [99, 120] measures the distance between each solution in the optimal Pareto front and its nearest neighbours in the Pareto front that is being evaluated. If all of the solutions within the optimal Pareto front are contained in the Pareto front being evaluated then $IGD(\mathcal{S}_a) = 0$, if they are not then $IGD(\mathcal{S}_a) > 0$. This measure gives a better indication of how much of the search space has been covered by a given Pareto front. However, it does give extra weight to extreme solutions within the search space. This is shown in figure



(a) *GD*



(b) *IGD*

Figure 4.10: Distances used to calculate *GD* and *IGD* for a constructed optimal Pareto front, $\mathcal{S}^*$, and a constructed Pareto front $\mathcal{S}_a$ that is being evaluated.

4.10b. It is defined formally as follows:

$$IGD(\mathcal{S}_a) = \frac{\sqrt{\sum_{i=1}^{N} \min \delta\left(\vec{s}_i^*, \vec{s}_{aj}\right)}}{N},\ \vec{s}_{aj} \in \mathcal{S}_a \tag{4.13}$$

where N is the number of solutions in $\mathcal{S}^*$.

The distances between solutions used by GD and IGD are calculated using the Euclidean distance. It is assumed that all of the solutions have been normalised before these measures are applied to evaluating sets of solutions.

4.3.4.1 Entropy

We can use the information entropy of the population as a measure of the diversity of a given $\mathcal{S}_a$ [124]. Here we define entropy as:

$$E\left(\mathcal{S}_a\right) = -\sum_{i=1}^{N} p_i \log_2 p_i \tag{4.14}$$

where p_i is the probability that $\vec{s}_{ai}$ or one solution identical to it is randomly drawn from $\mathcal{S}_a$.

Higher values indicate that $\mathcal{S}_a$ is diverse. Diverse $\mathcal{S}_a$ are desirable as they lead to a larger set of non-dominated $\mathcal{S}_a$ for a decision maker to choose from.

4.4 Representations of Clustering Solutions for Evolutionary Algorithms

Throughout previous studies it was shown that important areas to investigate were the representations and operators. Not all operators may be used with all representations. We will provide an overview of some previous studies in Section 4.5. In this section, we group the available options by the representation. For each representa-

tion we detail some suitable mutation operators and crossover operators. We also detail an initialisation routine that generates sets of random solutions that we use throughout the thesis for all the algorithms we use.

Evolutionary algorithms rely on a good representation of solutions that can be manipulated appropriately. For clustering problems several representations have been proposed [23, 70]. Popular methods include representing a solution as a set of medoids, a set of centroids, labelling each object in the data set or a graph representation [62]. In the following section we describe some implementations of these representations.

A major challenge for clustering in general is how to determine the number of clusters [20]. In this work we are interested in algorithms that can set the number of clusters as part of the optimisation process, hence we use representations that allow for solutions with a non-fixed number of clusters. To this end, we present modifications to established representations and operators where necessary. Most of the required modifications are focussed around the crossover operators.

4.4.1 Medoid Based Binary Encoding

A clustering solution may be represented as a set of cluster prototypes where each object from the data set is associated with its closest cluster prototype to form a clustering solution. This encoding is known as Medoid Based Binary Encoding (MBBE) [92], $\vec{m} = (m_1, \ldots, m_n)$. Each object from the data set, $\vec{x}_i$, is associated with an element of the encoding, m_i. The value of m_i is 1 to indicate that the associated object is a prototype cluster or it is 0 otherwise [126]. Clustering solutions are generated from the encoding by assigning objects from the data set to the cluster prototype that is closest to the object. The value of k (number of clusters) is equal to the number of cluster prototypes or medoids and can vary as part of the optimisation process. A minimum of one object is assigned to each cluster by definition. This avoids creating empty clusters.

A conceptually identical technique, that we do not use in this work, is the integer based medoid representation [133] where each solution is a list of the indexes of objects to be used as medoids. This technique increases the complexity of varying k as it leads to encodings of different lengths which require more elaborate crossover operators.

However, there are some advantages to using the MBBE representation. The representation is scalable as the search set for selecting k clusters is lower than the total number of partitions of a dataset into k clusters [75]. Medoids are also interpretable in circumstances where physical feasibility is crucial. Centroids derived from objects in the dataset may not themselves be valid objects that can be verified by a domain expert, for example a variable may need to be an integer to construct a real world valid object whereas a centroid may have a real value.

The mutation and crossover operators that are often used with MBBE are unguided operators, that is to say, they are not task dependent so they do not use knowledge of the data set or the quality of previous clustering solutions to guide the search towards solutions of higher quality. Since they operate on fixed length strings, they are relatively straightforward to implement.

To initialise a medoid vector, $\vec{m}$, the value of k must first be decided. We use a random value drawn from a uniform distribution running from two to $\frac{n}{10}$, where n is the number of objects in the dataset. We then randomly select k objects from the data set to be the initial cluster prototypes.

4.4.1.1 Mutation Operators

Several simple mutation operators are suitable for mutating a given medoid $\vec{m}$. They are as follows:

Individual Bit Mutation A medoid, m_i, is randomly selected and the value is inverted, as seen in figure 4.11. This has the effect of adding or removing a cluster prototype.

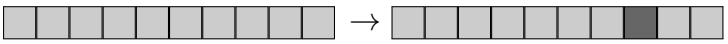


Figure 4.11: Individual Bit Mutation

Bezdek [11] used a mutation operator that added or removed one cluster from the solution. This mutation operator was functionally the same as the given mutation operator but was used on a matrix representation.

Multiple Bit Mutation First a threshold is set as $\frac{1}{n}$. For each m_i a random number in the interval $[0, 1)$ is generated. If it is smaller than that threshold then m_i is inverted, as in figure 4.12. This operator is more disruptive than the previous operator as it may lead to many clusters being added or removed from the clustering solution. Sheng [133] mutated solutions by flipping each feature in a solution.

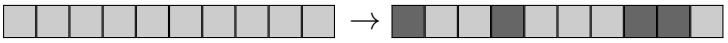


Figure 4.12: Multiple Bit Mutation

It is possible that this implementation may lead to a solutions that have a a value of k that is greater than $\frac{n}{10}$. This is undesirable as it will lead to a large number of medoids as this will lead to a large number of small clusters or even clusters that contain only single objects.

Invert Mutation All m_i in the solution are inverted to produce a solution that is the opposite of the original, as in figure 4.13. This operator is extremely disruptive to the solution, it may lead to a clustering solution with an abnormally high number of clusters. Clustering solutions generated from this technique are unlikely to be considered good clustering solutions. This mutation operator has not been used in any previous work and we do not expect it to generate good results.

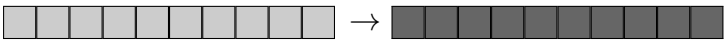


Figure 4.13: Invert Mutation

4.4.1.2 Crossover Operators

Crossover operators swap characteristics from two given parent solutions to produce two new child solutions that both have characteristics from each parent. The crossover operators we use with MBBE are as follows:

One Point Crossover A single point is randomly selected from the solutions and all of the values on a side of this point are exchanged to form two new solutions as seen in figure 4.14.

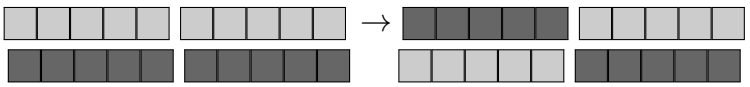


Figure 4.14: One Point Crossover

Two Point Crossover Two points within the solutions are selected and the values within the area between the points are exchanged to form two new solutions as seen in figure 4.15.

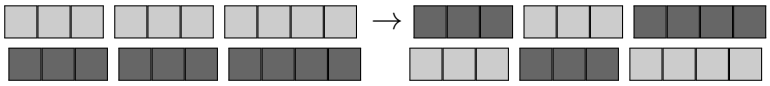


Figure 4.15: Two Point Crossover

Three Point Crossover Three points within the solutions are selected, the values contained in-between the first two points and the values in-between the third point and the end of the solutions are exchanged, as in figure 4.16.

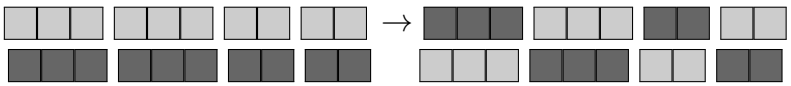


Figure 4.16: Three Point Crossover

Uniform Crossover For two solutions to be crossed, $\vec{m}$ and $\vec{m}'$, there is a 50% chance that m_i will be exchanged for m_i' [139]. This can be seen in figure 4.17.

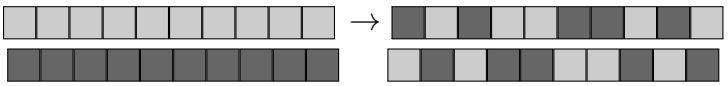


Figure 4.17: Uniform Crossover

4.4.2 Label Based Integer Encoding

Label Based Integer Encoding (LBIE) [90, 115, 100] represents a clustering solution by recording the cluster membership of each object from the data set. Each LBIE representation, $\vec{l}$, is a integer vector, $l_i \in [1, k]$, of length n. Each position, l_i, corresponds to an object in the data set $\vec{x}_i$ and defines the cluster the object belongs to. For example, the vector (111222233) describes a clustering solution for a data set of nine objects where there is a cluster containing three objects, a cluster containing four objects and a cluster containing two objects.

This representation is naturally redundant; for example (333111122) would generate a clustering solution that is identical to the previous example, even though the two solutions have completely redundant phenotypes. To eliminate this issue a renumbering procedure can be employed so that all permutations of a solution are treated as identical solutions. A suitable procedure is given in Algorithm 4.4.

where $\vec{l}$ is the vector to be renumbered, k is the number of clusters and n is the size of the dataset. $\vec{a}$ is a vector of size k that stores the order in which each cluster number is observed in the solution, in the solution (333111122) cluster 3 is the first observed cluster, cluster 1 is the second observed cluster and cluster 2 is the third observed cluster. This is populated by looping through the solution and using a counter, b, of the number of clusters we have not observed and added to $\vec{a}$ yet. $\vec{t}$ is then populated by looping through the observed clusters, this vector maps from the order the cluster was observed to the cluster number. Finally the algorithm loops through the solution again replacing the cluster numbers with the value in $\vec{t}$ which results in a renumbered vector.

This representation is advantageous as it can be used to represent a cluster of any shape. However, a disadvantage is that it may not scale well to large datasets

Algorithm 4.4 Renumbering Procedure

Require: $\vec{l} = (l_1, \ldots, l_n)$
  $\vec{a} \leftarrow (a_1, \ldots, a_k)$
  $\vec{t} \leftarrow (t_1, \ldots, t_k)$
  $b \leftarrow 1$
  for $i = 1$ to n **do**
    if $l_i \notin \vec{a}$ **then**
      $a_b \leftarrow l_i$
      $b \leftarrow b + 1$
    end if
  end for
  for $i = 1$ to k **do**
    $t_{a_i} \leftarrow i$
  end for
  for $i = 1$ to n **do**
    $l_i \leftarrow t_{l_i}$
  end for
  return $\vec{l}$

as each solution will have to be the same length as the number of objects in the dataset. Larger solutions will require more space for storage and require a greater amount of time for execution.

Krishna and Murty [90] describe an alternative matrix based binary encoding representation that is conceptually similar. The representation is a n by k sparse matrix of binary values where each row represents an object from the data set and each column represents a cluster, only one column from each row may be set to 1. This technique requires a pre-defined value of k and cannot be manipulated easily by common mutation and crossover operators, so in this work we use label based integer encoding. We did not use this alternative encoding in our main investigation and only present it here for comparison.

For our implementation, each $\vec{l}$ is randomly initialised. The value of k is selected from the range of integers $\left[2, \frac{n}{10}\right]$ and each position in the new $\vec{l}$ is set to a random integer value in the range $[1, k]$.

4.4.2.1 Mutation Operators

We experiment with mutation operators that manipulate multiple or single positions in an unguided or guided fashion giving rise to four mutation operators. For each invocation of a mutation operator that mutates multiple positions the probability of mutating multiple positions is first determined by a random value drawn from the range $[0, 1]$. For each position, further random values are drawn to determine if that position will be mutated. Where only one position is to be mutated a random position is selected from the representation. The mutation that is performed on each position is either guided or unguided, that is to say it has knowledge of the data set and present clustering solution or it does not.

Unguided Mutation To manipulate a position in an unguided fashion the value of the position is set to an integer drawn from $[1, k]$.

Guided Mutation Krishna [90] proposed a guided mutation operator where cluster memberships of objects are changed at random with a weighting towards clusters that are close to the object. We calculate the probability that the object in the i^{th} position is assigned to the g^{th} cluster as follows:

$$\Pr\{l_i = g\} = \frac{\delta_{max} - \delta(\vec{x}_i, \vec{c}_g)}{\sum_{j \in \mathcal{P}} (\delta_{max} - \delta(\vec{x}_i, \vec{c}_g))} \text{ where } \delta_{max} = \max \delta(\vec{x}_i, \vec{c}_g) \, \forall \mathcal{P}_g \in \mathcal{P} \quad (4.15)$$

where $\mathcal{P}$ is the clustering solution derived from the encoding, $\vec{l}$. From this we can then assign an object to a cluster in a biased fashion.

4.4.2.2 Crossover Operators

The encodings used for LBIE are fixed length encodings, that is, for a given data set all of the solutions are of the same length. Therefore, we can experiment using the standard one-point, two-point, three-point and uniform crossover operators previously defined in section 4.4.1.2.

4.4.3 Centroid Based Real Encoding

A Centroid Based Real Encoding (CBRE) is a set of cluster centroids that are not restricted to the values of the objects within the dataset, $\mathcal{R} = \{\vec{r}_1, \ldots, \vec{r}_r\}$. Each $\vec{r}_g$ represents a potential cluster centroid in the same space as $\mathcal{D}$, $\vec{r}_g = (r_{g1}, \ldots, r_{gd})$ and each $r_{gi} \in \mathbb{R}$.

To generate a clustering solution, $\mathcal{P}$, from an encoding, $\mathcal{R}$, the clustering solution is initialised so that the value of r, the number of centroids, is equal to k. Each object from the data set, $\vec{x}_i \in \mathcal{D}$, is assigned to the cluster, $\mathcal{P}_g$, whose centroid is closest to the object, $\min \delta(\vec{x}_i, \vec{r}_g) \forall \vec{r}_g \in \mathcal{R}$. Any cluster that is empty, $n_g = 0$, is removed so the solution is consistent with the rules defined in section 2.1.2. So, a $\mathcal{P}$ derived from a given $\mathcal{R}$ shall have $k \leq r$.

Each $\mathcal{R}$ is randomly initialised as follows. The value of r is an integer drawn randomly from the interval $[2, n]$, then r objects are randomly copied from $\mathcal{D}$ to become the initial set of cluster centroids in $\mathcal{R}$.

4.4.3.1 Mutation Operators

Previously defined mutation operators for CBRE are designed to operate on solutions with a fixed number of clusters. Because of this, they only change the values of the prototype centroids and do not vary r. We have defined some mutation and crossover operators that can vary r. They are as follows:

Swap Mutation The value of a cluster centroid in the solution, $\vec{r}_g$, is replaced with the value of an object randomly drawn from $\mathcal{D}$. This technique has been used by [43, 87].

Addition Mutation For each r_{gi} there is a chance that it may be mutated. Maulik and Bandyopadhyay [102] and Scheunders [131] use a probability of 0.05 to determine wether or not each r_{gi} should be mutated. To mutate an r_{gi} a value of

either one or negative one is added to that component with a 50% probability.

Bandyopadhyay and Maulik [6] used a similar technique where the value to be added takes into account the minimum and maximum values within the data set in the appropriate dimension to produce a scaled value to be added. The version used by Bandyopadhyay and Maulik used the value of a single objective to determine the value to be added and was not suitable for our multi-objective problems so we used the first implementation.

4.4.3.2 Crossover Operators

Several crossover operators exist within the literature that are suitable for a CBRE where the number of clusters has been pre-defined. However, in this work the number of clusters has not been pre-defined, so pre-existing techniques are not applicable or require modification.

Devising crossover operators for variable length solutions is a difficult task. Standard operators must be modified to take in to account variable lengths. Where the representation and operators allow solutions of variable length, sometimes solutions that are longer are favoured [18]. For clustering, this would favour solutions with a greater number of clusters.

The one point crossover operator described in section 4.4.1.2 is relatively easy to modify. It requires a cut off point to be chosen in each solution and the solutions then recombined appropriately. This is exemplified in figure 4.18. A one point crossover operator has been used with CBRE in several different implementations [6, 102, 131]. The variable length modification has been used in several genetic algorithms [23, 15, 49, 127].

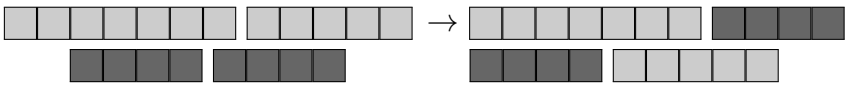


Figure 4.18: Variable Length One Point Crossover

A variation of the two point crossover operator was presented by Bezdek et al.

[11]. Brie and Morignot [15] experimented on a genetic planning problem and used a larger number of cut points in their crossover operator with poor results. In this work, therefore, we only use the one point crossover operator.

The uniform crossover operator may also be modified to work with solutions that are not of the same length [23, 92, 135]. For this, we can randomly select (with equal probability) if a centroid chromosome should now belong to the first or second new solution generated. As the order of the values in the solutions is unimportant in this representation the solutions that will be produced will still be valid. Speer el al. [135] used Uniform Crossover but had a fixed number of clusters. The crossover points are allowed to fall inside a cluster so it can be split into two clusters.

A further set of crossover operators are described by Frnti et al.[43]and Kivijrvi el al. [87]. We have produced slightly modified implementations for our experiments to allow them to be used with clustering solutions that are not the same size. The details of our implementations are as follows:

Centroid Distance Crossover The cluster centroids that are closest to the centroid of the data set are exchanged to form two new solutions. First we must order the cluster centroids by the distance between each cluster centroid and the centroid of the data set, $\vec{v}$. Then we exchange the $\frac{r}{2}$ cluster prototypes that are closest to $\vec{v}$ from encoding $\mathcal{R}$ with the $\frac{r'}{2}$ cluster prototypes that are closest to $\vec{v}$ from encoding $\mathcal{R}'$ resulting in two new sets of prototype clusters. In the original implementation r and r' are the same value and in our implementation they can differ.

Largest Partitions Crossover To perform crossover of two sets of centroid-based solutions, $\mathcal{R}$ and $\mathcal{R}'$, first we rank each cluster centroid by the number of data objects that are members of the associated cluster in the clustering generated from that solution. To generate the first new set of centroid-based clusters we remove the cluster centroid with the largest number of data objects from $\mathcal{R}$ and add it to the new solution. The data objects that were associated with this cluster centroid

are excluded from our calculations for the rest of this process. The cluster centroids for both $\mathcal{R}$ and $\mathcal{R}'$ are re-ranked taking into account the excluded data objects. A cluster prototype is then removed from $\mathcal{R}'$ in the same manner, this process continues until r or r' reach 0. To produce a second new solution $\mathcal{R}$ and $\mathcal{R}'$ are swapped and the whole process is repeated. In the original implementation r and r' are the same and here they may differ.

Multipoint Pairwise Crossover Each cluster centroid, $\vec{r}_g$, in a solution, $\mathcal{R}$, is associated with its nearest neighbour, $\vec{r}_h$, from another solution, $\mathcal{R}'$, that it is being crossed with, $\min \delta(\vec{r}_g, \vec{r}_h)$ where $\vec{r}_g \in \mathcal{R}$ and $\vec{r}_h \in \mathcal{R}'$. Cluster centroids that have been associated with each other are not placed in the same new solution. This prevents cluster centroids that are close together being in the same new solution. Cluster centroids are then randomly distributed to two new solutions while maintaining this rule.

4.5 An Overview of MOEAs for Clustering

To date there has been some work in the literature on applying MOEA to the clustering problem. A number of algorithms have been implemented and developed but so far there has been no developments that establish the best implementation of an MOEA for clustering.

Early attempts at applying genetic algorithms showed promise [11, 115]. The results of the genetic algorithm were better than those generated by the k-means algorithm on a single data set. A later attempt found similar results when applied to larger data sets but was not practical for real world usage as the algorithm took too long to execute [43]. Other studies performed in the same year used evolutionary algorithms as an initialisation procedure to determine an initial set of prototype centroids for classical clustering algorithms to cluster data [92] and to cluster regions of images [131]. Later algorithms [90, 100, 133] used hybrid strategies that combined

k-means with genetic algorithms to aid the search and found some promising results.

An early comparative study [23] focussed on small data sets and a limited number of representations. The findings indicated that the performance of these algorithms was not good as they took a large period of time to execute. It was shown that the choice of representation was important and the objective function was a main contributing factor to performance of the algorithm. It appeared that a representation based on labelling the cluster of individual solutions worked well. This representation has also been used in other algorithms since this study was published [96]. We described this representation in Section 4.4.2.

A number of experimental studies and literature reviews have focussed on applying MOEAs to the crisp clustering problem[63, 70, 123, 110]. Studies focussed on applying MOEA to the fuzzy clustering problem with some success [105].

In 2004 Handl designed and implemented an interesting algorithm called Voronoi Inisalized Evolutionary Nearest-Neighbour Algorithm (VIENNA) [56] that used an LBIE encoding in conjunction with the Pareto Evolutionary Strength Algorithm (PESA-II) algorithm [27, 26]. The algorithm used objectives based upon Connectivity that we previously described in Section 2.3.7. It did not have a crossover operator but used a mutation operator that moved several objects from one cluster to another cluster.

Handl later improved VIENNA with MOCK (Multi-Objective Clustering with automatic determination of the number of clusters) [57, 58, 59, 62, 61]. MOCK improved on VIENNA by introducing a novel adjacency graph based representation of a clustering solution that works well with Connectivity and also introduced the uniform crossover operator. MOCK uses the Gap statistic [141] to determine the solution that occurs at a 'knee' in the Pareto front as the final clustering solution. We have not implemented MOCK here as it has a very specific implementation of its representation. In 2005 Handl described MOCK-am [60], this version of the algorithm was based upon MOCK but used an MBBE representation instead of the graph based representation, while this implementation performed faster their later

experiments continued to use the graph based representation as it produced better clustering solutions.

MOCK has also inspired a number of other clustering algorithms. The graph based representation is now being used for applications within social networking to identify groups of users [82, 37]. A number of other algorithms have also been devised that either extend or slightly modify MOCK. Chen introduced a variation of MOCK called MOEAD [20] that used NSGA-II instead of PESA-II and used the CBRE to represent the clustering solutions. Shirakawa [134] introduced another variant of MOCK that specialised in identifying regions in images, this was based on SPEA2 and introduced a modified version of Connectivity called Edge designed to identify the boundaries between regions of colour. Qian implemented MECEA [119] which is the same as MOCK with the exception that it uses a novel technique for merging the Pareto solutions together to find edges in images.

In 2000 Maulik and Bandyopadhyay [102] introduced a genetic algorithm for clustering that used a version of Centroid Based Real Encoding we described in Section 4.4.3 to represent clusters, this implementation used a fixed number of clusters. They crossed over their solutions used the single point crossover and mutated individual values of each centroid by multiplying them with positive or negative values randomly drawn from a uniform distribution in the range [0,1]. Later in 2007 they expanded upon this with a novel Multi-objective algorithm for clustering, MOGA [5]. MOGA uses NSGA-II and is designed for detecting regions in satellite imagery. This algorithm differs from some of the other algorithms as it identifies fuzzy clusters. The algorithm used two objectives XB [149] and J_m. XB is a cluster quality measure for fuzzy clustering that uses the ratio between the total variation and minimum separation of the clusters which is similar to the measures defined for crisp clustering later by Halkidi that we defined in Section 2.3.4. The clustering solution with the highest value of the $\mathcal{I}$ index [103] was chosen as the final clustering solution. They later expanded upon this algorithm with MOGA-SVM [111], this iteration of the algorithm improved upon the previous version of the algorithm by using a novel

technique where an SVM [143] is combined with the results of NSGA-II to select the final clustering solution. MOGA-SVM has also been applied to bioinformatics problems [104]. The most recent version of this algorithm MOVGA [112, 105] now allows for the number of clusters to be varied. MOVGA uses updated versions of the objective functions and reverts to using the $\mathcal{I}$ index to select the final clustering solution.

4.6 Summary

In this chapter we have introduced the concept of multi-criteria decision making, dominance and the Pareto front. We then reviewed Genetic Algorithms and then Multi-Objective Evolutionary Algorithms. We also introduced a range of techniques that allow us to assess the quality of Pareto fronts: the volume of the dominated space, coverage, generational distance, inverted generational distance and a measure of entropy.

We then detailed some of the other work that has used MOEA to attempt to solve clustering problems. In particular we detailed three possible representations of the clustering problem with a range of mutation and crossover operators that work with these representations. Later we will perform experimental evaluations of various combinations of these operators and representations.

Chapter 5

A Novel Multi-Objective Evolutionary Clustering Algorithm

5.1 Introduction

Multi-Objective Evolutionary Algorithms (MOEAs) have some good potential for cluster analysis. Clustering algorithms optimise specific measures of cluster quality, such as compactness and separation. Many clustering algorithms have been defined in the literature [76] and they generally aim to optimise a single objective. Unfortunately, defining what constitutes a good clustering solution remains a difficult problem and no individual measure of clustering quality has emerged as the overall winner. In this context, MOEAs give us the opportunity to optimise several of these quality measures at once. Furthermore, they will deliver a number of clustering solutions representing trade-offs between the different quality measures.

Previous research into evolutionary algorithms for clustering has been conducted by Cole [23] who explored various techniques for representing clustering solutions and various objectives to be optimised. Handl and Knowles [56] and Chen and Wang

[20] have developed their own multi-objective clustering algorithms that operate with new cluster quality measures. These previous works have used different methods such as a graph based technique to assign objects to clusters. Here we will use a new centroid-based technique to establish cluster membership.

Broadly speaking, a MOEA consists of the following several components: a selection method; a strategy to manage the Pareto front; a fitness function; a crossover operator(s); a mutation operator(s). This research re-uses the selection method and the strategy to manage the Pareto front of NSGA-II, but the other three components (fitness, crossover and mutation operators) are new or new variations of existing operators introduced in this thesis.

In this chapter, we propose a new MOCA and evaluate its performance against the well known k-means algorithm, as an initial benchmark. In section 5.2 we propose a new Multi-Objective Cluster Algorithm; in section 5.3, we propose a method of assessing its quality; finally, we report our results in section 5.4 and give our conclusions in section 5.5.

5.2 The Proposed Multi-Objective Clustering Algorithm

Previously in Chapter 4 we reviewed Multi-Objective Evolutionary Algorithms. In Section 4.4 we reviewed a number of existing techniques for solving the clustering problem using MOEAs. Here we will propose a novel Multi-Objective Clustering Algorithm (MOCA) to solve the clustering problem.

MOEA is an evolutionary algorithm. More precisely, we have chosen to use NSGA-II [29], one of the best known MOEAs, as the underlying implementation for our MOCA. NSGA-II introduced techniques for producing a set of solutions that provide good coverage and convergence. To adapt NSGA-II for clustering we need to provide the following:

- an appropriate representation of a clustering solution,
- a set of evaluation functions for a clustering solution,
- an initialisation operator that creates valid solutions,
- a mutation operator,
- a crossover operator.

Additional parameters are used to define a minimum and maximum number of clusters allowed, $k_{\min}$ and $k_{\max}$ respectively. Sensible values are $k_{\min} = 2$ and $k_{\max} = n/2$ but the decision maker may use any values as long as $1 \leq k_{\min} \leq k_{\max} \leq n$.

5.2.1 Solutions Representation & Initialisation

Previously in Section 4.4.3 we reviewed Centroid Based Real Encoding. Here we describe an implementation of CBRE that we will use in our MOCA.

The solution representation consists of two sets of cluster prototypes: the set of selected prototypes $\mathcal{A} = \{\mathcal{A}(1), \ldots, \mathcal{A}(a)\}$ and the set of potential prototypes, not in use, $\mathcal{B} = \{\mathcal{B}(1), \ldots, \mathcal{B}(b)\}$. Therefore, each cluster $\mathcal{P}_g$ in the represented clustering solution is associated with a cluster prototype, $\mathcal{A}(g)$, from set $\mathcal{A}$.

To generate initial valid clustering solutions, the values of the cluster prototypes are drawn from $\mathcal{D}$, hence the initial prototypes are medoids. The lengths of $\mathcal{A}$ and $\mathcal{B}$ are required to create the initial solutions. The value of a is set to $k_{\min} + (k_{\max} - k_{\min})/2$ and b is set to $n - a$. Each object from the data set is then randomly added to either $\mathcal{A}$ or $\mathcal{B}$.

Once a set of selected prototypes has been defined, the distance between every object in the data set, $x \in \mathcal{D}$, and every cluster prototype, $\mathcal{A}(g) \in \mathcal{A}$, is calculated. x is added to the cluster that minimises $\delta(x, \mathcal{A}(g))$ to generate a clustering solution.

5.2.2 Mutation Operator

The novel mutation operator that we define encompasses three techniques for altering a solution. This mutation operator has not been defined in any previous work. These sub-operators use knowledge of clustering to encourage better clustering solutions so they differ from the more general mutation operators that are not problem specific. The techniques are defined as follows:

5.2.2.1 Decrease

The decrease sub-operator removes a cluster prototype from the solution that leads to a reduction in the number of clusters in the solution. A cluster prototype is moved from $\mathcal{A}$ to $\mathcal{B}$ decreasing the number of elements. To determine the prototype to remove, we first identify the nearest neighbour prototype, $\mathcal{A}(g)_{\mathcal{A}NN}$, of every cluster prototype $\mathcal{A}(g)$ in $\mathcal{A}$. We then move the prototype that minimises $\delta(\mathcal{A}(g), \mathcal{A}(g)_{\mathcal{A}NN})$, $\forall \mathcal{A}(g) \in \mathcal{A}$. The objects associated with the removed prototype, $\mathcal{A}(g)$, are likely to be associated with $\mathcal{A}(g)_{\mathcal{A}NN}$ after the removal.

5.2.2.2 Increase

The increase sub-operator adds a cluster prototype from the solution that leads to an increase in the number of clusters in the solution. A cluster prototype is moved from $\mathcal{B}$ to $\mathcal{A}$, increasing the number of clusters. The prototype drawn from $\mathcal{B}$ is the cluster prototype that is furthest away from any cluster prototype in $\mathcal{A}$. That is, for each cluster prototype, $\mathcal{A}(g) \in \mathcal{A}$, its furthest neighbour in $\mathcal{B}$, $\mathcal{A}(g)_{\mathcal{B}FN}$, is computed. The cluster prototype in $\mathcal{B}$ that maximises $\delta(\mathcal{A}(g), \mathcal{A}(g)_{\mathcal{B}FN})$ is moved to $\mathcal{A}$. This ensures that new cluster prototypes are not near pre-existing cluster prototypes so they should produce new and interesting clusters.

5.2.2.3 Recompute Prototypes

The values of the cluster prototypes are recomputed as the values of the centroids of the clusters with which they are associated. For example, the value of a cluster prototype, $\mathcal{A}(g)$, will be replaced with the value of $\mathcal{V}_g$ where $\mathcal{V}_g$ is the centroid of $\mathcal{P}_g$. This process is similar to a single iteration of the clustering algorithm, k-means.

5.2.2.4 Sub-Operator Selection

The alterations proposed are applied with a probability. We used a 50% probability of decreasing the number of prototypes in $\mathcal{A}$, a 25% probability of increasing the number of prototypes in $\mathcal{A}$, or a 25% probability of recomputing the cluster prototypes.

5.2.3 Crossover Operator

Our novel crossover operator works by exchanging clusters between two clustering solutions. This crossover operator has not been previously defined.

Given two clustering solutions, we first identify the clustering solution with the largest number of clusters, $\mathcal{P}^l$, and the clustering solution with the smallest number of clusters, $\mathcal{P}^s$. If the number of clusters is equal then this tie is broken at random.

For the smaller solution, $\mathcal{P}^s$, we then identify the largest cluster, $\mathcal{P}^s_g \in \mathcal{P}^s$, and its prototype, $\mathcal{A}(g)^s$.

For each object $x \in \mathcal{P}^s_g$ we determine the cluster in $\mathcal{P}^l$ in which it lies and the associated prototype in the larger solution. Let $\left\{\mathcal{A}(1)^l, \ldots, \mathcal{A}(o)^l\right\}$ denote this set of prototypes. The crossover operation then exchanges prototype $\mathcal{A}(g)^s$ in the small solution with all the prototypes $\left\{\mathcal{A}(1)^l, \ldots, \mathcal{A}(o)^l\right\}$ associated with it in the larger solution.

We must ensure that the size of the new $\mathcal{A}$ and $\mathcal{B}$ still sum to n after they have been generated. To ensure that $a+b=n$, we randomly remove the required number

of prototypes from the set $\mathcal{B}$ in the smaller solution and add them to the set $\mathcal{B}$ of the larger solution.

The resulting crossover is therefore an exchange of one cluster in one solution with the corresponding smaller clusters in the other solution.

5.2.4 Fitness Measures for MOCA

Previously in Chapter 2 we defined a number of Cluster Quality Measures. These are measures that assess and rank the quality of clustering solutions. Some of the CQMs measured different attributes of clustering solutions such as the density of clusters, separation of clusters or identified continuous shapes within the clustering solution.

In Chapter 3 we described how some CQMs may be better than others as part of an optimisation process as they may be more sensitive to changes in quality. Using those findings we have identified several CQMs that would be useful as fitness functions for our MOCA. These are as follows:

5.2.4.1 Homogeneity Based Fitness Measure

A common measure of the quality of a clustering solution is the density of the clusters. A clustering solution is homogeneous if the distances between the objects in each cluster are low. Previously in Chapter 3 we found that the Overall Deviation (Dev) was highly correlated with the degradation of clustering solutions so we will use this operator here. Overall Deviation was introduced in Section 2.3.7 as equivalent to the value of $\mathcal{P}^W$ that was previously defined in section 2.1.3. $\mathcal{P}^W$ can give us an indication of the homogeneity of a clustering solution but it does not take into account the value of k.

Here we will use a variation of the previous measure, the Average Within Group Sum of Squares. This also measures the homogeneity property of clustering solutions but takes the value of k into account. This can be measured by taking the average

of the distance between each object in the cluster and its centroid. The density of each cluster can then be summed to give the Average Within Group Sum of Squares for a given clustering solution:

$$\text{awgss}\left(\mathcal{P}\right) = \sum_{g=1}^{k} \frac{\sum_{i=1}^{|\mathcal{P}_g|} \delta\left(\mathcal{P}_g\left(i\right), \mathcal{V}_g\right)^2}{|\mathcal{P}_g|} \tag{5.1}$$

Values of this measure are high when the clusters are not very dense so this measure should be minimised.

This measure was not included in our previous experiment in Chapter 3. This CQM is measuring the density property that was shown to find good results in the previous experiment. We feel that taking the number of clusters into account in the measure will be beneficial when we come to compare clustering solutions that contain different numbers of clusters.

5.2.4.2 Separation Based Fitness Measure

Another method of assessing the quality of a clustering solution is the separation of clusters. A clustering solution is considered good if the clusters are well separated. Previously in section 2.1.3 we defined between-cluster variation, $\mathcal{P}^B$, which gives us an indication of how well separated a clusterings solution is. $\mathcal{P}^B$ is the opposite of $\mathcal{P}^W$, which we found was highly correlated with the degradation of clustering solutions in Chapter 3. By using two objectives that work in opposite ways we hope to generate a wide range of clustering solutions. $\mathcal{P}^B$ does not allow us to compare clustering solutions with a varied value of k so here we define the Average Between Group Sum of Squares.

The Average Between Group Sum of Squares of a clustering solution, $\text{abgss}\left(\mathcal{P}\right)$, is the average distance between the centroids of the clusters and the centroid of the data set:

$$\text{abgss}(\mathcal{P}) = \frac{\sum_{g=1}^{k} |\mathcal{P}_g| \, \delta(\mathcal{V}_g, \mathcal{V})^2}{k} \tag{5.2}$$

A low value of $\text{abgss}(\mathcal{P})$ would indicate that all of the cluster centroids are near the centroid of the data set and therefore also near each other, so the value of this measure should be maximised.

This measure was chosen because it behaves in the opposite way to the Average Within Group Sum of Squares that we introduced previously. This measure was also not in the previous experiment but is measuring the separation property of clustering solutions that was shown to be useful in the previous experiment.

5.2.4.3 Connectivity Based Fitness Measure

The concept of connectivity was introduced previously in section 2.3.7. We propose using the definition of connectivity introduced by Handl and Knowles [62]. Previously in Chapter 3 we discovered that the performance of Handl and Knowles' measure appeared to be identical to the slightly simpler definition by Chen and Wang [20]. We feel that there is a possibility that the definition given by Handl and Knowles may be beneficial for cases where very small clusters are formed as the penalties are scaled so clusters with less than l objects can exist whereas the version given by Chen and Wang used penalties of 1 and will not allow clusters with less than l objects.

The version of Connectivity we use as a fitness measure is as follows. Connectivity calculates the sum of the values of a penalty function for each object in the data set and its l nearest neighbours. A penalty for an object, x, and its m^{th} nearest neighbour, x_{mNN}, is 0 if they are contained in the same cluster and $\frac{1}{m}$ if they are not members of the same cluster. The quality measure we use as an objective is defined as follows:

$$\text{connectivity}\left(\mathcal{P}\right) = \sum_{i=1}^{n}\sum_{m=1}^{l}\text{penalty}\left(\mathcal{D}\left(i\right),\mathcal{D}\left(i\right)_m\right) \tag{5.3}$$

$$\text{where} \qquad \text{penalty}\left(x_{mNN}\right) = \begin{cases} \frac{1}{m} & \text{if } \nexists\, \mathcal{P}_g : x \in \mathcal{P}_g \wedge x_{mNN} \in \mathcal{P}_g, \\ 0 & \text{otherwise.} \end{cases} \tag{5.4}$$

5.2.5 Overview

Our algorithm uses the NSGA-II algorithm, described previously in Section 4.2.2.4, as its main framework. Our contributions are: the initialisation operator defined in Section 5.2.1, the mutation operator defined in Section 5.2.2 and the crossover operator defined in Section 5.2.3 which are all novel. We also provide fitness functions, defined in Section 5.2.4, that we have selected from the literature and implemented so they are compatible with the representation of a clustering solution defined. An overview of the MOCA components and the main steps from NSGA-II is given in Algorithm 5.1.

5.3 Preliminary Experimental Evaluation of MOCA

To initially test our MOCA we performed a comparison using k-means to cluster a large number of prefabricated data sets where a desired clustering solution exists. This initial comparison should enable us to test if it is producing correct clustering solutions and performing at least inline with the benchmark clustering algorithm. In later chapters we perform more complex experiments against other Multi-Objective Evolutionary Algorithms to determine if our MOCA is more efficient than other MOEA implementations for clustering.

First, we constructed a series of synthetic data sets; then we defined an experimental methodology for comparing the algorithm's performance against k-means; finally we report our results.

Algorithm 5.1 MOCA

- Initialise the population, $\mathcal{S}$.
- Randomly create solutions by drawing objects from $\mathcal{D}$ to form sets of medoids and insert into $\mathcal{S}_1$ to serve as the start population.
- $g = 1$.
- while $g <$ number of generations.
  - $\forall \vec{s} \in \mathcal{S}_g$
    - Calculate awgss $(\mathcal{P})$.
    - Calculate abgss $(\mathcal{P})$.
    - Calculate connectivity $(\mathcal{P})$.
  - Calculate dominance depth of solutions in $\mathcal{S}$ as in Algorithm 4.2.
  - Calculate crowding distance of solutions in $\mathcal{S}$ as in Algorithm 4.3.
  - Select solutions to mutate using binary tournament selection with $\prec_n$.
  - Mutate each selected solution to with a randomly selected a mutation sub-operator:
    - Decrease the number of clusters in the solution.
    - Increase the number of clusters in the solution.
    - Recompute the cluster prototypes.
  - Select solutions to crossover using binary tournament selection with $\prec_n$.
  - Crossover the selected solutions by exchanging cluster prototypes.
  - Add the mutated and crossed over solutions into the population.
  - Sort $\mathcal{S}$ with $\prec_n$.
  - Add the fittest solution from $\mathcal{S}_g$ to $\mathcal{S}_{g+1}$ until it is full.
  - $g = g + 1$.
- Return the final population.

5.3.1 Construction of Synthetic Data Sets

In Section 3.2 we described a method for generating a synthetic data sets based upon the work of Milligan and Cooper [107, 108]. The proposed method can be used to generate data sets where the following factors are varied: the number of naturally occurring clusters, the number of dimensions, the distribution of the membership of objects to clusters and the proportion of outliers that exist within the data set.

We varied each of the four factors to produce different data designs. A data set was generated from each design three times leading to twenty thousand and seven data sets for this experiment. Each data set contained five hundred objects. These data sets were newly generated and are not identical to those in Chapter 3.

5.3.2 Experimental Method

We set the population size for our version of NSGA-II to 100; the number of generations was set to 1,000; the mutation probability and the crossover probability were both set to 0.5. These choices were made based upon preliminary work where we experimented with mutation and crossover probabilities in the range of $[0.1 : 0.9]$ in increments of 0.1, the population size was varied in the range $[50 : 200]$ in increments of 10 and the number of generations was in the range of $[100 : 2000]$ in increments of 100.

Our MOCA was executed on each of the previously described synthetic data sets with these parameters. The result of this is a set of clustering solutions. We test each solution generated against the optimal clustering solution using the Rand Statistic, $\mathcal{R}$, previously defined in Section 2.4. We extract the highest, lowest and mean average values of $\mathcal{R}$ recorded for each Pareto set of solutions returned by an execution of MOCA. The value of k associated with the solutions that generated the minimum and maximum values of $\mathcal{R}$ and the average value of $\mathcal{R}$ are also reported.

We also make a comparison of performance against the algorithm k-means. For each synthetic data set, we execute the algorithm k-means for varying values of k

ranging from 2 to 40 in increments of 1. We report the highest value of $\mathcal{R}$ recorded for each pool of solutions associated with a data set and the associated k value.

5.3.3 Comparison to DBSCAN

We will also compare MOCA against another clustering algorithm, we have chosen to compare MOCA against k-means and DBSCAN [35], a clustering algorithm that is based upon density, discussed in Section 2.2.3.

We will draw a subset of the data sets described in Sections 3.2 and 5.3.1. The number of clusters in the data sets will be between two and twenty in increments of two. The number of dimensions will be between two and ten in increments of two. All three data set designs (df) are used; an even distribution of clusters is denoted as "a", a cluster consisting of 10% of the objects and the rest as evenly distributed as possible is denoted "b"; and a cluster consisting of 60% of the objects and the rest as evenly distributed as possible is denoted "c". The proportion of outliers is either 0% or 40% which is denoted as "a" and "b" respectively.

For each data set we execute the DBSCAN algorithm. This algorithm returns a single clustering solution without the need for a pre-determined value of k to be provided. We will then calculate the value of $\mathcal{R}$ of this solution compared to the intended clustering solution. We also report the value of k.

k-means is run in the same fashion as we described in Section 5.3.2. For each synthetic data set, we run the k-means algorithm with values of k from 2 to 40. Again we report the highest value of $\mathcal{R}$ for each pool of k-means solutions and the value of k associated with it.

MOCA is performed in almost exactly the same way as described in Section 5.3.2. The number of generations has been reduced to 100. Again, from each set of solutions returned by MOCA we extract the solutions with the highest and lowest values of $\mathcal{R}$ and report the value of k for this solution. We also report the mean and S.D. of k and $\mathcal{R}$ for each set of solutions returned by MOCA.

5.4 Preliminary Results with Synthetic Datasets

The results of our experiments are reported in table 5.1. The table contains columns summarising the best, worst and average solutions found by MOCA, as well as the best solutions found by k-means. When looking at the best solution reported by MOCA for each dataset (largest $\mathcal{R}$, reported as *MOCA Best* column in table 5.1) the optimal clustering solution, equivalent to $\mathcal{R} = 1$, was contained in the pool of solutions generated by MOCA at least once for 18.18% of the data sets. However, the optimal solution was drawn from the pool of solutions generated by k-means in only 4.09% of cases (k-means Best column in table 5.1). Furthermore, when looking at solutions close to the optimal solution ($\mathcal{R} \geq 0.9$) they were found by k-means in 21.37% of cases but by MOCA in 100% of cases.

We also extracted the worst solution from each pool of solutions generated by MOCA (the minimum value of $\mathcal{R}$, reported as *MOCA Worst*) and found that in 1.07% of cases this value was equal to 1. This shows that in 1.07% of cases the worst solution offered by MOCA was the optimal solution.

We extracted the average solutions reported by MOCA (average $\mathcal{R}$, reported as *MOCA Average*). In 30.11% of cases the average solution was close to the optimal solution. This shows that on average MOCA finds near optimal solutions more often than k-means.

We average our results and found that the average value of $\mathcal{R}$ from the best solutions generated by MOCA was 0.98. This was higher than the average equivalent generated by k-means which was 0.88. Also we calculated the average $\mathcal{R}$ from the worst and average solutions generated by MOCA. They were 0.56 and 0.89 respectively. Hence the average solution generated by MOCA is close to the best solution generated by k-means.

We also extracted the value of k associated with the solutions with the highest and lowest values of $\mathcal{R}$ generated by MOCA. Similarly, we extracted k associated with the solutions generated by k-means. We did not extract an average value of k

Table 5.1: Summary of Results

	MOCA			
	Best	Average	Worst	k-means Best
$\mathcal{R} \geq 0.9$ total	100%	30.11%	3.79%	21.37%
$\mathcal{R} = 1$ total	18.18%	1.07%	1.07%	4.09%
Max $\mathcal{R}$	1	1	1	1
Min $\mathcal{R}$	0.92	0.38	0.11	0.53
Average $\mathcal{R}$	0.98	0.89	0.56	0.88
StDev $\mathcal{R}$	0.02	0.03	0.11	0.05
Correct value of k	30.54%		1.41%	8.34%
Average difference of k	5.78	20.44	18.9	-2.42
StDev of difference of k	6.50	11.32	11.47	11.67

for MOCA as the mean value of k drawn from a set of solutions is unlikely to be an integer and is therefore never the correct value of k. We found that MOCA found the correct value of k in 30.54% of cases whereas k-means had the correct value in 8.34% of cases. The best solution drawn from the solutions generated by MOCA had 5.78 extra clusters on average and the worst solution had 18.9 extra clusters on average, whereas the best solution generated by k-means had 2.42 less clusters than the optimal number of clusters on average.

In Table 5.1 the number of runs of MOCA that have produced sets of solutions where $\mathcal{R} = 1$ for the average and worst cases are both 1.07%. For the worst solution in the set of solutions generated by MOCA to have a value of $\mathcal{R} = 1$ all of the solutions generated by that run of MOCA must have an $\mathcal{R}$ of 1. For the average $\mathcal{R}$ of a set of solutions to be 1 all the clustering solutions must have an $\mathcal{R}$ of 1 also. Therefore the runs of MOCA where the average value of $\mathcal{R} = 1$ and the runs where the worst value of $\mathcal{R} = 1$ are the same runs of MOCA.

In table 5.2 we report a subset of our comparison between MOCA and DBSCAN. The complete results are reported in Appendix B in Tables B.1 and B.2.

In table 5.2 we can see that DBSCAN has identified the correct number of clusters for all the synthetic data sets. This is true for all of the synthetic data sets reported in Tables B.1 and B.2. k-means successfully identifies the correct number of clusters

Table 5.2: Comparison of k-means, DBSCAN and MOCA on selected synthetic data sets where the intended value of k is 2 or 6 and there are no outliers.

Dataset			DBSCAN		k-means		MOCA Best		MOCA Worst		MOCA Average k		MOCA Mean $\mathcal{R}$	
k	d	df	k	$\mathcal{R}$	k	$\mathcal{R}$	k	$\mathcal{R}$	k	$\mathcal{R}$	Mean k	S.D.	Mean $\mathcal{R}$	S.D.
2	2	a	2	0.50100	2	1	2	1	19	0.57090	10.53623	1.01892	0.72273	0.01828
2	2	b	2	0.29946	2	1	2	1	17	0.36900	10.26471	0.81677	0.56809	0.02414
2	2	c	2	0.06148	2	1	2	1	31	0.12160	14.58667	2.01072	0.35144	0.02654
2	4	a	2	0.50100	2	1	2	1	3	0.90270	2.50000	0.35355	0.95135	0.03440
2	4	b	2	0.29946	2	1	2	1	2	1	2	0	1	0
2	4	c	2	0.06148	2	1	2	1	9	0.99910	5.07692	1.08807	0.99938	0.00008
2	6	a	2	0.50100	2	1	2	1	2	1	2	0	1	0
2	6	b	2	0.29946	2	1	2	1	2	1	2	0	1	0
2	6	c	2	0.06148	2	1	2	1	9	0.99910	5.25000	1.08253	0.99939	0.00008
2	8	a	2	0.50100	2	1	2	1	3	0.99800	2.50000	0.35355	0.99900	0.00071
2	8	b	2	0.29946	2	1	2	1	2	1	2	0	1	0
2	8	c	2	0.06148	2	1	2	1	2	1	2	0	1	0
2	10	a	2	0.50100	2	1	2	1	2	1	2	0	1	0
2	10	b	2	0.29946	2	1	2	1	2	1	2	0	1	0
2	10	c	2	0.06148	2	1	2	1	2	1	2	0	1	0
6	2	a	6	0.83500	6	0.87174	6	1	2	0.66600	13.59677	1.32121	0.93538	0.00575
6	2	b	6	0.83259	3	0.75106	6	1	2	0.58450	13.70149	1.13599	0.92616	0.00365
6	2	c	6	0.80926	5	0.89634	6	1	2	0.66870	11.35849	1.04964	0.93031	0.00337
6	4	a	6	0.83500	4	0.88956	6	1	2	0.60940	4.37500	0.92808	0.86669	0.04505
6	4	b	6	0.83259	4	0.85395	6	1	2	0.66700	6.88889	1.70370	0.92386	0.02155
6	4	c	6	0.80926	21	0.87966	6	1	2	0.52970	4.44444	0.85185	0.88350	0.03877
6	6	a	6	0.83500	39	0.79297	6	1	2	0.61210	3.85714	0.70193	0.84166	0.08676
6	6	b	6	0.83259	26	0.97076	6	1	2	0.62490	3.85714	0.70193	0.83661	0.08002
6	6	c	6	0.80926	37	0.88581	6	1	2	0.52970	3.83333	0.74846	0.83942	0.12644
6	8	a	6	0.83500	3	0.66867	6	1	2	0.61210	3.71429	0.64794	0.81809	0.07786
6	8	b	6	0.83259	20	0.86673	6	1	2	0.66700	4	0.89443	0.87082	0.09115
6	8	c	6	0.80926	20	0.87944	6	1	2	0.66870	3.66667	0.68041	0.84720	0.06977
6	10	a	6	0.83500	5	0.94478	6	1	2	0.66600	3.83333	0.74846	0.85152	0.07574
6	10	b	6	0.83259	36	0.90824	6	1	2	0.66700	4	0.89443	0.87082	0.09115
6	10	c	6	0.80926	11	0.96077	6	1	2	0.66810	3.57143	0.59394	0.83480	0.06301

when there are two clusters. For other numbers of clusters k-means is unsuccessful more often than not and tends towards high values of k. In all cases the worst solution (by $\mathcal{R}$) has two clusters and the best solution has broadly the correct number of clusters. The average number of clusters in the solutions maintained by MOCA is slightly lower than the intended number of clusters.

We can also see that DBSCAN cannot find high quality clustering solutions as we vary the data set design. When one cluster is significantly larger than the others DBSCAN performs worse. Varying the data set design also affects the performance of k-means. As one cluster becomes larger than all of the other clusters the value of k favoured by k-means appears to increase. Varying the design of the data set does not appear to have an effect on MOCA.

It appears that according to the value of $\mathcal{R}$ for the best solution, MOCA produces better solutions than the other techniques. The worst solution found by MOCA is of less quality than the solutions found by the other techniques. On average the solutions found by MOCA are of higher quality than those found by k-means and DBSCAN.

Varying the number of dimensions does not appear have to have an effect on any of the algorithms. As all of the algorithms are using the same distance metric it appears that any changes caused by the change in the number of dimensions affects all of the algorithms equally.

We can see in Tables B.1 and B.2 that k-means shows lower performance for data sets with a higher number of outliers. k-means is by definition susceptible to lower performance where there are a higher number of outliers. The other algorithms do not appears to have been affected by this factor.

5.5 Conclusions & Summary

In this chapter we have proposed a novel Multi-Objective Clustering Algorithm, which has been published independently in [85]. In the algorithm, we proposed a representation based on a set of prototypes. We also defined some guided operators including initialisation, mutation and crossover. We then proposed an experimental methodology to test the validity of the algorithm, using synthetic datasets where the optimal clustering solution is known by design.

We have shown that MOCA can generate a pool of clustering solutions that is more likely to contain the optimal clustering solution than the pool of solutions generated by k-means. The solutions in this pool are generally more similar to the optimal solution than the solutions generated by k-means. We have seen that it is more effective to use MOCA to find the optimal number of clusters than using k-means in a trial and error fashion. We therefore have proved the validity of the algorithm.

We also compared MOCA against another well known clustering algorithm, DBSCAN, and found that MOCA generally produces results of higher quality. We also have shown that MOCA is not been effected the design of the data set or the number of clusters.

There is still scope to investigate further configurations of MOCA in comparison

to more clustering algorithms on a data sets that can varied in a larger number of ways. Also, we have focussed on evaluating MOCA only in terms of the clustering solutions that it produces and not the Pareto fronts that it produces. Chapters 6 and 7 report studies that follow on from our initial findings with MOCA by investigating the effect of different configurations of the MOCA by also assessing of the Pareto fronts produced.

Future versions of the algorithm could modify the mutation operator so that the recompute sub-operator is executed after the increase or decrease operators have been used. This may lead to improved clustering solutions as the recompute sub-operation will be performed more often. It is also possible that other mutation and crossover operators may be more effective. We must now compare this algorithm to other MOEA algorithms and other configurations of operators.

Chapter 6

Experimental Comparison of Clustering Representations

6.1 Introduction

In Chapter 4 we introduced Multi-Objective Evolutionary Algorithms, and in Section 4.4 we discussed a number of representations that can be used to represent a clustering solution, and associated mutation and crossover operators. In summary,

- Section 4.4.1 introduced Medoid Based Binary Encoding (MBBE). This is a representation that uses a set of objects drawn from the data set being clustered as medoids to form clustering solutions.
- Section 4.4.2 introduced Label Based Integer Encoding (LBIE). This is a representation that assigns each member of the data set to a cluster within the clustering solution.
- Section 4.4.3 introduced Centroid Based Real Encoding (CBRE). This representation is similar to MBBE but uses centroids instead of medoids. Each centroid is encoded as a point within the space occupied by the data set and clustering solutions are generated from this.

In Chapter 4 we did not experiment with the different representations and operators available to use. In Chapter 5 we introduced a Multi-Objective Clustering Algorithm that used CBRE as the representation and novel mutation and crossover operators. We compared this novel algorithm against the standard k-means algorithm. We must now compare this algorithm to other configurations of Multi-Objective Algorithms for clustering. Here we will find how the configuration we proposed compares against other possible configurations that we reviewed in Chapter 4.

In this chapter, we present a framework for experimental evaluation of representations. First we define the data sets. Then we define the core MOEA components, that is the representation and operators that will be varied during the experiment. We then discuss how to evaluate and compare the quality of Pareto fronts. Finally, we report our results.

6.2 Experimental Design

In these experiments, a number of data sets for clustering problems are used. Each of these datasets has a classification which can be used as a reference clustering solution in the evaluation. The data sets used in this experiment are drawn from the UCI machine learning repository [2]; they are summarised in Table 6.1.

Table 6.1: Data Sets

Name	Dimensions	Objects	Classes (used as Clusters)
Balance Scale	4	625	3
Breast Cancer Wisconsin	10	569	2
Sonar	60	208	2
Vowel Recognition	10	528	10
Glass	9	214	7
Ionosphere	34	351	2
Iris	4	150	3
Pima Indians Diabetes	8	768	2
Heart Statlog	13	270	2
Vehicle Silhouettes	18	946	4
Zoo	17	101	7

We have decided to use the pre-defined class labels as the ground truth clusters for this experiment, trusting that these classifications provide a natural grouping of the data. However, we acknowledge that the ideal clustering of any data set is difficult to establish and may not coincide with the classification. Actually, in a classification dataset it is possible that some objects that belong to the same class are very far from each other in the data space, in which case assigning those objects to the same cluster would be a mistake. Investigating the extent to which those problem occurs in the datasets list in table 6.1 (and other classification datasets) is left for future research.

We have chosen to use data sets that are not overly large as we perform a large number of calculations in this experiment when we run the MOEAs. These include generating clustering solutions from sets of solutions, calculating the cluster quality measures on these clustering solutions and assessing the Pareto front at each iteration of the algorithm. Larger data sets would have required significantly more computational time.

We use the algorithm NSGA-II [29] as the MOEA which forms the basis for this set of experiments. Some parameters within NSGA-II need to be set and for this we will use fixed values. The number of generations will be set to 100; the population size will also be set to 100 and the mutation probability and crossover probability will both be 0.3. We determined these values from our preliminary work where we experimented with mutation and crossover probabilities in the range of $[0.1 : 0.9]$ in increments of 0.1, the population size was varied in the range $[50 : 200]$ in increments of 10 and the number of generations was in the range of $[100 : 2000]$ in increments of 100. We observed that as the number of generations approached 100 there was not a large amount of change in the results and we also saw little change as we increased the population size from 100. To initialise solutions we used a random initialisation as in this experiment we are concerned with the effect of the mutation and crossover operators.

For a full implementation, we define the data set to be clustered within a given

representation and provide a mutation operator, a crossover operator and a pair of fitness functions to the algorithm. We then run NSGA-II to generate a Pareto front.

The possible experimental configurations are summarised in Table 6.2. We will test all those configurations, that is, every combination of representation, mutation and crossover.

Table 6.2: Configurations

Representation	Mutation Operators	Crossover Operators
MBBE	Individual Bit Multiple Bit Invert	One Point Two Point Three Point Uniform
LBIE	Unguided (Single) Unguided (Multiple) Guided (Single) Guided (Multiple)	One Point Two Point Three Point Uniform
CBRE	Swap Addition MOCA	One Point Uniform MOCA Centroid Distance Largest Partition

All of the clustering quality measures defined in section 2.3 will be used as fitness functions in pairs. However, we will not pair the Dunn like indexes described in section 2.3.2 together as they are all very similar to one another. We average these results together so we can focus our study on the behaviour of the representation, mutation and crossover operators.

Each configuration and pair of objective functions will produce a Pareto Front or set of hopefully optimal solutions, $\mathcal{S}_a$. We will then normalise the values of the objective functions to allow us to meaningfully compare Pareto fronts. To do this we use the function $h_{\mathcal{Y}}(\vec{s}_{ai})$ described previously in section 4.3.

We can also compute the best obtainable Pareto front, $\mathcal{S}^*$, by combining all of the solutions generated for a particular dataset and pair of objective functions and keeping all those solutions that are non-dominated.

$$\mathcal{S}^* = \bigcup \vec{s}_{ai} \in \mathcal{S}_a \in \mathfrak{S} \text{ where } \exists \vec{s}_{bj} \in \mathcal{S}_b \in \mathfrak{S} : \vec{s}_{aj} \succeq \vec{s}_{bi} \tag{6.1}$$

Hence this constructed semi-optimal Pareto front will contain all of the best solutions obtained in different runs of the algorithm.

From the normalised solutions and semi-optimal Pareto front, $\mathcal{S}^*$, we then compute for each front the area of dominated space, the number of other fronts that are covered, the spread, the generational distance and the inverse generational distance. We use these values to determine which configuration most effectively explores the solution space by ranking configurations as follows. As the measures may not be in agreement we first rank each $\mathcal{S}_a$ in relation to the value of each of the measures of Pareto front quality, where one is the rank of the $\mathcal{S}_a$ that had the most desirable value for a given measure of quality, ties are allowed. For each $\mathcal{S}_a$, its measure of quality are then summed together and the $\mathcal{S}_a$ with the lowest summed rank is then considered the best configuration of the algorithm for the data set and the pair of fitness functions.

To determine the general performance of a configuration of the algorithm we sum together the number of times each configuration was considered the best configuration (best lowest summed rank), thus giving them a vote. If more than one configuration are tied then we reduce the influence of that configuration; that is, if two configurations have tied votes then we sum a $\frac{1}{2}$ vote instead for each. The voting gives a performance score for each configuration and allows us to summarise performance at different levels; for example we can summarise performance of configurations for different datasets or different pairs of objective functions. We can also determine the number of votes for each of the individual representations, crossover and mutation operators.

Most of the previous discussion relates to assessing the quality of the Pareto fronts generated, i.e. assessing the quality of solutions generated by the MO algorithm.

Additionally, it is important to evaluate all of the solutions within the Pareto front to assess the quality of the individual clustering solutions so we can identify which configuration produces the best individual clustering solutions, as well as the best Pareto front.

Two clustering solutions, $\mathcal{P}$ and $\mathcal{P}'$, may be compared to each other using the Rand Index that we described previously in section 2.4. To establish the worth of individual clustering solutions, we will record the value of $\mathcal{RI}(\mathcal{P}, \mathcal{P}')$, where $\mathcal{P}'$ is the "perfect" or reference clustering solution known for each dataset.

We can also modify the voting technique presented earlier to use $\mathcal{RI}(\mathcal{P}, \mathcal{P}')$ to determine which $\mathcal{S}_a$ contains the best clustering (i.e. the one closest to the reference clustering solution) for a given $\mathfrak{S}$. Now we will rank solutions $\mathcal{S}_a$ according to the value of $\mathcal{RI}(\mathcal{P}, \mathcal{P}')$. The rest of the voting procedure remains the same.

6.3 Results

We calculated the frequency with which each possible configuration of the algorithm has been judged as the best configuration by assessing the quality of the Pareto fronts generated for each possible combination of a data set and a pair of fitness functions, summarising results with the voting procedure described earlier. This is reported in Table 6.3. We also report these results based on the quality of the individual clustering solutions produced; this is reported in Table 6.4. These results are summarised to show the performance of the mutation operators and the crossover operators as judged by the quality of the front and by the quality of the generated solutions in Tables 6.5, 6.6, 6.7 and 6.8 respectively.

Table 6.3: Results of the Proposed MOEA for Clustering by Front Quality

Rep	Crossover	Mutation	Score
MBBE	Three Point	Invert	113.0
CBRE	One Point	MOCA	108.8
MBBE	One Point	Invert	85.5
MBBE	Uniform	Individual Bit	80.7
MBBE	One Point	Individual Bit	77.7
MBBE	Two Point	Invert	70.8
CBRE	Centroid Distance	Addition	69.7
MBBE	Three Point	Individual Bit	68.0
MBBE	Two Point	Individual Bit	60.5
MBBE	Uniform	Invert	52.5
MBBE	Uniform	Multiple Bit	51.0
CBRE	Uniform	MOCA	49.8
MBBE	Three Point	Multiple Bit	46.0
MBBE	Two Point	Multiple Bit	40.8
MBBE	One Point	Multiple Bit	37.8
CBRE	MOCA	MOCA	36.9
CBRE	Multipoint Pairwise	MOCA	35.6
CBRE	Largest Partitions	MOCA	33.6
CBRE	Centroid Distance	MOCA	32.6
CBRE	Largest Partitions	Addition	28.3
CBRE	One Point	Addition	28.3
CBRE	One Point	Swap	24.8
CBRE	MOCA	Addition	20.3
CBRE	Multipoint Pairwise	Addition	20.0
CBRE	Uniform	Swap	18.8
CBRE	Uniform	Addition	15.1
CBRE	Multipoint Pairwise	Swap	13.0
CBRE	MOCA	Swap	11.3
CBRE	Largest Partitions	Swap	10.8
CBRE	Centroid Distance	Swap	7.8
LBIE	One Point	Guided (Individual)	6.0
LBIE	Two Point	Guided (Individual)	5.5
LBIE	Three Point	Guided (Individual)	5.5
LBIE	One Point	Unguided (Individual)	4.0
LBIE	Three Point	Unguided (Multiple)	3.0
LBIE	Uniform	Unguided (Multiple)	3.0
LBIE	Three Point	Unguided (Individual)	2.0
LBIE	One Point	Guided (Multiple)	2.0
LBIE	One Point	Unguided (Multiple)	2.0
LBIE	Uniform	Guided (Multiple)	1.0
LBIE	Two Point	Guided (Multiple)	1.0
LBIE	Two Point	Unguided (Individual)	1.0
LBIE	Two Point	Unguided (Multiple)	0.0
LBIE	Three Point	Guided (Multiple)	0.0
LBIE	Uniform	Guided (Individual)	0.0
LBIE	Uniform	Unguided (Individual)	0.0

LBIE as the representation, with all its configurations, performed poorly in terms of both quality of the Pareto front and quality of the clustering solutions obtained. This can be seen in tables 6.3 and 6.5 respectively where LBIE ranked low. In all cases the quality of the Pareto fronts produced was lower than all of the other possible configurations, which implies that the Pareto fronts were far from the optimal front and explored a limited part of the objective space. The fronts also contained clustering solutions that were less similar to the intended clustering solutions than most of the other configurations.

Configurations using the MBBE representation generally perform well when performance is judged by the quality of the Pareto Fronts produced as can be observed in Table 6.3. On the other hand, when judged using the quality of the clustering solutions CBRE configurations performed better, as can be observed in Table 6.4. Hence , overall, MBBE produces good Pareto fronts, whereas CBRE produces better individual solutions.

Table 6.4: Results of the Proposed MOEA for Clustering by $\mathcal{RI}(\mathcal{P}, \mathcal{P}')$

Rep	Crossover	Mutation	Score
CBRE	MOCA	MOCA	112.5
CBRE	Centroid Distance	MOCA	103.8
CBRE	MOCA	Addition	96.0
CBRE	Uniform	Addition	95.5
CBRE	Largest Partitions	MOCA	92.0
CBRE	Multipoint Pairwise	MOCA	88.1
CBRE	Multipoint Pairwise	Addition	85.0
CBRE	Largest Partitions	Addition	82.0
CBRE	Centroid Distance	Addition	74.5
CBRE	Multipoint Pairwise	Swap	64.0
CBRE	One Point	Addition	63.0
CBRE	Uniform	MOCA	48.5
CBRE	Centroid Distance	Swap	46.5
CBRE	One Point	MOCA	35.5
CBRE	Largest Partitions	Swap	27.0
MBBE	Uniform	Invert	24.0
CBRE	MOCA	Swap	23.5
MBBE	One Point	Invert	22.0
MBBE	Two Point	Invert	20.0
MBBE	Three Point	Invert	20.0
MBBE	One Point	Individual Bit	18.3
CBRE	One Point	Swap	16.0
MBBE	Three Point	Individual Bit	14.0
MBBE	Two Point	Individual Bit	13.3
LBIE	One Point	Guided (Individual)	13.0
LBIE	Three Point	Guided (Individual)	12.0
MBBE	Uniform	Individual Bit	11.5
LBIE	One Point	Unguided (Individual)	10.0
MBBE	Two Point	Multiple Bit	7.0
LBIE	One Point	Guided (Multiple)	7.0
LBIE	Three Point	Unguided (Individual)	6.0
LBIE	Two Point	Guided (Individual)	5.0
CBRE	Uniform	Swap	4.5
MBBE	Uniform	Multiple Bit	4.0
LBIE	Three Point	Guided (Multiple)	4.0
LBIE	Three Point	Unguided (Multiple)	3.0
MBBE	One Point	Multiple Bit	3.0
MBBE	Three Point	Multiple Bit	3.0
LBIE	Two Point	Unguided (Individual)	3.0
LBIE	Two Point	Guided (Multiple)	2.0
LBIE	One Point	Unguided (Multiple)	2.0
LBIE	Two Point	Unguided (Multiple)	1.0
LBIE	Uniform	Guided (Individual)	0.0
LBIE	Uniform	Unguided (Multiple)	0.0
LBIE	Uniform	Unguided (Individual)	0.0
LBIE	Uniform	Guided (Multiple)	0.0

Tables 6.5 and 6.6 summarise the results for mutation operators and show us that in terms of Pareto front quality, the Invert mutation operator used with MBBE had the highest evaluation of the mutation operators with our method. The invert mutation operator is the most disruptive operator in terms of changing the clustering solution. The MOCA mutation operator with CBRE and Individual Bit mutation operator within MBBE also performed well. The MOCA mutation operator is somewhat disruptive, hence disruptive mutation mechanisms appear advantageous for exploration of the search space, in order to to obtain good Pareto fronts. Table 6.6 shows us that Addition and MOCA mutation operators performed significantly better than all of the other operators when the quality of the clustering solutions produced is being judged by the Rand Index. The MOCA mutation operator performed well both in terms of quality of Pareto fronts and quality of individual solutions, so it seems a good compromise.

Table 6.5: Results of the Proposed MOEA for Clustering for Mutation by Front Quality

Rep	Mutation	Score
MBBE	Invert	321.8
CBRE	MOCA	297.4
MBBE	Individual Bit	286.8
CBRE	Addition	181.8
MBBE	Multiple Bit	175.7
CBRE	Swap	86.6
LBIE	Guided (Individual)	17.0
LBIE	Unguided (Multiple)	8.0
LBIE	Unguided (Individual)	7.0
LBIE	Guided (Multiple)	4.0

Table 6.6: Results of the Proposed MOEA for Clustering for Mutation by $\mathcal{RI}(\mathcal{P}, \mathcal{P}')$

Rep	Mutation	Score
CBRE	Addition	496.2
CBRE	MOCA	480.3
CBRE	Swap	181.5
MBBE	Invert	86.0
MBBE	Individual Bit	57.0
LBIE	Guided (Individual)	30.0
LBIE	Unguided (Individual)	19.0
MBBE	Multiple Bit	17.0
LBIE	Guided (Multiple)	13.0
LBIE	Unguided (Multiple)	6.0

The guided mutation operators did not perform well, which initially seems surprising, given that they are specifically designed for the clustering task. A likely explanation for these results is that the guided mutation operators were only used with the LBIR encoding, which was the worst encoding that we experimented with. Considering only the results for LBIE encoding in Tables 6.5 and 6.6, we can observe that, in the context of LBIE, the Guided (Individual) mutation was the most successful operator, outperforming the unguided operators. However, the Guided (Multiple) mutation operator was not successful as it performed worse than the unguided mutation operators. It is possible that when multiple changes were made to the solutions these changes may somehow work against each other. Perhaps this causes some clusters to loose or gain too many members in one iteration.

Tables 6.7 and 6.8 summarise results for crossover operators. Crossover operators used with MBBE have again outperformed the other operators in terms of Pareto front quality (Table 6.7), whereas in terms of clustering solution quality, as shown in Table 6.8, operators associated with CBRE perform better. The operators used with a LBIE have performed poorly. There does not appear to be a clearly superior crossover operator, but for CBRE the guided operators perform better. Simple crossover schemes such as the one point crossover have performed well in this experiment.

Table 6.7: Results of the Proposed MOEA for Clustering for Crossover by Front Quality

Rep	Crossover	Guided?	Score
MBBE	Three Point	No	227.0
MBBE	One Point	No	201.0
MBBE	Uniform	No	184.2
MBBE	Two Point	No	172.1
CBRE	One Point	No	161.9
CBRE	Centroid Distance	Yes	110.1
CBRE	Uniform	No	83.8
CBRE	Largest Partitions	Yes	72.8
CBRE	Multipoint Pairwise	Yes	68.6
CBRE	MOCA	No	68.5
LBIE	One Point	No	14.0
LBIE	Three Point	No	10.5
LBIE	Two Point	No	7.5
LBIE	Uniform	No	4.0

Table 6.8: Results of the Proposed MOEA for Clustering for Crossover by $\mathcal{RI}(\mathcal{P}, \mathcal{P}')$

Representation	Crossover	Guided?	Score
CBRE	Multipoint Pairwise	Yes	237.1
CBRE	MOCA	Yes	232.0
CBRE	Centroid Distance	Yes	224.8
CBRE	Largest Partitions	Yes	201.0
CBRE	Uniform	No	148.5
CBRE	One Point	No	114.5
MBBE	One Point	No	43.3
MBBE	Two Point	No	40.3
MBBE	Uniform	No	39.5
MBBE	Three Point	No	37.0
LBIE	One Point	No	32.0
LBIE	Three Point	No	25.0
LBIE	Two Point	No	11.0
LBIE	Uniform	No	0.0

6.4 Conclusion & Summary

This chapter has presented an experimental study on components of Multi-Objective Evolutionary Algorithms for clusterings. In Chapter 4, we described how a good

configuration for a MOEA should lead to a set of diverse solutions that are close to the Pareto front. Using the techniques we described in Section 4.3, we evaluated the sets of solutions (or Pareto fronts) that were generated.

The experimental results show that CBRE and MBBE are superior clustering representations to LBIE as they generate better Pareto fronts and better individual solutions.

The CBRE representation is likely to produce a Pareto Front containing good clustering solutions. The results also show that this representation in-conjunction with the MOCA mutation operator can find good Pareto fronts, so it appears as a good compromise. Part of our motivation for this study was to assess how well our proposed implementation of a Multi-Objective Evolutionary Algorithm for clustering, including the MOCA operators, worked in relation to other implementations. The results validate our approach.

We also found that in terms of mutation, operators such as Addition or MOCA that manipulate the values of the centroids provide good results. They outperform others such as the swap operator that only exchanges centroids. Furthermore, mutation operators that are very disruptive, that is to say change the solutions drastically, cause the algorithm to explore a larger area of the objective space and return better Pareto fronts. More constructive mutation operators that modify the solution with respect to the problem with little disruption such as: Addition, Swap or MOCA tend to produce clustering solutions that are more similar to the desired clustering solutions. Such information provides useful guidance to produce better mutation operators.

In terms of exploring the objective space and producing a good Pareto front, MBBE was better, particularly when the very disruptive Invert mutation operator was used. The Invert operator drastically changes the solution and may lead to solutions that are very different from the pre-existing population, hence why the space may be well explored. However, Invert mutation can lead to a great increase in the number of clusters, hence further research could be performed with mutation

operators that radically change the resulting clustering solutions without causing a large increase in the number of clusters.

In the results, we did not find that there was a clearly superior crossover operator for both producing high quality clustering solutions and investigating the objective space. This is an area that may require further investigation, for example, designing a crossover operator that attempts to improve the Pareto front and the clustering quality of the solutions in the population using a combination of disruptive and constructive techniques when appropriate.

In fact, future work on Multi-objective algorithms for clustering could focus upon medoid or centroid based representations of a clustering solution and should investigate the effects of including an initial exploration phase with disruptive operators followed by a local search phase with more constructive operators. This may be key to investigating the space fully while also producing high quality clustering solutions.

The LBIE performed poorly. This may be because it is difficult to randomly produce solutions where the clusters are continuous shapes within the space. Clusters that are not continuous will obviously be of poor quality. This representation may still be worth considering as it allows the production of arbitrarily shaped clusters. This technique may show an improved performance if there was a stage of the algorithm that increased the quality of the clustering solutions that were found. Future work could also include enhancements that establish rules about clusters being continuous. This could involve constraints using the K-nearest neighbours of objects as a method of forming clusters, which has been used previously by [62].

We observed that the Guided (Individual) mutation operator outperformed the Guided (Multiple) mutation operator for the LBIE. This was unusual as many previous approaches to clustering change many cluster memberships at each iteration so we had not expected this bad result. Similarly, guided mutation operators for the other encodings should also be investigated.

Chapter 7

Experimental Comparison of New Mutation Operators

7.1 Introduction

In this chapter we expand upon the experiment that we presented in Chapter 6. Previously we concluded that we had seen promising results using Centroid Based Real Encoding as the representation for a Multi-Objective Clustering Algorithm. We found that mutation operators that were very disruptive explored the objective space well and found good Pareto fronts but did not find the highest quality clustering solutions. We also found that mutation operators that manipulated the clustering solutions directly lead to highest quality clustering solutions but did not explore the Pareto front as well as the disruptive operators. We therefore suggested that further research on the performance of the operators used within the algorithm should be conducted. Here we will present an investigation into three mutation operators designed to emphasise these properties.

In this chapter we report on an investigation into the performance of three different mutation operators used in conjunction with CBRE. We first explain in section 7.2 the main configuration of the algorithm used: the representation, crossover op-

erator and MOEA.

In Section 7.2.3 we describe three mutation operators that attempt to promote different aspects of a solution: front quality, clustering solution quality and a hybrid combination of these. The mutation operator designed to promote front quality is a variation of the mutation operator that we proposed in Chapter 5. The mutation operator designed to promote the quality of the clustering solutions is loosely based upon the clustering algorithm k-means. The final choice is a combination of the two other mutation operators.

We present the experimental setup in section 7.3. The results are presented in section 7.4. Supporting figures have been placed in Appendix A.

7.2 Multi-Objective Clustering Algorithm

The algorithm we use here is similar to one we have presented in previous work [85], Chapter 5 and Chapter 6. Again we use NSGA-II [29] with 100 generations, population size of 100 and mutation and crossover probability of 0.3. For a full implementation, we used a representation, a mutation operator, a crossover operator described in Section 7.2.2 and a pair of fitness functions. We use the Centroid Based Real Encoding (CBRE) previously defined in Section 4.4.3 as the representation; the uniform crossover operator; and for the fitness functions we use average within group sum of squares and $\mathcal{C}onn$, previously described and used in Chapter 5. We do not use the between group sum of squares as a third objective as our preliminary work suggested that using it and the within group sum of squares together did not change the results significantly. Using two objectives instead of three made the task of analysing the results more manageable.

7.2.1 Representation

For this experiment we will be using a Centroid Based Real Encoding. This representation of a clustering solution is based on storing a set of prototype cluster centroids. Modifications to the solution can be made by moving the prototype centroids into new positions and by changing the number of prototype centroids in the solution. Clustering solutions are then derived from the representation by assigning each object in the data set to the closest prototype centroid. We previously fully defined CBRE in Section 4.4.3. Formally we define a CBRE as $\mathcal{R} = \{\vec{r}_1, \ldots, \vec{r}_r\}$ where each $\vec{r}_g, g = 1, ..., r$, represents an individual prototype cluster centroid.

We have chosen to use CBRE as the representation based on our previous work reported in Chapter 6, which highlighted it as leading to high quality individual solutions. However, this representation was not the most effective in terms of improving the quality of the Pareto front. Here, we further investigate its performance.

7.2.2 Crossover

To crossover two CBRE solutions, $\mathcal{R}$ and $\mathcal{R}'$, we will use a uniform crossover operator. A simple uniform crossover has been widely used in conjunction with a CBRE[23, 92, 135] and we also found that it performed fairly well in our previous experiments in Chapter 6. The Centroid Distance crossover operator defined in Section 4.4.3.2 was better in some instances. However, we have chosen to not use that crossover operator here, as it is complex and is very disruptive to the clustering solutions and could be sensitive to the nature of the data.

The implementation of the uniform crossover operator is as follows. From two solutions, $\mathcal{R}$ and $\mathcal{R}'$, we produce two new solutions $\mathcal{R}''$ and $\mathcal{R}'''$. For each $\vec{r}_g \in \mathcal{R} \bigcup \mathcal{R}'$ there is a 50% chance that $\vec{r}_g$ will be included in either $\mathcal{R}''$ or $\mathcal{R}'''$. $\mathcal{R}''$ and $\mathcal{R}'''$ will be valid solutions as the order of the prototypes is unimportant. If identical prototypes exist within a new solution only one is used when a clustering solution is generated. This crossover operator has been modified to work with solutions that

are not of equal length unlike their original definitions.

7.2.3 Mutation Operators

Our main contribution in this chapter is to investigate the performance of three mutation operators: one that is entirely based on randomness, one that refines solutions in a form of local search using concepts from k-means and a hybrid combination of them.

Previously in Chapter 6 we found that the Pareto front was best explored by the mutation operators that were the most disruptive to the solutions. We also found that the mutation operators that directly manipulated the clustering solutions produced higher quality clustering solutions. Here we aim to test this hypothesis further by proposing two mutation operators that we have designed to emphasise these two characteristics.

The initial assumption is that the Randomness Mutation operator will better explore the search space by producing very disruptive changes in solutions, whereas the k-means Like Mutation operator will improve the quality of individual solutions by performing a form of local search. The hybrid mutation operator is expected to combine the exploration of the Pareto front with the exploitation of good solutions. They are as follows:

7.2.3.1 Randomness Mutation (RM)

Previously in Section 5.2.2 we defined a mutation operator designed as part of MOCA [85]. In Section 5.4 we showed that MOCA appeared to be working correctly. In Chapter 6 we performed an experiment to compare the different characteristics of representations of the clustering problem and we found that the MOCA mutation operator showed good performance both in terms of the quality of the Pareto fronts generated and the quality of the clustering solutions produced.

We felt that this mutation operator promoted randomness within the clustering

solution as it is very disruptive to the structure of the solutions. The operator combines three tasks (each considered a "sub-operator" in Section 5.2.2): decreasing the number of prototypes; increasing the number of prototypes; and modifying the prototypes. Here we present a refined version of this operator: now each sub operation is performed with equal probability. Previously we biased the algorithm to decrease the number of clusters in each solution as our preliminary work had found that the algorithm tended to increase the number of clusters in the clustering solutions.

We have changed the implementation of the evolutionary algorithm to use a single set of prototype centroids, $\mathcal{R}$, instead of a more complex solution where two sets of cluster prototypes were used. We therefore present the increase and decrease sub operators here again. These operators are conceptually very similar to the sub operators we presented in 5.2.2.

The third sub operator we used previously was a recompute operation that recalculated the prototype centroids using a method similar to a single iteration of the k-means algorithm. Here we use a modification operator instead. This operation is based upon moving the centroids around the solution space randomly instead of refining the solution.

Here, we redefine the operator as follows:

- All three tasks defined bellow are equiprobable.
- We use only one set of prototype centroids, $\mathcal{R}$, instead of two.
- We redefine the tasks that can be performed, and, in particular, the recalculation of cluster prototypes.

The tasks that can be performed are:

Decrease To decrease the number of prototypes, first the pair of prototypes that are closest together must be found, computing $\min \delta\left(\vec{r}_g, \vec{r}_h\right)\ \forall \vec{r}_g, \vec{r}_h \in \mathcal{R}$. For each

prototype in this pair, the second closest prototype is found. The prototype of the original pair that has the closest neighbour is removed. That is, $\min \delta\left(\vec{r}_g, \vec{r}_i\right) \forall \vec{r}_i \in \mathcal{R} < \min \delta\left(\vec{r}_h, \vec{r}_j\right) \forall \vec{r}_j \in \mathcal{R}$ then $\vec{r}_g$ is removed, else $\vec{r}_h$ is removed. This sub-operator is conceptually the same as the sub-operator defined in section 5.2.2.1.

Increase To increase the number of prototypes, a new prototype is drawn from $\mathcal{D}$. The $\vec{x}_i$ within $\mathcal{D}$ that is furthest away from any prototype is inserted into $\mathcal{R}$, computing $\max \delta\left(\vec{x}_i, \vec{r}_i\right) \forall \vec{x}_i \in \mathcal{D} \wedge \forall \vec{r}_i \in \mathcal{R}$. This sub-operator is conceptually the same as the sub-operator defined in section 5.2.2.2.

Modification For each dimension of each prototype, r_{gi}, there is a chance that this dimension maybe modified. Scheunders [131] and Maulik [102] use a probability of 0.05. To mutate a dimension, a negative or positive value is added with an equal chance to the dimension. We use the technique inspired by the technique that Bandyopadhyay [6] used. We calculate a scaled value by using the minimum and maximum values of the dimension within $\mathcal{D}$. The value that is added, or subtracted is equal to 1% of the difference between the minimum and maximum values in that dimension of the data set.

7.2.3.2 k-Means Like Mutation (KMLM)

An iteration of the k-means algorithm can be used as a mutation operator by re-calculating the cluster prototypes. The mutation operator we previously used in Section 5.2.2 used a single iteration of k-means as a sub operation to refine the clustering solution. Here we use this sub operation as a mutation operator in its own right. This operator is intended to refine the quality of the clustering solutions. This type of mutation has been used before with successful results [90, 100].

To perform the mutation, first a clustering solution, $\mathcal{P}$, is derived from $\mathcal{R}$. If $r \neq k$ because there are identical prototypes then r is changed to k by adding additional centroids. Each new prototype $\vec{r}_g$ is set to the value of a random centroid

$\vec{c}_g$ from the generated clustering $\mathcal{P}$.

7.2.3.3 Hybrid Mutation (HM)

The hybrid mutation operator combines the other operators with a linearly varying probability of application. Initially, the probability of using RM is higher to explore the solution space; later in the search, the probability of using KMLM becomes higher in order to refine solutions. The probability of application of KMLM over RM is calculated as g/m where g is the current generation and m is the number of generations in total. This mutation operator relies on the maximum number of generations as a stopping criterion so there must be a value of m.

7.3 Experimental Setup

We compare the performance of the mutation operators in terms of both individual solution quality and Pareto front quality, as described in section 2.4 and 4.3. As a benchmark, we also generate Pareto fronts using the k-means algorithm described in section 2.2.1.1.

For the implementations based upon NSGA-II we keep a record of all the solutions in the current population at each generation to see how the quality of the population changes as the algorithm progresses.

For the k-means algorithm, an execution begins with a single solution that is modified incrementally over a number of iterations. Typically k-means is executed several times and CQMs are used to select a solution. We run P instances of k-means in parallel for G iterations, where P is the population size and G is the number of generations used for NSGA-II. The k-means algorithm typically converges well before all G iterations have occurred. In this case P and G are both set to 100 but they could be set to different values in future work. Therefore, the solutions present at the first iterations are equivalent to the first generation of a genetic algorithm and so on. When all of the instances of k-means have finished executing we return

the non-dominated solutions. We did not vary these values so that we could create create a simple environment for the experimental configuration as these values have been shown to be sensible choices in our preliminary work.

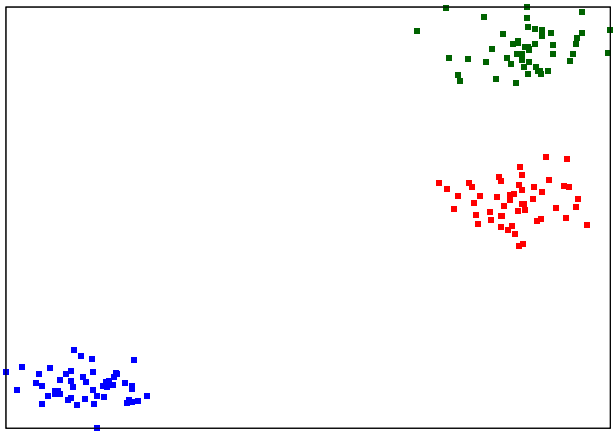
Each run of NSGA-II (with a different mutation operator) starts with identical populations to ensure differences in performance are not due to the random generation of the start populations. We use the same populations to supply centroids to the k-means instances. This ensures a variety of start populations while isolating performance differences to the mutation operators.

We use nine data sets, described in Table 7.1. Six are popular benchmark data sets for clustering problems drawn from the UCI Machine Learning Repository [1] and three have been generated for this experiment, and are visualised in Figure 7.1 as follows. The data sets drawn from the UCI repository in this experiment differ from the data sets that we used in our previous experiment that we described in Table 6.1. For this experiment we chose to eliminate some of the larger data sets as performing experiments upon them took significantly longer than the data sets that we retained. Data set **g** is designed to represent a typical clustering solution where the clusters are roughly spherical and well separated, data set **h** is designed to represent two continuous clusters that cannot be easily defined by their centroids as they are approximately the same and data set **i** is designed to show two continuous clusters that are separated but also cannot be well described by their centroids. To cluster the data we use all of the variables within the data sets, other than the variable that represents the correct class. As before the class labels are assumed to be representative of the expected clustering solution as we explained in Section 6.2.

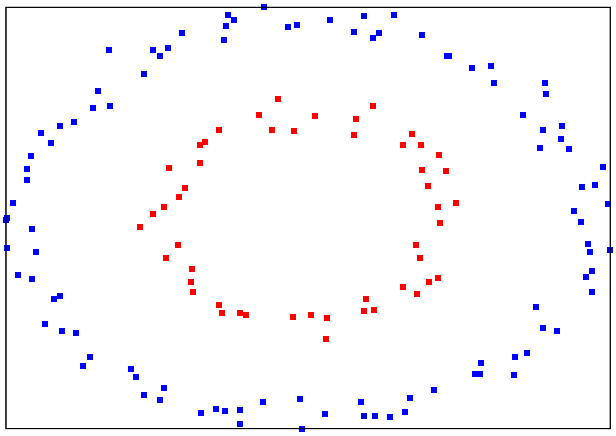
For each dataset we perform 100 executions of NSGA-II and $100 \times P$ executions of k-means, each with a distinct random seed used to create the initial population. We record all solutions ever discovered to generate $\mathcal{S}^*$, the simulated optimal front described in section 4.3.

For each generation of solutions we calculate: Volume of Dominated Space;

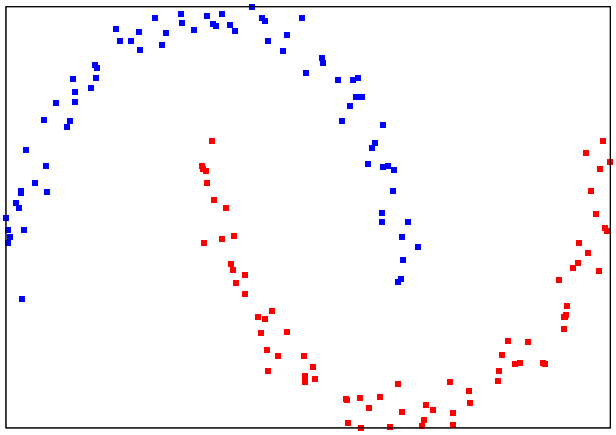
[1] `http://archive.ics.uci.edu/ml/datasets.html`



(a) Data Set **g**



(b) Data Set **h**



(c) Data Set **i**

Figure 7.1: Visual Representation of Constructed Data Sets

Table 7.1: Data Sets

Data Set	Name	Dimensions	Objects	Classes (used as Clusters)
UCI Data Sets				
a	Balance Scale	4	625	3
b	Breast Cancer Wisconsin	10	569	2
c	Glass	9	214	7
d	Heart Statlog	13	270	2
e	Iris	4	150	3
f	Zoo	17	101	7
Constructed Data Sets				
g	Blobs	2	150	3
h	Circles	2	150	2
i	Moons	2	150	2

Spread; GD; IGD; Entropy; and the average Rand Index for each clustering solution. We calculate the arithmetic mean of these across 100 executions. This averaging gives us a good description of the algorithms whilst eliminating any anomalies that may be introduced by randomly generating the start populations.

We will also calculate a statistical test to determine if one representation is significantly different from the others. Here we will use the Friedman test [24] for this task. The Friedman test is a non parametric statistical test that we will use to see if any of the representations produces statistically different rankings of the results for each measure of quality.

To perform the Friedman test first for each measure of quality we are using we rank the best representation as 1 and the worst representation as 4 for each data set. We then calculate the value of the Friedman test as described by Demšar [31]. We use a critical value of 3.01 for the 0.05 significance level. If the result of the Friedman test is statistically significant then we perform a post-hoc test to determine which representation produced significantly different results to the others. To perform the post-hoc test we use the Nemenyi test with a critical value of 1.563 for the 0.05 significance level. The results of post-hoc tests are presented as critical difference diagrams as proposed by Demšar.

7.4 Results

Here we present a comparison of the performance of the benchmark k-means Algorithm (KMA) against the implementations of NSGA-II that use RM, KMLM and HM that we proposed in Section 7.2.3. The results are presented graphically as the algorithms progress and as tables summarising the values of each measure at the final, most important, Pareto front in Appendix A. We assess a number of Pareto front quality measures as they may indicate different aspects of quality. We also assess the quality of individual solutions with the Rand Index. Throughout we observe that KMA converges quickly to a fixed set of solutions where no further improvements are made and the implementations of NSGA-II continue to change the population of the solutions throughout.

7.4.1 Volume of Dominated Space

Previously in Section 4.3.1 we introduced how we could use the volume of the space dominated by a set of solutions as a measure of performance. Figures A.1 through A.9 show the change in the volume of the space dominated by the Pareto front as the algorithms progress. In general as each algorithm progresses the volume of space that the Pareto front dominates increases.

Generally KMA improves initially and then stabilises without further change. In the case of Figure 7.2, we can see that the results actually got worse after an initial improvement when executing the algorithm on dataset **e**. We did not observe this behaviour for any other data sets used in conjunction with KMA.

The algorithm using RM generally improves early on in the process and then no further improvements are seen. In the case of datasets; **a**, **b**, **d**, **h** and **i** we see that there is a gradual improvement throughout the execution of the algorithm, but the volume of the data set that is dominated is lower than that of the other implementations OF NSGA-II. In the case of data set **g** there is a slight loss in the volume of the space dominated but this is very small.

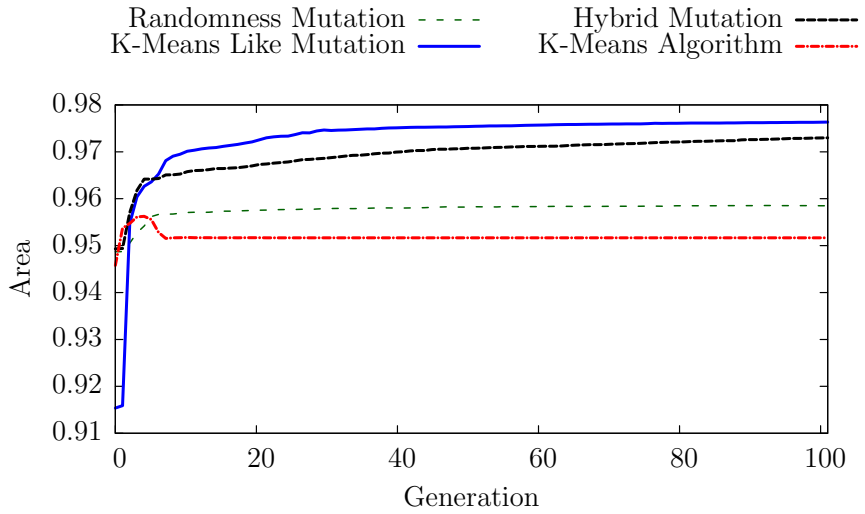


Figure 7.2: Change in Volume of Dominated Space for Dataset **e** over 100 generations

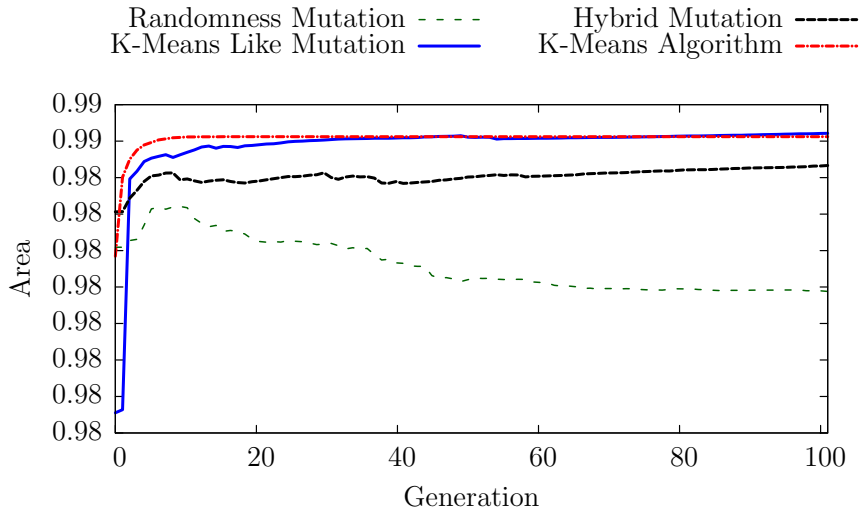


Figure 7.3: Change in Volume of Dominated Space for Dataset **g** over 100 generations

In the majority of cases the area dominated by the implementations using KMLM and HM increases as the algorithm progresses. KMLM increases at the quickest rate but in most cases HM and KMLM converge to similar values, though KMLM is often slightly higher. Dataset **g** shows a case where KMLM and KMA performed almost identically and were better than HM, this can be seen in Figure 7.3.

Table 7.2 shows that KMLM dominates a larger volume of the objective space

Table 7.2: Volume of the Dominated Space of the final Pareto front (best results highlighted)

Data Set	RM	KMLM	HM	KMA
a	0.6001	**0.6728**	0.6672	0.6436
b	0.7945	0.8791	**0.8796**	0.8468
c	0.8241	**0.9090**	0.8937	0.8672
d	0.7965	**0.8364**	0.8341	0.7767
e	0.9585	**0.9763**	0.9730	0.9517
f	0.8785	**0.9308**	0.9011	0.8890
g	0.9819	**0.9862**	0.9853	0.9861
h	0.7620	**0.7864**	0.7785	0.7446
i	0.8949	**0.9188**	0.9136	0.8574

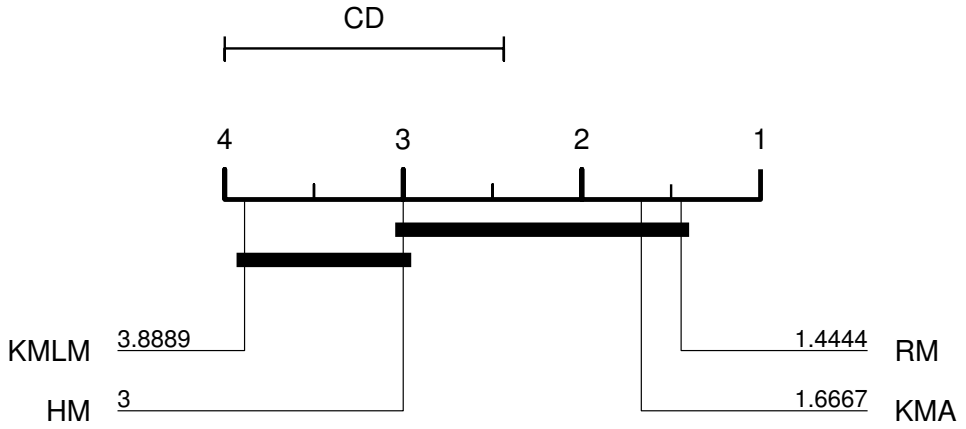


Figure 7.4: Critical difference diagram for Volume of the Dominated Space

at the final generation for most of the datasets. HM showed better performance by this method of assessment only in the case of dataset **b**. None of the other implementations showed better performance after 100 generations.

We performed the Friedman test on the data presented in Table 7.2 and obtained a value of 13.13 which shows statistical significance. We performed the Nemenyi post-hoc test and we report the results of this in Figure 7.4. We can see that KMLM produces statistically different results to KMA and RM. We cannot say that the other pairings are statistically different to one another. The results of KMLM and KMA are statistically different from one another and the results for KMLM are more desirable so KMLM has performed better than KMA when assessed with the

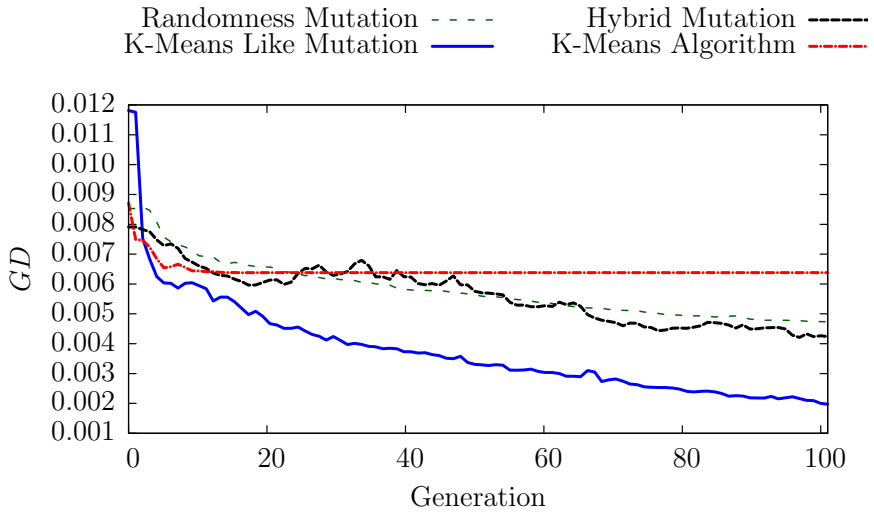


Figure 7.5: Change in GD for Dataset **g** over 100 generations

volume of the dominated space.

7.4.2 *GD & IGD*

The Generational Distance (GD) is the distance between the optimal Pareto front, $\mathcal{S}^*$, and a given Pareto front. As the algorithm converges GD should reduce. This measure was previously introduced in Section 4.3.4. We show our results in Figures A.10 through A.18. In general the distance decreases as we would expect.

KMLM shows continuous improvement of GD and therefore of the Pareto front. For all of the data sets we studied, the value of GD is initially very high, consistently with a poor initial Pareto front. Most improvement occurs early in the search. In the case of dataset **g** (Figure 7.5) the initial decrease in the value of GD is less pronounced as the values found later continue to improve at a faster rate than the other implementations.

The convergence to $\mathcal{S}^*$ in terms of GD by HM is similar, though sometimes slower than KMLM. In the cases of dataset **a** and **b** (Figures 7.6 and 7.7) the results actually get slightly worse initially before slowly improving as the algorithm progresses. The value of GD appears erratic on dataset **g** but it is very small and may not therefore

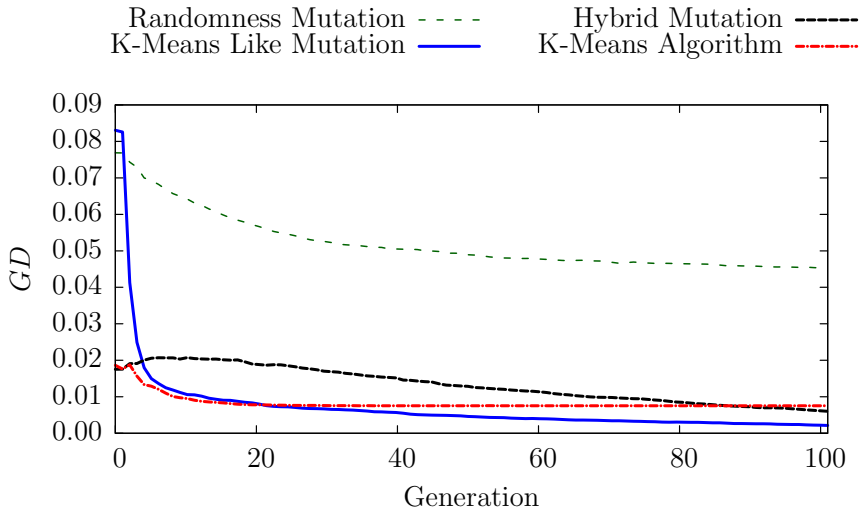


Figure 7.6: Change in GD for Dataset **a** over 100 generations

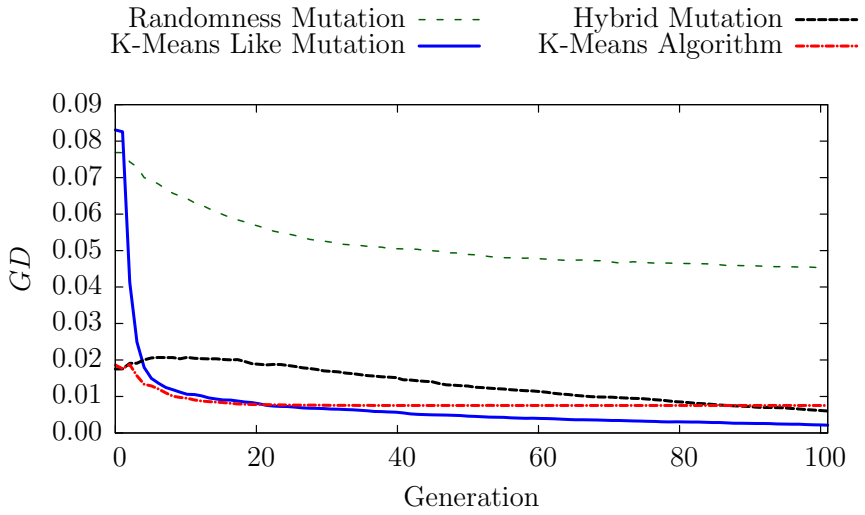


Figure 7.7: Change in GD for Dataset **b** over 100 generations

be significant.

We observe that initially KMA performs similarly to KMLM, but it does not show continuous improvement with more generations. The implementation of the k-means algorithm that we used in this experiment converges quickly for all of the datasets. KMLM exhibits a behaviour where early populations are significantly further away from $\mathcal{S}^*$ than the solutions maintained by KMA. This is not an artefact

Table 7.3: GD of the final Pareto front (best results highlighted)

Data Set	RM	KMLM	HM	KMA
a	0.0453	**0.0022**	0.0060	0.0075
b	0.0604	0.0040	**0.0032**	0.0205
c	0.0452	**0.0048**	0.0091	0.0340
d	0.0358	**0.0047**	0.0065	0.0259
e	0.0206	**0.0015**	0.0041	0.0530
f	0.0318	**0.0002**	0.0060	0.0242
g	0.0047	**0.0020**	0.0042	0.0064
h	0.0312	**0.0103**	0.0191	0.0193
i	0.0392	**0.0114**	0.0196	0.0355

of the population's initialisation process as starting populations are shared by each algorithm across each run. The initial behaviour of KMLM is generally similar to KMA once the solutions have improved from their initial low quality until KMA stops improving. This shows that KMLM acts very similarly to KMA but is not constrained by the same covering conditions. As the mutation operator is essentially k-means any further improvements after KMA stops improving should be attributed to the MOEA.

RM shows consistently less desirable values of GD that the other implementations. The value of GD continues to improve as the algorithm is executed, but in the cases of datasets **a** and **b** it is considerably worse than the other implementations. However, it does show better performance that KMA for datasets **e** and **g**.

The final value of GD shown in Table 7.3 is higher (worse) for KMA with respect to KMLM at the end of the experiment. RM shows the worst performance. It does not produce a Pareto front that is closest to the optimal Pareto front in any of the cases and is the worst in most of them. In all but one of the cases the distance between the final Pareto front and $\mathcal{S}^*$ is best for KMLM and KMA. KMLM obtained the best result in eight out of the nine datasets.

We performed a Friedman test on this data and obtained a result of 65.63 which shows that the representations are statistically different to one another. We performed a Nemenyi test and report the results in Figure 7.8. The results for KMLM

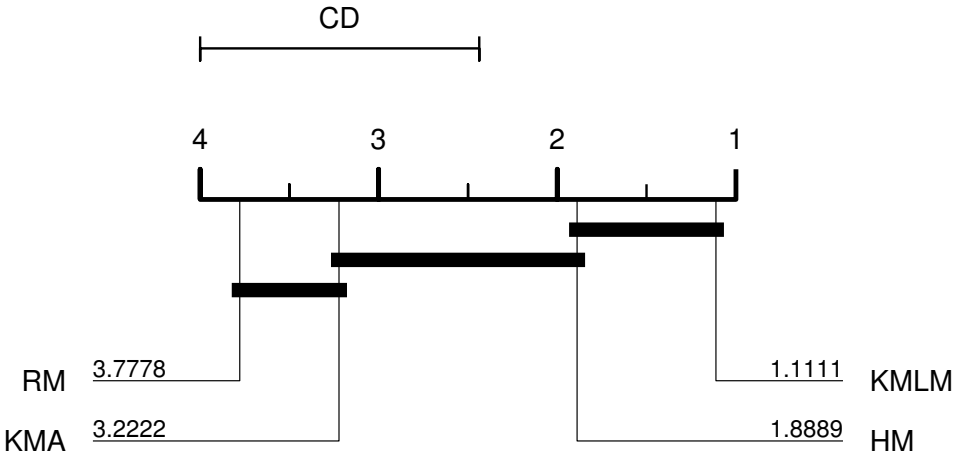


Figure 7.8: Critical difference diagram for GD

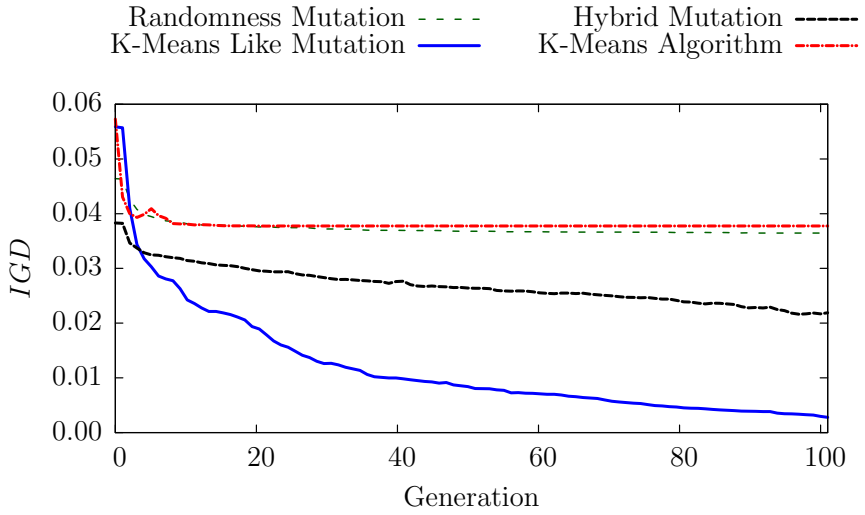


Figure 7.9: Change in IGD for Dataset **e** over 100 generations

are statistically different from RM and KMA. RM is also statistically different from HM. KMLM has produced more desirable values than KMA and RM so we can say that KMLM has performed better than KMA and RM when measured with GD. HM has also performed better than RM when measured with GD.

Previously in Section 4.3.4 we introduced IGD, the inverse of GD. The evaluation of Pareto quality by IGD and GD are different for the problems we have used. If all of the solutions in a Pareto front are in $\mathcal{S}^*$ but the tail ends of $\mathcal{S}^*$ are not included then it would have a low GD and high IGD. Conversely, a high GD and

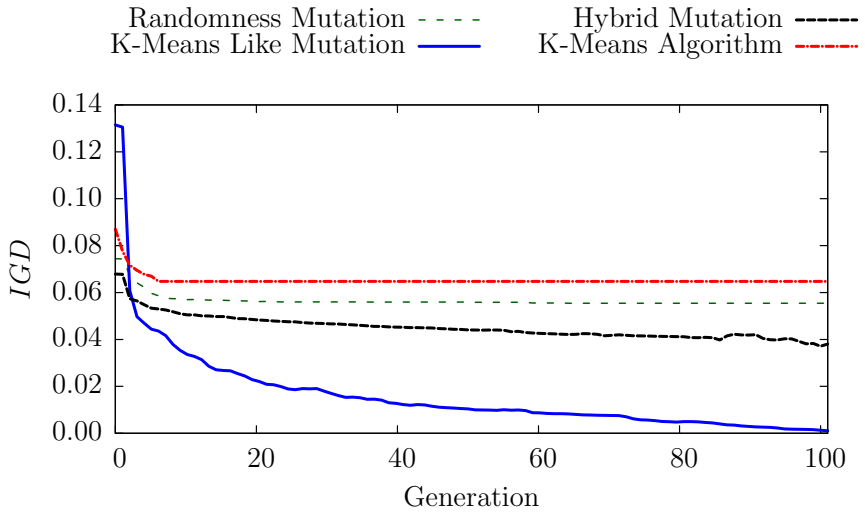


Figure 7.10: Change in IGD for Dataset **f** over 100 generations

low IGD may be achieved if the front contained the extreme solutions but not the solutions in the middle of the front. We see these effects in Figures A.19 through A.27 and Table 7.4, where the values of IGD for KMA are much higher than the values of GD presented previously.

The performance of HM and KMLM are less similar when measured with IGD, particularly in the case of datasets **e** and **f** (Figures 7.9 and 7.10.) where the final value for HM is much higher. KMLM is superior to the other implementations in eight out of nine datasets, HM shows slightly more favourable results on dataset **h** (Figure 7.11). These results imply that for some data sets KMLM is more effective at locating the tail ends of $\mathcal{S}^*$.

KMA shows the worst performance in the majority of datasets when assessed by IGD. This shows that KMA does not locate the solutions at the tail ends of the Pareto front, so most of the solutions it identifies are compromise solutions in the middle of the front and possibly quite similar.

In the final Pareto front generated for each dataset (Table 7.4) the IGD values of HM and KMLM are much lower than the values of KMA and RM. This implies HM and KMLM are better at converging their Pareto fronts towards $\mathcal{S}^*$ than RM

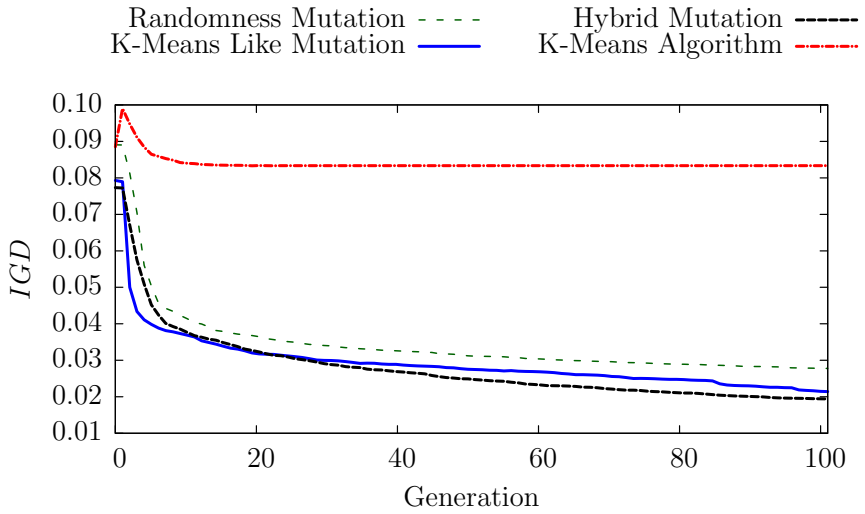


Figure 7.11: Change in IGD for Dataset **h** over 100 generations

and KMA.

We performed a Friedman test on this data and obtained a result of 93.25 so the results for the different representations are statistically different to one another. We performed a post-hoc test using the Nemenyi test and found that KMLM is statistically different from RM and KMA. We also see that HM and KMA are also statistically different. These results are slightly different to the results we observed of GD as HM is not statistically different from RM but it is statistically different to RM when measured with IGD. When compared to HM RM seems to perform better when measured with GD and KMLM performs better when measured with IGD. This would suggest that RM has found the solutions in the middle of the optimal Pareto front whereas HM has found the tail ends of the Pareto front.

7.4.3 Spread

The Spread of a set of solutions is an indicator of how well separated a set of solutions are and how near these solutions are to the extreme ends of the optimal Pareto front. We defined this previously in Section 4.3.3. Lower values of spread are desirable. Generally we observe that the values of the Spread reduce as the algorithms are

Table 7.4: *IGD* of the final Pareto front (best results highlighted)

Data Set	RM	KMLM	HM	KMA
a	0.0503	**0.0111**	0.0126	0.0485
b	0.0550	**0.0075**	0.0100	0.0976
c	0.0553	**0.0092**	0.0141	0.0717
d	0.0285	**0.0080**	0.0089	0.0796
e	0.0365	**0.0028**	0.0219	0.0377
f	0.0554	**0.0011**	0.0380	0.0647
g	0.0102	**0.0066**	0.0088	0.0203
h	0.0278	0.0214	**0.0194**	0.0833
i	0.0378	**0.0159**	0.0252	0.0845

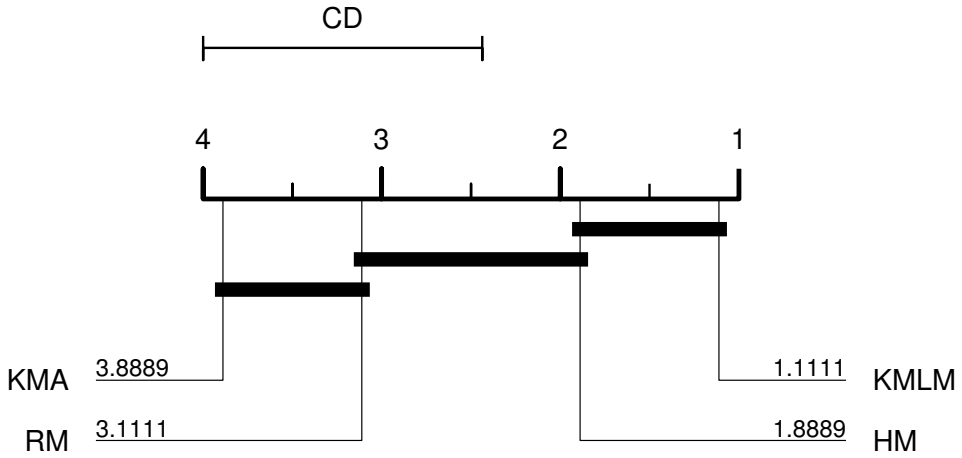


Figure 7.12: Critical difference diagram for *IGD*

executed. This is shown in Figures A.28 through A.36 and Table 7.5.

The solutions generated by RM become well Spread quickly in most cases and then stabilise. Dataset **g** (Figure 7.13) shows some fluctuation in the value of Spread as the algorithm progresses, but very erratic and unusual behaviour is observed on all of the implementations we are investigating, so this is probably related to a property of the dataset. For dataset **e** (Figure 7.14) there is no improvement or change in the value of Spread as the RM algorithm progresses. These results imply that the solutions generated using RM are distinct from each other and show a consistent tradeoff in the values of the objective functions. The solutions must also be fairly close to the tail ends of the optimal Pareto front to obtain low values

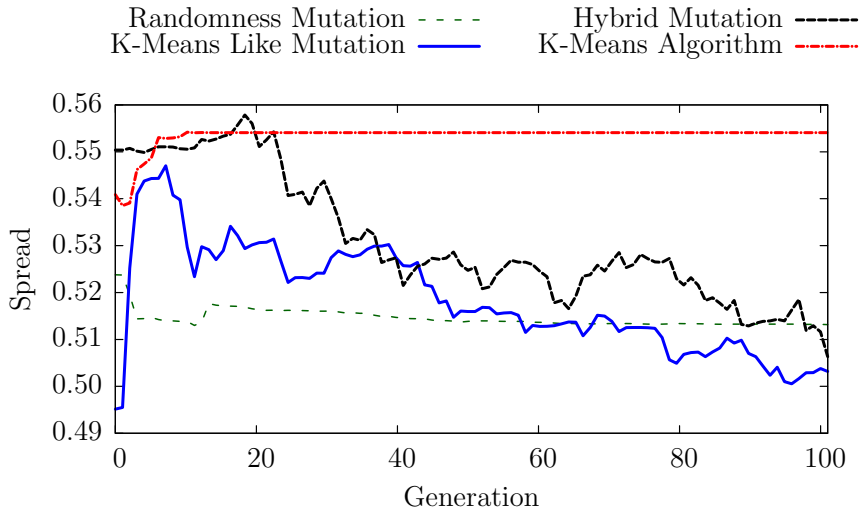


Figure 7.13: Change in Spread for Dataset **g** over 100 generations

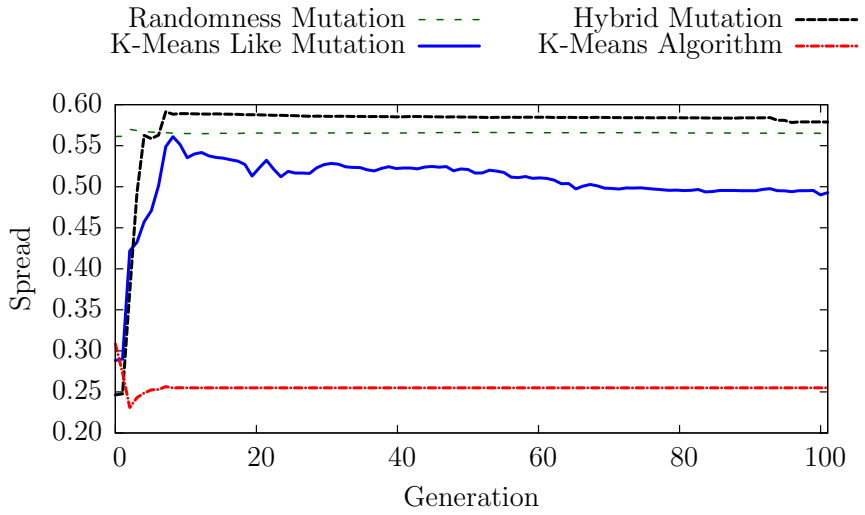


Figure 7.14: Change in Spread for Dataset **e** over 100 generations

of Spread. Previously, when measuring with IGD, our results implied that the solutions generated may be far from the tail ends of the optimal Pareto front. This suggests that the solutions are evenly spaced and near the tail ends of the optimal Pareto front and converge towards the middle of the optimal Pareto front as the algorithm executes.

For most of the datasets we studied the behaviour of HM and KMLM appeared

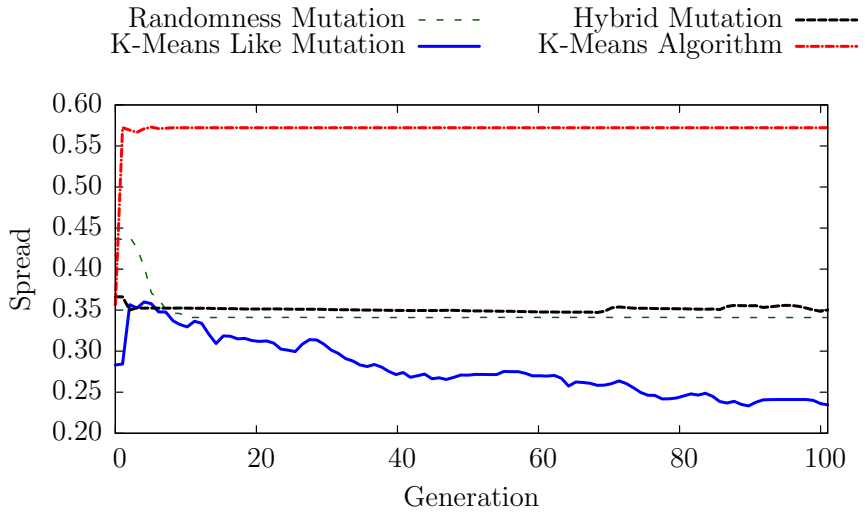


Figure 7.15: Change in Spread for Dataset **f** over 100 generations

to be similar as the algorithm progressed, but were very different at the start of execution as you would expect, because HM becomes KMLM as it progresses. KMLM showed low values of Spread before getting worse whereas HM showed high values of Spread before improving. The behaviour of KMLM and HM then converges after approximately ten generations as the behaviour of the mutation operators become more similar. The behaviour of HM initially is quite similar to RM because it is initially based on RM. RM seems to be encouraging well Spread solutions to be found, whereas KMLM is encouraging solutions to be very similar to each other. Unusually, for dataset **f** (Figure 7.15) the solutions found using KMLM become much more spread (lower values of the Spread measure) than those found using the other implementations. There does not appear to be a clear reason for this. KMLM and HM exhibit erratic behaviour in the case of dataset **g**. This was also seen for RM, so it must be caused by the structure of the dataset.

Overall, solutions found using KMA are generally less spread (larger values of the spread measure) than those generated using any of the NSGA-II implementations (Table 7.5). This implies that KMA produces solutions that are very similar to each other. Given that the algorithm is progressing until one objective is minimised this is to be expected. RM is producing sets of solutions that are generally very well

Table 7.5: Spread of the final Pareto front (best results highlighted)

Data Set	RM	KMLM	HM	KMA
a	**0.0851**	0.1176	0.1094	0.1959
b	**0.1050**	0.1820	0.2045	0.5333
c	**0.2281**	0.3048	0.3126	0.4147
d	**0.1254**	0.2157	0.2119	0.4506
e	0.5651	0.4928	0.5788	**0.2550**
f	0.3410	**0.2346**	0.3502	0.5722
g	0.5132	**0.5032**	0.5063	0.5541
h	**0.1637**	0.2307	0.2047	0.4646
i	**0.2104**	0.2502	0.2503	0.5616

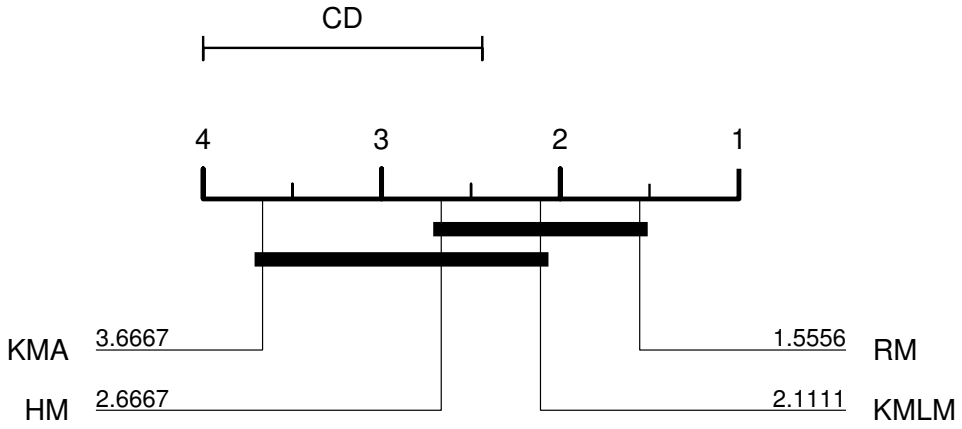


Figure 7.16: Critical difference diagram for Spread

spread compared to the sets of solutions generated using the other implementations. This shows that RM is good at exploring the objective space. RM obtained the best results in eight of the nine datasets.

Again, we assessed these results using the Friedman test and obtained a result of 7.58 which shows statistical difference. We report the results of the post-hoc test in Figure 7.16. We can see that RM is statistically different to KMA, the other pairs are not statistically different. This confirms that RM has performed better than KMA when measured with the Spread measure.

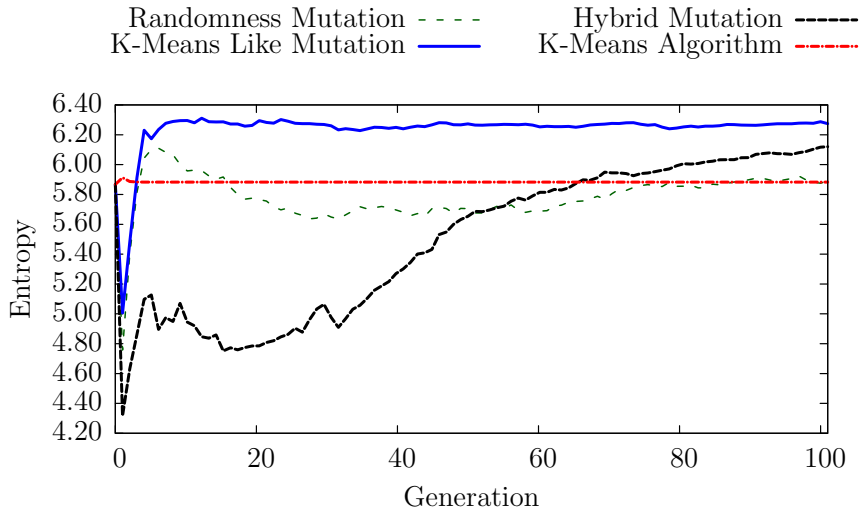


Figure 7.17: Change in Entropy for Dataset **a** over 100 generations

7.4.4 Entropy

A large Entropy indicates that the Pareto front is more diverse, there are less duplicate solutions and, where there are duplicates, there are less copies of them. We previously fully defined Entropy in Section 4.3.4.1. Generally the starting populations are initially diverse then the sets of solutions immediately become significantly less diverse before quickly regaining their diversity, this is shown in Figures A.37 through A.44 A.45.

The solutions found by KMA are the least diverse. The lack of diversity is not surprising as the algorithm has only one objective, so the solutions are all drawn towards on ideal solution. The Entropy of the populations maintained by the other techniques tends to initially rise quickly before stabilising.

The sets of solutions found using KMLM are generally very diverse. The changes in the diversity during execution of the algorithm are less significant than the changes observed when using HM and RM. This implies that the populations maintained by the implementations using HM and RM are changing frequently. For datasets **a** and **b** (Figures 7.17 and 7.18) the diversity of the population becomes significantly lower and takes a large amount of iterations of the algorithm to improve for the HM

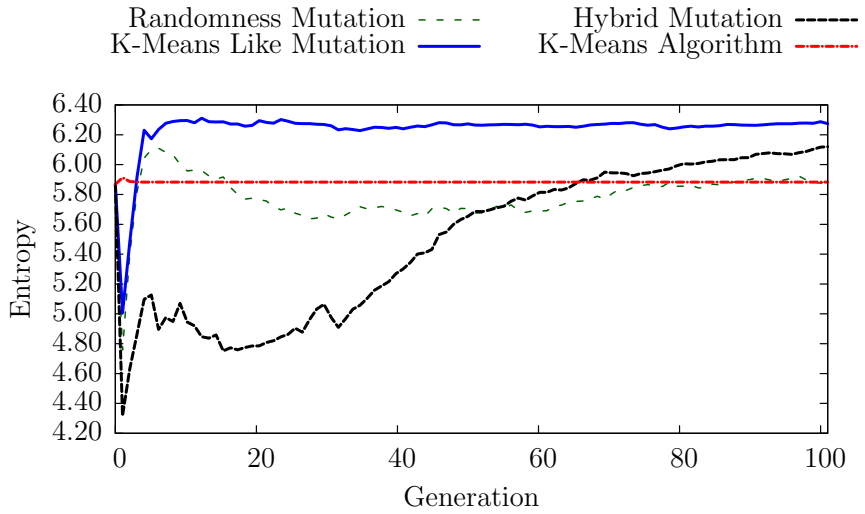


Figure 7.18: Change in Entropy for Dataset **b** over 100 generations

Table 7.6: Entropy of the final Pareto front (best results highlighted)

Data Set	RM	KMLM	HM	KMA
a	5.8897	**6.2744**	6.1201	5.8831
b	5.8445	**6.0879**	6.0583	6.0375
c	**5.6126**	5.5089	5.3285	4.9315
d	**5.6671**	5.4663	5.5959	4.8916
e	5.0365	**5.6119**	5.1939	4.5209
f	4.4439	**5.1797**	4.8716	4.0788
g	5.0758	**5.5078**	5.3445	4.3125
h	4.9235	**5.3053**	5.1261	4.0508
i	4.9646	**5.6359**	4.7627	4.4176

representation. This behaviour is not observed in the other implementations. These results imply that there are some duplicate solutions in the populations maintained by the implementations using HM and RM.

Table 7.6 shows the Entropy of the Pareto fronts generated using implementations of NSGA-II were higher at the final Pareto front than those found using KMA. For seven of the nine datasets the sets of solutions found using KMLM were the most diverse, followed by HM in most cases. This implies that using KMLM as the mutation operator leads to sets of solutions that contain unique solutions.

The value of the Friedman test was calculated as 15.55 which means that the rep-

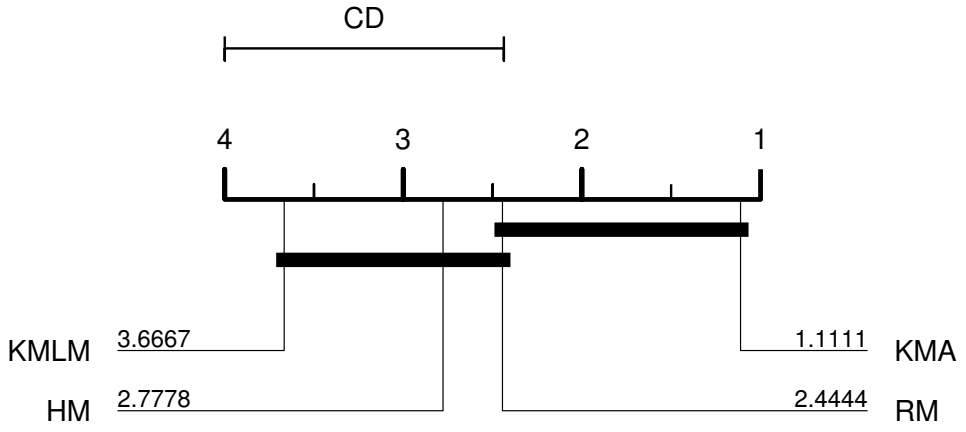


Figure 7.19: Critical difference diagram for Entropy

resentations show statistically different performance to one another. We performed a post-hoc test and report out results in Figure 7.19. Our results show that KMA is significantly different from HM and KMLM, the other pairs are not statistically different. This confirms that the solutions found by KMA were less diverse than those found by KMLM and HM.

7.4.5 Average Rand Index

The Rand Index was introduced in Section 2.4. The Rand Index differs from the previous measures of quality as it does not assess the quality of the Pareto front. It assesses the quality of individual clusterings with respect to some pre-defined ideal clustering. As we are interested in finding techniques that deliver high quality Pareto fronts and high quality clustering solutions this is therefore useful. High values close to 1 indicate that a clustering solution is similar to a pre-defined ideal clustering solution, whereas values close to 0 show that it is not similar. Here we averaged the similarity of each solution in a Pareto front to determine how similar each front is to an ideal solution. The similarities that we observed varied from dataset to dataset significantly, this is show in Figures A.46 through A.54 and Table 7.7.

The solutions found using KMA quickly converge steadily towards the intended clustering solution and do not continue to change as the algorithm has converged.

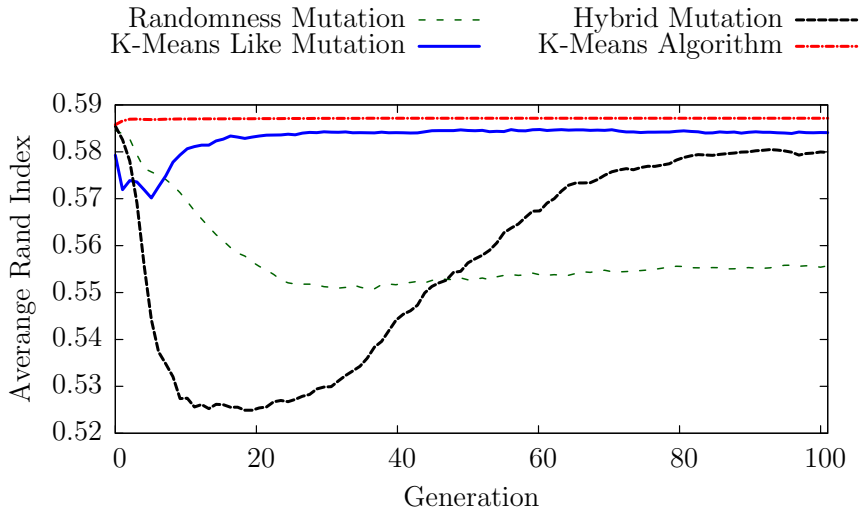


Figure 7.20: Change in Average Rand Index for Dataset **a** over 100 generations

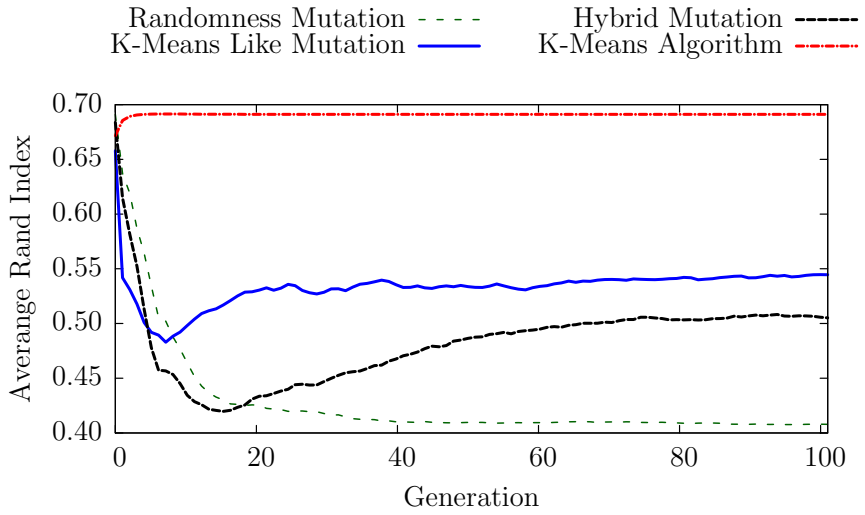


Figure 7.21: Change in Average Rand Index for Dataset **c** over 100 generations

The results we observed were good for datasets: **a**, **b**, **c**, **f**, **g** and **i**. This implies that the solutions generated using KMA were often close to the solution that we intended to discover.

KMLM does not show consistent behaviour, for datasets **a** (Figure 7.20), **b** and **e** the performance of KMLM is very similar to KMA. For datasets **c** (Figure 7.21), **f**, **g** and **i** the solutions found become more dissimilar rapidly before steadily improving

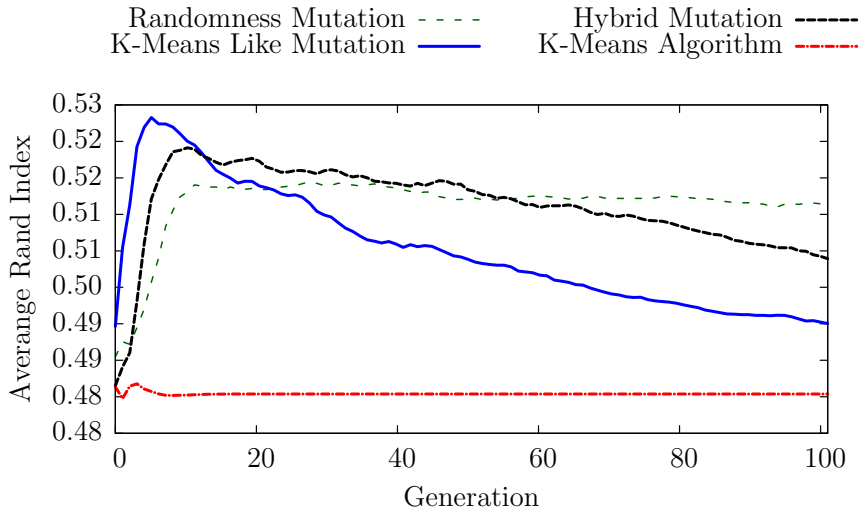


Figure 7.22: Change in Average Rand Index for Dataset **h** over 100 generations

as the algorithm executes. In the case of dataset **h** (Figure 7.22) the similarity to the intended solution increases initially before steadily declining as the algorithm executes, this change in value is small but noticeable.

The sets of solutions found by RM either start similar to the intended solution and then become less similar as the algorithm progresses, as demonstrated in the case of **a**, **b**, **c**, **f** and **i**; or they start dissimilar to the intended solution and gain similarity as the algorithm progresses as in **d**, **e**, **g** and **h**. There appears to be no obvious explanation for this phenomena. Data sets **g** and **h** have been visualised in Figure 7.1 and it can be seen that they are very different to each other, we would have expected a difference between these data sets and the others. These results imply that clusters have been formed in data set **g** that cover more than one of the clusters in the original solution which seems very unintuitive.

The behaviour shown by HM tends to be similar, though slight worse, in comparison to the behaviour exhibited by KMLM. In the case of datasets **a** and **b** the similarity gets significantly worse early on. This behaviour is not exhibited by KMLM or RM or upon any other datasets, so this may be an anomaly.

Generally there does not seem to be a best operator arising from the experiments

Table 7.7: Average Rand Index in the final Pareto front (best results highlighted)

Data Set	RM	KMLM	HM	KMA
a	0.5557	0.5841	0.5799	**0.5872**
b	0.7111	0.7137	**0.7439**	0.5544
c	0.4079	0.5446	0.5052	**0.6912**
d	**0.5127**	0.5123	0.5124	0.5098
e	0.7920	0.7945	**0.8220**	0.7777
f	0.4315	**0.7993**	0.6866	0.7852
g	0.7971	0.8131	0.8161	**0.8299**
h	**0.5114**	0.4950	0.5039	0.4854
i	0.5634	0.5924	0.5671	**0.6191**

in terms of individual solution quality. The solutions found by KMA are often more similar to the intended solution, with KMA obtaining the best result in four out of the nine datasets.

We performed the Friedman test on this data and we cannot say that any one representation is statically different to any of the others. We obtained a value of 0.95 which is not statistically significant at the levels we are testing so we have not performed a post-hoc test. We cannot say that any representation is statistically better than another representation when assessed with the Rand Index.

7.5 Summary & Conclusions

We have investigated the performance of mutation operators that promote random changes, solution quality and a hybrid combination. We have compared their performance against a benchmark, the k-means algorithm for clustering. We have also shown how to investigate the performance difference between these techniques using a variety of measures of performance that assess Pareto front quality and individual solution quality. Part of our contribution is to have created an experimental set-up for testing both individual solution quality and Pareto front quality through the execution of the algorithms. This allows for more experimentation with crossover, mutation and other multi-objective algorithm parameters.

We have performed the Friedman test on each set of results that we have generated and found that the representations were statistically different to each other for all of our results except the Rand Index. We have seen that KMLM and KMA are statistically different to each other when measured with the volume, GD, IGD and Entropy. KMA performed better when measured with Entropy and the KMLM mutation performed better for the other measures. They were not statistically different when measured with Spread. It appears that KMLM mutation is producing solutions more similar to the optimal Pareto front than KMA but KMA has produced more diverse solutions.

We have seen that KMLM performs best for Pareto front quality by a number of measures. The Pareto fronts generated dominate the largest area of the objective space; they lie on a simulated optimal Pareto front; they are closer to the extreme ends of the simulated optimal Pareto front and contain a larger proportion of unique solutions. However, we have also seen that these solutions are not always evenly spread throughout the objective space and that the similarity to the clustering solutions that we desired, that is, the quality of individual solutions, varies from data set to data set.

We have also shown that using RM produced poorer Pareto fronts that do not dominate all of the objective space, do not lie on the optimal Pareto front and exclude the extreme ends of the optimal Pareto front. However, the Pareto fronts contain a large quantity of unique and varied solutions that are evenly spaced. The mutation operator that promotes random chances appears to offer some advantages in terms of diversity and should be investigated further. We have shown that this implementation of a mutation operator designed to disrupt the solutions may not be as disruptive as we desired.

We postulate that KMLM is performing some form of local search on specific solutions which leads to faster convergence to the optimal Pareto front. We also note that the use of KMLM does not lead to the diverse Pareto front found by RM even though it converges better to the optimal Pareto front than KMLM.

HM, as we implemented it, did not represent a good improvement with respect to KMLM. This may be because our initial assumption that the random operator would promote better Pareto fronts, whereas KMLM may provide some form of local optimisation of solutions did not prove correct with our implementation of the random operator. A better operator or combination of operators that consistently delivers diverse and improved Pareto fronts may be needed.

As we have produced an experimental set-up which allows us to observe the quality of both the Pareto front and the individual solutions over a number of generations, other operators can now be investigated. We expect that some form of adaptive mechanism which switches the emphasis of the search from the quality of the Pareto front to the quality of individual solutions or to the diversity in the population may present advantages for this and other multi-objective problems. Our experimental set-up should allow for further parameter experimentation and eventually deliver more effective and efficient MO algorithms. In future we should also look at ways of improving the technique to identify which representation is producing very different results from the others. We performed a statistical test on the results of the values of the final front and we did not find any of these to be statistically different from one another.

Chapter 8

Conclusions and Further Work

The overall objective of this thesis was to investigate how Multi-Objective Evolutionary Algorithms can be applied to the clustering problem. First we needed to investigate ways of assessing clustering solutions. We performed an experimental study in Chapter 3 that produced a number of recommendations about possible objectives. Secondly we needed to investigate how to represent and manipulate clustering solutions in the context of MOEAs. In Chapter 5 we proposed our own algorithm using a pre-existing representation and our own operators to manipulate the solutions. In Chapter 6 we extensively investigated the performance of different representations and operators from the literature. Finally, in chapter 7 we specifically investigated the performance of mutation operators. Overall we have given advice on which objective functions can be used for this problem, how the problem can be represented in an MOEA context and how the solutions to the problem can be manipulated.

8.1 Summary & Contributions

In Chapter 2 we introduced the general clustering problem and important related concepts: data sets, distance measures, characteristics of clustering solutions, classic clustering algorithms and Clustering Quality Measures (CQM). We discussed the

difficulties of establishing what constitutes a good clustering solution to a problem. In reality, the evaluation of clustering solutions may be subjective, that is to say, the best clustering solution to a given clustering problem may be dependent on the decision maker, the intended use of the clustering solution or other factors. In this context, we proposed the use of Multi-Objective Evolutionary Algorithms (MOEA) for clustering because they allow for the incorporation of multiple objectives to be optimised and provide a set of trade-off solutions which are hopefully optimal which respect to those objectives.

In order to investigate the many CQM proposed for clustering, we designed some experiments, which are presented on Chapter 3. The results of those experiments were published in [83]. We discussed that measures could be grouped together according to the properties they measure, e.g: separation, density or connectivity. For a Multi-Objective clustering algorithm, it may be desirable to include as objectives a number of measures representing different properties. We proposed that the most useful measures should show a steady change in their value in relation to a steady change in the quality of a clustering. To test which measures may exhibit this behaviour, we produced a number of synthetic clustering solutions and deteriorated them steadily by randomly assigning objects to other clusters. We then observed how the CQM behaved. We found that connectivity based measures and some simple measures based on the separation and density of clusters, such as Overall Deviation and R-Squared, showed the most favourable behaviour. These findings informed our choice of objective functions for the Multi-Objective Clustering Algorithm described in later chapters.

We have contributed a novel Multi-Objective Clustering Algorithm (MOCA) in Chapter 5. This was published independently in [85]. We performed an initial experimentation to validate our implementation and found that our MOCA was more likely than the standard k-means algorithm to produce high quality clustering solutions.

Since other MOCAs have been defined in the literature, with different repre-

sentations and operators, in Chapter 6 we performed an experimental comparison. Instead of implementing specific algorithms, we focused on the components of the algorithms to find the best possible configuration for a MOCA. We compared a large number of possible configurations, including a number of representations combined with related mutation and crossover operators proposed by others. Furthermore, in order to assess the real power of a Multi-Objective algorithm, it is important to be able to compare not only the quality of the individual solutions produced but also the quality of the Pareto Fronts produced. To this aim, we implemented a number of measures of Pareto front quality and ranked the various configurations according to all those measures. We then discussed which representations and operators lead to the best Pareto fronts and to the best individual clustering solutions. We found that representations based on centroids or medoids generally produced good individual solutions and good Pareto fronts, whereas representations based on Label Based Integer Encoding gave poor results for both. A centroid based representation such as the one we proposed appeared to show some advantage for finding individual clustering solutions of high quality whereas the medoid based representation showed an advantage for finding good Pareto fronts. Our own implementation with the MOCA mutation operator we devised did well for both. We noticed also with this set of experiments that certain characteristics of the mutation operator could have specific effect on the search. For example, disruptive mutation operators may encourage better exploration of the search space.

Our observations on the mutation operators in Chapter 6 led us to another set of experiments that we published separately [84]. We attempted to investigate how mutation operators could be used to improve both the exploratory and exploitative phases of the search by promoting diversity of solutions in the Pareto front and quality of individual solutions. For this, we devised a mutation operator that promoted random changes in order to diversify the search; one based on one iteration of the k-means algorithm that promoted solution improvement to intensify the search; and a hybrid combination of both which was supposed to change the emphasis of the search between diversification and intensification We were able to confirm with

our results that some operators promote Pareto front quality and some operators promote clustering solution quality.

Overall, we believe that the use of MOEA for clustering is justified. We have investigated a number of Cluster Quality Measures as potential objectives for MOEAs. Measures based upon the concept of connectivity were shown to be very good at identifying good clustering solutions, i.e. those that were similar to the expected clustering solution according to the Rand Index. We believe this is because they promote solutions that contain clusters that are continuous regions. We have also recommended measures based upon the concepts of density and separation. These are simple and effective measures. The cluster quality measures that mixed different concepts are not suitable in the MOEA context as each objective in an MOEA should attempt to measure only one quality of a clustering solution.

Our investigations have also produced recommendations on how to represent a clustering solution in a MOEA context. We have found that representations based around centroids and medoids have produced the most promising results here. This implies that clustering solutions that contain clusters that are hyper-spherical have been well evaluated. It may be that clustering is inherently biased towards hyper-spherical clusters.

We have also investigated operators for manipulating clustering solutions. We found that operators that were very disruptive discovered a large number of varied clustering solutions. We also found that operators that used information about a clustering solution represented by an individual led to higher quality clustering solutions but were less capable of investigating the Pareto front. We attempted to combine these characteristics unsuccessfully.

8.2 Further Work

There is a significant number of measures of cluster quality that we have investigated but others may not have been covered in our work and may deserve further research.

Future work should continue to assess new Cluster Quality Measures as they emerge and should also focus on how the measures interact with each other when they are combined in an MOEA context.

It is worth noting that although our experiments point at centroid or medoid based representations as being better, these types of representation can lead to finding hyper-spherical clusters and may not therefore be optimal for all problems. Therefore, there is still scope for other representations such as the one use by Handl [61], which have been shown to produce good results.

Our attempt to create a hybrid operator did not provide us with good results, hence an improved algorithm which switches the search from exploration to exploitation would require further research. A more sophisticated adaptive mechanism guided by measures of solution quality would probably be necessary to deliver improved results and should be the subject of further research. It may also be possible to design an MOEA that automatically changes between solution representations, mutation operators and crossover operators as the algorithm executes to create a self-adaptive MOEA for clustering.

We believe there is also further scope for the development of specific mutation operators tailored to the clustering problem that could use a hybrid approach that is different from the one we proposed to discover high quality clustering solutions within diverse Pareto fronts. Further investigations into hybrid mutation operators may also have applications to other problems that can be solved with MOEA algorithms. We have only briefly investigated the performance of specific crossover operators, the further development of specific clustering crossover operators for the clustering problem may lead to more effective implementations.

A major limiting factor we encountered during our investigations was the execution time of MOEAs attempting to solve clustering problems. Anecdotally, the evaluation of the quality measures of a clustering solution was generally slow. The time factor may be improved with the use of more specific implementations of CQMs. Throughout our experiments, we did not create specific optimised implementations

of CQMs. An efficient implementation, for example, could retain information from one generation to the next so the values of the CQMs would need to be regenerated less often. Novel implementations of MOEAs specifically for generating clustering solutions may also be able to generate interesting results. A large number of possible representations of clustering solutions exist, not all of these have been investigated in this thesis and new representations will be developed in time. Future work should continue to investigate and propose different methods of representing clustering solutions as they emerge.

Appendices

Appendix A

Graphs for Mutation Operator Comparison

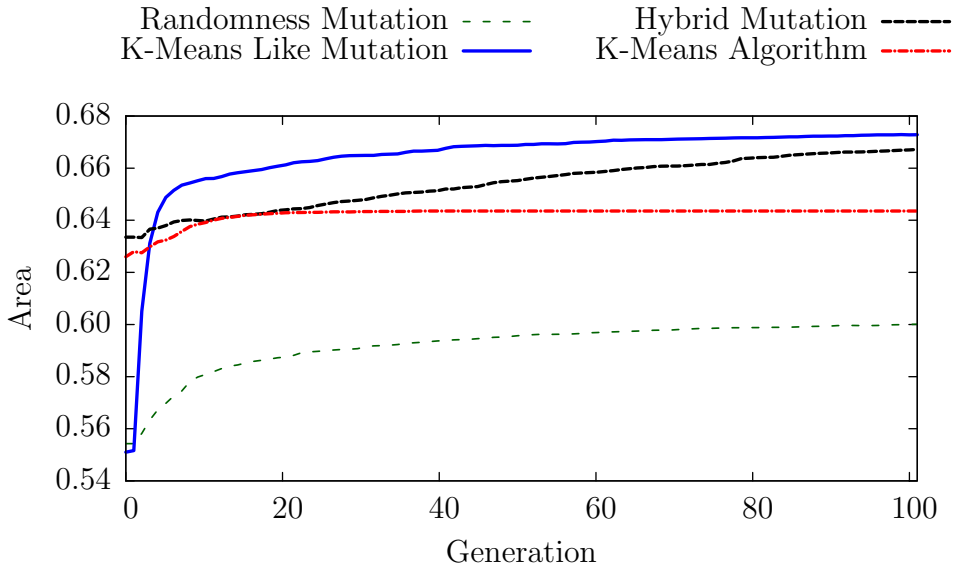


Figure A.1: Change in Volume of Dominated Space for Dataset **a** over 100 generations

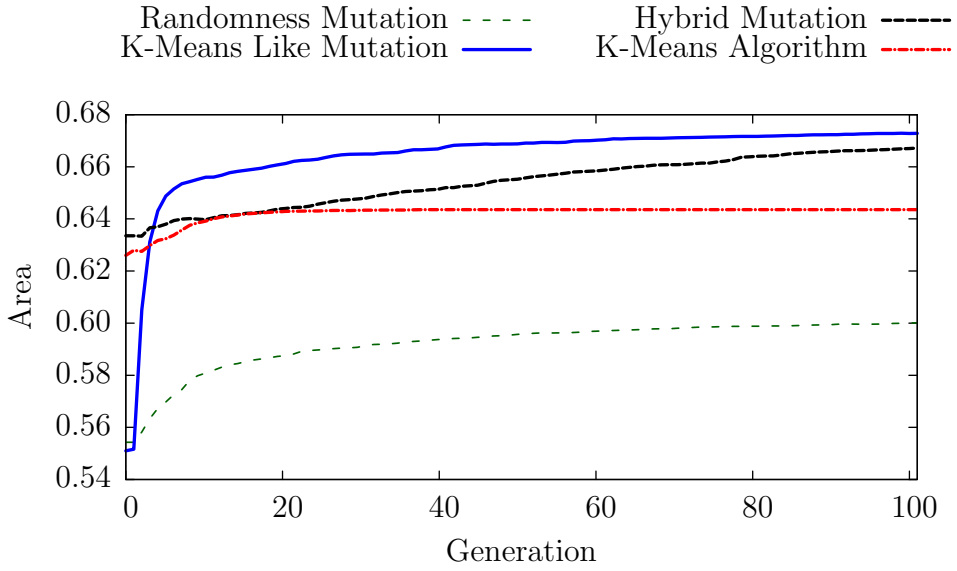


Figure A.2: Change in Volume of Dominated Space for Dataset **b** over 100 generations

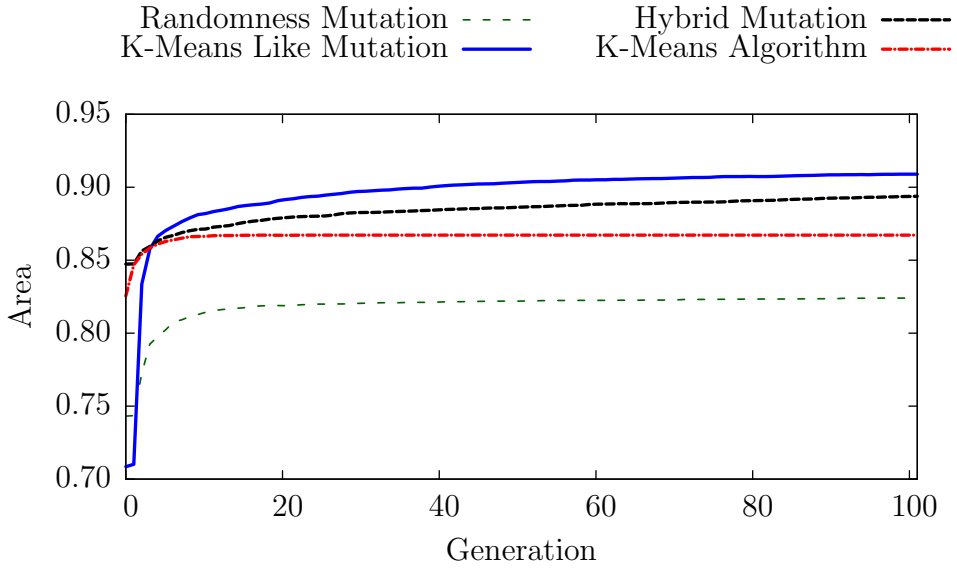


Figure A.3: Change in Volume of Dominated Space for Dataset **c** over 100 generations

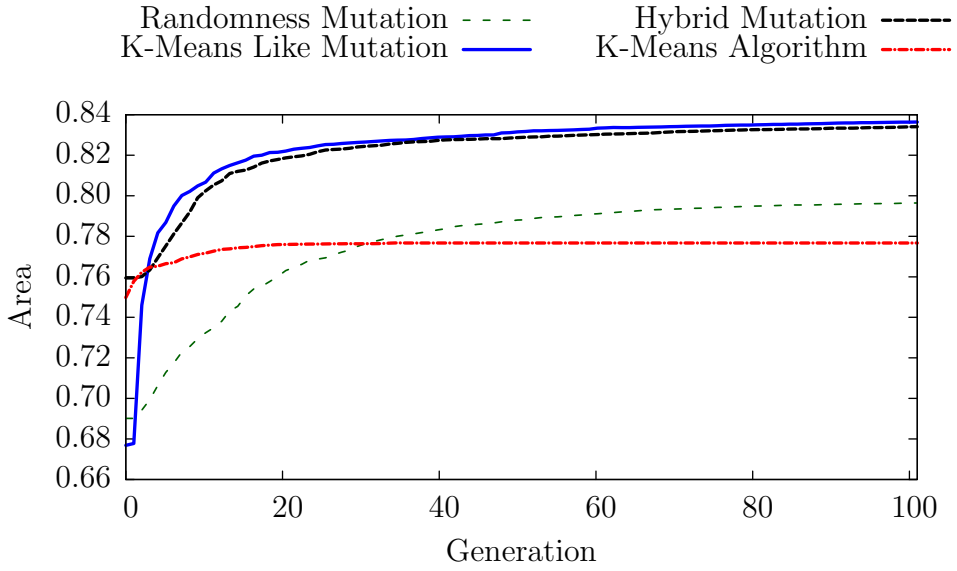


Figure A.4: Change in Volume of Dominated Space for Dataset **d** over 100 generations

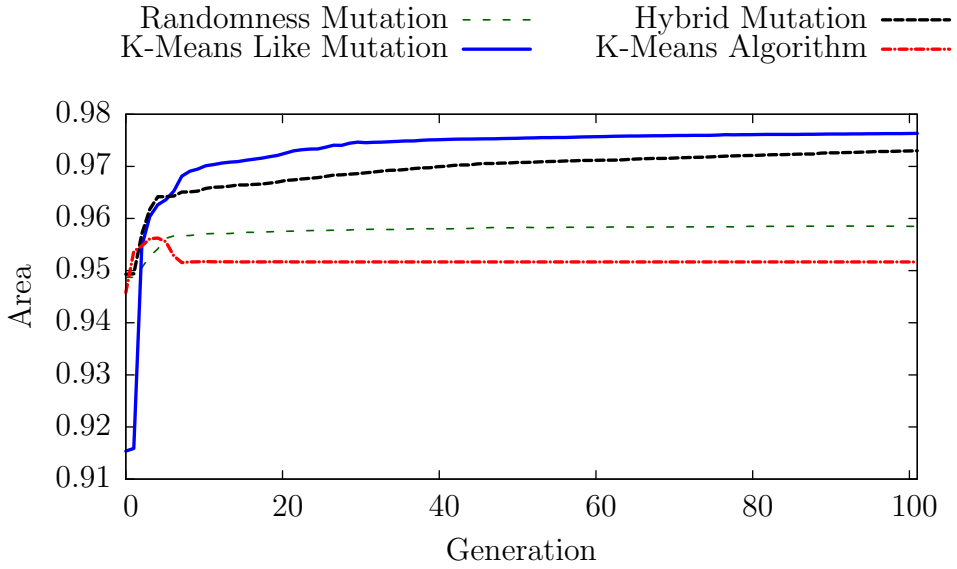


Figure A.5: Change in Volume of Dominated Space for Dataset **e** over 100 generations

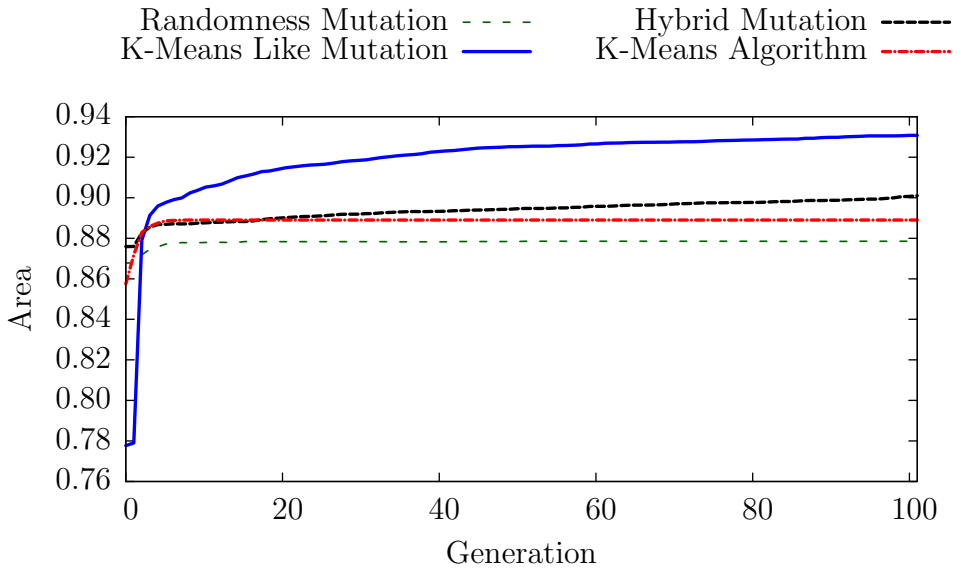


Figure A.6: Change in Volume of Dominated Space for Dataset **f** over 100 generations

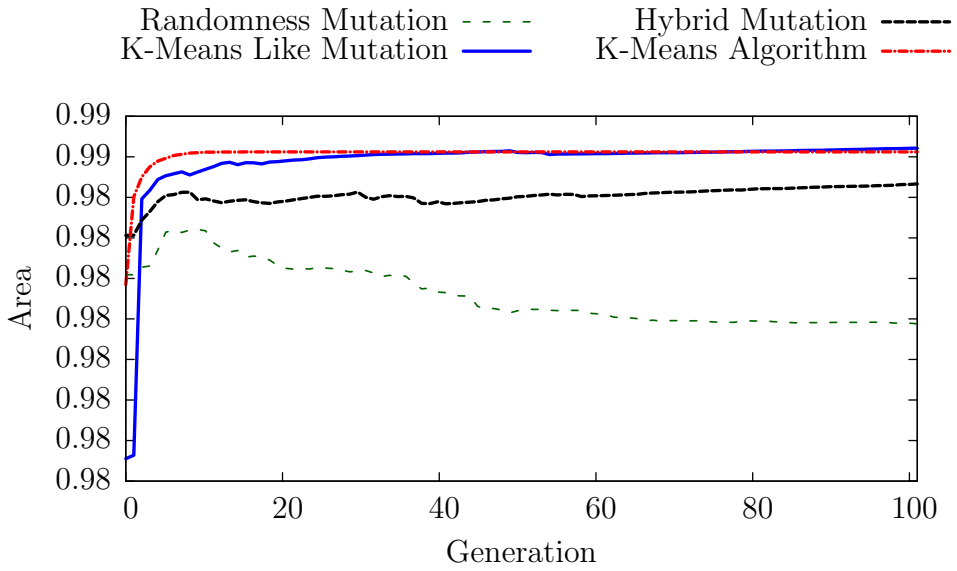


Figure A.7: Change in Volume of Dominated Space for Dataset **g** over 100 generations

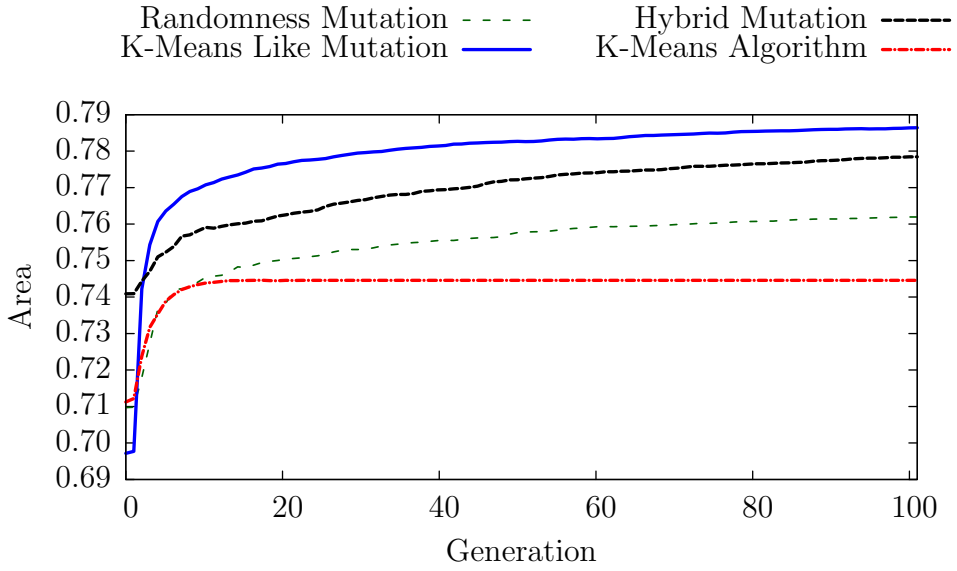


Figure A.8: Change in Volume of Dominated Space for Dataset **h** over 100 generations

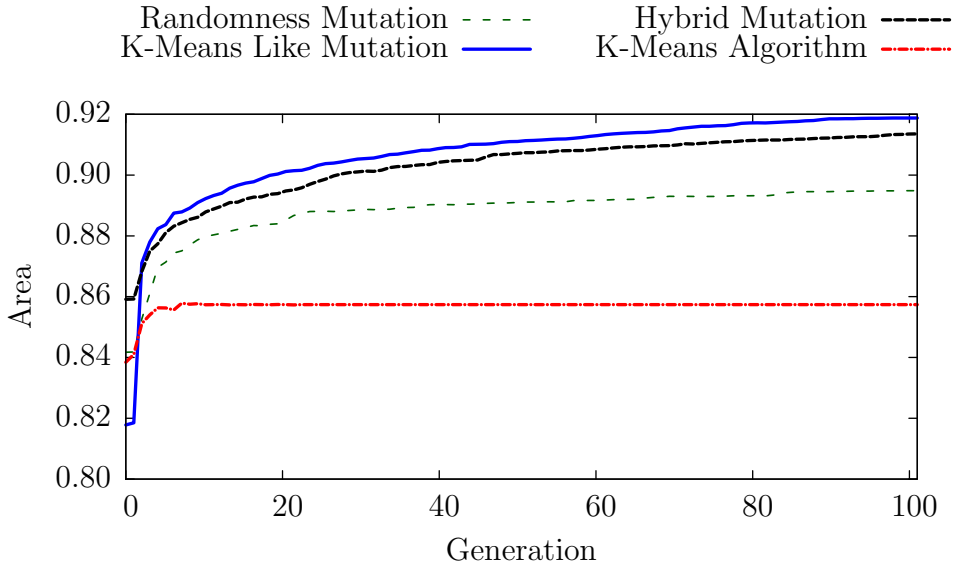


Figure A.9: Change in Volume of Dominated Space for Dataset **i** over 100 generations

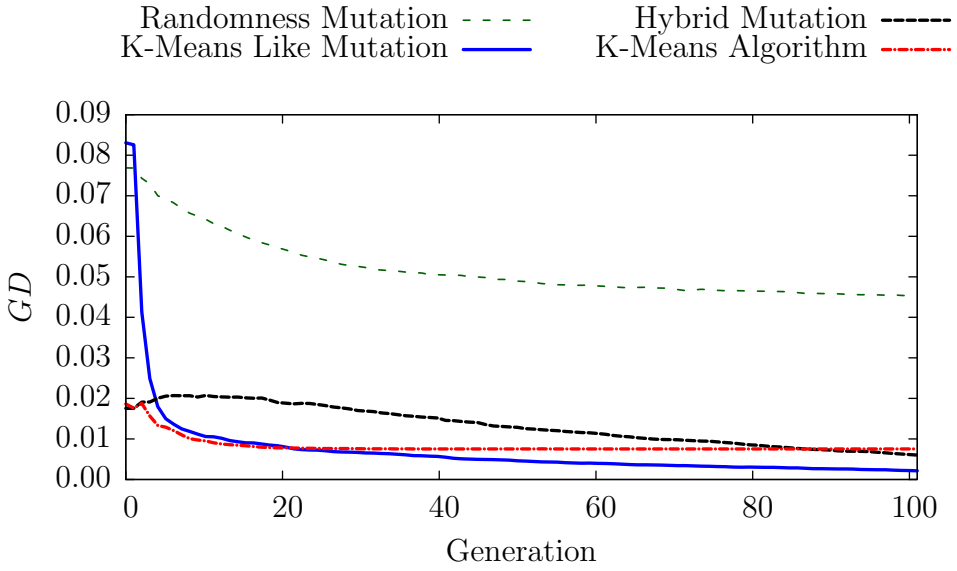


Figure A.10: Change in GD for Dataset **a** over 100 generations

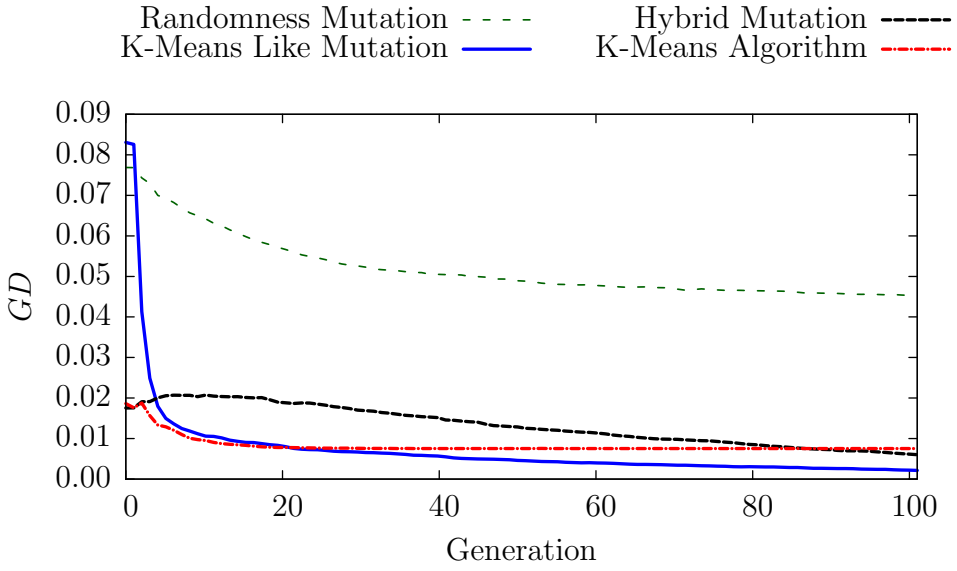


Figure A.11: Change in GD for Dataset **b** over 100 generations

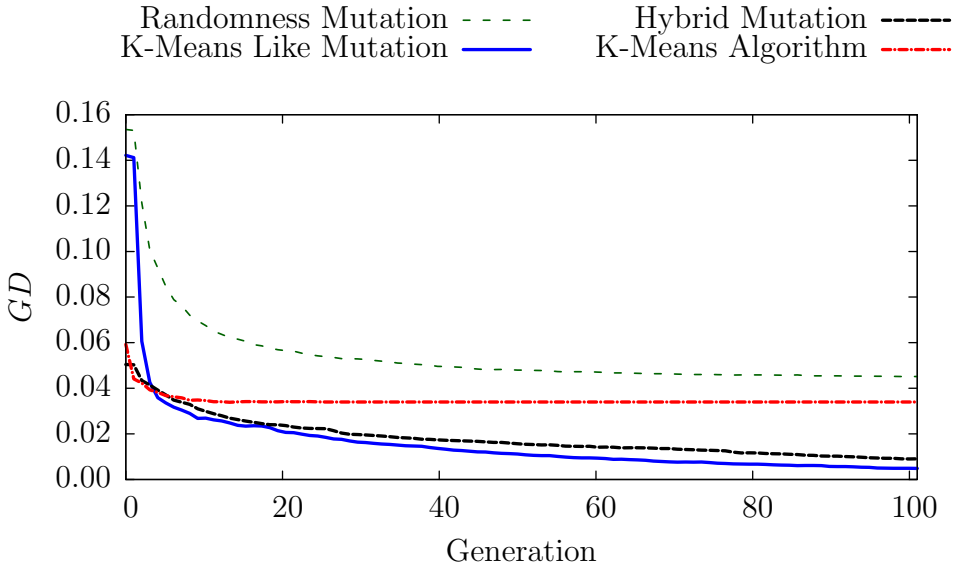


Figure A.12: Change in GD for Dataset **c** over 100 generations

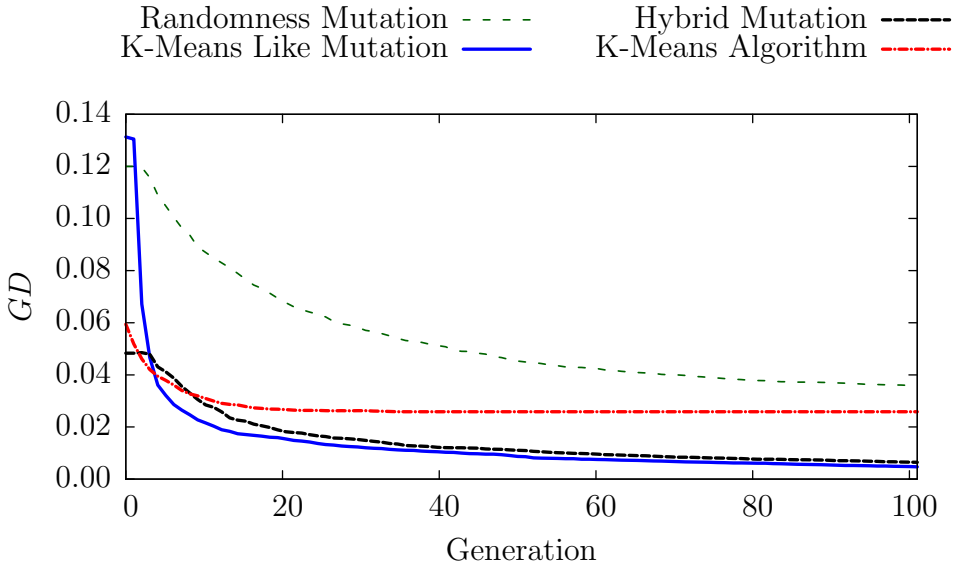


Figure A.13: Change in GD for Dataset **d** over 100 generations

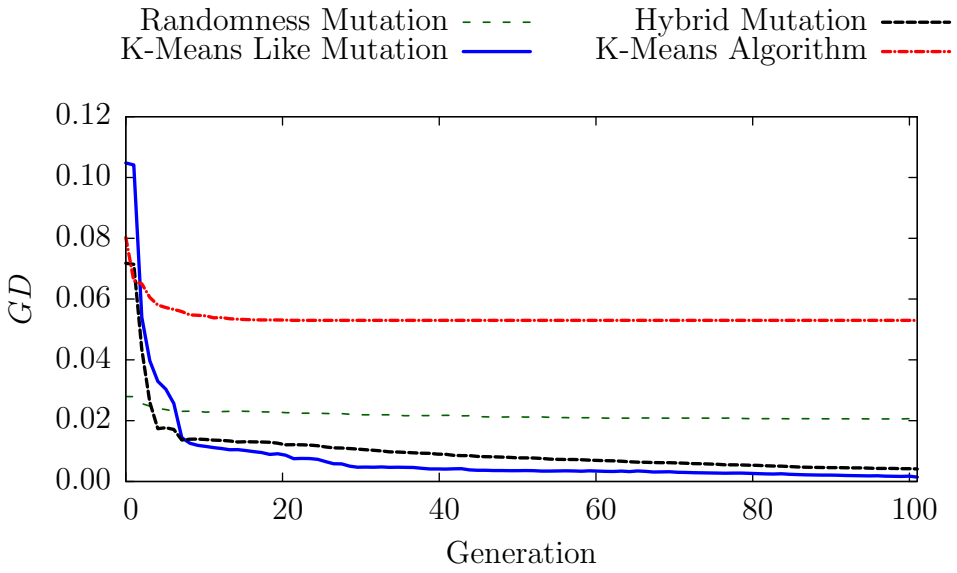


Figure A.14: Change in GD for Dataset **e** over 100 generations

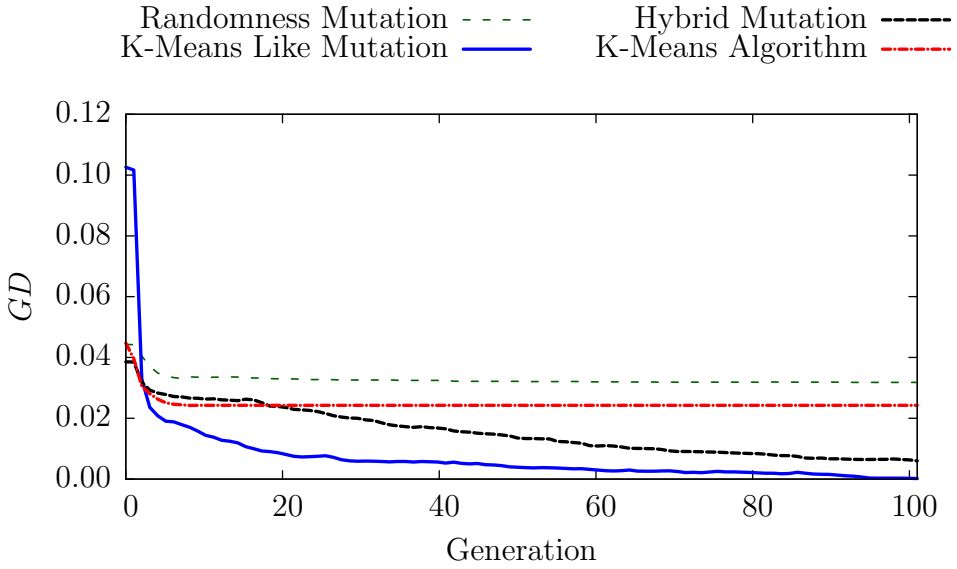


Figure A.15: Change in GD for Dataset **f** over 100 generations

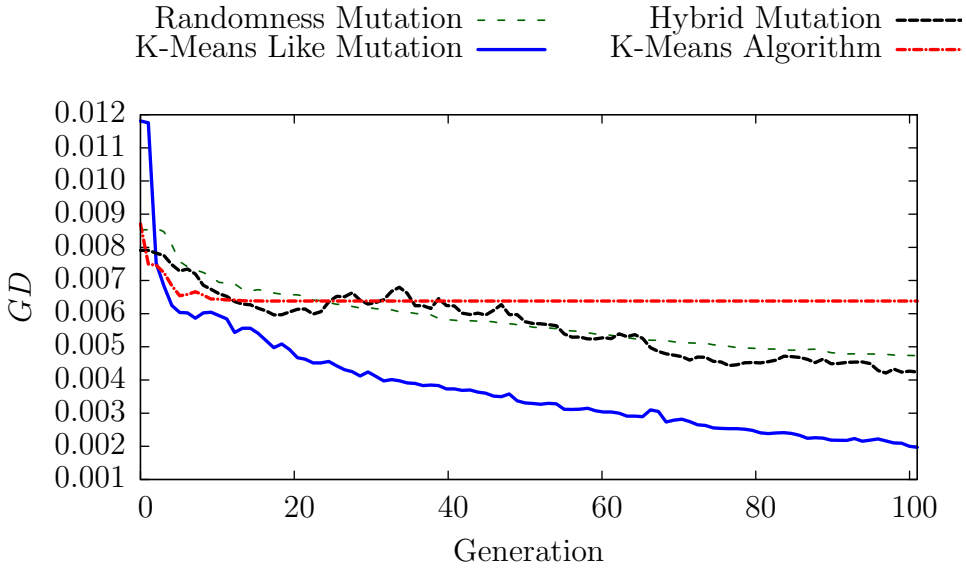


Figure A.16: Change in *GD* for Dataset **g** over 100 generations

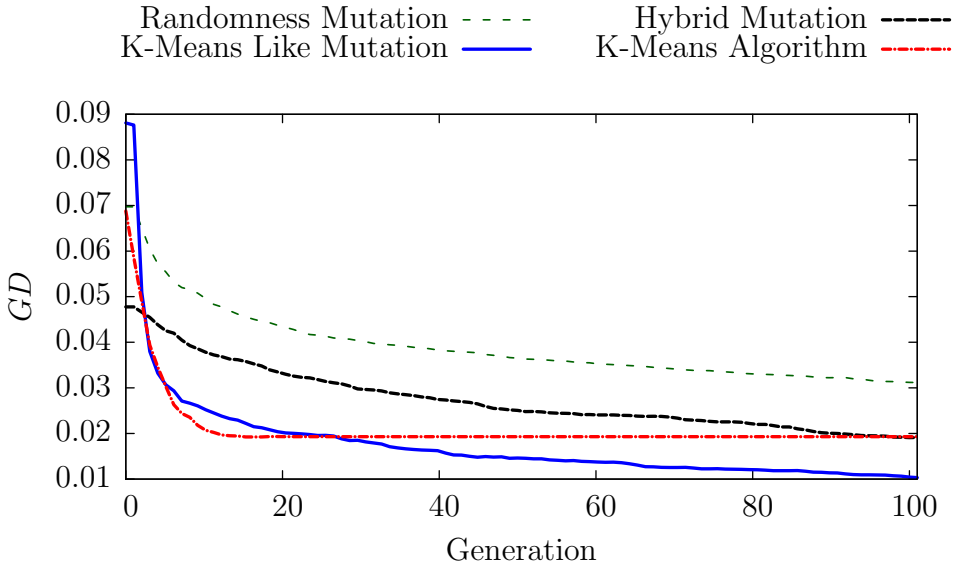


Figure A.17: Change in *GD* for Dataset **h** over 100 generations

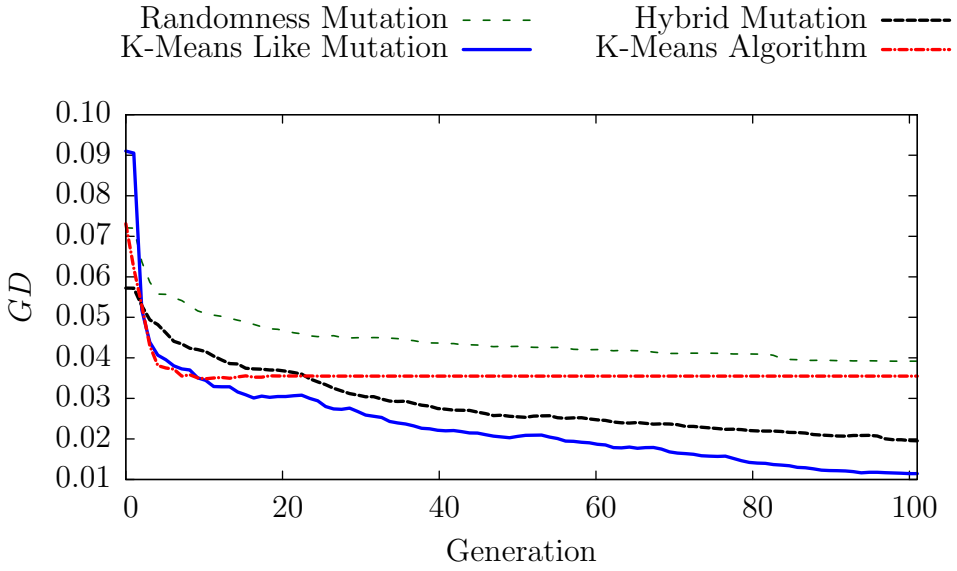


Figure A.18: Change in GD for Dataset **i** over 100 generations

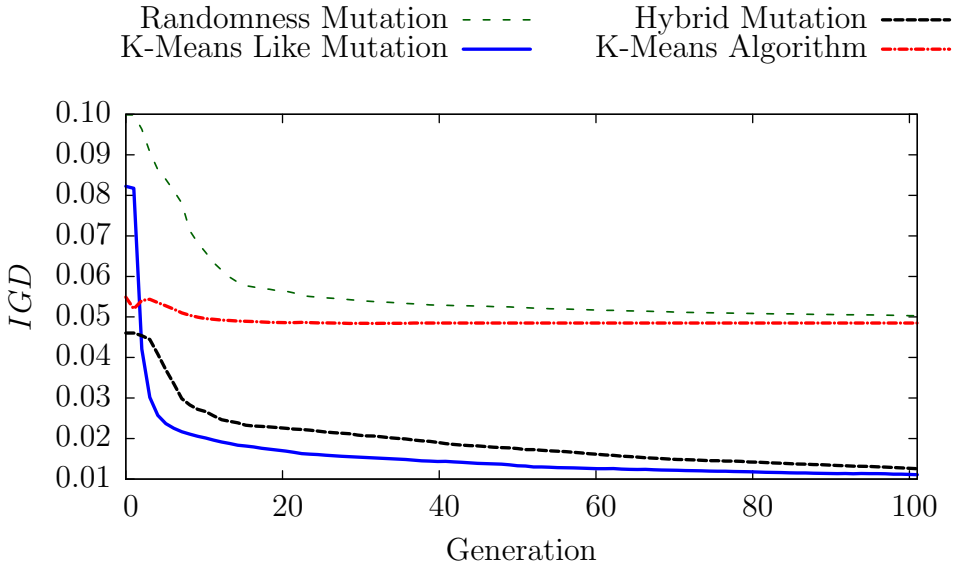


Figure A.19: Change in IGD for Dataset **a** over 100 generations

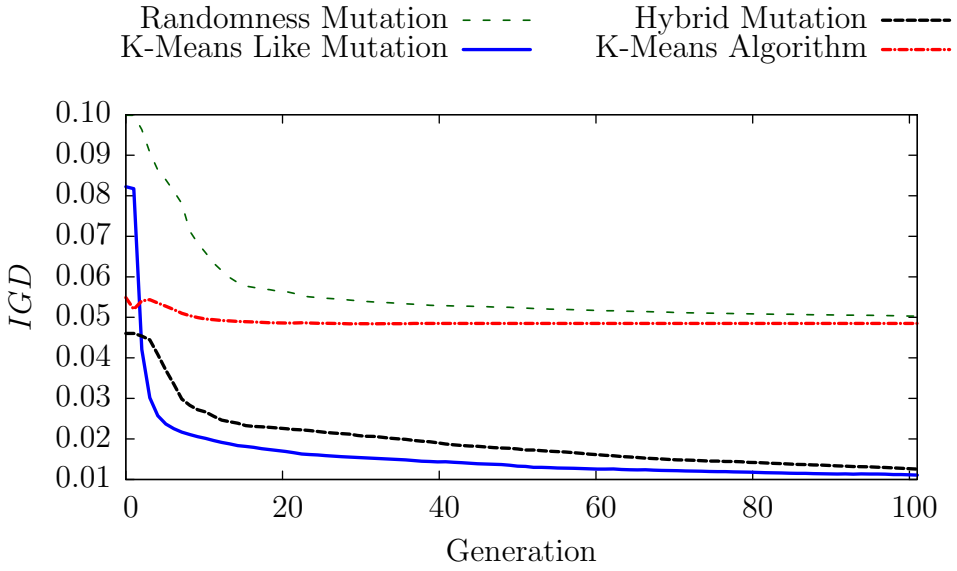


Figure A.20: Change in IGD for Dataset **b** over 100 generations

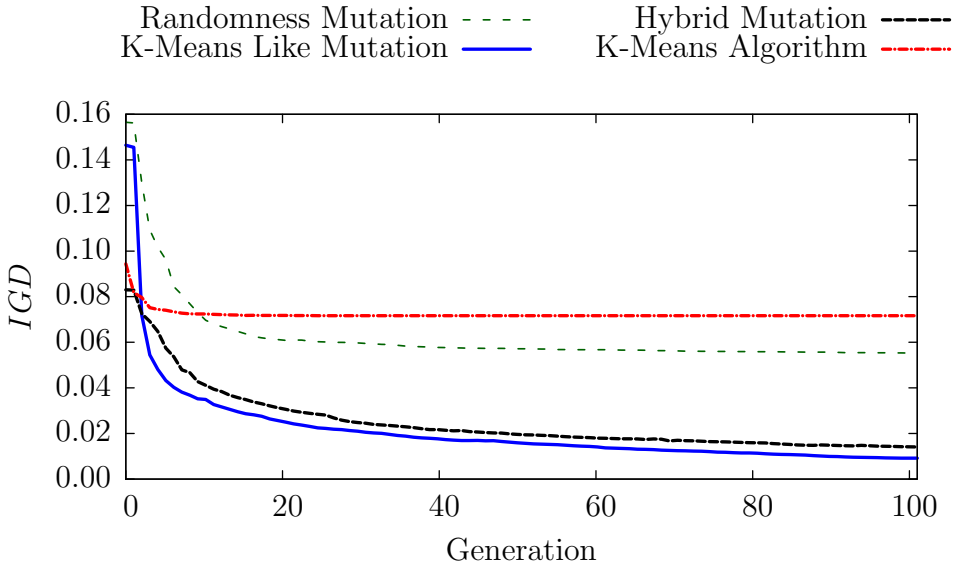


Figure A.21: Change in IGD for Dataset **c** over 100 generations

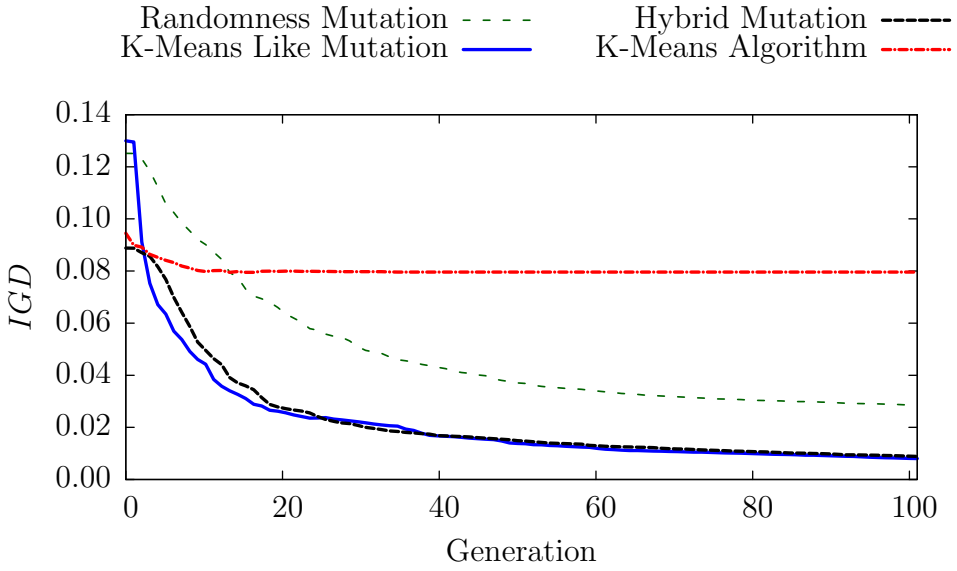


Figure A.22: Change in IGD for Dataset **d** over 100 generations

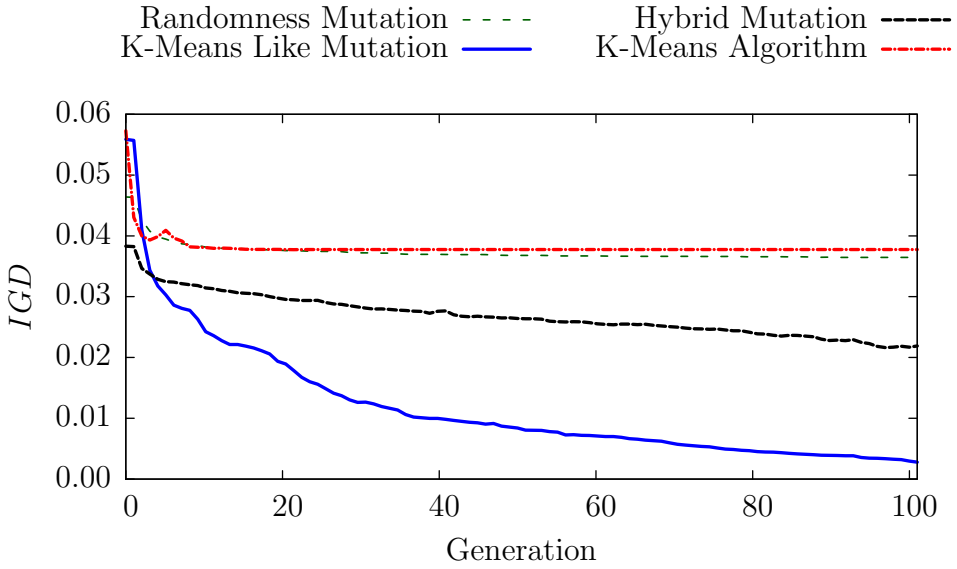


Figure A.23: Change in IGD for Dataset **e** over 100 generations

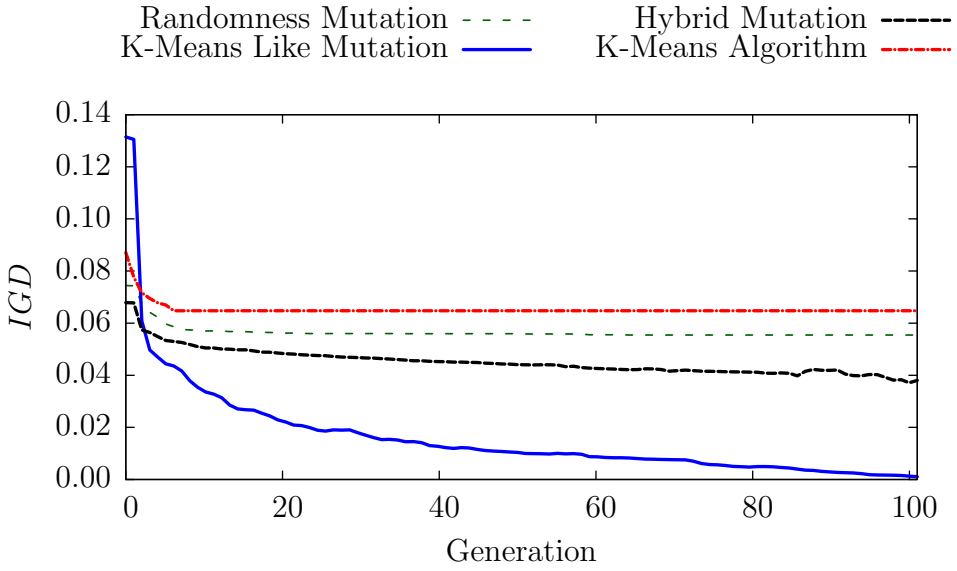


Figure A.24: Change in IGD for Dataset **f** over 100 generations

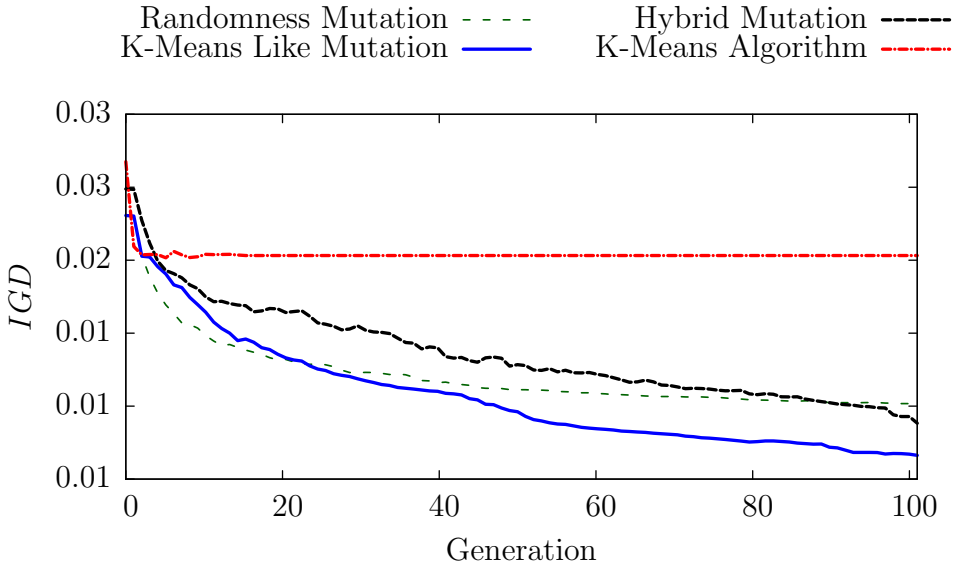


Figure A.25: Change in IGD for Dataset **g** over 100 generations

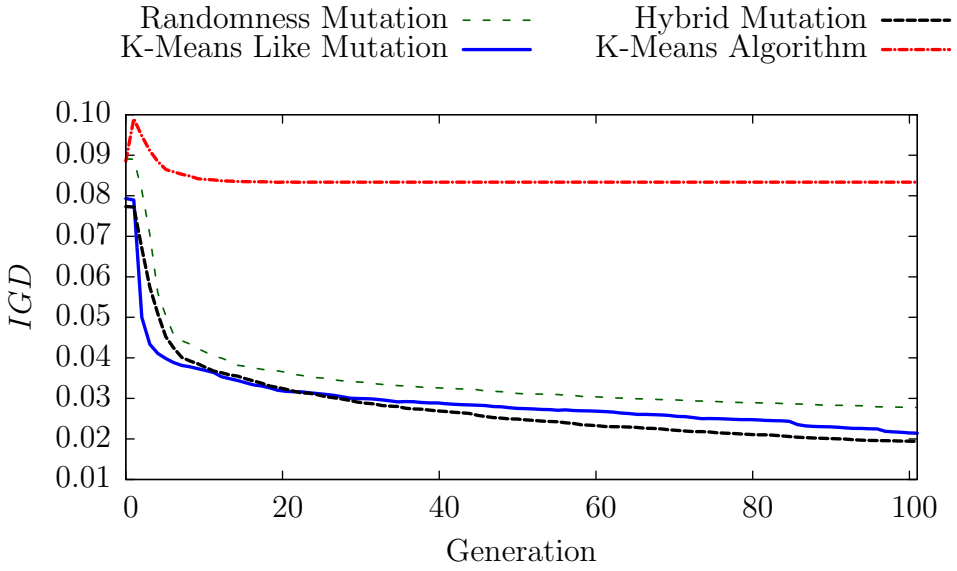


Figure A.26: Change in IGD for Dataset **h** over 100 generations

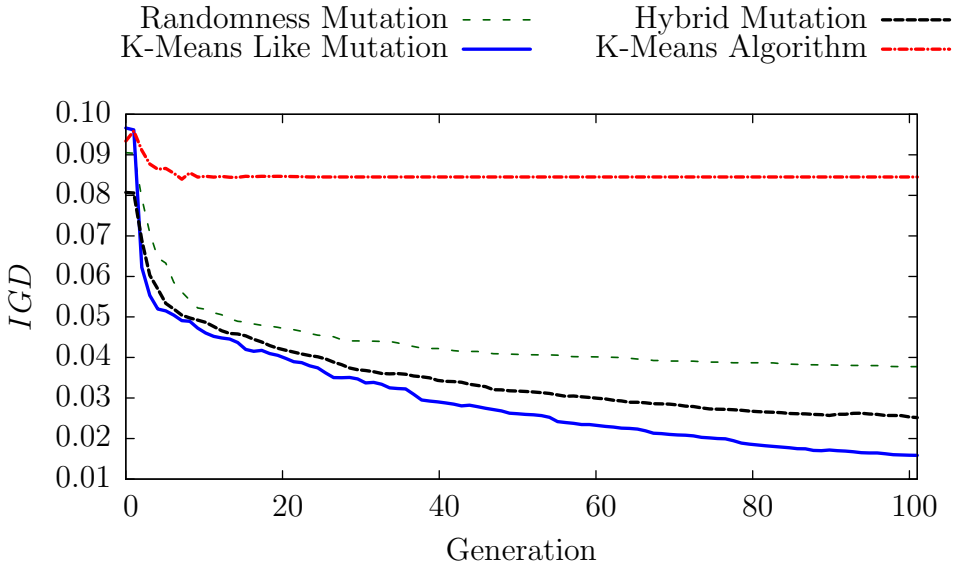


Figure A.27: Change in IGD for Dataset **i** over 100 generations

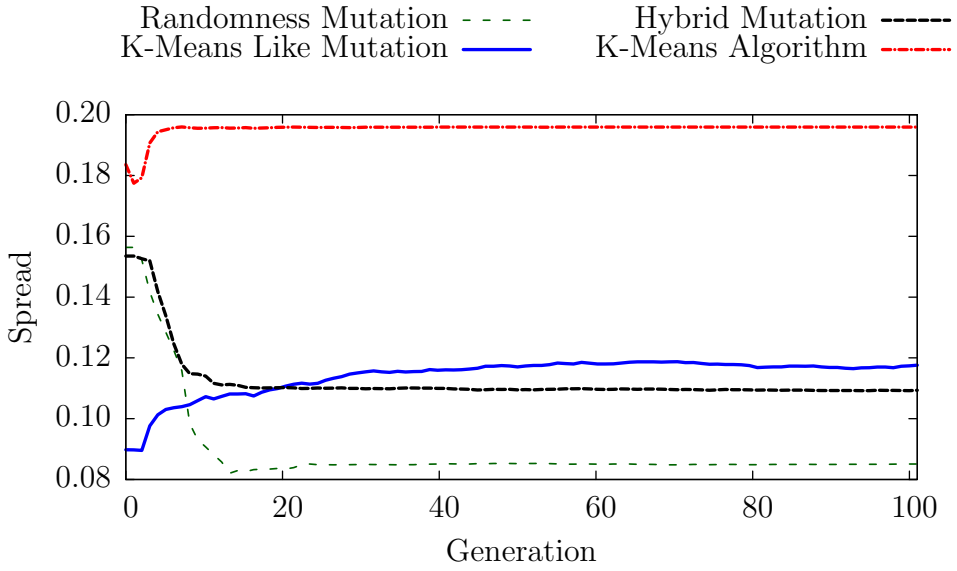


Figure A.28: Change in Spread for Dataset **a** over 100 generations

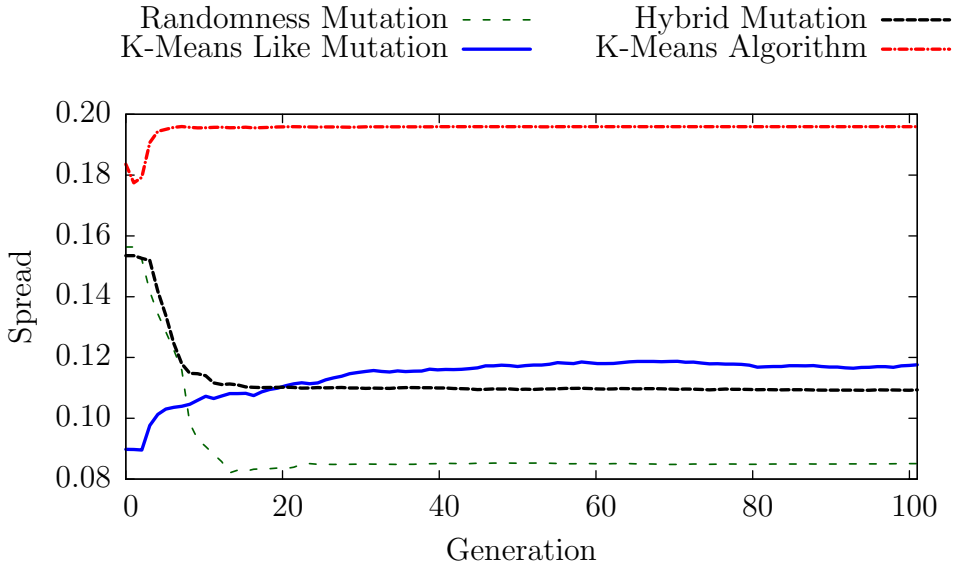


Figure A.29: Change in Spread for Dataset **b** over 100 generations

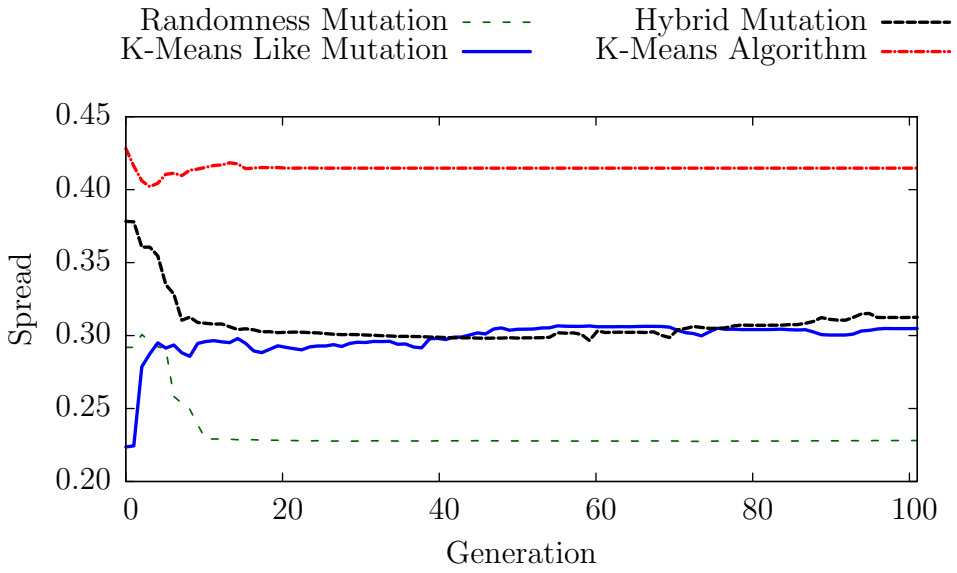


Figure A.30: Change in Spread for Dataset **c** over 100 generations

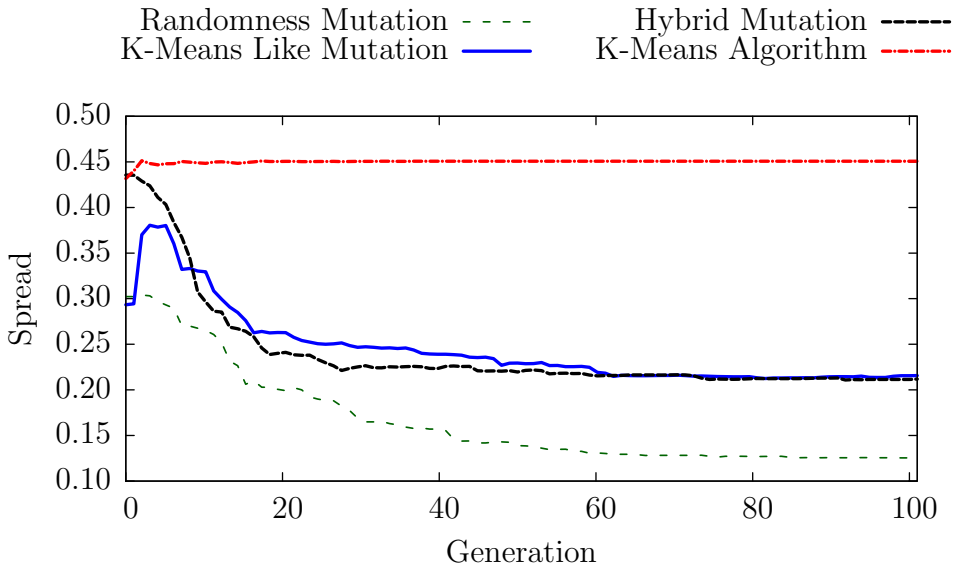


Figure A.31: Change in Spread for Dataset **d** over 100 generations

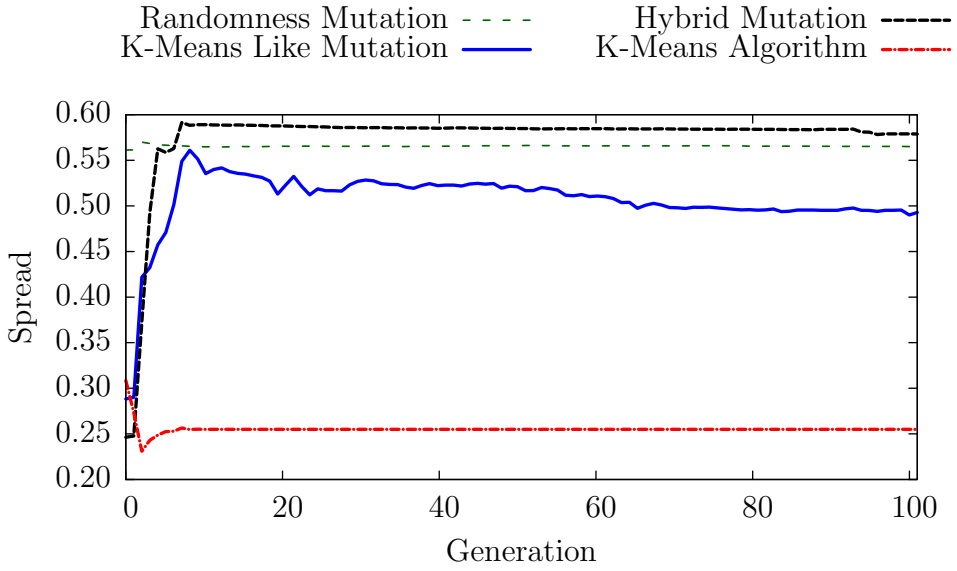


Figure A.32: Change in Spread for Dataset **e** over 100 generations

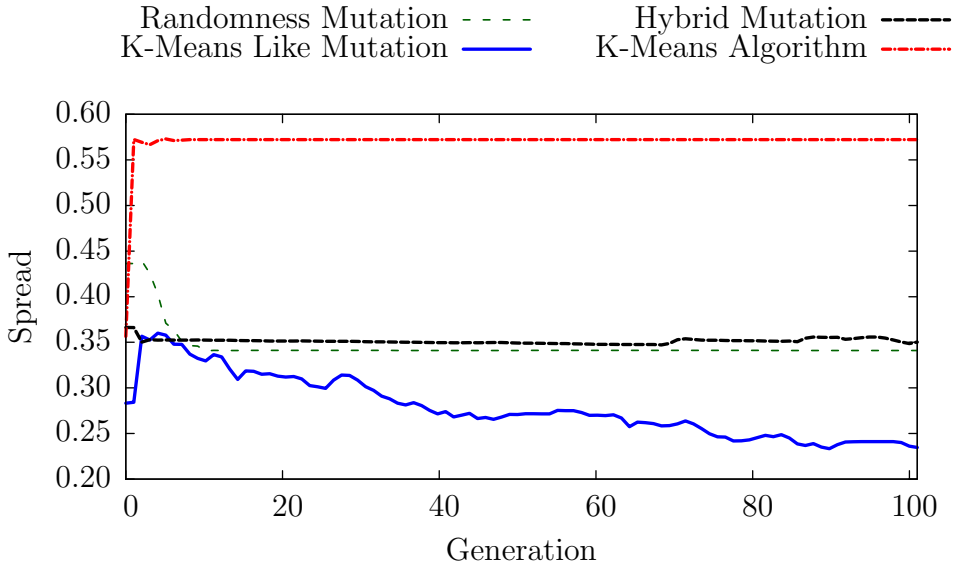


Figure A.33: Change in Spread for Dataset **f** over 100 generations

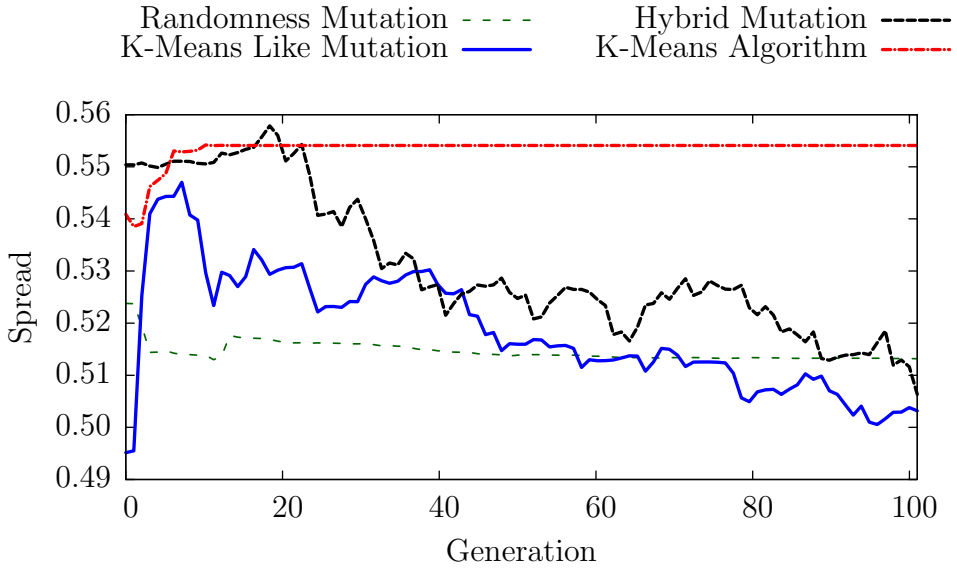


Figure A.34: Change in Spread for Dataset **g** over 100 generations

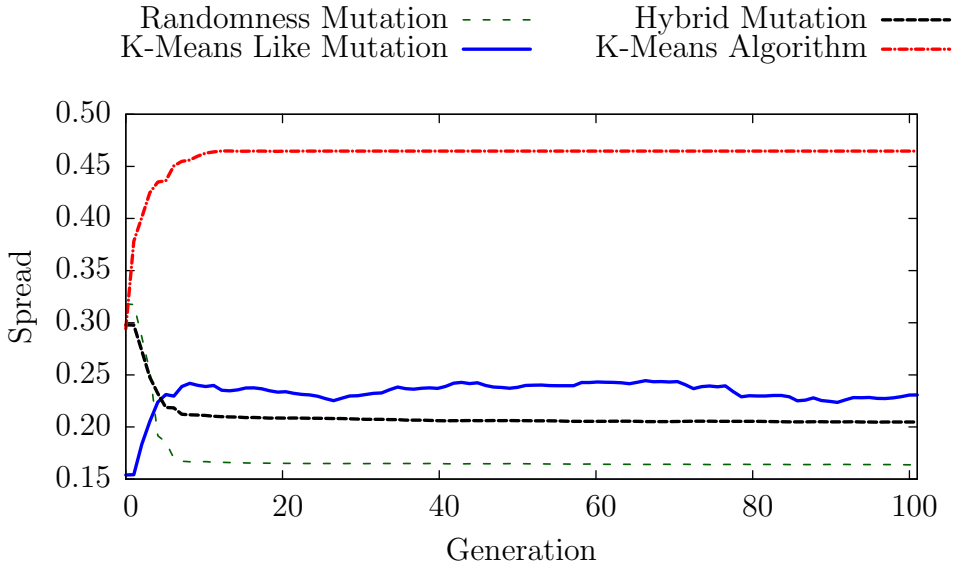


Figure A.35: Change in Spread for Dataset **h** over 100 generations

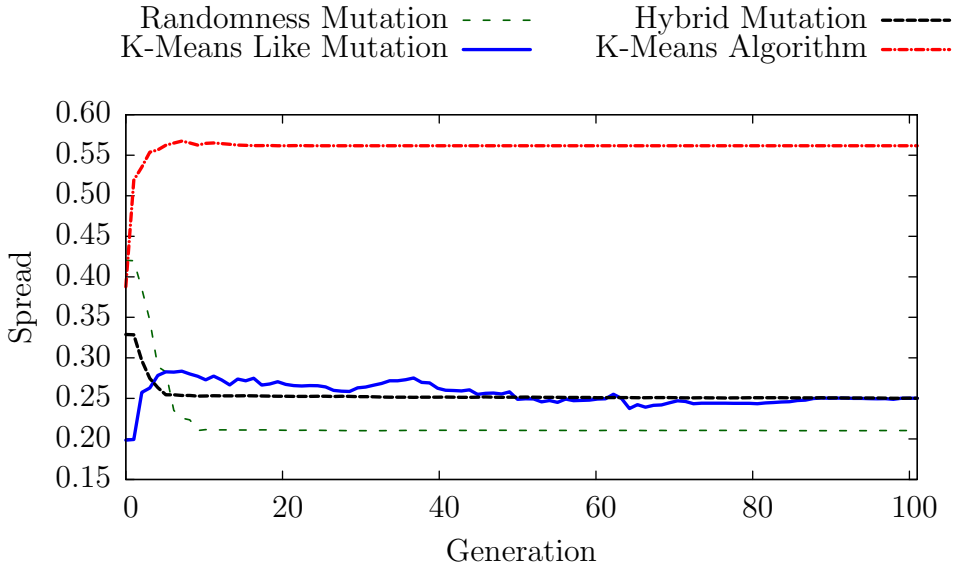


Figure A.36: Change in Spread for Dataset **i** over 100 generations

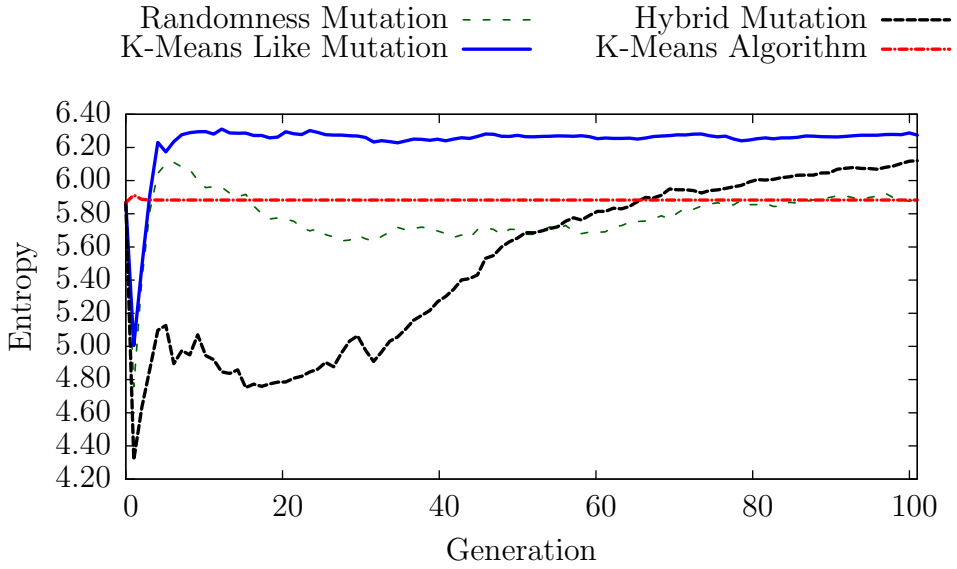


Figure A.37: Change in Entropy for Dataset **a** over 100 generations

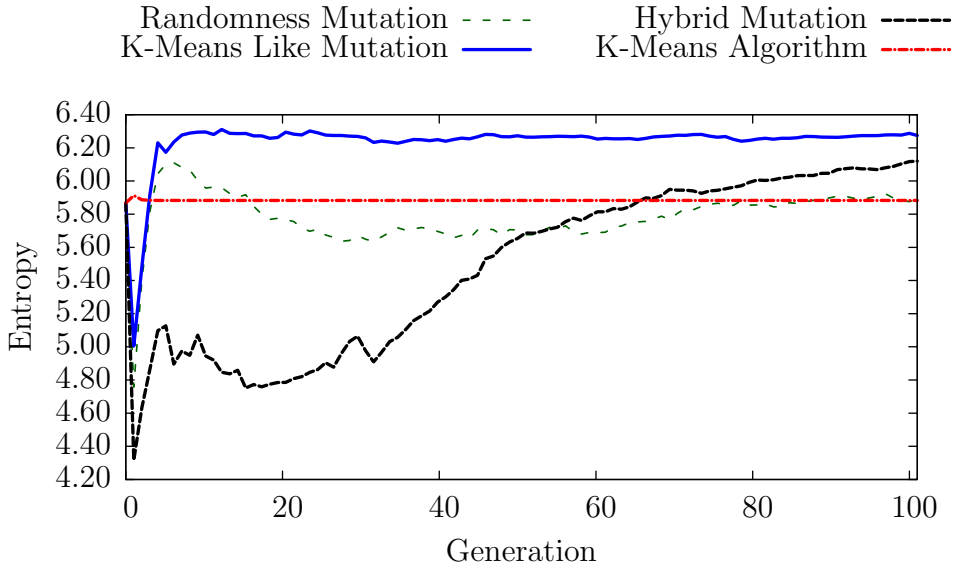


Figure A.38: Change in Entropy for Dataset **b** over 100 generations

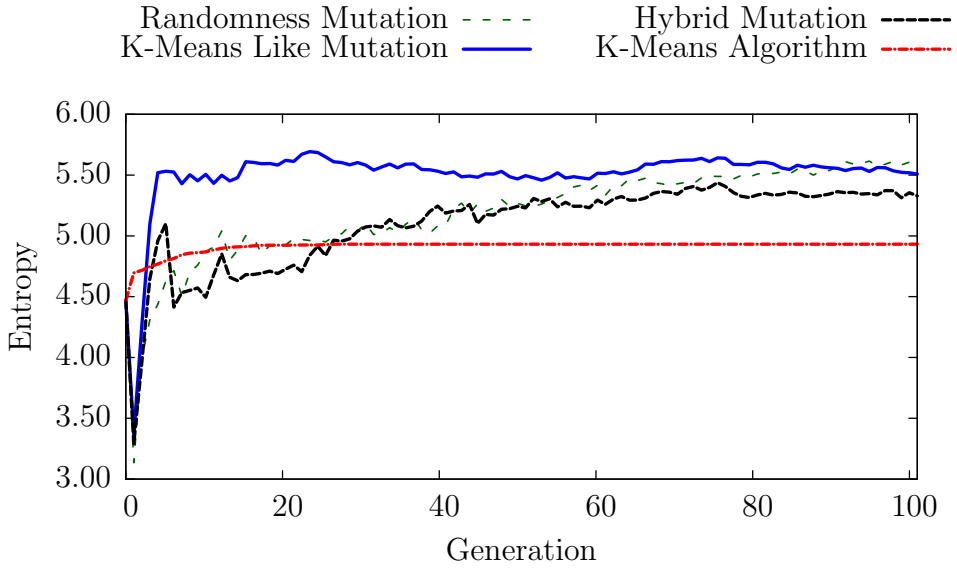


Figure A.39: Change in Entropy for Dataset **c** over 100 generations

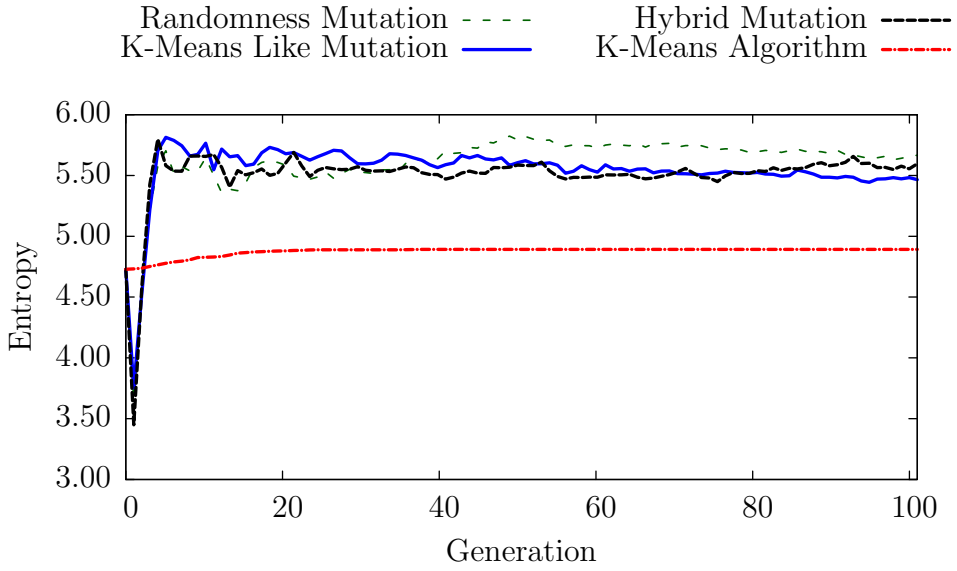


Figure A.40: Change in Entropy for Dataset **d** over 100 generations

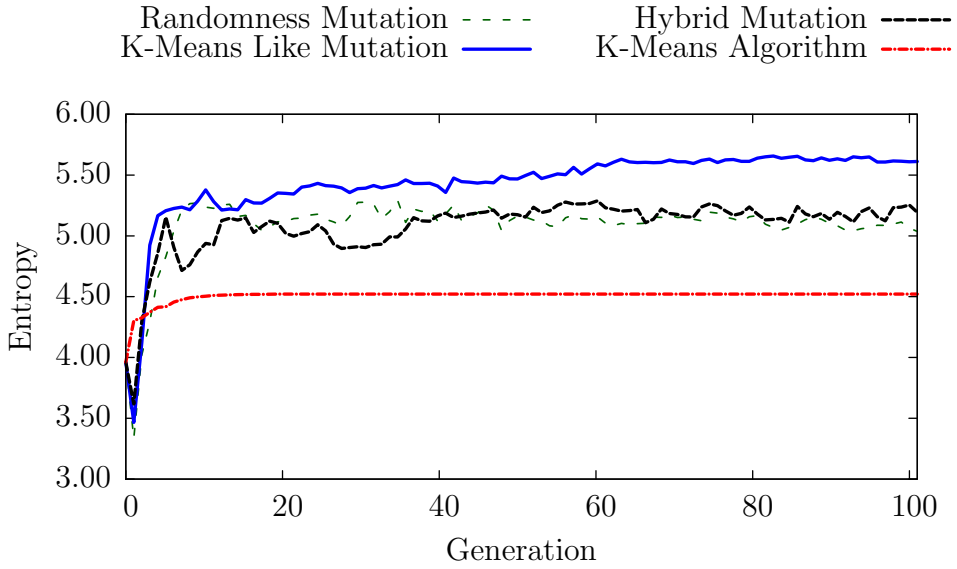


Figure A.41: Change in Entropy for Dataset **e** over 100 generations

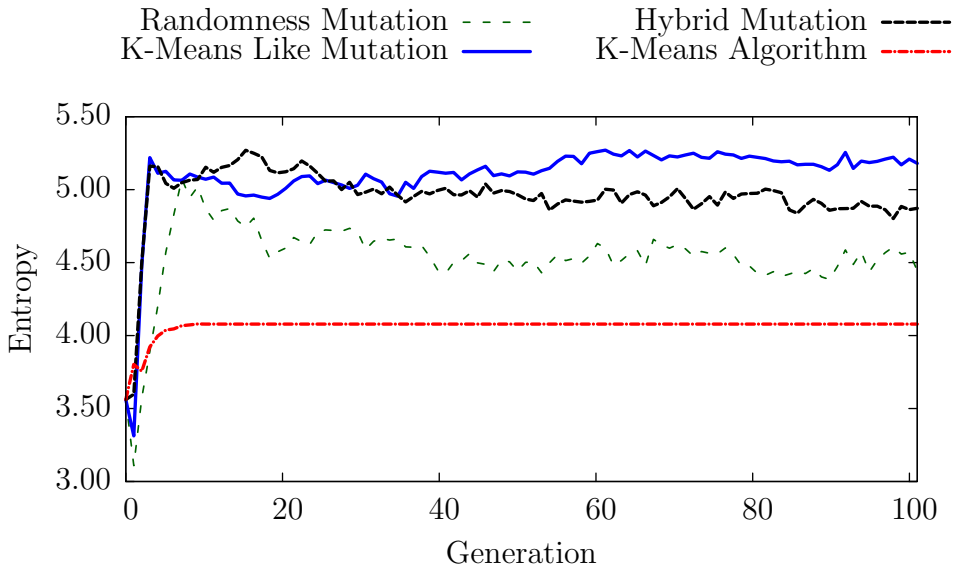


Figure A.42: Change in Entropy for Dataset **f** over 100 generations

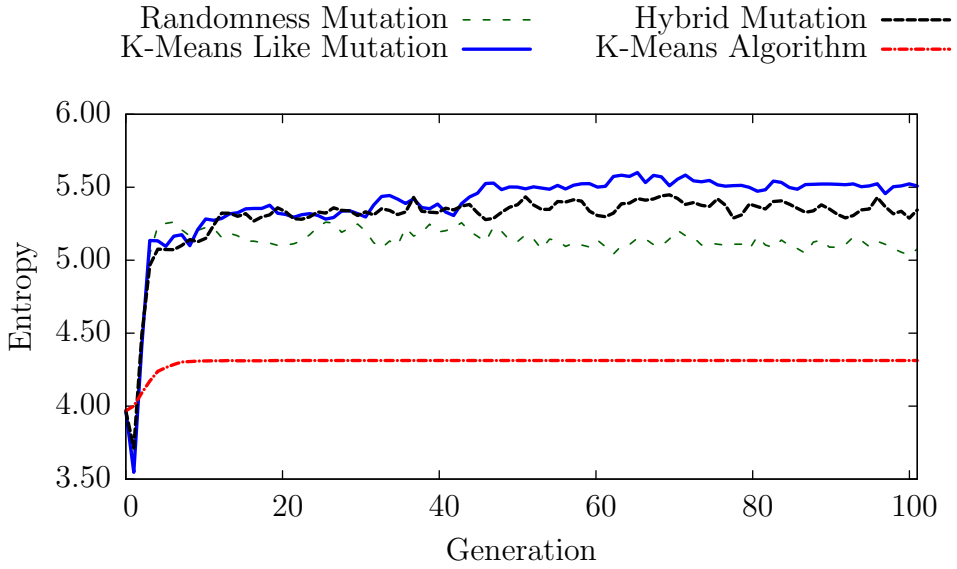


Figure A.43: Change in Entropy for Dataset **g** over 100 generations

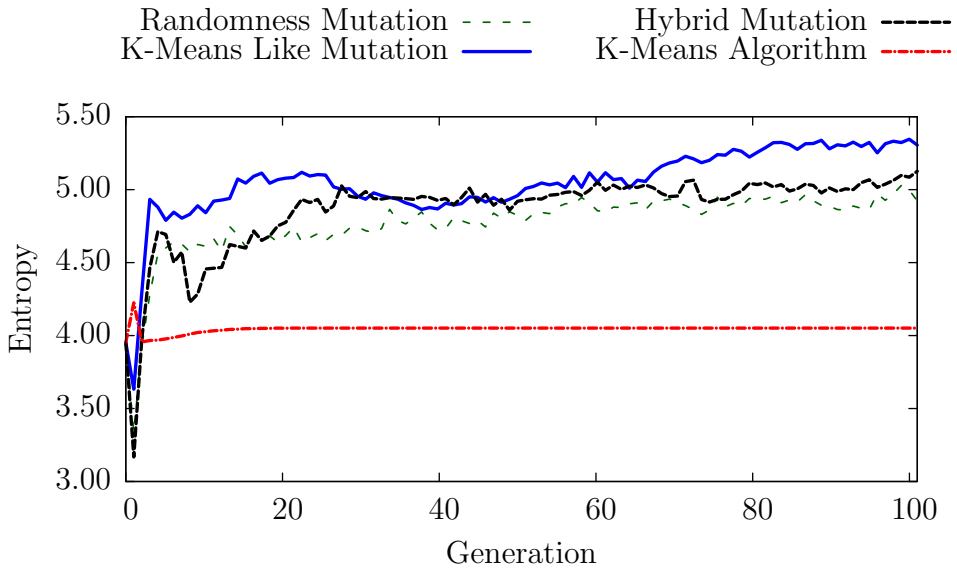


Figure A.44: Change in Entropy for Dataset **h** over 100 generations

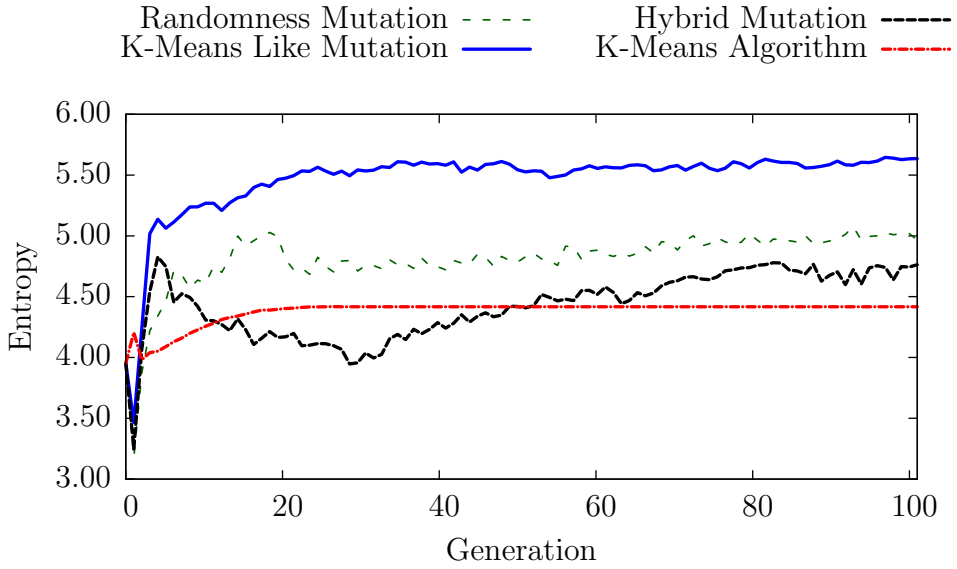


Figure A.45: Change in Entropy for Dataset **i** over 100 generations

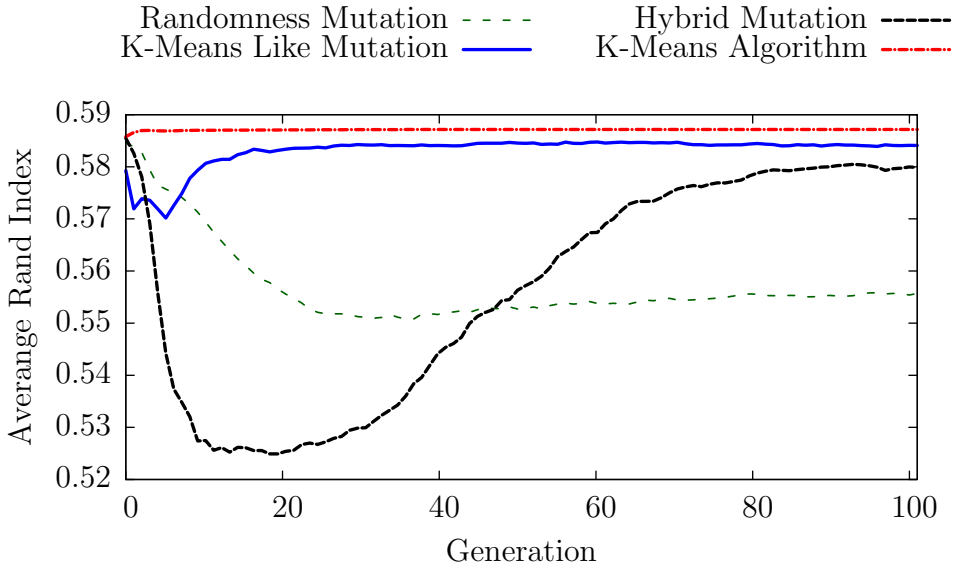


Figure A.46: Change in Average Rand Index for Dataset **a** over 100 generations

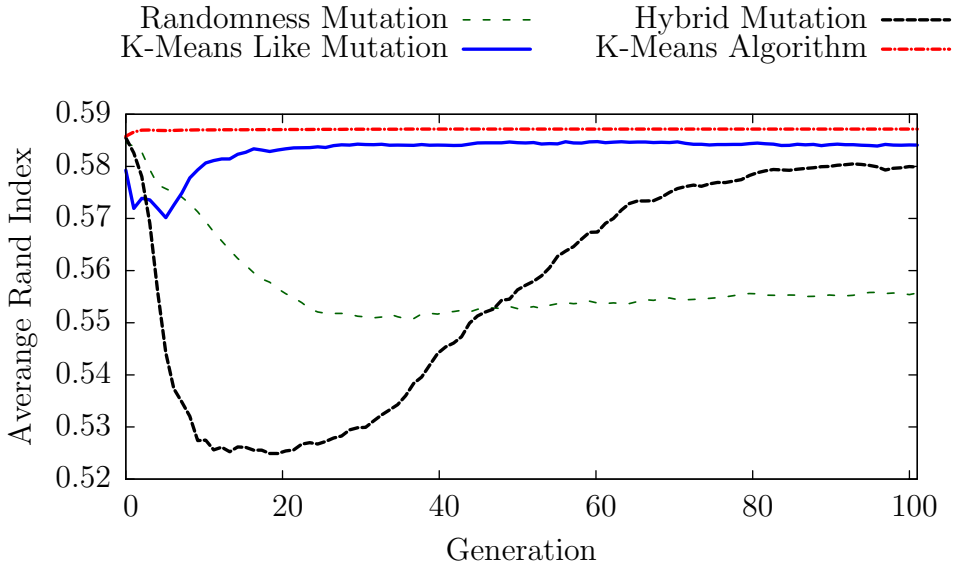


Figure A.47: Change in Average Rand Index for Dataset **b** over 100 generations

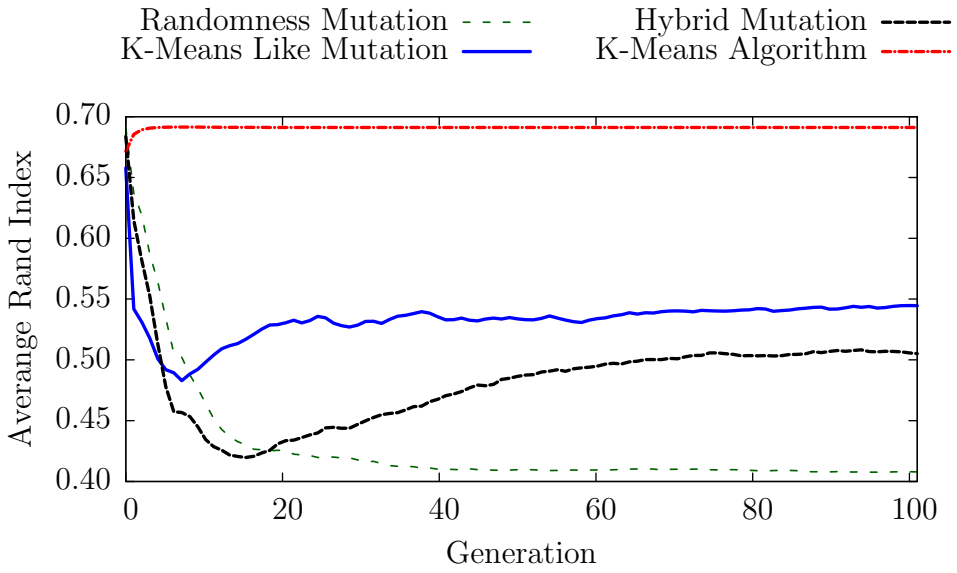


Figure A.48: Change in Average Rand Index for Dataset **c** over 100 generations

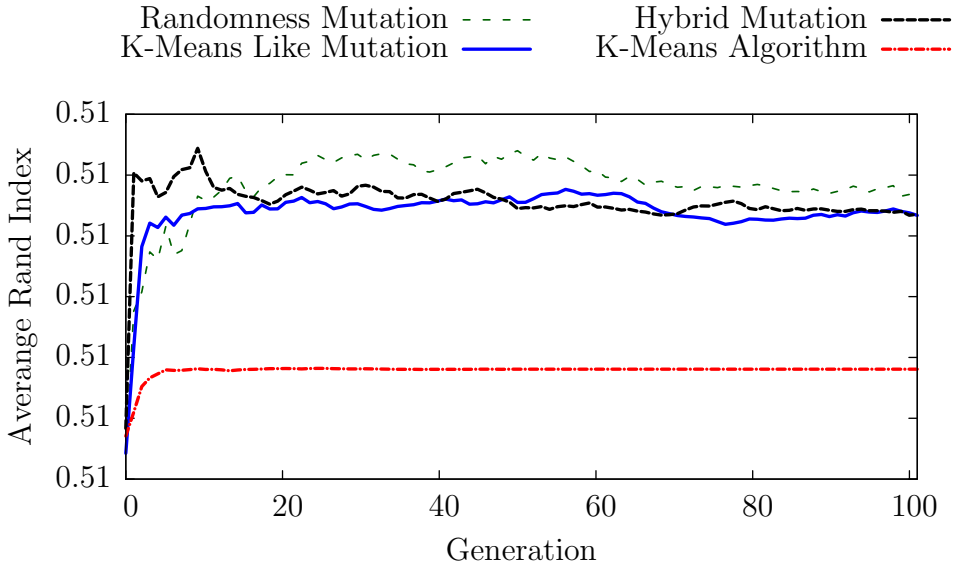


Figure A.49: Change in Average Rand Index for Dataset **d** over 100 generations

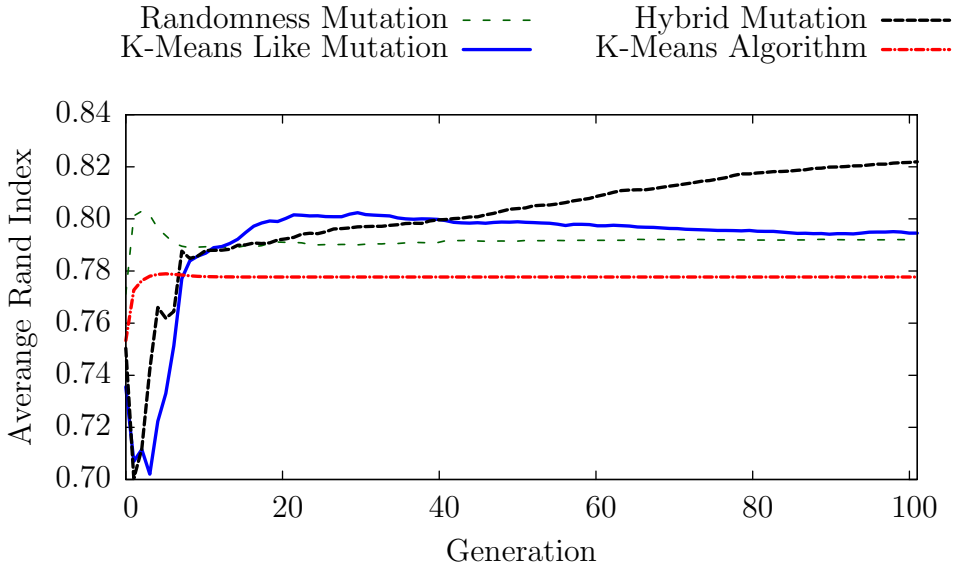


Figure A.50: Change in Average Rand Index for Dataset **e** over 100 generations

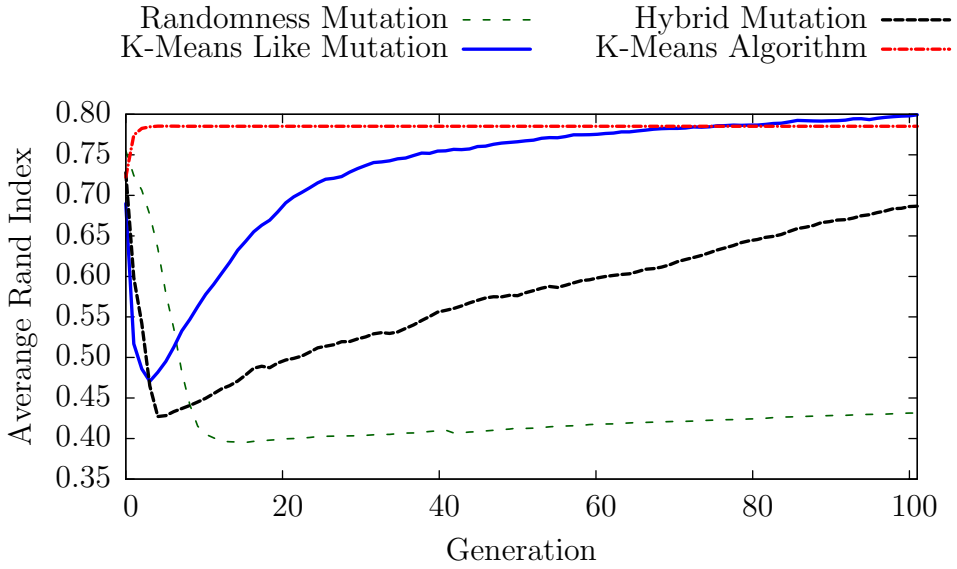


Figure A.51: Change in Average Rand Index for Dataset **f** over 100 generations

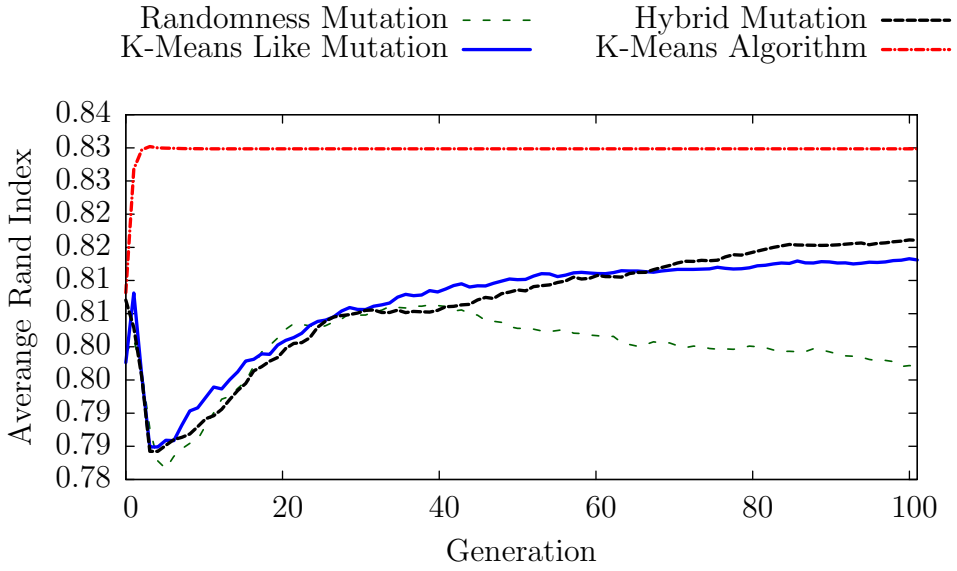


Figure A.52: Change in Average Rand Index for Dataset **g** over 100 generations

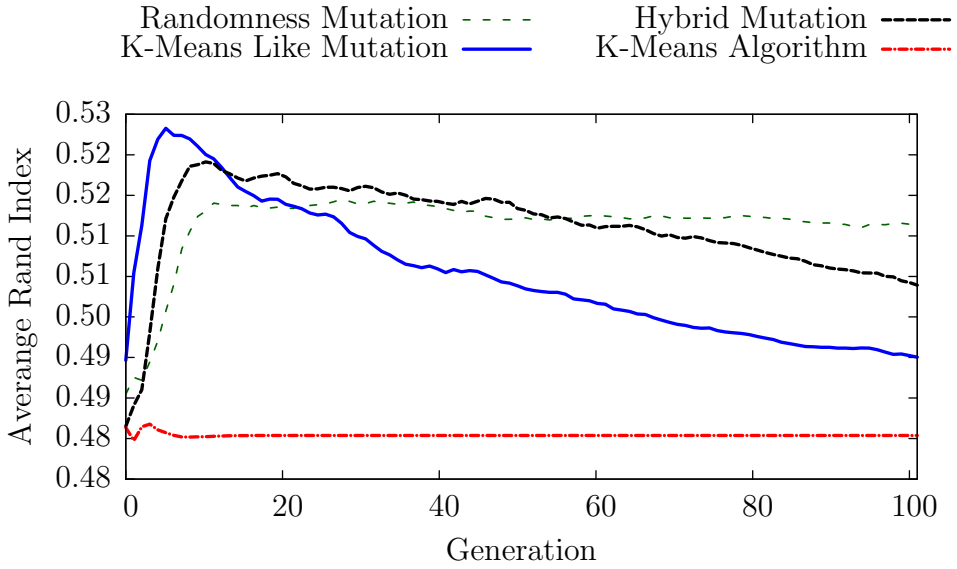


Figure A.53: Change in Average Rand Index for Dataset **h** over 100 generations

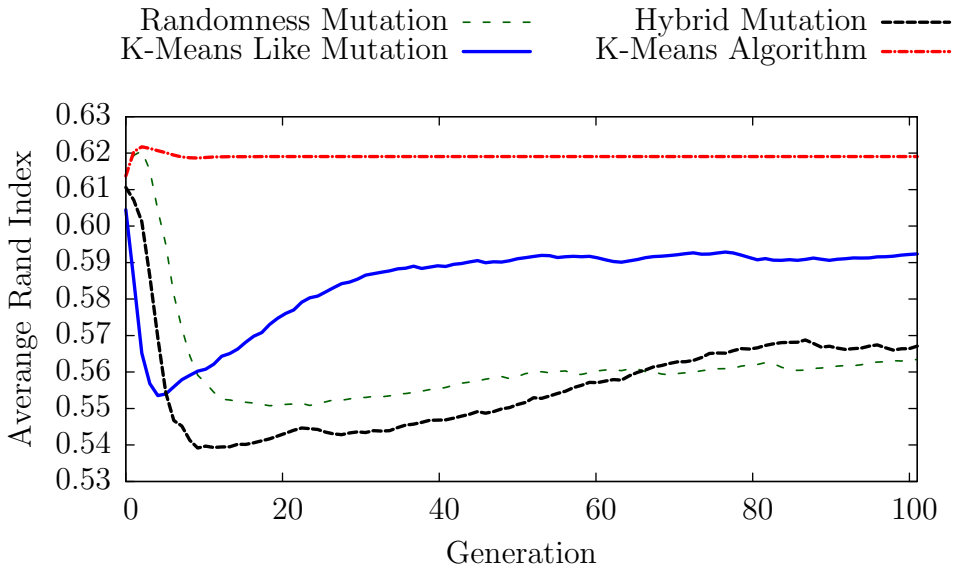


Figure A.54: Change in Average Rand Index for Dataset **i** over 100 generations

Appendix B

Additional MOCA Experimental Results

Table B.1: Comparison of k-means, DBSCAN and MOCA on selected synthetic data sets where the intended value of $k \leq 6$.

Dataset				DBSCAN		k-means		MOCA Best		MOCA Worst		MOCA Average k		MOCA Mean $\mathcal{R}$	
k	d	df	o	k	$\mathcal{R}$	k	$\mathcal{R}$	k	$\mathcal{R}$	k	$\mathcal{R}$	Mean k	S.D.	Mean $\mathcal{R}$	S.D.
2	2	a	a	2	0.50100	2	1	2	1	19	0.57090	10.53623	1.01892	0.72273	0.01828
2	2	a	b	2	0.50100	2	0.96845	2	0.97230	17	0.62840	10.50000	1.03940	0.74461	0.02013
2	2	b	a	2	0.29946	2	1	2	1	17	0.36900	10.26471	0.81677	0.56809	0.02414
2	2	b	b	2	0.29946	2	0.96404	2	1	18	0.38540	10.28571	1.10204	0.57940	0.02771
2	2	c	a	2	0.06148	2	1	2	1	31	0.12160	14.58667	2.01072	0.35144	0.02654
2	2	c	b	2	0.06148	2	1	2	1	20	0.22000	6.29412	3.32416	0.91222	0.16789
2	4	a	a	2	0.50100	2	1	2	1	3	0.90270	2.50000	0.35355	0.95135	0.03440
2	4	a	b	2	0.50100	2	0.95306	2	0.93790	2	0.93790	2	0	0.93790	0
2	4	b	a	2	0.29946	2	1	2	1	2	1	2	0	1	0
2	4	b	b	2	0.29946	2	0.99270	2	1	2	0.99270	3	0.50000	0.99598	0.00049
2	4	c	a	2	0.06148	2	1	2	1	9	0.99910	5.07692	1.08807	0.99938	0.00008
2	4	c	b	2	0.06148	2	0.96876	2	1	2	0.97650	2.25000	0.37500	0.99210	0.00385
2	6	a	a	2	0.50100	2	1	2	1	2	1	2	0	1	0
2	6	a	b	2	0.50100	2	1	2	1	2	1	2	0	1	0
2	6	b	a	2	0.29946	2	1	2	1	2	1	2	0	1	0
2	6	b	b	2	0.29946	2	1	2	0.99270	2	0.99270	2	0	0.99270	0
2	6	c	a	2	0.06148	2	1	2	1	9	0.99910	5.25000	1.08253	0.99939	0.00008
2	6	c	b	2	0.06148	5	0.98388	2	1	3	0.53210	2.33333	0.38490	0.84140	0.17857
2	8	a	a	2	0.50100	2	1	2	1	3	0.99800	2.50000	0.35355	0.99900	0.00071
2	8	a	b	2	0.50100	2	1	2	1	2	1	2	0	1	0
2	8	b	a	2	0.29946	2	1	2	1	2	1	2	0	1	0
2	8	b	b	2	0.29946	2	0.99270	2	0.99270	2	0.99270	2	0	0.99270	0
2	8	c	a	2	0.06148	2	1	2	1	2	1	2	0	1	0
2	8	c	b	2	0.06148	2	1	2	1	3	0.99980	2.50000	0.35355	0.99990	0.00007
2	10	a	a	2	0.50100	2	1	2	1	2	1	2	0	1	0
2	10	a	b	2	0.50100	2	0.99600	2	0.98020	2	0.98020	2	0	0.98020	0
2	10	b	a	2	0.29946	2	1	2	1	2	1	2	0	1	0
2	10	b	b	2	0.29946	2	0.99270	2	0.99270	3	0.99140	2.50000	0.35355	0.99205	0.00046
2	10	c	a	2	0.06148	2	1	2	1	2	1	2	0	1	0
2	10	c	b	2	0.06148	2	1	2	1	4	0.99960	3	0.57735	0.99980	0.00012
6	2	a	a	6	0.83500	6	0.87174	6	1	2	0.66600	13.59677	1.32121	0.93538	0.00575
6	2	a	b	6	0.83500	6	0.93088	6	0.98450	2	0.44240	6.05556	1.16542	0.86817	0.01931
6	2	b	a	6	0.83259	3	0.75106	6	1	2	0.58450	13.70149	1.13599	0.92616	0.00365
6	2	b	b	6	0.83259	8	0.92471	6	0.99200	2	0.45840	5.85294	0.88271	0.87563	0.01641
6	2	c	a	6	0.80926	5	0.89634	6	1	2	0.66870	11.35849	1.04964	0.93031	0.00337
6	2	c	b	6	0.80926	13	0.96613	6	0.98770	2	0.66870	7.55882	1.96215	0.92766	0.00399
6	4	a	a	6	0.83500	4	0.88956	6	1	2	0.60940	4.37500	0.92808	0.86669	0.04505
6	4	a	b	6	0.83500	6	0.93288	6	0.99600	2	0.66210	4.12500	0.66291	0.85879	0.04805
6	4	b	a	6	0.83259	4	0.85395	6	1	2	0.66700	6.88889	1.70370	0.92386	0.02155
6	4	b	b	6	0.83259	25	0.91290	6	0.99200	2	0.66500	4.24000	0.75200	0.86000	0.02592
6	4	c	a	6	0.80926	21	0.87966	6	1	2	0.52970	4.44444	0.85185	0.88350	0.03877
6	4	c	b	6	0.80926	4	0.91313	6	0.99150	2	0.50680	4.46154	1.25874	0.86625	0.02919
6	6	a	a	6	0.83500	39	0.79297	6	1	2	0.61210	3.85714	0.70193	0.84166	0.08676
6	6	a	b	6	0.83500	6	0.94147	6	0.99740	2	0.66600	3.71429	0.26997	0.84083	0.02250
6	6	b	a	6	0.83259	26	0.97076	6	1	2	0.62490	3.85714	0.70193	0.83661	0.08002
6	6	b	b	6	0.83259	20	0.96804	6	0.98910	2	0.66700	4	0.26726	0.86226	0.01856
6	6	c	a	6	0.80926	37	0.88581	6	1	2	0.52970	3.83333	0.74846	0.83942	0.12644
6	6	c	b	6	0.80926	11	0.96794	6	1	2	0.67630	4	0.37796	0.87203	0.01350
6	8	a	a	6	0.83500	3	0.66867	6	1	2	0.61210	3.71429	0.64794	0.81809	0.07786
6	8	a	b	6	0.83500	13	0.98218	6	0.99470	2	0.66600	4	0.66667	0.86803	0.04222
6	8	b	a	6	0.83259	20	0.86673	6	1	2	0.66700	4	0.89443	0.87082	0.09115
6	8	b	b	6	0.83259	14	0.97387	7	0.99500	2	0.66700	5.56250	0.85938	0.90980	0.02070
6	8	c	a	6	0.80926	20	0.87944	6	1	2	0.66870	3.66667	0.68041	0.84720	0.06977
6	8	c	b	6	0.80926	20	0.87797	5	0.99040	2	0.66870	4.16667	1.15670	0.86573	0.04975
6	10	a	a	6	0.83500	5	0.94478	6	1	2	0.66600	3.83333	0.74846	0.85152	0.07574
6	10	a	b	6	0.83500	13	0.97643	6	0.98970	2	0.44510	3.46154	0.70404	0.77478	0.05961
6	10	b	a	6	0.83259	36	0.90824	6	1	2	0.66700	4	0.89443	0.87082	0.09115
6	10	b	b	6	0.83259	5	0.93777	12	0.98710	2	0.66700	7.85714	1.94382	0.88117	0.03554
6	10	c	a	6	0.80926	11	0.96077	6	1	2	0.66810	3.57143	0.59394	0.83480	0.06301
6	10	c	b	6	0.80926	5	0.92194	6	0.98690	2	0.52970	3.37500	0.40625	0.77336	0.05148

Table B.2: Comparison of k-means, DBSCAN and MOCA on selected synthetic data sets where the intended value of $k \geq 10$.

10	2	a	a	10	0.90180	6	0.89941	10	1	2	0.59920	13.25000	1.54161	0.95494	0.00023
10	2	a	b	10	0.90180	6	0.89957	13	0.98210	2	0.41880	9.06897	1.28706	0.88141	0.01742
10	2	b	a	10	0.90186	5	0.82442	11	0.99550	2	0.52650	14.22222	0.91662	0.94725	0.00197
10	2	b	b	10	0.90186	7	0.89340	10	0.98770	2	0.51310	9.84783	1.64430	0.90803	0.00899
10	2	c	a	10	0.89458	4	0.78231	10	1	2	0.60260	13.12727	1.19640	0.94611	0.00146
10	2	c	b	10	0.89458	38	0.91231	10	0.99090	2	0.59820	7.03846	1.36527	0.88811	0.01869
10	4	a	a	10	0.90180	5	0.85972	10	1	2	0.59920	6.53333	1.41149	0.89934	0.02452
10	4	a	b	10	0.90180	5	0.85162	10	0.98990	2	0.27860	5.07018	0.00929	0.79066	0.01055
10	4	b	a	10	0.90186	4	0.80491	10	1	2	0.59920	5.88235	0.45654	0.87738	0.00994
10	4	b	b	10	0.90186	16	0.88718	11	0.99520	2	0.51580	5.44444	0.10692	0.82153	0.01658
10	4	c	a	10	0.89458	5	0.87335	10	1	2	0.59880	6.78571	1.12632	0.90622	0.02351
10	4	c	b	10	0.89458	6	0.86350	10	0.98640	2	0.55820	5.93750	0.16573	0.87431	0.00233
10	6	a	a	10	0.90180	24	0.85044	13	0.99320	2	0.57920	7.05263	0.44676	0.87206	0.00958
10	6	a	b	10	0.90180	15	0.85033	10	0.99450	3	0.55910	4.80645	0.21437	0.81182	0.01825
10	6	b	a	10	0.90186	6	0.90245	10	1	2	0.59920	6	1.33333	0.87597	0.09226
10	6	b	b	10	0.90186	26	0.90950	10	0.99220	2	0.59920	5.36842	0.54335	0.84662	0.02104
10	6	c	a	10	0.89458	26	0.85767	10	1	2	0.55820	5.87500	0.03125	0.87181	0.00737
10	6	c	b	10	0.89458	5	0.84186	11	0.99230	2	0.54780	5.82759	1.14619	0.83142	0.02978
10	8	a	a	10	0.90180	34	0.89021	10	1	2	0.57920	5.58333	0.74574	0.84168	0.02828
10	8	a	b	10	0.90180	24	0.94120	10	0.99920	2	0.59840	5.07143	0.39146	0.83701	0.01875
10	8	b	a	10	0.90186	33	0.80789	10	1	2	0.52650	5.61538	1.00273	0.84700	0.07341
10	8	b	b	10	0.90186	28	0.86588	11	0.99340	2	0.57700	5.97222	0.00463	0.87913	0.00554
10	8	c	a	10	0.89460	37	0.91683	10	1	2	0.57110	9.77273	1.65715	0.92504	0.07546
10	8	c	b	10	0.89458	38	0.88075	13	0.99020	2	0.54620	8.16667	1.61063	0.87056	0.02796
10	10	a	a	10	0.90180	4	0.79960	12	0.99080	2	0.57920	6.09091	1.78166	0.86606	0.03761
10	10	a	b	10	0.90180	4	0.83630	10	0.99840	2	0.57920	6.85714	1.34048	0.90279	0.02051
10	10	b	a	10	0.90186	16	0.83257	10	1	2	0.57700	5.75000	1.08253	0.86943	0.08442
10	10	b	b	10	0.90186	20	0.91100	10	0.99420	2	0.59920	7	1.33333	0.87658	0.01924
10	10	c	a	10	0.89458	5	0.78627	10	1	2	0.54620	5.57143	0.95450	0.84637	0.06707
10	10	c	b	10	0.89458	36	0.92884	10	0.99610	2	0.59660	5.70000	0.60374	0.85802	0.02447
14	2	a	a	14	0.93042	5	0.77715	14	1	2	0.57050	14.68750	1.63282	0.95535	0.00497
14	2	a	b	14	0.93042	22	0.89251	14	0.98070	2	0.47130	10.01563	1.37305	0.89293	0.01036
14	2	b	a	14	0.92900	38	0.88997	15	0.99530	2	0.55580	11.73684	1.50268	0.92736	0.00987
14	2	b	b	14	0.92900	18	0.91172	15	0.99230	2	0.55440	8.87273	1.23072	0.88575	0.01395
14	2	c	a	14	0.92752	40	0.84899	14	1	2	0.55140	13.51220	1.32557	0.94419	0.00636
14	2	c	b	14	0.92753	20	0.85389	17	0.98160	2	0.50860	13.16129	1.24952	0.91920	0.00744
14	4	a	a	14	0.93042	5	0.83719	14	1	2	0.56130	8.15000	0.63728	0.89118	0.01700
14	4	a	b	14	0.93042	24	0.88244	16	0.98530	2	0.47360	8.43137	1.05982	0.87581	0.01533
14	4	b	a	14	0.92900	38	0.85650	16	0.99580	2	0.55580	10.17391	1.42334	0.91853	0.01570
14	4	b	b	14	0.92900	7	0.89221	15	0.99640	2	0.56740	7.59459	1.54624	0.85479	0.02226
14	4	c	a	14	0.92752	40	0.83532	13	0.99730	2	0.57180	8.10526	2.27001	0.89746	0.02251
14	4	c	b	14	0.92752	20	0.80091	14	0.99150	2	0.55230	6.93651	0.13399	0.84374	0.00987
14	6	a	a	14	0.93042	38	0.84372	13	0.99020	2	0.56130	7.52941	1.09854	0.87904	0.03874
14	6	a	b	14	0.93042	20	0.87131	15	0.99380	2	0.55900	7.84848	0.32178	0.89060	0.00103
14	6	b	a	14	0.92900	11	0.83861	15	0.99840	2	0.56720	7.70000	1.85594	0.87063	0.02806
14	6	b	b	14	0.92900	6	0.85831	14	0.99800	2	0.54480	8.40000	0.86298	0.89992	0.04495
14	6	c	a	14	0.92752	5	0.80098	13	0.98860	2	0.50970	7	1.06600	0.83308	0.05662
14	6	c	b	14	0.92752	7	0.84437	15	0.99160	2	0.57160	8.42553	0.22966	0.90184	0.00783
14	8	a	a	14	0.93042	5	0.78756	14	1	2	0.57050	8.27778	0.40593	0.89788	0.01427
14	8	a	b	14	0.93042	29	0.91129	12	0.97780	2	0.55900	5.75439	0.03253	0.83998	0.00558
14	8	b	a	14	0.92899	14	0.84873	14	1	2	0.57120	7.23077	0.63361	0.88422	0.01793
14	8	b	b	14	0.92900	12	0.89231	14	0.99860	2	0.55650	7.44444	0.09259	0.87412	0.00975
14	8	c	a	14	0.92752	40	0.86078	14	0.99110	2	0.56950	8.08696	0.22665	0.90150	0.00102
14	8	c	b	14	0.92752	39	0.91661	13	0.99340	2	0.54430	6.45455	0.17904	0.83310	0.00144
14	10	a	a	14	0.93042	25	0.86358	14	1	2	0.55900	7.48000	0.49600	0.87112	0.00112
14	10	a	b	14	0.93042	15	0.85352	14	0.99720	2	0.53130	7.21053	1.10140	0.87819	0.01931
14	10	b	a	14	0.92900	14	0.87436	15	0.99770	2	0.46890	7.38889	0.85115	0.85862	0.02536
14	10	b	b	14	0.92900	40	0.89401	18	0.99460	2	0.55520	8.13158	1.60087	0.88033	0.01854
14	10	c	a	14	0.92752	40	0.87218	16	0.99480	2	0.56900	9	0.81650	0.89292	0.01628
14	10	c	b	14	0.92752	40	0.86121	14	0.99360	2	0.54190	7.78378	0.52874	0.87960	0.01496
18	2	a	a	18	0.94632	7	0.87695	18	0.98610	2	0.52770	12.06667	1.48077	0.91408	0.01009
18	2	a	b	18	0.94632	38	0.89513	17	0.98010	2	0.54650	9.08475	1.03048	0.88581	0.01228
18	2	b	a	18	0.94346	31	0.89783	18	0.97820	2	0.54460	10.84091	1.68230	0.91268	0.00923
18	2	b	b	18	0.94346	8	0.84937	16	0.98040	2	0.55600	10.85000	1.43946	0.90152	0.00929
18	2	c	a	18	0.94498	32	0.82390	16	0.98450	2	0.55410	12.97368	1.13982	0.93414	0.00739
18	2	c	b	18	0.94498	40	0.89787	18	0.97980	2	0.55240	10	1.08012	0.89000	0.01366
18	4	a	a	18	0.94632	23	0.86244	16	0.98440	2	0.55460	8.81481	1.11906	0.87833	0.03111
18	4	a	b	18	0.94632	28	0.89122	17	0.98550	2	0.54750	8.45455	1.15227	0.87875	0.01439
18	4	b	a	18	0.94346	19	0.85807	17	0.98750	2	0.55600	8.76000	0.35200	0.89610	0.00194
18	4	b	b	18	0.94346	30	0.85331	19	0.96970	2	0.52210	11.61818	1.26504	0.89738	0.00962
18	4	c	a	18	0.94499	6	0.74852	16	0.98580	2	0.55260	11	1.29777	0.91199	0.01158
18	4	c	b	18	0.94499	12	0.87149	18	0.98760	2	0.55560	9.59091	0.91200	0.90531	0.00972
18	6	a	a	18	0.94632	34	0.84968	17	0.99060	2	0.53130	10.83333	2.27939	0.91444	0.01510
18	6	a	b	18	0.94632	34	0.89125	18	0.98930	2	0.55330	8.86538	0.98939	0.86855	0.01590
18	6	b	a	18	0.94346	36	0.84474	17	0.99450	2	0.52210	9.18919	0.19550	0.89457	0.00339
18	6	b	b	18	0.94346	9	0.88771	17	0.97980	2	0.52210	8.86885	0.40090	0.88219	0.00895
18	6	c	a	18	0.94498	37	0.82842	17	0.98990	2	0.55480	9.04167	1.82861	0.88312	0.02169
18	6	c	b	18	0.94498	10	0.87986	15	0.98220	2	0.50660	7.98387	0.50595	0.85907	0.00824
18	8	a	a	18	0.94632	20	0.85008	17	0.98630	2	0.54930	10.44000	0.71200	0.91506	0.01129
18	8	a	b	18	0.94632	12	0.88769	17	0.98350	2	0.54650	10.01724	0.78557	0.89909	0.01099
18	8	b	a	18	0.94346	9	0.80996	16	0.98900	2	0.49180	8.88571	0.82584	0.89355	0.01603
18	8	b	b	18	0.94346	29	0.84521	18	0.98520	2	0.44410	8.72727	0.50259	0.87558	0.00562
18	8	c	a	18	0.94498	19	0.83494	17	0.99000	2	0.55480	9.44444	1.45407	0.91430	0.01457
18	8	c	b	18	0.94499	32	0.87670	16	0.98940	2	0.54550	8.83871	0.61452	0.89889	0.01242
18	10	a	a	18	0.94632	33	0.83083	16	0.98720	2	0.54930	9.90909	0.28781	0.91245	0.00010
18	10	a	b	18	0.94632	19	0.89929	14	0.96680	2	0.52770	7.85965	0.24632	0.86876	0.00152
18	10	b	a	18	0.94346	14	0.83039	17	0.97790	2	0.54590	9.90625	1.25401	0.89715	0.01428
18	10	b	b	18	0.94346	18	0.84141	17	0.98680	2	0.55600	9.42308	0.08000	0.89644	0.00633
18	10	c	a	18	0.94498	36	0.90478	19	0.98780	2	0.55410	8.67742	1.85399	0.89609	0.01647
18	10	c	b	18	0.94498	35	0.87163	18	0.98750	2	0.55510	8.28889	0.10601	0.87117	0.00735

Bibliography

[1] Mihael Ankerst, Markus M. Breunig, Hans-Peter Kriegel, and Jörg Sander. OPTICS: Ordering points to identify the clustering structure. *Special Interest Group on Management of Data*, 28(2):49–60, June 1999.

[2] K. Bache and M. Lichman. UCI machine learning repository, 2013.

[3] Johannes Bader and Eckart Zitzler. HypE: An algorithm for fast hypervolume-based many-objective optimization. *Evolutionary Computation*, 19(1):45–76, February 2011.

[4] Kenneth Bailey. *Typologies and taxonomies : an introduction to classification techniques*. Sage Publications, Thousand Oaks, California, 1994.

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

[6] Sanghamitra Bandyopadhyay and Ujjwal Maulik. An evolutionary technique based on K-means algorithm for optimal clustering in $\mathbb{R}^n$. *Information Sciences*, 146(1-4):221–237, oct 2002.

[7] LucasS. Batista, Felipe Campelo, FredericoG. Guimarães, and JaimeA. Ramírez. Pareto cone ε-dominance: improving convergence and diversity in multiobjective evolutionary algorithms. In RicardoH.C. Takahashi, Kalyanmoy Deb, ElizabethF. Wanner, and Salvatore Greco, editors, *Evolutionary Multi-Criterion Optimization*, volume 6576, pages 76–90. Springer, 2011.

[8] P. Berkhin. A survey of clustering data mining techniques. In Jacob Kogan, Charles Nicholas, and Marc Teboulle, editors, *Grouping Multidimensional Data*, pages 25–71. Springer Berlin Heidelberg, 2006.

[9] Nicola Beume, Boris Naujoks, and Michael Emmerich. SMS-EMOA: Multiobjective selection based on dominated hypervolume. *European Journal of Operational Research*, 181(3):1653–1669, 2007.

[10] James C Bezdek, Robert Ehrlich, and William Full. FCM: The fuzzy c-means clustering algorithm. *Computers & Geosciences*, 10(2):191–203, 1984.

[11] J.C. Bezdek, S. Boggavarapu, L.O. Hall, and A. Bensaid. Genetic algorithm guided clustering. In *In Proceedings of the First IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence*, pages 34–39, 1994.

[12] J.C. Bezdek and N.R. Pal. Some new indexes of cluster validity. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 28(3):301–315, 1998.

[13] Gabriel Richard Bitran and Thomas L Magnanti. The structure of admissible points with respect to cone dominance. *Journal of Optimization Theory and Applications*, 29(4):573–614, 1979.

[14] Tobias Blickle and Lothar Thiele. A comparison of selection schemes used in evolutionary algorithms. *Evolutionary Computation*, 4(4):361–394, 1996.

[15] A.H. Brie and P. Morignot. Genetic planning using variable length chromosomes. In *15th International Conference on Automated Planning and Scheduling*, pages 320–329, 2005.

[16] T. Caliński and J. Harabasz. A dendrite method for cluster analysis. *Communications in Statistics-Simulation and Computation*, 3(1):1–27, 1974.

[17] Erick Cantú-Paz. A survey of parallel genetic algorithms. *Calculateurs paralleles, reseaux et systems repartis*, 10(2):141–171, 1998.

[18] R. Cavill, S.L. Smith, and A.M. Tyrrell. Variable length genetic algorithms with multiple chromosomes on a variant of the onemax problem. In *Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pages 1405–1406. ACM, 2006.

[19] Vladimír Černỳ. Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. *Journal of optimization theory and applications*, 45(1):41–51, 1985.

[20] E. Chen and F. Wang. Dynamic clustering using multi-objective evolutionary algorithm. *Computational Intelligence and Security*, pages 73–80, 2005.

[21] Keh-Shih Chuang, Hong-Long Tzeng, Sharon Chen, Jay Wu, and Tzong-Jer Chen. Fuzzy c-means clustering with spatial information for image segmentation. *computerized medical imaging and graphics*, 30(1):9–15, 2006.

[22] Carlos A Coello Coello Coello. A short tutorial on evolutionary multiobjective optimization. In *Evolutionary Multi-Criterion Optimization*, pages 21–40. Springer, 2001.

[23] R.M. Cole. Clustering with genetic algorithms. Master's thesis, University of Western Australia, 1998.

[24] William J Conover and Ronald L Iman. Rank transformations as a bridge between parametric and nonparametric statistics. *The American Statistician*, 35(3):124–129, 1981.

[25] R.M. Cormack. A review of classification. *J. Roy. Statistical Society*, 134(3):321–367, 1971.

[26] David Corne, Joshua D. Knowles, and Martin J. Oates. The pareto envelope-based selection algorithm for multi-objective optimisation. In *Proceedings of the 6th International Conference on Parallel Problem Solving from Nature*, PPSN VI, pages 839–848, London, UK, UK, 2000. Springer-Verlag.

[27] D.W. Corne, N.R. Jerram, J.D. Knowles, M.J. Oates, et al. Pesa-ii: Region-based selection in evolutionary multiobjective optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2001*, pages 283–290, 2001.

[28] William HE Day and Herbert Edelsbrunner. Efficient algorithms for agglomerative hierarchical clustering methods. *Journal of classification*, 1(1):7–24, 1984.

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

[30] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler. Scalable test problems for evolutionary multiobjective optimization. *Evolutionary Multiobjective Optimization*, pages 105–145, 2005.

[31] Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine Learning Research*, 7:1–30, 2006.

[32] C. Ding and X. He. K-means clustering via principal component analysis. In *Proceedings of the twenty-first international conference on Machine learnin*, pages 29–45. ACM New York, NY, USA, 2004.

[33] J.C. Dunn. Well-separated clusters and optimal fuzzy partitions. *Cybernetics and Systems*, 4(1):95–104, 1974.

[34] Michael Emmerich, Nicola Beume, and Boris Naujoks. An EMO algorithm using the hypervolume measure as selection criterion. In *Evolutionary Multi-Criterion Optimization*, pages 62–76. Springer, 2005.

[35] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Knowledge Discovery and Data Mining*, volume 96, pages 226–231, 1996.

[36] J.E. Fieldsend, R.M. Everson, and S. Singh. Using unconstrained elite archives for multiobjective optimization. *IEEE Transactions on Evolutionary Computation*, 7(3):305–323, 2003.

[37] Francesco Folino and Clara Pizzuti. A multiobjective and evolutionary clustering method for dynamic networks. In *International Conference on Advances in Social Networks Analysis and Mining (ASONAM), 2010*, pages 256–263. IEEE, 2010.

[38] Carlos M Fonseca, Peter J Fleming, et al. Genetic algorithms for multiobjective optimization: Formulationdiscussion and generalization. In *Proceedings of the 5th International Conference on Genetic Algorithms*, volume 93, pages 416–423. ACM, 1993.

[39] C.M. Fonseca, L. Paquete, and M. López-Ibánez. An improved dimension-sweep algorithm for the hypervolume indicator. In *IEEE Congress on Evolutionary Computation*, pages 1157–1163. IEEE, 2006.

[40] Stephanie Forrest. Genetic algorithms: principles of natural selection applied to computation. *Science*, 261(5123):872–878, 1993.

[41] E.B. Fowlkes and C.L. Mallows. A method for comparing two hierarchical clusterings. *Journal of the American Statistical Association*, 78(383):553–569, 1983.

[42] Chris Fraley and Adrian E Raftery. How many clusters? Which clustering method? Answers via model-based cluster analysis. *The Computer Journal*, 41(8):578–600, 1998.

[43] P. Fränti, J. Kivijärvi, T. Kaukoranta, and O. Nevalainen. Genetic algorithms for large-scale clustering problems. *The Computer Journal*, 40(9):547, 1997.

[44] Alex A. Freitas. A critical review of multi-objective optimization in data mining: a position paper. *ACM SIGKDD Explorations*, pages 77–86, 2004.

[45] T. Friedrich, K. Bringmann, T. Voß, and C. Igel. The logarithmic hypervolume indicator. In *Proceedings of the 11th workshop proceedings on Foundations of genetic algorithms*, pages 81–92, 2011.

[46] Ashish Ghosh and Satchidananda Dehuri. Evolutionary algorithms for multi-criterion optimization: A survey. *International Journal of Computing & Information Sciences*, 2(1):38–57, 2004.

[47] Fred Glover. Tabu search—part i. *ORSA Journal on computing*, 1(3):190–206, 1989.

[48] Fred Glover. Tabu search—part ii. *ORSA Journal on computing*, 2(1):4–32, 1990.

[49] D. Goldberg, K. Deb, and B. Korb. Messy genetic algorithms: Motivation, analysis, and first results. *Complex systems*, 3(3):493–530, 1989.

[50] David E Goldberg. Genetic algorithms: A tutorial. *Computer Design*, 1995.

[51] David E Goldberg and Kalyanmoy Deb. A comparative analysis of selection schemes used in genetic algorithms. *Urbana*, 51:61801–2996, 1991.

[52] David E Goldberg and John H Holland. Genetic algorithms and machine learning. *Machine learning*, 3(2):95–99, 1988.

[53] M. Halkidi and M. Vazirgiannis. Clustering validity assessment: Finding the optimal partitioning of a data set. In *Industrial Conference on Data Mining*, pages 187–194, 2001.

[54] M. Halkidi and M. Vazirgiannis. Clustering validity assessment using multi representatives. In *Proceedings of the Hellenic Conference on Artificial Intelligence, SETN*, pages 237–249, 2002.

[55] M. Halkidi, M. Vazirgiannis, and Y. Batistakis. Quality scheme assessment in the clustering process. *Principles of Data Mining and Knowledge Discovery*, pages 265–276, 2000.

[56] J. Handl and J. Knowles. Evolutionary multiobjective clustering. In *Parallel Problem Solving from Nature-PPSN VIII*, pages 1081–1091, 2004.

[57] J. Handl and J. Knowles. Multiobjective clustering with automatic determination of the number of clusters. *University of Manchester Institute of Science and Technology Report TR-COMPSYSBIO-2004-02*, 2004.

[58] J. Handl and J. Knowles. Exploiting the trade-off: the benefits of multiple objectives in data clustering. In *Proceedings of the Third International Conference on Evolutionary Multicriterion Optimization*, pages 547–560. Springer, 2005.

[59] J. Handl and J. Knowles. Improvements to the scalability of multiobjective clustering. In *Evolutionary Computation, 2005. The 2005 IEEE Congress on*, volume 3, 2005.

[60] J. Handl and J. Knowles. Multiobjective clustering around medoids. In *Evolutionary Computation, 2005. The 2005 IEEE Congress on*, volume 1, 2005.

[61] J. Handl and J. Knowles. *Multiobjective clustering and cluster validation.* Springer Series on Computational Intelligence. Springer, 2006.

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

[63] J. Handl and J. Knowles. Clustering criteria in multiobjective data clustering. In *Parallel Problem Solving from Nature-PPSN XII*, pages 32–41. Springer, 2012.

[64] J. A. Hartigan and M. 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.

[65] J.A. Hartigan. Statistical theory in clustering. *Journal of classification*, 2(1):63–76, 1985.

[66] John H Holland. Genetic algorithms. *Scientific american*, 267(1):66–72, 1992.

[67] Jeffrey Horn, Nicholas Nafpliotis, and David E Goldberg. A niched pareto genetic algorithm for multiobjective optimization. In *In Proceedings of the First IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence*, pages 82–87. IEEE, 1994.

[68] Christian Horoba and Frank Neumann. Benefits and drawbacks for the use of epsilon-dominance in evolutionary multi-objective optimization. In *Proceedings of the 10th annual conference on Genetic and evolutionary computation*, pages 641–648. ACM, 2008.

[69] R. Howard. Classifying a population into homogeneous groups. *Operational Research in the Social Sciences. Tavistock Publ., London*, pages 585–594, 1966.

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

[71] Brian J Hunt and Margaret M Wiecek. Cones to aid decision making in multicriteria programming. *Multi-Objective Programming and Goal Programming*, pages 153–158, 2003.

[72] Ching-Lai Hwang, Abu Syed Md Masud, Sudhakar R Paidy, and Kwangsun Paul Yoon. *Multiple objective decision making, methods and applications: a state-of-the-art survey*, volume 164. Springer, 1979.

[73] Hisao Ishibuchi and Tadahiko Murata. Multi-objective genetic local search algorithm. In *Proceedings of IEEE International Conference on Evolutionary Computation*, pages 119–124. IEEE, 1996.

[74] Paul Jaccard. Étude comparative de la distribution florale dans une portion des alpes et du jura. *Bulletin de la Société Vaudoise des Sciences Naturelles*, 57:547–579, 1901.

[75] A.K. Jain and R.C. Dubes. *Algorithms for Clustering Data*. Advanced Reference Series. Prentice-Hall, 1988.

[76] A.K Jain, MN Murty, and PJ Flynn. Data clustering: a review. *ACM Computing Surveys (CSUR)*, 31(3):264–323, 1999.

[77] Stephen C Johnson. Hierarchical clustering schemes. *Psychometrika*, 32(3):241–254, 1967.

[78] Abimbola M Jubril. A nonlinear weights selection in weighted sum for convex multiobjective optimization. *Facta universitatis-series: Mathematics and Informatics*, 27(3):357–372, 2012.

[79] L. Kaufman and P. Rousseeuw. Clustering by means of medoids. *Statistical Data Analysis Based on the L1 Norm and Related Methods*, pages 405–416, 1987.

[80] Leonard Kaufman and Peter J. Rousseeuw. *Finding Groups in Data*. John Wiley & Sons, Inc., 1990.

[81] J.R. Kettenring. The practice of cluster analysis. *Journal of classification*, 23(1):3–30, 2006.

[82] Keehyung Kim, Robert Ian McKay, and Byung-Ro Moon. Multiobjective evolutionary algorithms for dynamic social network clustering. In *Proceedings of the 12th annual conference on Genetic and evolutionary computation*, pages 1179–1186. ACM, 2010.

[83] O. Kirkland and B. de la Iglesia. Experimental evaluation of cluster quality measures. *UK Workshop on Computational Intelligence 2013*, 2013.

[84] O. Kirkland and B. de la Iglesia. Moea for clustering: Comparison of mutation operators. In *Genetic and Evolutionary Computation Conference 2013*, 2013.

[85] O. Kirkland, V. Rayward-Smith, and B. de la Iglesia. A novel multi-objective genetic algorithm for clustering. *International Conference on Intelligent Data Engineering and Automated Learning 2011*, pages 317–326, 2011.

[86] Scott Kirkpatrick, MP Vecchi, et al. Optimization by simmulated annealing. *science*, 220(4598):671–680, 1983.

[87] J. Kivijärvi, P. Fränti, and O. Nevalainen. Self-adaptive genetic algorithm for clustering. *Journal of Heuristics*, 9(2):113–129, 2003.

[88] Abdullah Konak, David W Coit, and Alice E Smith. Multi-objective optimization using genetic algorithms: A tutorial. *Reliability Engineering & System Safety*, 91(9):992–1007, 2006.

[89] F. Kovacs, C. Legány, and A. Babos. Cluster validity measurement techniques. In *Proceedings Sixth International Symposium Hungarian Researchers on Computational Intelligence (CINTI)*, 2005.

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

[91] Mirko Křivánek and Jaroslav Morávek. Np-hard problems in hierarchical-tree clustering. *Acta Informatica*, 23(3):311–323, 1986.

[92] L.I. Kuncheva and J.C. Bezdek. Selection of cluster prototypes from data by a genetic algorithm. *Proceedings of the 5th European Congress on Intelligent Techniques and Soft Computing*, 1997.

[93] G.N. Lance and W.T. Williams. A general theory of classificatory sorting strategies. *The computer journal*, 9(4):373, 1967.

[94] BS Landau, S. Landau, and M. Leese. *Cluster Analysis*. Arnold, London, 2001.

[95] Marco Laumanns, Lothar Thiele, Kalyanmoy Deb, and Eckart Zitzler. Combining convergence and diversity in evolutionary multiobjective optimization. *Evolutionary computation*, 10(3):263–282, 2002.

[96] Martin HC Law, Alexander P Topchy, and Anil K Jain. Multiobjective data clustering. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 2, pages II–424. IEEE, 2004.

[97] J.S. Lee. *Preserving nearest neighbor consistency in cluster analysis*. PhD thesis, Iowa State University, 2009.

[98] S. Lee, P. von Allmen, W. Fink, A.E. Petropoulos, and R.J. Terrile. Comparison of multi-objective genetic algorithms in optimizing q-law low-thrust orbit transfers. In *Genetic and Evolutionary Computation Conference Late-breaking Paper, Washington, DC*, 2005.

[99] H. Li and Q. Zhang. Multiobjective optimization problems with complicated pareto sets, MOEA/D and NSGA-II. *IEEE Transactions on Evolutionary Computation*, 13(2):284–302, 2009.

[100] Y. Lu, S. Lu, F. Fotouhi, Y. Deng, and S.J. Brown. FGKA: a fast genetic K-means clustering algorithm. In *ACM Symposium On Applied Computing*, pages 622–623, 2004.

[101] JB MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability*, 1967.

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

[103] Ujjwal Maulik and Sanghamitra Bandyopadhyay. Performance evaluation of some clustering algorithms and validity indices. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(12):1650–1654, 2002.

[104] Ujjwal Maulik, Anirban Mukhopadhyay, and Sanghamitra Bandyopadhyay. Combining pareto-optimal clusters using supervised learning for identifying co-expressed genes. *BMC Bioinformatics*, 10(1):27, 2009.

[105] Mukhopadhyay Anirban Maulik Ujjwal, Bandyopadhyay Sanghamitra. *Multi-objective Genetic Algorithms for Clustering*. Applications in Data Mining and Bioinformatics. Springer, 2011.

[106] Achille Messac, Cyriaque Puemi-Sukam, and Emanuel Melachrinoudis. Aggregate objective functions and pareto frontiers: required relationships and practical implications. *Optimization and Engineering*, 1(2):171–188, 2000.

[107] G.W. Milligan. A Monte Carlo study of thirty internal criterion measures for cluster analysis. *Psychometrika*, 46(2):187–199, 1981.

[108] G.W. Milligan. An algorithm for generating artificial test clusters. *Psychometrika*, 50(1):123–127, 1985.

[109] G.W. Milligan and M.C. Cooper. An examination of procedures for determining the number of clusters in a data set. *Psychometrika*, 50(2):159–179, 1985.

[110] A Mukhopadhyay, U. Maulik, S. Bandyopadhyay, and C.AC. Coello. Survey of multiobjective evolutionary algorithms for data mining: Part ii. *IEEE Transactions on Evolutionary Computation*, 18(1):20–35, Feb 2014.

[111] Anirban Mukhopadhyay and Ujjwal Maulik. Unsupervised pixel classification in satellite imagery using multiobjective fuzzy clustering combined with svm classifier. *IEEE Transactions on Geoscience and Remote Sensing*, 47(4):1132–1138, 2009.

[112] Anirban Mukhopadhyay and Ujjwal Maulik. A multiobjective approach to mr brain image segmentation. *Applied Soft Computing*, 11(1):872–880, 2011.

[113] Deep Malya Mukhopadhyay, Maricel O Balitanas, Alisherov Farkhod, Seung-Hwan Jeon, and Debnath Bhattacharyya. Genetic algorithm: A tutorial review. *International Journal of of Grid and Distributed Computing*, 2(3):25–32, 2009.

[114] Fionn Murtagh. A survey of recent advances in hierarchical clustering algorithms. *The Computer Journal*, 26(4):354–359, 1983.

[115] C.A. Murthy and N. Chowdhury. In search of optimal clusters using genetic algorithms. *Pattern Recognition Letters*, 17(8):825–832, 1996.

[116] A.J. Nebro, F. Luna, E. Alba, B. Dorronsoro, J.J. Durillo, and A. Beham. AbYSS: Adapting scatter search to multiobjective optimization. *IEEE Transactions on Evolutionary Computation*, 12(4):439–457, 2008.

[117] Raymond T. Ng and Jiawei Han. Clarans: A method for clustering objects for spatial data mining. *IEEE Transactions on Knowledge and Data Engineering*, 14(5):1003–1016, 2002.

[118] Conor O'Mahony and Nic Wilson. Sorted-pareto dominance: an extension to the pareto dominance relation and its application in soft constraints. In *11th Workshop on Preferences and Soft Constraints*, page 91, 2011.

[119] Xiaoxue Qian, Xiangrong Zhang, Licheng Jiao, and Wenping Ma. Unsupervised texture image segmentation using multiobjective evolutionary clustering ensemble algorithm. In *IEEE Congress on Evolutionary Computation, 2008. CEC 2008.(IEEE World Congress on Computational Intelligence)*, pages 3561–3567. IEEE, 2008.

[120] I. Radziukynienė and A. Žilinskas. Evolutionary methods for multi-objective portfolio optimization. In *Proceedings of the World Congress on Engineering*, 2008.

[121] M. Ramze Rezaee, BPF Lelieveldt, and JHC Reiber. A new cluster validity index for the fuzzy c-mean. *Pattern Recognition Letters*, 19(3-4):237–246, 1998.

[122] W.M. Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical association*, 66(336):846–850, 1971.

[123] Ramachandra Rao Kurada, K Karteeka Pavan, and AV Rao. A preliminary survey on optimized multiobjective metaheuristic methods for data clustering using evolutionary approaches. *International Journal of Computer Science & Information Technology*, 5(5):57–77, 2013.

[124] A.P. Reynolds and B. de la Iglesia. Managing population diversity through the use of weighted objectives and modified dominance: An example from data mining. In *IEEE Symposium on Computational Intelligence in Multicriteria Decision-Making*, pages 99–106. IEEE, 2007.

[125] A.P. Reynolds and B. De la Iglesia. A multi-objective grasp for partial classification. *Soft Computing-A Fusion of Foundations, Methodologies and Applications*, 13(3):227–243, 2009.

[126] Kazi Shah Nawaz Ripon and Mia Nazmul Haque Siddique. Evolutionary multiobjective clustering for overlapping clusters detection. In *IEEE Congress on Evolutionary Computation, 2009. CEC'09*, pages 976–982. IEEE, 2009.

[127] J. Riquelme, MA Ridao, EF Camacho, and M. Toro. Using genetic algorithms with variable-length individuals for planning two-manipulators motion. In *Proceedings of the International Conference on Artificial Neural Networks and Genetic Algorithms*, pages 26–30, 1997.

[128] Joseph Lee Rodgers and W. Alan Nicewander. Thirteen ways to look at the correlation coefficient. *The American Statistician*, 42(1):59–66, 1988.

[129] P.J Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65, 1987.

[130] J. David Schaffer. Multiple objective optimization with vector evaluated genetic algorithms. In *Proceedings of the 1st International Conference on Genetic Algorithms*, pages 93–100, Hillsdale, NJ, USA, 1985. L. Erlbaum Associates Inc.

[131] P. Scheunders. A genetic c-means clustering algorithm applied to color image quantization. *Pattern Recognition*, 30(6):859–866, 1997.

[132] Subhash Sharma. *Applied multivariate techniques*. John Wiley & Sons. New York, USA, 1995.

[133] W. Sheng and X. Liu. A hybrid algorithm for K-medoid clustering of large data sets. In *Congress on Evolutionary Computation, 2004*, volume 1, pages 77–82, 2004.

[134] S. Shirakawa and T. Nagao. Evolutionary image segmentation based on multiobjective clustering. In *IEEE Congress on Evolutionary Computation, 2009*, pages 2466–2473, May 2009.

[135] N. Speer, P. Merz, C. Spieth, and A. Zell. Clustering gene expression data with memetic algorithms based on minimum spanning trees. In *Evolutionary Computation, 2003. CEC'03. The 2003 Congress on*, volume 3, pages 1848–1855, 2003.

[136] Nidamarthi Srinivas and Kalyanmoy Deb. Muiltiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary computation*, 2(3):221–248, 1994.

[137] K. Stoffel and A. Belkoniene. Parallel k/h-means clustering for large data sets. *Lecture notes in computer science*, pages 1451–1454, 1999.

[138] C.A. Sugar and G.M. James. Finding the number of clusters in a dataset. *Journal of the American Statistical Association*, 98(463):750–763, 2003.

[139] Gilbert Syswerda. Uniform crossover in genetic algorithms. In *Proceedings of the 3rd International Conference on Genetic Algorithms*, pages 2–9. Morgan Kaufmann Publishers, Inc., 1989.

[140] P.N. Tan, M. Steinbach, and V. Kumar. *Introduction to Data Mining*. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA, 2005.

[141] Robert Tibshirani, Guenther Walther, and Trevor Hastie. Estimating the number of clusters in a data set via the gap statistic. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 63(2):411–423, 2001.

[142] D.A. Van Veldhuizen and G.B. Lamont. Multiobjective evolutionary algorithm research: A history and analysis. *Air Force Institute Technology, Dayton, OH*, 1998.

[143] Vladimir Naumovich Vapnik and Vlamimir Vapnik. *Statistical learning theory*, volume 2. Wiley New York, 1998.

[144] T. Velmurugan and T. Santhanam. Computational complexity between K-Means and K-Medoids clustering algorithms for normal and uniform distributions of data points. *Journal of Computer Science*, 6(3):363–368, 2010.

[145] L. Vendramin, R. Campello, and E. Hruschka. A robust methodology for comparing performances of clustering validity criteria. *Advances in Artificial Intelligence*, pages 237–247, 2008.

[146] L. Vendramin, R.J.G.B. Campello, and E.R. Hruschka. On the comparison of relative clustering validity criteria. In *Proceedings of the 2009 Society for Industrial and Applied Mathematics International Conference on Data Mining*, volume 733–744, 2009.

[147] J.H. Ward Jr. Hierarchical grouping to optimize an objective function. *Journal of the American statistical association*, 58(301):236–244, 1963.

[148] Lyndon While. A new analysis of the lebmeasure algorithm for calculating hypervolume. In *Evolutionary Multi-Criterion Optimization*, pages 326–340. Springer, 2005.

[149] 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.

[150] R. Xu, D. Wunsch, et al. Survey of clustering algorithms. *IEEE Transactions on Neural Networks*, 16(3):645–678, 2005.

[151] E. Zitzler and L. Thiele. Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271, 1999.

[152] Eckart Zitzler, Marco Laumanns, and Stefan Bleuler. A tutorial on evolutionary multiobjective optimization. In *Metaheuristics for multiobjective optimisation*, pages 3–37. Springer, 2004.

[153] Eckart Zitzler, Marco Laumanns, Lothar Thiele, Eckart Zitzler, Eckart Zitzler, Lothar Thiele, and Lothar Thiele. SPEA2: Improving the strength pareto evolutionary algorithm, 2001.

[154] Eckart Zitzler and Lothar Thiele. Multiobjective optimization using evolutionary algorithms—a comparative case study. In *Parallel problem solving from nature-PPSN V*, pages 292–301. Springer, 1998.