ENSEMBLE LEARNING FOR OPTIMAL CLUSTER ESTIMATION

Submitted in partial fulfillment of the requirement of Doctor of Philosophy

by

Afees Adegoke ODEBODE

Department of Computer Science

Brunel University London

Abstract

This thesis addresses the importance of understanding the underlying structure of high-dimensional datasets through clustering, considering the vast amount of unlabelled available content on the internet and electronic sources. While clustering ensembles have been proposed in the past, the potential of heuristic search-based ensembles has been relatively unexplored. The thesis presents a novel computational method that combines heuristic search and clustering ensembles, focusing on two crucial issues. Firstly, it establishes a representative solution by effectively subsetting ensembles. The thesis introduces a Gray code implementation that maximises the spread across subsets while minimising differences between them. Secondly, the exhaustive search for the best solution from the representative pool becomes computationally expensive as the dimension and volume increase. An alternative approach based on heuristic search is suggested. This approach evaluates subsets incrementally, similar to the implementation of Gray code, resulting in significant speed gain. However, random mutation hill climbing (RMHC) in heuristic search suffers from finding a suitable solution without guidance, particularly in larger search spaces. The thesis presents an innovative seeding technique that leverages Fiedler vector decomposition and minimum spanning tree (MST) to address this challenge. This technique significantly improves both the quality of solutions and computational efficiency. The proposed methodology is extensively evaluated using simulated and benchmark clustering datasets, employing theoretical and empirical examples. The results demonstrate the high effectiveness of the proposed approach. The key contributions of this thesis include the introduction of Gray code subsetting of ensembles, the incorporation of heuristic-search-based techniques into clustering ensembles, and the novel improvement in search space convergence through effective seeding.

Acknowledgements

I would like to express my profound gratitude to the individuals who have played a crucial role in making this work possible. Firstly, I am immensely grateful to my supervisor, Dr Stephen Swift, whose unwavering support and guidance have been invaluable throughout this research journey. Despite facing personal challenges and the additional difficulties posed by the COVID-19 pandemic, Dr Swift's patience and encouragement have been instrumental in helping me overcome obstacles and achieve my research objectives. I owe a great deal of my research accomplishments to his mentorship and the resources he provided, which have allowed me to develop my research skills.

I am also indebted to my second supervisor, Dr Allan Tucker, for his invaluable assistance in various aspects of my research. His passion for scientific inquiry and wide-ranging knowledge across different fields have greatly enriched my thesis. I also want to express my gratitude to Dr Mahir Arzoky, my mentor, whose meticulousness and insightful suggestions have been integral to the success of this work. Additionally, I am thankful to my committee members, including Professor Steve Counsell and Dr Stasha Lauria, for their constructive feedback and guidance throughout the research process.

I extend my heartfelt appreciation to the members of the IDA group for their warm welcome and support since my first day at the university. The department's opportunities for enhancing my teaching skills through Graduate Teaching Assistantship roles have been beneficial to both my research and academic development. I am grateful for the support and meaningful conversations shared with my colleagues, including Ashley Mann, Faisal Maramazi, Ben Evans, Fawzia Kara-Isitt, and many others, as they have profoundly impacted my research.

I would also like to thank my friends, including Zear Ibrahim, Anas, Futra, and others, for their continuous encouragement and assistance throughout this endeavour. Furthermore, I am grateful for the love and support of my students and colleagues at BPC and Regent. I want to express my profound appreciation to my parents and siblings for their unwavering love and support, which has been a constant source of motivation and inspiration.

Last but not least, I am incredibly thankful to my wife and children, whose boundless patience, prayers, and love have propelled me towards the finish line. Their unwavering support has been an endless source of strength and encouragement throughout this journey.

Declaration

The following papers have been submitted/accepted for publication as a direct result of the research discussed in this thesis:

- Afees Adegoke Odebode, Allan Tucker, Mahir Arzoky, Stephen Swift, 2022 “Estimating the Optimal Number of Clusters from Subsets of Ensembles” Proceedings of the 11^{th} International Conference on Data Science, Technology and Applications.
- Faisal Maramazi, Afees Adegoke Odebode, Stephen Swift, Ashley Mann, Mahir Arzoky, 2023 “How Starting Points and Representations Affect Software Modularisation: An Empirical Analysis” Intelligent System Conference (IntelliSys) 7- 9 September, 2023.
- Afees Adegoke Odebode, Allan Tucker, Mahir Arzoky, Ashley Mann, Faisal Maramazi, Stephen Swift, 2023 “Using Clustering Ensembles and Heuristic Search To Estimate The Number Of Clusters In Datasets” Intelligent System Conference (IntelliSys) 7- 9 September, 2023.

Contents

Contents v

List of Figures x

List of Tables xii

List of Equations xiv

1 Introduction 1

1.1 Motivation 2

1.2 Methodology 3

1.2.1 Overview of the critical stages in the thesis 4

1.3 Contributions 7

1.4 Summary of Thesis 9

2 Literature Review 12

2.1 Introduction 12

2.2 Clustering 12

2.2.1 Clustering Applications 14

2.2.2 Applications 15

2.2.3 Measures of Similarity 17

2.3 Clustering Methods 20

2.3.1 Partitioning Around Medoids (PAM) 23

2.3.2 X-Means 25

2.3.3 Clustering Large Applications (CLARA) 26

2.3.4 Fuzzy Clustering 28
2.3.5 Hierarchical Clustering 29
2.3.6 Density-based Methods 32
2.3.7 ccfkms:Clustering with Conjugate Convex Function . 34
2.4 Cluster Validity: Evaluating Clustering 34
2.4.1 Cluster Validity Index (CVI) 35
2.4.2 Evaluating Cluster Validity Index 36
2.4.3 Gap Statistics 37
2.4.4 The Average Silhouette 37
2.5 Clustering Ensemble 43
2.5.1 Advantages of Using Ensemble for Clustering 45
2.5.2 Clustering Ensemble Representation 47
2.5.3 Search Space 49
2.6 Search Strategies: Exhaustive Search 51
2.7 Heuristic Search Algorithms 53
2.7.1 Genetic Algorithms and Exploratory Data Analysis . . 54
2.7.2 Greedy Search 55
2.7.3 Hill Climbing Algorithms 56
2.7.4 Fitness in a Search Space 59
2.7.5 Convergence in a Search Space 61
2.7.6 Simulated Annealing 62
2.7.7 Genetic Algorithms 64
2.8 Evolutionary Approaches to Clustering 65
2.9 Related Studies 66

3 Description of Datasets 68

3.1 Introduction 68
3.2 Dataset Description 68
3.3 Data Collection and Pre-Processing 69
3.4 Dataset Characteristics and Features 71

3.5 Clustering Ensemble Generation 78
3.5.1 R and The List of Packages 79
3.6 Variability by Method and Dataset 81

4 ESTIMATING NUMBER OF CLUSTERS USING THE ENSEMBLE FRAMEWORK 85

4.1 Introduction 85
4.2 Background 86
4.2.1 Why Gray Codes? 88
4.2.2 How Does a Cluster Estimator Work? 89
4.3 Estimating the Number of Clusters 90
4.4 The Ensembles Framework 94
4.4.1 Subsets Generation 95
4.4.2 Determining The Best Subset 97
4.5 Quality Function Description 98
4.5.1 The Quality Function 98
4.5.2 The Update Quality Function ($\hat{Q}$) 99
4.6 Results and Discussions 102
4.6.1 Estimated Errors 103
4.6.2 Quality Vs Update Quality 106
4.7 Recommendations 108

5 HEURISTIC SEARCH BASED CLUSTERING ENSEMBLE 110

5.1 Introduction 110
5.2 Background to The Study 111
5.3 The Ensemble Framework 112
5.3.1 Generation of the Base Clustering 113
5.3.2 Construction of The W_k Agreement Matrix 115
5.3.3 Sub-setting the W_k Agreement Matrix 116
5.3.4 Experimental data 116

5.3.5 Selection of The Best Subset 118
5.3.6 The Fitness Function 118
5.4 Methods 120
5.4.1 The Exhaustive Approach 120
5.4.2 Random Mutation Hill Climbing (RMHC) 121
5.5 Experimental Procedure 123
5.6 Results and Discussions 125
5.7 Recommendations 131

6 Seeding Using The Fiedler Vector Decomposition for Clustering Ensemble Search 133

6.1 Introduction 133
6.2 Minimum Spanning Tree 135
6.3 Fiedler Vector Decomposition 136
6.4 Motivation for Seeding 138
6.5 Arnoldi Iteration 140
6.6 Average Quality of the Search Space 140
6.6.1 Empirical Proof of the Average Quality 141
6.6.2 Mathematical proof of Average Quality 142
6.7 Experimental Data 147
6.8 Methods 148
6.9 Experiments 148
6.10 Results and Discussions 150
6.10.1 Results by Quality 150
6.10.2 Results by Convergence Points 152
6.10.3 Results by State 153
6.11 Recommendations 157

7 Conclusions **159**

7.1 Contributions 162

7.1.1 Gray Code Subsetting 162

7.1.2 Assessing Subsets Based on Quality 163

7.1.3 Quality Metric Framework 163

7.1.4 Seeded RMHC 164

7.1.5 Proof of the Search Space Fitness Average 164

7.2 Summary 165

7.3 Limitations 166

7.3.1 Generating Ensembles 167

7.3.2 Seeded RMHC 167

7.4 Further Work 168

Appendices **171**

Appendix A: Results from Simulated Datasets 171

Appendix B : Datasets Description 172

Appendix C: Partioning adjacency matrix using FVD 173

Bibliography **176**

List of Figures

1.1 Basic Framework of the Ensemble 6

2.1 Figure Showing a Dendrogram of Iris Dataset 33
2.2 Steps in a General Exhaustive Search. 53
2.3 State Space Landscape of a Hill Climbing Algorithm. [1] . . . 58

3.1 Stages in Data Preprocessing 71
3.2 Visualisation of Atom Dataset 73
3.3 Visualisation of Hapta Dataset 74
3.4 Visualisation of Lsun Dataset 75
3.5 Visualisation of Tetra Dataset 75
3.6 Visualisation of LongSquare Dataset 77
3.7 Shapes Dataset Visualisation 78
3.8 Variability of Methods based on Weighted Kappa Values . . . 82
3.9 Variability of Datasets based on their Weighted Kappa values 83

4.1 Representation of Optimal Number of Clusters in a Dataset. . 90
4.2 Ensemble Framework 94
4.3 Normalised Average Errors on the Twenty-Seven Datasets . 104
4.4 The speed gain for Exhaustive 106
4.5 The speed gain of Exhaustive Vs $\hat{Q}$ 107

5.1 The Modified Ensemble Framework 113

5.2 A plot Showing the Linear Regression Model by Average and Maximum Convergence Points. 129
5.4 The Accuracy of RMHC Compared with Exhaustive 130
5.3 RMModel vs Exhaustive Convergence point 130

6.1 A Simple Graph with Five Nodes 136
6.2 The Minimum Spanning Tree of The Graph 136
6.3 Fiedler vector decomposition 137
6.4 The Variability as Average Search Space Converges to Zero (Simulated Dataset for Matrix Size 10 and 11) 142
6.5 Quality Value for Methods: Shows the Average Quality Values for Each Method. 153
6.6 Convergence Values Methods: The figure shows the convergence values for each method in the legend on average from data size 13 to 30 154
6.7 The figure shows the number of outcomes for State 3 of the FVD for all datasets 155
6.8 The Figure Shows the Efficiency by Dataset Size 156

List of Tables

2.1 Showing list of methods for determining the optimal number of clusters in datasets 38
2.2 The Weighted Kappa guideline 48

3.1 Dataset by rows, columns and number of clusters 72
3.2 Description of methods used in clustering the dataset 81

4.1 A summary of the indices implemented in NBClust [2] 92
4.2 Errors from the Eight Methods and the Average Ensemble . . 102
4.3 Errors for the Ensembles(Fair, Moderate, Median and Average) 103

5.1 Dataset, Attributes and Number of Clusters with k_{max} 117
5.2 Table of Experiments with the Methods, Datasets, Number of Iterations, and Repeats. 125
5.3 A summary of the Average and Standard Deviation by Dataset 126
5.4 A Summary of the Average and Standard Deviation by Input Size 128

6.1 FVD table of experiments: Showing the methods, datasets, number of iterations, and repeats. 150
6.2 Table showing the quality values for dataset size 5 to 30 based on the methods 151
6.3 State Distributions 155

List of Algorithms

2.1 The K-means Algorithm 22
2.2 Partitioning Around Medoids (PAM) 25
2.3 CLARA (Clustering Large Applications) 27
2.4 Single Linkage Hierarchical Clustering 30
2.5 Complete Linkage Hierarchical Clustering 31
2.6 Average Linkage Hierarchical Clustering 32
2.7 Gap Statistics 37
2.8 The Average Silhouette Algorithm 38
2.9 Random Mutation Hill Climbing (RMHC) 59
2.10 Simulated Annealing 63
2.11 The Genetic Algorithm 65
4.12 Gray Code Implementation of Exhaustive Search Algorithm . 97
5.13 Smallchange between Subsest 115
5.14 Random Mutation Hill Climbing (RMHC) 123
6.15 Prims Minimum Spanning Tree (MST) 137

List of Equations

2.1 Euclidean Distance 18
2.2 Scaled Euclidean Distance 18
2.3 Minkowski Distance 19
2.4 Weighted Kappa 47
2.5 Adjusted Rand Index 48
4.1 Quality Function 99
4.2 Average Fitness Function 99
4.3 Update Quality Function 101
4.6 Error Estimate 105
6.4 Average Quality For $\bar{Q}$ 144

Notations

V	Set of objects to be clustered
n	Number of columns in the dataset, $n = \|V\|$
C	Clustering arrangement
m	The number of rows in a dataset
k	Number of clusters
r	Number of input methods (clustering arrangements)
E	Set of input cluster
e_i	Individual input cluster
$\|X\|$	Size of object X
i, j	Index variable for notation
W_k	Weighted Kappa matrix of size r by r
w_{ij}	Weighted Kappa between cluster i and cluster j
$Q(s, W, \theta)$	Quality metric of subsets s applied to W with threshold θ
θ	Quality function threshold
D	Dataset of size n by m
$\hat{s}$	Number of comparison when evaluating Q on s
$\mu(X)$	Average of object X
$\hat{Q}(s, W, \theta)$	Updated version of Q
p_o	Relative observed agreement among raters
p_e	Probability of chance agreement
$G(D)$	Gold standard number of clusters for dataset D
$H(X, D)$	Predicted number of clusters for method X on dataset D
A	Agreement Matrix
E_d	Euclidean Distance
B_k	Between cluster sum of squares
C_k	Cluster k
U	Number of vertices
$C(V, k)$	Number of combinations of selecting k elements from a set of V elements

x, y	x and y are vectors of equal dimensions
α	weight in the scaled euclidean distance
$trace$	The sum of diagonal elements of a square matrix.
$DIFF$	Difference in the function when the number of groups in the partition is increased
$ptbiserial$	Point biserial correlation
$dCor$	The distance correlation between two variables x and y
cc_i	The cluster centroid of cluster c_i
$J_e(2)$	Error rate of the second largest eigenvalue
$J_e(1)$	Error rate of the largest eigenvalue
C_v	Matrix representing the variability within the clusters for the k clusters
C_p	The observed within cluster dispersion matrix for the k clusters
S_i	Silhoutte score for the i^{th} data point
δ_{kl}	The distance between the two centroids
$d(C_i, C_j)$	The dissimilarity function between two clusters C_i and C_j
$diam(C_k)$	Diameter of cluster C_k

Abbreviations

CVI	Cluster Validity Index
BIC	Bayesian Information Criterion
MST	Minimum Spanning Trees
FVD	Fiedler Vector Decomposition
CH	Calinski and Harabasz Index
$Duda$	Duda and Hart
KL	Krzanowski and Lai
ARI	Adjusted Rand Index
FCM	Fuzzy C-means
CCC	Cubic Clustering Criterion
KL	Krzanowski and Lai
$Trcovw$	Trace of Covariance Matrix
COV	Covariance
$Marriot$	Marriot index
$Tracew$	Tracew index (assesses the quality of clustering in hierachical clustering
$Friedman$	Friedman index
$sDbw$	Scatter Density between clusters index
$SDindex$	Scatter and Distance index
$Dunn$	Dunn index

Chapter 1

Introduction

The exponential increase in data collection and generation from various sources, such as mobile devices, sensors, and video surveillance, has led to unprecedented growth of data available for analysis. With increased accessibility to storage and data collection on cost-effective electronic media [3], new opportunities arise for identifying the structures in datasets through the number of clusters to understand the data selecting the right number of clusters is important for algorithms such as k-means, and it can also affect the interpretability of the clustering result. Cluster analysis is an essential technique for pre-processing high-dimensional datasets. Likewise, most datasets require labelling or identifying the underlying structure; for instance, data generated by the healthcare industry or data from social media sources [4]. However, finding patterns or clusters within datasets is still challenging, and it remains an unresolved issue in data analysis due to the unambiguous definition of a “cluster”. The most intuitive way to extract information from this unlabelled data is to establish similarity between the objects and group objects into clusters based on their similarity, known as data clustering. The main idea is that the objects are similar within a group, and there are clear differences between groups [5].

In the recent past, several single clustering algorithms have been developed and applied in different contexts and application areas, such as bioinformatics [6], machine learning [7], data mining [8], market basket analysis, pattern recognition and for image processing [9]. Single clustering algorithms perform well in some contexts or data types. They may perform poorly in others because different clustering algorithms have different clustering criteria besides being context or data-sensitive.

Additionally, varying the methods' parameters produces different results for the same dataset. Besides, the clustering process may involve choosing between clustering techniques and setting the correct parameter, which is often difficult for a single clustering method. Therefore, it is hard for practitioners to determine which algorithm is most suitable for determining the number of clusters in a dataset, especially for single clustering algorithms. In recent times, there has been growing interest in the use of ensemble techniques for cluster analysis. Ensemble clustering [10], rather than relying on a single algorithm, evaluates clustering from algorithms or multiple instances of an algorithm to analyse clusters and estimate the number of clusters in a dataset.

1.1 Motivation

This study is partly motivated by an earlier research on detection of outliers for large smart meter datasets titled:"A Sampling-Based Clustering Scheme for Large Datasets [11]".The objective of this study is to analyse customers' electricity consumption patterns by clustering them into distinct labels identified within the dataset. With the help of smart meters, which collect millions of records daily, it is crucial for Distribution Network Operators (DNOs)

to gain a comprehensive understanding of these records through analysis. This understanding is essential to offer appropriate and efficient consumer services. The study combines the k-means algorithm and k-nearest neighbour to detect anomalies in the dataset. The study found that a single clustering algorithm may not effectively handle large datasets and may fail to detect anomalies in some cases. The current study expands the idea and seeks more accurate and efficient methods for determining the number of clusters in datasets using ensembles instead of a single clustering algorithms. This research is further motivated by the need to improve the clustering solution's quality and robustness across a wide range of high-dimensional datasets. This thesis further proved the efficacy of combining a heuristic optimisation technique with clustering ensembles to improve cluster estimates of high-dimensional datasets.

This work illustrates how combining heuristic optimisation with a clustering ensemble can effectively estimate the number of clusters in datasets. Clustering ensembles allow flexible definitions of the objective function to match various criteria and goals. Through rigorous experimentation and a sound mathematical framework, the efficacy of clustering ensembles was established, and new methods were designed for determining the number of clusters in high-dimensional datasets.

1.2 Methodology

This thesis examines the development of a robust clustering ensemble framework that correctly estimates the number of clusters in a dataset, carefully examining key components such as dataset selection, validation, generation of base clusterings, transformation into an agreement matrix, creation of subsets, formulation of quality metrics, and the establishment of a novel evaluation framework for subset quality. Refer to Figure 1.1 for a comprehensive process diagram illustrating the research stages.

1.2.1 Overview of the critical stages in the thesis

The following sections provide a brief overview of the critical stages involved in this thesis:

Database Selection and Preprocessing

Like every data mining and exploratory data analysis, the initial phase of the clustering ensemble involves the meticulous selection and preprocessing of datasets. This process prioritises diversity, size, type, and domain relevance to ensure a comprehensive dataset representation encompassing various clustering problems and scenarios. Benchmark datasets are included for analysis, and to establish a solid foundation, the reported number of clusters for each dataset is cross-validated across multiple repositories, ensuring reliability and correctness. Appendix B contains a list of datasets used in the analysis, including metadata and sources.

Generation of Base Clusterings

Base clusterings serve as the foundation of any clustering ensemble system. Diverse clustering algorithms are applied to the selected datasets to enhance performance and reliability, generating varied base clusterings. This diversity aims to capture different facets of the underlying data structure, increasing the likelihood of uncovering latent patterns and improving overall clustering accuracy and the number of clusters in the dataset. Similar concepts have been explored in works by Ayed et al. [12] and Swift et al. [13].

Transformation into Agreement Matrix

Creating an agreement matrix is crucial for consolidating diverse perspectives within the base clusterings. This matrix quantifies the agreement

among individual clustering results, providing a foundation for subsequent ensemble processing. Consensus functions and similarity measures are vital in transforming the base clusterings into a unified representation.

Creation of Subsets

Utilising the agreement matrix as a foundation, subsets are systematically generated, each offering a unique perspective on the dataset. The Gray code subsetting methodology ensures minimal changes between subsets, contributing to the overall robustness and stability of the clustering ensemble used in this research. Chapter 4 and 5 provide a more detailed discussion on subsetting, including alternative methods adopted to improve search strategy for heuristic search algorithms.

Formulation of Quality Metrics

Quality metrics are formulated to comprehensively evaluate various aspects, including clustering accuracy and cluster separation across different subsets generated from the base clusterings. These metrics offer insights into how well the ensemble captures intrinsic structures within the dataset and the reported number of clusters.

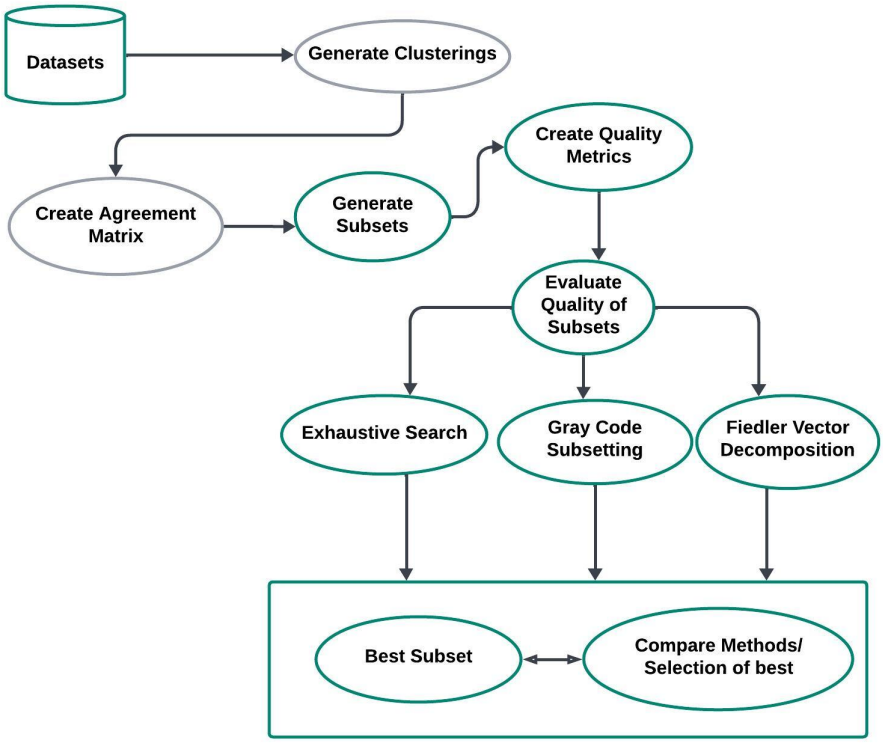


Figure 1.1: Basic Framework of the Ensemble

Evaluation Framework

The thesis adopts a rigorous evaluation framework to thoroughly assess the quality and effectiveness of different subsets in uncovering latent patterns and enhancing clustering performance. This framework serves as a critical tool for gauging the success of the clustering ensemble system. The evaluation framework is based on quality function further described in chapters 4, 5, and 6.

Method Comparison/Selection of Best Subset

A crucial aspect of this research involves a comparative analysis of various methods employed for evaluating subsets. The optimal subset selection is determined through a meticulous examination of performance metrics,

enabling the identification of the most effective clustering ensemble configuration for the given dataset, the index of the best subset corresponding to the number of clusters in the dataset. The comparison encompasses a range of standard techniques used to determine the number of clusters compared with the clustering ensemble techniques developed in this thesis. The outcomes of these comparisons are detailed in Chapters 4, 5, and 6.

1.3 Contributions

The first significant contribution is the introduction of the Gray Code subsetting methodology. This approach systematically generates potential solutions by creating subsets of ensembles derived from clusterings. These subsets are meticulously constructed to exhibit minimal differences between them. An advantage of Gray Code sub-setting lies in its ability to establish meaningful relationships among the generated subsets. This relationship-based framework facilitates the calculation of fitness values for each subset, significantly enhancing the efficiency of the search process.

The second key contribution is the development of an enhanced fitness function, which is pivotal in ensuring each subset's quality and accuracy. The fitness function is essential for assessing the quality of subsets used to estimate the number of clusters in a dataset. It functions as a quantitative measure, evaluating how well a subset represents the underlying structure of the dataset. The fitness function assigns higher scores to subsets that closely match the actual distribution of clusters, signifying their effectiveness in estimating the number of clusters. Unlike conventional methods relying on statistical formulas influenced by data characteristics, our proposed approach eliminates such dependencies. The method provides a robust and

objective measure of subset quality, incorporating the fitness function in the evaluation process and utilising the agreement matrix. This innovative approach introduces flexibility and may mitigate the limitations typically associated with statistical formulas used in traditional methods.

A third noteworthy contribution is the creation of a Quality Metric Framework, establishing a mathematical basis for scoring subsets within the evaluation process. This framework systematically evaluates weaker and stronger subsets, considering a threshold value for relative strength comparison. Its flexibility allows adjustments to align with the unique characteristics of the search space, offering a comprehensive and adaptable evaluation process. This framework gives researchers and practitioners a structured approach to score subsets and evaluate their strengths.

The final major contribution addresses the inefficiencies of traditional Random Mutation Hill Climbing (RMHC) as explained in Chapter 6. A Seeded RMHC approach is proposed, utilising Fiedler vector decomposition for seeding. This technique significantly improves search space exploration by addressing limitations such as slow convergence and sensitivity to the initial starting point.

In the context of traditional RMHC, one of the limitations is its susceptibility to local optima, where the algorithm may become trapped in suboptimal solutions, restricting its ability to explore the global search space effectively [14]. Additionally, the reliance on random mutations in RMHC may hinder thorough exploration, potentially overlooking diverse regions in the search space [15]. The proposed Seeded RMHC technique aims to overcome these limitations, demonstrating its effectiveness in achieving better solutions in the search for optimal subsets.

1.4 Summary of Thesis

This thesis is delivered in seven chapters; and organised as follows:

Chapter 1 presents a concise overview of the thesis, including the motivation behind the research and a general introduction to the main contents. It summarises the entire thesis, providing readers with a comprehensive understanding of the work.

Chapter 2 provides a literature review of this research project's concepts, techniques and methods. The initial focus is on providing an overview of clustering techniques, including partitional and hierarchical methods. Additionally, various similarity metrics used in the clustering process are explained. Next, methods for estimating the optimal number of clusters and cluster validity are described, followed by a review of heuristic search algorithms and the ensemble technique applied to improve the search for the optimal number of clusters. Finally, elements of meta-heuristic search utilised for this research are described.

Chapter 3 outlines and describes in detail the datasets' unique clustering characteristics, the techniques employed to generate the clusterings, and the pre-determined parameters for selecting the dataset applied in later chapters. A comprehensive analysis of the datasets was conducted, sourced from multiple repositories and consisting of both artificial and standard clustering benchmark datasets. Selection of the datasets and reasons for eliminating some of the datasets from experiments are provided in this chapter. Equally, in the chapter, a variability analysis was conducted on the selected datasets to serve as a guide in evaluating the stability and robustness of the clustering results. Also, variability analysis helps estimate the number of clusters in the datasets and supports understanding dataset patterns providing valuable insights into the datasets' hidden structures.

Chapter 4 introduces the ensemble framework for determining the number of clusters in a dataset. The chapter is based on a published work [16], introducing Gray code sub-setting and its use in evaluating the subset's quality for estimating the optimal number of clusters in datasets. Lastly, this chapter compares methods for combining the ensembles to achieve maximum diversity with standard methods for estimating the number of clusters.

Chapter 5 introduces a heuristic search-based ensemble built on top of the framework in Chapter 4. The aim is to further improve the speed of execution for the same accuracy as the exhaustive search. The chapter introduces specific algorithms and their application to enhance the ensemble search for the best subset. The chapter detailed the search space and different methods to improve efficiency and accuracy. The algorithm used on or modified to meet the specific requirement of the search space includes Hill Climbing and Simulated Annealing (SA), both of which have earlier been introduced in Chapter 2.

Chapter 6 presents the introduction of seeding in the ensemble search space as a means to improve the search for optimal solutions. The chapter explores the utilisation of concepts such as the Fiedler vector decomposition and Minimum Spanning Trees (MST) to identify an appropriate starting point in the search and guide it towards optimal solutions. The effectiveness of this approach is demonstrated through benchmarking, simulations, and rigorous experimentation. Mathematical proofs are provided to establish the average fitness value, which approximates zero. Furthermore, the chapter concludes with a comprehensive comparison of exhaustive search, SA, RMHC, and seeded Fiedler vector approaches. The results highlight that guided search significantly enhances both the outcome and computational efficiency when determining the optimal number of clusters in datasets.

Chapter 7 summarises the thesis and reports the main findings and contributions. This chapter also acknowledges potential limitations inherent in the proposed methods and suggests promising avenues for future research. The chapter provides a concise overview of the key points and outcomes presented throughout the thesis, ensuring a comprehensive understanding of the research conducted. It highlights the significance of the research, emphasising its relevance and implications for future work. Limitations of the current approaches are critically assessed, acknowledging potential constraints or drawbacks that could have impacted some results or conclusions. Finally, the chapter concludes by outlining possible directions for future research work.

Chapter 2

Literature Review

2.1 Introduction

This chapter is a review of concepts applied in this research. The first part of the review is an introduction to clustering, followed by methods for estimating the optimal number of clusters and cluster validity. The second part reviews clustering ensembles and key terminologies. The last section of the review is on the heuristic search algorithms applied to the ensemble technique that has been used to improve the search for the optimal number of clusters in the selected datasets

2.2 Clustering

Clustering can be described as an unsupervised learning process that groups data into clusters so that data objects within a cluster are similar but dissimilar to objects in other clusters [5]. Clustering is also the formal study of methods and algorithms for grouping objects according to measured or perceived intrinsic characteristics or similarities [5, 17, 18]. In grouping objects, clustering organises objects into patterns based on similarity; the pat-

terns are usually points in multidimensional space where patterns that belong to the same cluster are closely similar compared to patterns that belong to another cluster [5].

Clustering is an unsupervised technique that utilises observation values to uncover the inherent structure within a dataset [19]. However, it is essential to acknowledge that different clustering techniques can yield varying results based on how the cluster patterns are formed and interpreted, as pointed out by Christen [20]. To select an appropriate clustering methodology, researchers should understand and consider the underlying data characteristics exhibited by the clusters they aim to identify. Clustering plays a pivotal role in exploratory data analysis (EDA) by facilitating the identification of inherent structures and patterns within datasets. As part of EDA, clustering methods help reveal similarities and groupings among data points, providing insights into the underlying distribution and relationships within the data. Clustering algorithms, such as k-means, hierarchical clustering, and DBSCAN, assign data points to clusters based on their intrinsic characteristics, assisting analysts in discerning natural divisions in the dataset. These unsupervised learning technique is particularly valuable for understanding complex datasets where relationships among variables may not be immediately apparent. For instance, in a study by [3], hierarchical clustering was employed to explore patterns in gene expression data, revealing distinct gene expression profiles. Clustering, thus, serves as a powerful tool in the exploratory phase, aiding researchers and analysts in uncovering meaningful structures that may inform subsequent analyses or guide further investigations [5].

Therefore, in Chapter 3, a variability analysis was conducted on the dataset to gain insights into the effect of these characteristics. It is crucial to note that the identified clusters should be viewed as valuable constructs that facilitate

clarification and transparent comparison of cluster estimates in the dataset rather than being perceived as the absolute truth in clustering [21]. Thus, validation may be necessary to further verify the reliability and accuracy of the clustering results.

2.2.1 Clustering Applications

A clustering arrangement can be defined as follows:
Let D be a dataset consisting of m data points, denoted as $D = d_1, d_2, ..., d_m$, where each d_i is a data point in the dataset. A clustering arrangement is a partitioning of D into k clusters, denoted as $C = \{C_1, C_2, ..., C_k\}$, such that the following conditions holds:

1. Each cluster C_i is non-empty, meaning that $C_i \neq \emptyset$ for all $i = 1, 2, ..., k$.
2. The union of all clusters covers the entire dataset, meaning that

$$C = \bigcup_{i=1}^{k} C_i$$

   for all $i = 1, 2, ..., k$.

3. The clusters are pairwise disjoint, meaning that $C_i \cap C_j = \emptyset$ for all $i \neq j$, where $i, j = 1, 2, ..., k$.
4. Within each cluster, the data points share common characteristics or features, while dissimilarity is observed between data points belonging to different clusters. Thus, the data points in cluster C_i are less similar to those in cluster C_j when $i \neq j$. This requirement ensures that the clustering arrangement effectively differentiates between distinct groups of data points within the dataset.

In summary, a clustering arrangement groups the data points into k clusters, ensuring that each cluster is non-empty and the clusters collectively

cover the entire dataset with no overlap (crisp clustering). Relaxing the requirement for disjoint clusters can provide more flexibility and adaptability in clustering, particularly when dealing with complex, multidimensional, or overlapping data sets. However, this relaxation can introduce challenges in interpreting results and assessing the clustering quality.

2.2.2 Applications

Clustering is crucial in numerous applications, particularly when dealing with high-dimensional datasets. Its versatility and usefulness extend across diverse fields and industries, providing valuable insights and practical solutions to a wide range of problems. Below are some prominent areas where clustering finds significant application.

Gene Expression Analysis

Molecular biology generates vast amounts of data comprising gene expression levels measured under various conditions, experimental parameters, and environments, as well as from different individuals. Employing clustering approaches can be instrumental in validating the presence of diverse measures, such as proteins or regulatory RNAs, within cells or membranes [22]. Uncovering underlying patterns within these datasets is pivotal in biological research. For example, in gene expression data, each column represents an individual, providing insights into their health status and the presence of metabolic diseases. Clustering the data based on genetic relationships and leveraging phenotypic features makes it possible to identify specific disorders, enabling personalised treatment approaches [23] and facilitating early disease detection [24].

Software Modularisation

Software clustering is another important application. It is a method used in software engineering to cluster related software modules or components into groups based on their similarity in terms of functionality or other relevant characteristics; entities within the system with common features are grouped, and entities with different features are kept apart. More details and examples can be found in the following references [25, 26, 27]. Software clustering can cluster related software components together, allowing developers to better manage and maintain them as a group, reducing the risk of errors and simplifying the debugging process. Through software clustering, software systems scale more efficiently by allowing developers to add or remove components as needed. This can help prevent performance issues and ensure the system handles the increased workload. Finally, Clustering will enable developers to reuse components across different software systems, reducing development time and costs.

Customer Recommendation Systems

Another application of clustering is seen in customer recommendation systems, also known as recommender systems. In such scenarios, customers can express their preferences for various products within a company by providing votes or ratings. Nowadays, recommendation systems are widely deployed on numerous websites, catering to millions of customers and enhancing targeted marketing efforts [28, 29].

Text Documents Organisation

The huge collection of documents accessible from the Internet, digital libraries and repositories are piling up at an astronomic rate. These require the effective and efficient organisation of text documents [30]. Clustering

methods can be applied to organise a collection into meaningful groups. Collections of text documents such as pages on the Web or data collected through Web Scrapping can be transformed into high-dimensional feature vectors, which may contain thousands of attributes grouped into different themes. For example, online stores cluster customer responses to provide collaborative recommendations [31]. The other application area includes image processing [32] and anomaly detection [33].

2.2.3 Measures of Similarity

A similarity measure is a quantitative measure that determines how related two objects are in a cluster [34]. It is commonly used in machine learning, natural language processing, image recognition, and information retrieval to compare and classify data. The similarity of objects within a cluster plays a vital role in the clustering process. A well-defined similarity measure is fundamental to identifying a cluster; distance measures differ for different data requirements due to the variety of feature types and scales. An essential characteristic of a distance measure is that it should be symmetric [34]. The focus in this review of similarity measures will be limited to datasets that exhibit continuous features, while relevant citations will be provided for other measures. The standard similarity measures are Hausdorff distance [35], Modified Hausdorff (MODH) [36], HMM-based distance [37], Euclidean distance [38], Manhattan distance [39], Minkowski Distance [3], Hamming distance [40] and Pearson Correlation [41]. Two measures will be described here because of their relevance to the current research.

Euclidean Distance

The Euclidean distance E between two points, say (x_1, y_1) and (x_2, y_2) is the geometric distance between the points and is defined as:

$$E(x, y) = \sqrt{(x_2 - x_i)^2 + (y_2 - y_1)^2} \tag{2.1}$$

The formula can be extended to higher-dimensions, and it has many advantages; one is that adding a new object to the dataset does not affect the existing distance between objects. However, the difference in scale can strongly affect the dimension from which the distances are computed, and it may not be suitable for some types of data. Euclidean distance is also referred to as the $L2 - Norm$ because it is calculated using the squared value of the vector components. There is a variation of Euclidean distance called the Scaled Euclidean Distance (SED) [42]; it considers the scale or weight of each feature or dimension in a vector. Incorporating these scales into the distance calculation gives higher scales more importance and less to dimension with lower scales. The formula for calculating scale distance is:

$$\text{SED}(x, y) = \sqrt{(\alpha_1 \cdot (x_1 - y_1))^2 + \ldots + (\alpha_n \cdot (x_n - y_n))^2} \tag{2.2}$$

where x and y represent the vectors with corresponding feature values $(x_1, x_2, \ldots, x_n)$ and $(y_1, y_2, \ldots, y_n)$, respectively. α is the vector of weights or scales $(\alpha_1, \alpha_2, \ldots, \alpha_n)$ assigned to each dimension. The formula calculates the Scaled Euclidean distance by summing the squared differences between each corresponding feature value, scaled by its respective weight, and taking the square root of the sum.

Minkowski Distance

Minkowski distance is a metric used to measure the distance between two vectors in a Euclidean space of any dimension and a generalisation of the Euclidean and the Manhattan distance ($L1 - norm$). The Minkowski distance between two vectors $X = (x_1, x_2, x_3, ..., x_n)$ and $Y = (y_1, y_2, y_3, ..., y_n)$ is defined as follows:

$$\left(\sum_{i=1}^{n} |x_i - y_i|^p\right)^{1/p} \quad \text{where} \quad p \geq 1 \tag{2.3}$$

X and Y are vectors of equal dimensions and p controls the behaviour of the distance metric. When $p = 1$, it is referred to as the Manhattan distance, and when it is equal to 2, it is referred to as the Euclidean distance and when $p = \infty$ it is called the Chebychev metric [34].

Distance Correlation

Distance correlation is a statistical measure that quantifies the dependence between two random variables or datasets [43]. It evaluates the similarity of the underlying relationship by considering both linear and nonlinear associations. Unlike traditional correlation measures that only capture linear dependencies, distance correlation considers all types of associations, including nonlinear and higher-order relationships. The distance correlation between two random variables X and Y denoted as Corr(X, Y), is calculated based on the distances between the observations in the joint space of X and Y [44]. It is defined as the normalized covariance of the distances:

$$\text{Corr}(X, Y) = \frac{\text{cov}(x, y)}{\sqrt{\text{var}(x) \cdot \text{var}(y)}}$$

X and Y are matrices of pairwise distances between the observations in x

and y, respectively.

The distance correlation ranges from -1 to 1, where 0 indicates no dependence or independence between the variables, and 1 represents a perfect dependence or relationship. A distance correlation close to 0 suggests a weak or negligible relationship, while a value close to 1 indicates a strong association. Distance correlation is a versatile measure applied to various data types, including continuous, discrete, and mixed-type variables. It is useful in many areas of statistical analysis, such as feature selection, dimensionality reduction, and exploring complex relationships in data.

2.3 Clustering Methods

Clustering methods comprise various approaches, including partitional and hierarchical clustering based on specific clustering characteristics [17, 21, 5]. Additionally, there are other families of techniques, such as density-based and graph-based clustering, among others.

Hierarchical clustering groups data objects into a sequence of partitions, from singleton clusters to a cluster including all individuals, referred to as agglomerative hierarchical clustering. In contrast, divisive hierarchical clustering repeatedly partitions data into its components; both types of hierarchical clustering build a tree-like representation called a dendrogram [34]. Furthermore, when data objects are directly divided into some predefined number of clusters without creating any hierarchical structure, it is called a partitional method [21]. Partitional clustering seeks k-partitions of dataset, say D, where the partitions or clusters C is defined $C = C_1, ..., C_k$ $(k \leq V)$, such that the following condition holds:

1. $c_i \neq \varnothing, i = 1, .., k$;
2. $\bigcup_{i=1}^{k} c_i = C$;

3. $c_i \cap c_j = \varnothing, i, j = 1, ..., k$ and $i \neq j$

The partitional methods are based on the iterative relocation of data points between clusters using the similarity or characteristics uncovered from the dataset. The aim is to produce clusters with a high degree of similarity within members in the group and a low degree of similarity between the groups. Partitioning methods finds a specified number of groups (k), say: $c_1, c_2 ... c_k$ of the input dataset D that optimises a specific criterion. The criterion is usually of the form:

$$\sum_{i=1}^{k} \sum_{x=s_i}^{k} p(x, \mu_i)$$

Where μ_i represents the cluster centroid of c_i and $p(x, \mu_i)$ represents the distance metric (for example, Euclidean distance) between x and μ_i. Partitional clustering methods are popular due to their simplicity, efficiency, and ability to handle large datasets. They are suitable when the number of clusters is known in advance or when a specific number of clusters needs to be determined. However, they have limitations, such as sensitivity to the initial parameter(s) values, convergence to local optima, and difficulty handling noisy or overlapping clusters.

After outlining the concept of similarity measures as the foundation for identifying distinct groups or clusters within datasets, the subsequent section provides examples of clustering algorithms. One such algorithm is the k-means clustering [45], which is considered one of the most straightforward partitional clustering algorithms [5]. The simplified version of the algorithm is described in Algorithm 2.1. K-means algorithm takes input dataset D consisting of m data points, with K cluster centroids $c_1, \ldots, c_K$, and the maximum number of iterations J as described in Algorithm 2.1. It operates by iteratively updating cluster assignments and centroid locations. At each iteration, the algorithm computes the Euclidean distance between each data point and the cluster centroids, assigning the point to the cluster with the

Algorithm 2.1 The K-means Algorithm

Input: Dataset $D = d_1, \ldots, d_P$, initialisation for cluster centroids $c_1, \ldots, c_K$, and maximum number of iterations J

for $j = 1, \ldots, J$ **do**
  for $p = 1, \ldots, P$ **do** ▷ Update cluster assignments
    $E_d = \arg\min_{k=1,\ldots,K} \|c_k - d_p\|_2$
  end for
  for $k = 1, \ldots, K$ **do** ▷ Update centroid locations
    Let $S_k =$ the index set of points d_p
    c_k = $\frac{1}{|S_k|} \sum_{p \in S_k} d_p$ ▷ Update c_k
  end for
end for
for $p = 1, \ldots, P$ **do** ▷ Update cluster assignments using final centroids
  $E_d = \arg\min_{k=1,\ldots,K} \|c_k - d_p\|_2$
end for

Output: C - Clusters and Cluster centroids

closest centroid. The centroid locations are then recalculated by computing the mean of the data points in each cluster. This process is repeated for a specified number of iterations. Finally, the algorithm returns the cluster assignments and centroid locations as the output. By minimising the within-cluster sum of squares, the k-means algorithm optimises the cluster assignments and centroid positions, offering a simple and efficient method for clustering analysis. The three most popular version of k-means are: Lloyd/Forgy[46], the MacQueen [47] and Hartigan and Wong algorithms [48].

Lloyd/Forgy Algorithm

The Lloyd and Forgy's algorithms are both batch and centroid-based algorithms. The Lloyd and the Forgy algorithms differ in how they treat data distribution; for the Lloyd algorithm, data distribution is discrete, while the Forgy algorithm considers the distribution continuous but they are both slow to converge and can result in empty clusters.

Hartigan/Wong's algorithm

Hartigan and Wong's algorithm uses the within-cluster sum of squares to allocate data to partitions, and the centroid is updated and data redistributed to their nearest centroid. The iteration continues until convergence to avoid a case where a further change would make the clusters more internally distinct and more externally similar.

MacQueen algorithm

MacQueen's is a partitional algorithm that follows an iterative optimisation approach to find the optimal number of clusters in datasets; unlike Lloyd's algorithm, the centroids are recalculated after each iteration and updated more regularly than the Hartigan. However, Hartigan and MacQueen store the nearest clusters, thus the initial convergence can be fast [49].

k-means clustering is very useful in exploratory data analysis, and its ease of implementation, low memory consumption and computational efficiency have kept the k-means algorithm very popular. The next section introduces an alternative algorithm to K-means known as Partitioning Around Medoids (PAM). Although both algorithms share the common objective of partitioning a dataset into clusters, notable distinctions exist between them. One significant difference lies in the choice of representative points, with PAM utilising medoids instead of K-means' centroids.

2.3.1 Partitioning Around Medoids (PAM)

PAM is a modification of the k-medoids clustering algorithm proposed by Kaufman and Rousseeuw [50]. Kaufman and Rousseeuw described a medoid as the representative object of a cluster where the average dissimilarity to all the objects is the least. Object dissimilarity can be calculated using metrics such as Euclidean distance. PAM and other k-medoid algo-

rithms can handle outliers much better than k-means algorithms [5, 51]. The following are the key steps in PAM:

1. Randomly select k objects as medoids to represent initial clusters (centres) for the dataset;
2. Assign other objects to their nearest medoids;
3. Evaluate the clustering quality using the objective function. The clustering solution that minimises the objective function is the best;
4. Determine the swapping cost for each pair of non-medoid and medoid. Swapping replaces a medoid with a non-medoid only if it improves the objective function; otherwise, the medoid is retained;
5. Repeat steps 2, 3 and 4 until convergence (no more changes).

PAM has several favourable properties and advantages. Firstly, PAM is flexible in choosing a distance metric, thereby allowing different metrics to be applied in different contexts or clustering applications. This is not possible with a k-means algorithm. Secondly, clusters are identified by the medoids, which are robust representations of the cluster centres and are less sensitive to outliers than other cluster profiles, such as the cluster means of k-means [52]. However, the computational complexity of PAM is higher than that of k-means, hence unsuitable for clustering large datasets. The full algorithm is described in Algorithm 2.2 below. Unlike K-means, the following section describes the X-means algorithms, which automate the number of cluster determinations by iteratively refining them based on statistical criteria.

Algorithm 2.2 Partitioning Around Medoids (PAM)

Input: Dataset D, Number of clusters k

Initialize: Randomly select k data points from D as initial medoids.
repeat
    Assign: Assign each data point to the nearest medoid.
    Update: Calculate the total dissimilarity of the current clustering.
    for each non-medoid point p **do**
        for each current medoid q **do**
            Swap: Replace medoid q with point p.
            Calculate the total dissimilarity of the new clustering.
            Newcludiss = New Clustering dissimilarity
            Curcludiss = Current Clustering dissimilarity
            if *Newcludiss* $<$ *Curclusdiss* **then**
                Accept the swap and update medoids.
            end if
        end for
    end for
until No more improvements can be made
return Final medoids, clustering of data points.

2.3.2 X-Means

X-means is an extension of k-means, introducing a method to determine the number of clusters through the iterative splitting of the parent cluster based on local decisions about the current centroid subset to be split [53]. This algorithm efficiently explores the optimal value of k by specifying a range of values in which the true k is likely to exist, relying on the computed Bayesian Information Criterion (BIC) score for guidance.

The Bayesian Information Criterion (BIC), also known as the Schwarz criterion, is a statistical measure for model selection among a set of candidate models. It balances the model's goodness of fit and complexity, penalising overly complex models [54]. In the context of X-means, the BIC score is utilised to guide the algorithm in determining the most appropriate number of clusters.

The X-means algorithm alternates between two key steps. First, it runs

conventional k-means to optimally identify clusters for a chosen value of k. Subsequently, it assesses where a new centroid should appear by splitting existing clusters based on local decisions about the subsets of the parent cluster that best fit the data. This iterative process refines the clustering and enhances the algorithm's adaptability to the underlying structure of the data.

The next clustering algorithm to be described is Clara; it excels in computational efficiency and noise robustness, which is particularly suitable for larger datasets. Unlike PAM, which concentrates on medoid-based clustering with a user-specified cluster count, Clara employs a sampling strategy for scalability, efficiently managing large datasets without needing a predefined number of clusters. Additional details about Clara are provided in the following section.

2.3.3 Clustering Large Applications (CLARA)

CLARA was proposed by Kaufman and Rousseeuw [55]. It is a k-med oid clustering algorithm that builds on a major drawback of PAM. The main drawback of PAM is that it iteratively replaces the current medoid with one of the non-medoid points until it attains the optimal medoid with a high computational cost. CLARA reduces the search space through multiple sampling to lower the computational cost. Rather than go through the entire search space iteratively, CLARA first takes a sample from the original m data points and then calls PAM on the sampled data points to find k-medoid. Once the optimal k-medoid is found, the remaining (nonsampled) data points are assigned to one of these k clusters based on their distance to the k-medoid. Conceptually, both algorithms are graph-searching problems while PAM greedily searches the graph until it cannot find a better neighbour as medoid. CLARA reduces the search space by searching a subgraph induced by the

sampled data points. Technically, CLARA can deal with much larger dataset than PAM [56], but the accuracy depends on the quality and size of sample [21]. The full algorithm is in Algorithm 2.3, and the key steps in CLARA are as follows:

1. Select randomly from the original data;
2. Partition the objects of the selected sample into k clusters using PAM;
3. Assign each object in the rest of the dataset to the nearest cluster;
4. Repeat the above process for a number of iterations (predetermined).

Algorithm 2.3 CLARA (Clustering Large Applications)

Input: dataset D, *numSamples X*, *maxIter*
Initialize: Select *numSamples* random subsets of D.
bMedoid $=$ *ClusterMedoids*(*samples*, X)
for iteration $= 1$ to *maxIter* **do**
    Assign: Assign data point to the closest medoid.
    Update: Calculate the total dissimilarity of the current clustering.
    for each non-medoid point p **do**
        for each cMedoid q **do**
            Swap: Replace medoid q with point p.
            Calculate the total dissimilarity after the medoid swap.
            if *newCluDis* $<$ *cCluDis* **then**
                Accept the swap and update cMedoids.
            end if
        end for
    end for
    if *tDiss*(*cMedoids*) $<$ *tDiss*(*bMedoid*) **then**
        Update *bMedoid*
    end if
end for
Output: *bMedoid* and the clustering of data points

The algorithm to be discussed next is Fuzzy clustering, distinguished among other clustering algorithms by its distinctive approach to handling data assignments. Unlike conventional methods that assign each data point to a single cluster, Fuzzy clustering permits partial memberships, indicating

the degree to which a point belongs to multiple clusters simultaneously. This flexibility is particularly valuable in capturing complex relationships within the data, allowing data points to share characteristics with more than one cluster. A detailed description of the Fuzzy clustering algorithm follows.

2.3.4 Fuzzy Clustering

Fuzzy clustering is a clustering algorithm that allows data points to belong to multiple clusters simultaneously with varying degrees of membership. Unlike traditional partitional clustering algorithms, where each data point is assigned to a single cluster, fuzzy clustering assigns objects to each cluster, indicating the degree of belongingness [57].

In fuzzy clustering, the goal is to find the best fuzzy partition of the data where each data point's membership values sum up to 1.0 across all clusters. The membership values represent the degree of association or similarity between a data point and each cluster. Higher membership values indicate a stronger association, while lower ones indicate a weaker association.

The most widely used fuzzy clustering algorithm is the Fuzzy C-means (FCM) algorithm [58]. FCM extends the traditional k-means algorithm by allowing data points to have partial membership in multiple clusters. The algorithm iteratively updates the cluster centres and the membership values until convergence. The update of cluster centres is similar to k-means. Nevertheless, the update of membership values is based on the distances between data points and cluster centres, considering the fuzzy membership concept. Fuzzy clustering is beneficial when data points exhibit overlapping characteristics or uncertainty in their membership assignments. It provides a more nuanced representation of cluster membership, allowing for more

flexible and granular clustering results. Fuzzy clustering has applications in various fields, including pattern recognition, image segmentation, decision-making systems, and data analysis where probabilistic or soft clustering is desired.

The hierarchical clustering algorithm described next adopts a distinct approach by constructing a tree-like structure of nested clusters, organising data points into a hierarchy according to their similarities. In contrast to Fuzzy clustering, which allows flexibility in handling overlapping memberships, hierarchical clustering prioritises a structured, tree-based representation of relationships within the data. The following section provides a detailed exploration of hierarchical clustering.

2.3.5 Hierarchical Clustering

Hierarchical clustering methods work by grouping data objects into a tree-like structure. It produces a nested series of partitions based on a criterion that merges or splits clusters using a similarity measure. Hierarchical clustering techniques that start with the whole data object and split it are divisive. In contrast, those that begin with singleton clusters and merge them are called agglomerative, i.e. the hierarchy is formed in a bottom-up fashion [59]. The clusters from hierarchical clustering suffer from inability to adjust once a merge or a split has been concluded. It cannot backtrack in case a poor choice has been made in the hierarchical decomposition, and it is not scalable for large dataset. The process of allocating or dividing the cluster into different groups is based on the closeness between the two clusters. Well-known examples of hierarchical clustering include: BIRCH [60], clustering Using REpresentatives–CURE [61] and CHAMELEON [62]. Several common measures or distance metrics are used to determine the proximity between clusters. Some of the most widely used measures of cluster proximity in hierarchical clustering include:

Single-linkage Hierarchical Clustering

Single-linkage Hierarchical Clustering is a bottom-up agglomerative clustering algorithm that forms clusters based on the similarity between data points [5]. It operates by iteratively merging the two closest clusters until all data points belong to a single cluster.

The algorithm starts by treating each data point as a separate cluster. It then calculates the pairwise distances between all clusters using a chosen distance metric, such as Euclidean distance. The algorithm merges the two clusters with the smallest distance, forming a new cluster. This process is repeated until all data points are in a single cluster. The output of Single-linkage Hierarchical Clustering is a dendrogram, which is a binary tree representation of the merging process.

Each node in the dendrogram represents a cluster, and the node's height indicates the distance at which the clusters were merged. Single-linkage Hierarchical Clustering has several advantages, including its simplicity and ability to handle non-convex clusters. However, it is sensitive to noise and can suffer from the "chaining" effect, where clusters are elongated due to the single-linkage criterion. The pseudocode for Single-linkage Hierarchical Clustering is is described in Algorithm 2.4.

Algorithm 2.4 Single Linkage Hierarchical Clustering

Input: Dataset D

```
Initialise each data point as a separate cluster
while Number of clusters > 1 do
    Calculate pairwise distances between clusters
    Merge the two clusters with the smallest distance
end while
Build the dendrogram based on the merging process
```

Output: dendrogram representing the clustering hierarchy

Complete Linkage Hierarchical Clustering

Complete linkage Hierarchical Clustering is a bottom-up agglomerative clustering algorithm that forms clusters based on the maximum dissimilarity between data points. It operates by iteratively merging the two clusters with the smallest maximum dissimilarity until all data points belong to a single cluster.

It then calculates the pairwise dissimilarities between all clusters using a chosen dissimilarity measure, such as Euclidean distance. The algorithm merges the two clusters with the smallest maximum dissimilarity, forming a new cluster. This process is repeated until all data points are in a single cluster. Complete linkage Hierarchical Clustering output is a dendrogram, a binary tree representation of the merging process. Complete linkage Hierarchical Clustering has several advantages, including its ability to handle non-convex clusters and its robustness to noise. The pseudocode for Complete linkage Hierarchical Clustering is as shown in Algorithm 2.5.

Algorithm 2.5 Complete Linkage Hierarchical Clustering

Input: Dataset D

```
Initialise each data point as a separate cluster
while Number of clusters > 1 do
    Calculate pairwise dissimilarities between clusters
    Merge the two clusters with the smallest maximum dissimilarity
end while
Build the dendrogram based on the merging process
```

Output: dendrogram representing the clustering hierarchy

Average Linkage Hierarchical Clustering

Average Linkage Hierarchical Clustering is a bottom-up agglomerative clustering algorithm that forms clusters based on the average dissimilarity between data points. The average linkage is referred to as the minimum variance method and can be described as the mid-point of the two other mea-

sures because it considers the distance between two clusters to be equal to the average distance from any member of one cluster to any member of the other cluster [59]. It operates by iteratively merging the two clusters with the smallest average dissimilarity until all data points belong to a single cluster.

The process is similar to the Single and Complete linkage; however, in average-linkage clustering, the dissimilarity between two clusters is defined as the average dissimilarity between all pairs of points belonging to different clusters. Average Linkage Hierarchical Clustering has several advantages, including its ability to handle non-convex clusters, its resistance to outliers, and its tendency to produce compact and spherical clusters. The pseudocode for Average Linkage Hierarchical Clustering is described in Algorithm 2.6:

Algorithm 2.6 Average Linkage Hierarchical Clustering

Input: Dataset D

```
Initialise each data point as a separate cluster
while Number of clusters > 1 do
    Calculate pairwise dissimilarities between clusters
    Merge the two clusters with the smallest average dissimilarity
end while
Build the dendrogram based on the merging process
```

Output: dendrogram representing the clustering hierarchy

Using the popular Iris dataset widely used in machine learning and statistics, the three different species: Setosa, Versicolor, and Virginica, can be represented in a dendrogram as shown in Figure 2.1

2.3.6 Density-based Methods

Density-based clustering is centred on the concept of how closely interconnected points are with one another. This approach is non-parametric, meaning it makes no assumptions about the number of clusters or their distribution.

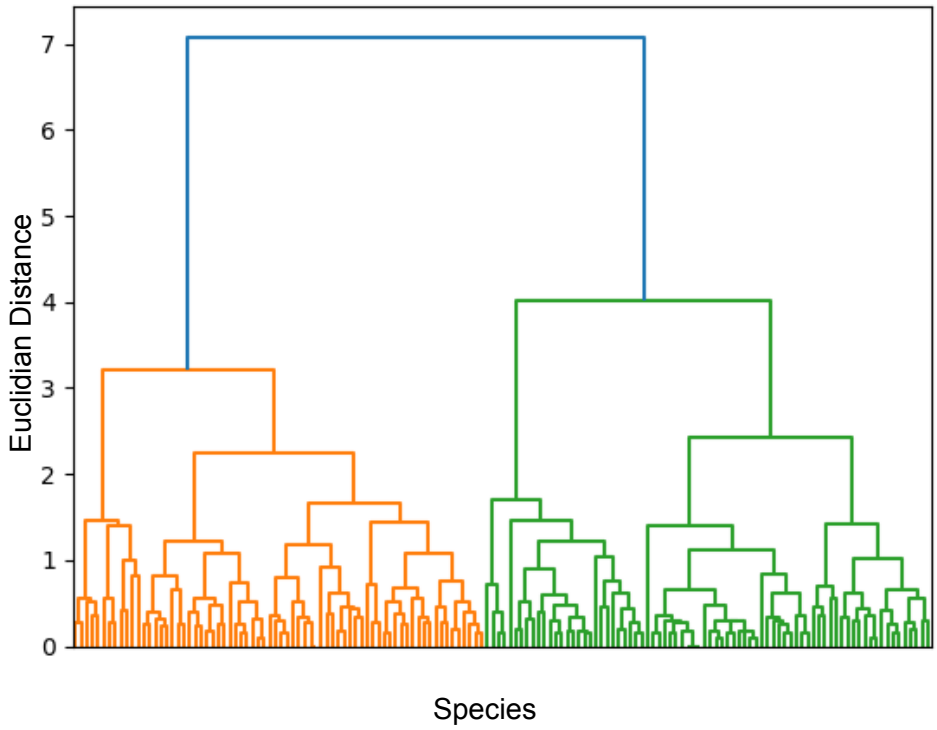


Figure 2.1: Figure Showing a Dendrogram of Iris Dataset

In data space, density-based clusters are contiguous, densely populated regions separated by sparser areas. In this method, the density in the noise areas is assumed to be lower than the density in any of the clusters. The computation of density-based clusters can be done efficiently by performing at most one region query per database object. Any sparse regions in the data space are considered noise and are not assigned to any cluster.

In contrast to partition-based methods, if the density in the neighbourhood exceeds a particular threshold, the cluster will continue to grow. For a given radius, each data point in a cluster is a member of the neighbourhood if it contains at least a minimum number of points. Density-based methods help identify and filter possible outliers in a cluster and can identify clusters of unusual shapes [63]. OPTICS is an example of a density-based clustering algorithm, while DBSCAN (Density-Based Spatial Clustering of

Applications with Noise) is another example. DBSCAN is suitable for large spatial databases and can identify clusters of arbitrarily shaped data points [64].

2.3.7 ccfkms:Clustering with Conjugate Convex Function

The ccfkms clustering algorithm uses conjugate convex functions [65], derived from prototype vectors' norm powers or a logarithmic norm transformation, aiming to enhance partition robustness. Internally, the algorithm employs sparse data structures optimally, avoiding computations with zero data values and eliminating the need to center the data. This approach, however, may exhibit some inefficiency for dense data. When initial prototypes are excluded, the user must specify the number of prototypes drawn from the data without replacement or chosen with retries for the best solution. Furthermore, the algorithm incorporates tie-breaking mechanisms for determining cluster memberships. These are just a few examples of clustering methods and their applications to different datasets and data characteristics.

2.4 Cluster Validity: Evaluating Clustering

Cluster validity pertains to evaluating the quality and usefulness of the results obtained from a clustering algorithm [66]. This review focuses on techniques used to assess cluster validity, encompassing standard methods and publicly available indices. Although newer methods in the literature may not have readily accessible tools or implementations [66], the emphasis here is primarily on well-established approaches.

According to [67], there are three main approaches to testing cluster validity: external criteria, internal criteria, and relative criteria.

External criteria involve comparing the clustering result with the dataset's external known labels or ground truth. These criteria measure how well the clusters align with the pre-existing class labels. On the other hand, **internal criteria** rely on information derived from the clustering process to evaluate the quality of the result. These measures assess compactness, separation, and cohesion within clusters. **Relative criteria** compare the clustering result with alternative clustering schemes. These criteria examine how the clusters generated by the algorithm compare to other plausible clustering solutions.

While some clustering algorithms can automatically estimate the number of clusters, many others require the user to specify this parameter. The performance of such algorithms heavily depends on the characteristics of the dataset and the input parameters chosen. Incorrectly specifying the number of clusters can result in clusters that do not accurately represent the underlying structure of the dataset. Therefore, it is crucial to have reliable guidelines for evaluating the clusters and determining the optimal number of clusters for a given dataset. Hence the need for cluster validity indices (CVIs).

Cluster validity indices provide quantitative measures to assess the quality and coherence of clusters. These indices evaluate various aspects, such as cluster compactness, separation, and overall structure. Calculating these indices for different clusters can help identify the number of clusters that best fit the dataset.

2.4.1 Cluster Validity Index (CVI)

Most clustering algorithms require some tuning of input parameters to determine the optimal number of clusters. For instance, the k-means algorithm requires the user to specify the number of clusters, denoted as k, and the

output quality strongly depends on the correctness of this value. To address this challenge, cluster validity indices (CVIs) have recently emerged as a valuable tool to guide the selection of input parameters, thus improving the accuracy of the resulting clusters [68]. By selecting an appropriate CVI, the input parameters can be tuned to the optimal number of clusters, thereby enhancing the quality of the resulting clusters. Estimating the optimal number of clusters is challenging, and various methods have been proposed. In this thesis, the NbClust package [66] in the R programming language was used; the package implements well-known indices for determining the optimal number of clusters. The methods were used as a benchmark to compare against our ensemble methods. The NbClust [66] package is publicly available on the CRAN website and offers implementations of all indices reviewed in this study.

2.4.2 Evaluating Cluster Validity Index

The evaluation of cluster validity involves two primary criteria: compactness and separability, as discussed by Arbelaitz et al. [68]. Compactness measures the proximity of objects within a cluster, typically assessed by variance or the degree of dissimilarity among objects. Lower variance indicates a higher level of compactness within the cluster. Separability, on the other hand, gauges the distinctness of clusters from one another.

To further categorise cluster validity indices (CVIs), they can be based on the ratio of the intra-cluster distance to the inter-cluster distance. Prominent examples include Dunn [69], Davies-Bouldin (DB) [70], and Calinski-Harabasz [71]. Another category is determined by the weighted sum of these two distances: inter-cluster and intra-cluster distances. Notable examples in this category include SD [72] and S_Dbw [73].

Table 2.1 provides a comprehensive list of methods for evaluating CVIs,

accompanied by a brief description of the formulas employed for their calculation.

2.4.3 Gap Statistics

Gap statistics are a popular method for estimating the optimal number of clusters in a dataset [74]. It compares the observed data's within-cluster dispersion with reference null datasets. If the clustering structure is meaningful, the observed data should have a smaller within-cluster dispersion compared to random reference data. The algorithm is described in 2.7.

Algorithm 2.7 Gap Statistics

Input: Dataset D, K_{max}, B
1: Initialize arrays: W_{obs}, W_{ref}
2: **for** $k = 1$ to K_{max} **do**
3: $W_{\text{obs}}[k] \leftarrow$ Within-cluster dispersion of D with k clusters
4: **for** $b = 1$ to B **do**
5: Generate reference dataset D_{ref} with the same dimensions as D
6: $W_{\text{ref}}[b, k] \leftarrow$ Within-cluster dispersion of D_{ref} with k clusters
7: **end for**
8: $W_{\text{avg}}[k] \leftarrow \frac{1}{B} \sum_{b=1}^{B} W_{\text{ref}}[b, k]$
9: $G[k] \leftarrow \log(W_{\text{avg}}[k]) - \log(W_{\text{obs}}[k])$
10: **end for**
11: $K_{\text{opt}} \leftarrow \arg\max_k G[k]$
Output: Optimal number of clusters K_{opt}

The algorithm takes a dataset D, the maximum number of clusters K_{max}, and the number of reference datasets B as inputs. It calculates the within-cluster dispersion for the observed data and reference datasets, computes the gap statistic for different numbers of clusters, and returns the optimal number of clusters based on the maximum gap statistic.

2.4.4 The Average Silhouette

The average silhouette method is a clustering validation technique used to determine the optimal number of clusters in a dataset. It involves applying a

clustering algorithm with varying numbers of clusters, calculating silhouette scores for each data point to assess cohesion and separation within clusters, and then computing the average silhouette score for each clustering solution. The algorithm can be computed as follows:

Algorithm 2.8 The Average Silhouette Algorithm

Input: Dataset D, number of clusters K
1: Initialize array to store silhouette scores: S
2: **for** $k = 2$ to K **do**
3: Perform clustering on D with k clusters
4: Calculate the silhouette score for each data point
5: Calculate the average silhouette score $S[k]$ for k clusters
6: **end for**
7: Find the optimal number of clusters: $K_{\text{opt}} \leftarrow \arg\max_k S[k]$
8: $S \leftarrow S[K_{\text{opt}}]$
Output: Average silhouette score S

This algorithm takes a dataset D and the number of clusters K as inputs as described in Algorithm 2.8. It performs clustering with different numbers of clusters, calculates the silhouette score for each data point, and computes the average silhouette score for each number of clusters. Finally, it determines the optimal number of clusters based on the maximum average silhouette score and returns the average silhouette score S.

Table 2.1: Showing list of methods for determining the optimal number of clusters in datasets

Index	**Formulae**	**Ref**
Calinski and Harabasz	CH = $\frac{\frac{trace(B_k)}{k-1}}{\frac{trace(C_k)}{n-k}}$, where $trace$ is the sum of diagonal elements of a square matrix.	**[75]**
Cubic Clustering Criterion	CCC = $\frac{\sum_{i=1}^{k}\sum_{\mathbf{x}\in C_i}\|\mathbf{x}-\mathbf{cc}_i\|^2}{\sum_{i=1}^{k}\sum_{j=1}^{k}\sum_{\mathbf{x}\in C_i}\sum_{\mathbf{y}\in C_j}\|\mathbf{x}-\mathbf{y}\|^2}$, where $\mathbf{cc}_i$ is the cluster centroid of c_i	**[76]**

Continued on next page

Table 2.1 – *Continued from previous page*

Index	**Formulae**	**Ref**
Duda and Hart (Pseudot2)	Duda = $\frac{J_e(2)}{J_e(1)}$, where $J_e(2) =$ Error rate of the second largest eigenvalue $J_e(1) =$ Error rate of the largest eigenvalue	**[77]**
Krzanowski and Lai (KL)	KL = $\frac{DIFF_k}{DIFF_k+1}$, where $DIFF_k$ is a measure of difference between successive partitions obtained at different values of k.	**[78]**
Gamma	Gamma = $\frac{s(+)-s(-)}{s(+)+s(-)}$, where s(+) = number of concordant comparisons, s(-) = number of discordant comparisons.	**[79]**
Gap	Gap(k) = $\frac{1}{B}\sum \log C_v - \log C_p$ where B = Number of referenced dataset generated. C_v = matrix representing the variability within the clusters for the k clusters. C_p = the observed within-cluster dispersion matrix for the k clusters.	**[74]**
Silhouette	Silhoutte = $\frac{\sum_{i=1}^{n} S_i}{n}$ S_i represents the Silhouette score for the i^{th} data point. where $S_i = \frac{b(i)-a(i)}{max(z(i);b(i))}$ b_i measures how well the data point fits into its neighboring cluster. a_i measures the compactness of the data point within its assigned cluster. z_i is the maximum value between a_i and b_i	**[80]**
Hartigan	Hartigan = $\frac{trace(C_k)}{trace(C_{k+1})} - 1)(n-k-1)$, where $k \in 1, .., n-2$	**[81]**

Continued on next page

Table 2.1 – *Continued from previous page*

Index	**Formulae**	**Ref**
Cindex	Cindex = $\frac{S_w - S_{min}}{S_{max} - S_{min}}), S_{min} \neq S_{max},$ $Cindex \in (0, 1)$, where S_w number of concordant pairs S_{min} number of pairs with tied predictions S_{max} the total number of pairs	**[82]**
DB	DB = $\frac{1}{k}\sum_{k=1}^{k}$ max $\frac{\delta_k + \delta_l}{\delta_{kl}}$ where $k, l = 1, ..., k =$ cluster number, and δ_k is the average distance between each points in cluster k and its centroid. δ_{kl} is the distance between the two centroids.	**[70]**
Ratkowsky	Ratkowky = $\frac{\bar{S}}{k^{1/2}}$, where $\bar{S}^2$ is the average sum of squares.	**[83]**
Scott	Scott = $n \log\left(\frac{det(T)}{det(C_k)}\right)$, Where $det(T)$ is determinant of the total variance-covariance matrix. $det(C_k)$ is determinant of the within-cluster variance-covariance matrix for k partitions.	**[84]**
Marriot	Marriot = $k^2 det(C_k)$	**[85]**
Ball	Ball = $\frac{C_k}{k}$	**[86]**
Trcovw	Trcovw = trace$(COV(C_k))$	**[87]**
Tracew	Tracew = trace (C_k)	**[87]**
Friedman	Friedman = trace ${C_k}^{-1} B_c$	**[88]**
Rubin	Rubin =$\frac{det(T)}{det(C_k)}$, where T is the total scatter matrix.	**[88]**

Continued on next page

Table 2.1 – *Continued from previous page*

Index	Formulae	Ref
Beale	Beale = $F \equiv \frac{p}{q}$ where p = ratio of the between-cluster variability to the sum of within cluster variability and the between-cluster variability q = represents measures of compactness	**[89]**
Ptbiserial	Ptbiserial = $\frac{\bar{S_b} - \bar{S_w}(N_w N_b / N_t^2)^{1/2}}{s_d}$ where $\bar{S_w} = S_w/N_w$, $\bar{S_b} = S_b/N_b$ s_d = standard deviation of all distances	**[90]**
Gplus	Gplus = $\frac{2s(-)}{n(n-1)}$, where $s(-)$= the number of discordant comparison	**[91]**
Frey	Frey = $\frac{\bar{S_{b_{j+1}}} - \bar{S_{b_j}}}{\bar{S_{w_{j+1}}} - \bar{S_{w_j}}}$ where $\bar{S_b}$= mean between-cluster distance. $\bar{S_w}$ = mean within-cluster distance.	**[70]**
Tau	Tau = $\frac{s(+) - s(-)}{[(n(n-1)/2 - t)(n(n-1)/2]^{1/2}}$, where t = the number of tied pairs	**[66]**
McClain	McClain = $\frac{\bar{S_w}}{\bar{S_b}}$, where $\bar{S_w}$ = mean within-cluster distance $\bar{S_b}$ = mean between-cluster distance	**[92]**
Dindex	Dindex = $\frac{1}{k}\sum_{q=1}^{k} \frac{1}{n} \sum_{x_i \in cc} d(x_i, c_k)$	**[93]**
Dunn	Dunn = $\frac{\min d(C_i, C_j)}{\max diam(C_k)}$ where $d(C_i, C_j)$ is the dissimilarity function between two clusters C_i and C_j. $diam(C_k)$ = diameter of cluster C_k	**[70]**

Continued on next page

Table 2.1 – *Continued from previous page*

Index	**Formulae**	**Ref**
SDindex	SDindex(q) =$\alpha Scat(q) + Dis(q)$ where α= weighing parameter. $Scat(q)$ = scatter index $Dis(q)$= Distance index.	**[72]**
sDbw	sDbw =$Scat(q) + bw(q)$ where $Scat$(q) is the same computed in SDindex bw(q) is the inter-cluster density.	**[72]**

2.5 Clustering Ensemble

Clustering Ensemble (CE) is a robust methodology combining multiple clustering results to produce a cohesive set of clusters or reconcile clustering information from diverse methods [94, 13]. Recognised as an NP-complete optimisation technique, CE operates on the principles of the median partition, demonstrating its computational complexity [95]. The methodology involves four key stages: member generation, consensus function formulation, optimisation, and evaluation.

In the initial member generation stage, an agreement matrix is systematically constructed to portray the extent of agreement between different input methods. This matrix serves as a foundational representation of the relationships and discrepancies among the various clustering solutions. The subsequent optimisation stage employs a fitness function applied to the agreement matrix to explore and identify an optimal clustering arrangement. CE excels at formalising the integration of diverse clustering arrangements into a consensus representation, thereby unveiling the latent clustering structure inherent in the dataset.

A rich body of literature, including theoretical reviews and applications, contribute to the understanding of clustering ensembles [96, 13, 7, 97]. Further practical implementations showcase the versatility of CE, with notable examples such as the R package diceR [98] and its related paper, which explores the application of ensemble clustering methods. The authors focus on developing an information-based distance metric that addresses the challenges inherent in clustering evaluation. They introduce a metric that quantifies dissimilarity by considering shared and lost information between clusterings, offering a robust solution rooted in information theory.

The Matlab package LinkCLuE also presents another avenue for practical

implementation, offering users an efficient toolkit for clustering ensemble analysis [10]. The toolkit was based on exploring the concept of clustering ensembles, focusing on developing models that capture both consensus and weak partitions. The authors delve into the theoretical foundations of clustering ensembles and propose models to address the challenges of combining diverse clustering solutions. They emphasize the importance of consensus in achieving robust clustering results and introduce methods to handle weak partitions effectively.

The advantages of employing a clustering ensemble extend beyond those typically associated with classification ensembles, which aim to enhance predictive accuracy [99]. Some notable benefits of a clustering ensemble include enhanced robustness, increased stability, and improved generalisation capabilities. By leveraging the collective insights from multiple clustering algorithms, CE can effectively mitigate the impact of individual algorithmic biases, resulting in more reliable and comprehensive clustering solutions [7, 99].

Practical applications of clustering ensembles span various domains, such as bioinformatics, image segmentation, and social network analysis. In bioinformatics, CE has been employed for the identification of gene expression patterns and the delineation of biological pathways [100]. Image segmentation tasks benefit from the ensemble approach by achieving more accurate delineation of object boundaries and regions of interest [101]. Social network analysis utilizes clustering ensembles to uncover community structures, enhancing our understanding of complex network interactions [97].

In summary, with its robust theoretical foundation and practical implementations, the clustering ensemble methodology proves to be a valuable tool for addressing the inherent challenges in clustering analysis. Its versatility and

its advantages make CE a compelling choice for researchers and practitioners seeking comprehensive and reliable clustering solutions across various applications. Listed below are some of its application areas:

2.5.1 Advantages of Using Ensemble for Clustering

The following are some of the motivations for using a clustering ensemble.

Enhanced quality of the solution

Unlike a single clustering solution, the ensemble approach considers the biases and limitations of different clustering algorithms, leading to improved and more accurate clustering results [96].

Robust clustering

Single clustering algorithms often assume specific underlying data characteristics, which may limit the quality of the clustering outcome. By selecting and aggregating results from multiple approaches, ensemble clustering can improve the robustness and quality of the clustering process. Empirical results in document clustering [102] have demonstrated the effectiveness of ensemble clustering in enhancing clustering quality.

Knowledge Reuse

Combining individual clusterings into ensembles is geared towards achieving a conclusive clustering solution characterized by stability and accuracy while concurrently promoting knowledge reuse [96]. Knowledge reuse within clustering ensembles involves strategically leveraging insights from various clustering runs or algorithms to inform subsequent analyses and decision-making processes. This strategic approach entails capitalising on

previously acquired knowledge about cluster assignments, structures, or similarities between data points [10]. By integrating information derived from diverse clustering sources, knowledge reuse contributes to a more profound understanding of the underlying patterns present in the data, enriching the overall analytical process [7].

Increased stability and reliability

Clustering ensembles can provide excellent stability and reliability in the clustering process. Ensemble methods can identify the common structures or patterns consistently present across different algorithms because they consider multiple clustering solutions, thereby reducing the impact of outliers or algorithm-specific requirements.

Handling uncertainty and ambiguity

Ensemble clustering methods prove highly beneficial in handling uncertainty and ambiguity in clustering tasks, capturing diverse perspectives to reveal intricate patterns when the underlying clustering structure is uncertain [96, 10]. Integrating multiple clustering outcomes enhances stability, reliability, and generalisation, making them robust in scenarios where the optimal clustering solution is unclear [103]. Furthermore, ensemble clustering serves as a risk mitigation strategy for unstable or noisy data, contributing to removing uncertainties associated with variability in the dataset [7]. This comprehensive approach fosters a nuanced interpretation of complex datasets, reducing uncertainty and providing a reliable foundation for subsequent analyses. Ensemble clustering is particularly adept at handling ambiguity in data, offering an effective strategy to navigate uncertainty and unveil latent structures embedded in complex datasets [7].

2.5.2 Clustering Ensemble Representation

The clustering ensemble problem can be described as follows: Suppose we are given a set of v data points in a dataset $D = d_1, d_2, ...d_v$ and a set of partition C defined as $C = \{c_1, c_2, ..., c_k\}$. The different partitions of D will return a set of labels for each point D_i, $i = 1, ..., v$: The task of a clustering ensemble is to find the partition C^* that aggregates the ensemble members over a function μ such that C^* is best in terms of consistency and quality [10].

There are two primary approaches to achieving a consensus solution and assessing the quality of each partition. The first approach involves employing the maximum likelihood formulation, which compares the labels assigned by individual solutions to the true consensus labels. Alternatively, one can aim for a consensus that aligns most closely with the original clusterings [99]. The second approach involves measuring the similarity between different clusterings using well-known metrics such as the Adjusted Rand Index (ARI) [94], Weighted Kappa, Normalized Mutual Information (NMI) [96], and Variation of Information (VI) [104]. The following subsection describes some of the metrics.

Weighted Kappa

The Kappa measure(w_k) [105, 106] is a metric that can be used to compare an expected accuracy with an observed accuracy based on the agreement between the two raters. It is generally a more robust measure than a simple percentage agreement because it accounts for the possibility of the agreement occurring by chance. It is usually expressed, as shown in the equation below:

$$w_k = \frac{p_0 - p_e}{1 - p_e} = 1 - \frac{1 - p_0}{1 - p_e}, \tag{2.4}$$

Where p_o is the relative observed agreement among raters (identical to accuracy), and p_e is the probability of chance agreement. If the raters are in complete agreement then $w_k = 1$. If there is no agreement among the raters, then $w_k = 0$. If Kappa is negative, it implies no effective agreement between the two raters or is worse than random. The Weighted Kappa derives from the Kappa metric; it allows weight to be assigned to disagreement between two raters. It has an agreement strength between poor and very good and is equivalent to the Adjusted Rand Index. The full guideline is shown in Table 2.2 reproduced from [13].

Table 2.2: The Weighted Kappa guideline

Weighted Kappa	Agreement Strength
$0.0 \leq$K ≤ 0.2	Poor
$0.2 <$ K ≤ 0.4	Fair
$0.4 <$ K ≤ 0.6	Moderate
$0.6 <$ K ≤ 0.8	Good
$0.8 <$ K ≤ 1.0	Very good

Adjusted Rand Index

The Adjusted Rand index (ARI) proposed by Hubert and Arabie [94] measures the similarity between two clusterings, which considers chance agreements. The ARI is defined as:

$$ARI = \frac{\sum_{i,j}\binom{n_{i,j}}{2} - [\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2}]/\binom{n}{2}}{\frac{1}{2}[\sum_i \binom{a_i}{2} + \sum_j \binom{b_j}{2}] - [\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2}]/\binom{n}{2}} \tag{2.5}$$

where n is the total number of objects being clustered, a_i is the number of objects in cluster C_i in the first clustering, b_j is the number of objects in cluster C_j in the second clustering, and $n_{i,j}$ is the number of objects that are in both C_i and C_j. The term $\binom{n_{i,j}}{2}$ represents the number of pairs of objects in the intersection of C_i and C_j that can be selected. The value ranges between -1 and 1, with 1 indicating identical clusterings and 0 indi-

cating random clustering. A higher value of ARI indicates a higher degree of similarity between the two clusterings and the Adjusted Rand Index is equivalent to the Weighted Kappa.

Agreement Matrix

An agreement matrix represents the level of agreement or similarity between different clustering solutions or clusterings generated by multiple clustering algorithms. It is a square matrix that quantifies the degree of concurrence between pairs of data points across the ensemble. The agreement matrix, denoted as A, is an $m \times m$ square matrix, where each entry $A_{i,j}$ represents the level of agreement or similarity between data points i and j across the ensemble. The agreement matrix quantitatively measures agreement or similarity between different clustering solutions [107]. The agreement matrix determines members' agreement represented by specific numbers, each corresponding to a label or cluster in the dataset. The Weighted Kappa Agreement matrix comprises the agreement matrix calculated from the clustering ensemble.

2.5.3 Search Space

The notion of the search space transcends various domains, spanning computer science, optimization, artificial intelligence, and algorithms [108, 109]. This conceptual framework encompasses the potential solutions that an algorithm or search process explores, necessitating evaluating these possibilities to identify those that meet specific criteria or constraints.

In computer science, challenges associated with the search space are pervasive, arising in algorithmic problem-solving, pathfinding, and optimisation tasks. For example, in algorithm design, the search space encompasses all possible combinations of inputs and outputs, requiring an efficient naviga-

tion strategy to pinpoint optimal solutions. Similarly, optimisation problems entail exploring the entire search space to discover the configuration that optimally aligns with a defined objective function.

Within the realm of a search space, several key concepts play a crucial role in optimisation and problem-solving:

Neighbourhood: Defined as the set of solutions closely related to the current solution, the Neighbourhood in the search space is a pivotal aspect of optimisation algorithms. Exploring this proximity involves scrutinising nearby solutions to identify potential enhancements, with the specific definition of closeness dependent upon the problem domain and algorithmic methodology.

Objective Function: Also referred to as a fitness or evaluation function, the objective function quantifies the quality or desirability of a solution within the search space. This numerical measure determines how well a solution aligns with the goals or constraints of the problem, guiding search algorithms in evaluating and comparing different solutions.

Solutions: These are specific points or configurations within the search space, each characterised by a unique combination of parameters or variables. The overarching objective of the search process is to identify an optimal or satisfactory solution based on the criteria defined by the objective function.

The integration of the search space concept extends beyond traditional algorithms; it is also integral to heuristic search algorithms. These algorithms leverage the search space, representing the states or configurations available for exploration, and employ heuristics to guide the process intelligently.

2.6 Search Strategies: Exhaustive Search

In machine learning and AI, the term “search” pertains to exploring and evaluating various configurations, models, or parameters to discover an optimal solution for a given task or problem. This search aims to identify the most favourable settings or combinations that produce the desired outcome or achieve the highest performance [110].

Search in machine learning, and AI encompasses a range of approaches tailored to specific contexts and goals. It involves systematically examining the space containing potential solutions and assessing their quality using predefined criteria or metrics. The search process typically involves several steps. Firstly, the search space is defined, which entails specifying the possible configurations, models, or parameters that can be explored. The definition may include determining the range or set of values for each component that can be adjusted or varied during the search. Secondly, an appropriate search strategy or algorithm is selected to explore the search space efficiently. Different strategies possess distinct characteristics in terms of exploration and exploitation. Common options include grid search [111], random search, Bayesian optimisation [112], Genetic Algorithms [113], and reinforcement learning-based search [114]. Grid search involves exhaustively evaluating predefined combinations within a specified range, while random search randomly samples configurations [111]. Bayesian optimisation employs probabilistic models to guide the search towards promising regions in the configuration space [115]. Genetic Algorithms utilise evolutionary principles, evolving a population of candidate solutions over multiple generations [113]. Reinforcement learning-based search, exemplified by deep reinforcement learning, involves agents learning optimal configurations through trial and error [116].Thirdly, an evaluation metric or criterion is established to measure the quality or performance of the evaluated solu-

tions. This metric can include accuracy, precision, recall, F1-score, mean squared error, or other suitable measures depending on the specific problem. Subsequently, the search is executed by systematically exploring different configurations, models, or parameters based on the chosen search strategy. Execution involves training and evaluating models, adjusting parameters, and assessing their performance using the defined evaluation metric. Finally, the performance of different configurations or models is compared to identify the most favourable solution. A common search approach is to use brute force, considering all possible combinations.

The exhaustive search [117], also known as brute-force search, is a straightforward algorithmic technique that systematically explores all potential solutions within a search space to identify the optimal solution. This method examines every possible combination, permutation, or configuration of elements while adhering to the given problem constraints. In the present scenario, the candidate solutions for each dataset are derived from subsets of the Weighted Kappa agreement matrix. Although the exhaustive search is conceptually simple and often yields successful results, it is sometimes criticised for its lack of elegance in problem-solving [117] hence the need for a heuristic search approach. The process of determining the optimal subsets follows the steps illustrated in Figure 2.2 of the diagram.

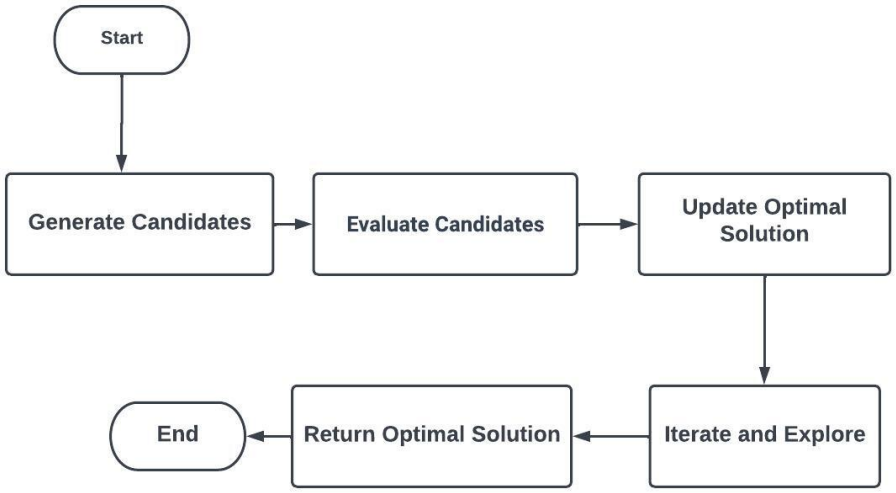


Figure 2.2: Steps in a General Exhaustive Search.

Exhaustive search algorithms are guaranteed to find the optimal solution within the search space because they explore every possible solution. However, they can be computationally expensive and impractical for large search spaces due to their time and resource requirements. The time complexity of exhaustive search algorithms is often exponential, growing rapidly with the size of the search space.

2.7 Heuristic Search Algorithms

Heuristic search algorithms are a family of search algorithms that use heuristics, which are problem-specific rules or knowledge, to guide the search for a solution. Unlike exhaustive search algorithms, which examine every possible solution, heuristic search algorithms can quickly narrow the search space and find a solution more efficiently.

2.7.1 Genetic Algorithms and Exploratory Data Analysis

Genetic Algorithms (GAs), rooted in natural selection and genetics principles, constitute a robust methodology for addressing intricate optimization problems. These evolutionary algorithms iteratively advance a population of potential solutions, employing genetic operators such as selection, crossover, and mutation to dynamically adapt and refine the search strategy (Holland, 1975). GAs find widespread applications across diverse domains, encompassing optimisation problems (Goldberg, 1989), machine learning (Mitchell, 1996), combinatorial optimisation (De Jong, 1975), and parameter tuning (Eiben and Smith, 2003).

Conversely, Exploratory Data Analysis (EDA) is crucial for comprehending and extracting meaningful insights from data. EDA is centered around systematically visualising, summarising, and interpreting datasets to unveil inherent patterns, relationships, and structures (Tukey, 1977). This analytical approach is pivotal in preprocessing data before optimisation, ensuring a comprehensive understanding of the problem space and facilitating informed decision-making.

In contrast, Hill Climbing Algorithms represent optimization techniques that iteratively traverse toward the peak (maximum) or valley (minimum) of a function. These algorithms incrementally adjust the solution space to reach the optimal point within the given constraints (Russell and Norvig, 2010).

Examples of Heuristic Algorithms

There are several types of heuristic search algorithms, some of which are described below [118]:

Best-First Search

The best-first algorithm explores a search space while prioritising the “best” node using a heuristic evaluation function. The evaluation function estimates the cost of reaching the goal from the current state [119]. It aims to find the optimal solution efficiently by focusing on the most promising paths rather than exhaustively searching all possible paths.

A* Search

This is a best-first search algorithm that combines the cost from the start node and the heuristic evaluation function. A* search is widely used in pathfinding and route planning problems. The heuristic function in A* never overestimates the actual cost to reach the most promising paths [119].

2.7.2 Greedy Search

This informed search algorithm makes locally optimal decisions at each step by choosing the most promising path based on a heuristic evaluation. It focuses on the immediate benefits by selecting the most promising node at every step. Essentially, this algorithm expands the node that appears to be closest to the goal state according to the heuristic evaluation function. However, this algorithm can get stuck in local optima [119].

Beam Search

This algorithm is a variant of best-first search that expands a fixed number of the most promising nodes at each level of the search tree. Beam search is commonly used in natural language processing and machine translation tasks, for generating sequences such as sentences or translations. It is a heuristic search algorithm that explores the most promising paths while

keeping a limited number of the best candidate solutions, called the beam width or beam size. The main characteristic of Beam search is that it maintains a limited number of candidate solutions at each step, discarding less-promising candidates. It keeps only a subset of candidates; the search space is pruned, thus allowing the algorithm to focus on the most promising paths [120].

Iterative Deepening A*

This algorithm combines A* and depth-first searches. It first performs a depth-limited search and gradually increases the depth limit until the goal is found [121]. The key advantage of Iterative Deepening Search (IDS) is that it combines the completeness of the Best First Search with the efficiency of the Depth First Search. IDS avoids the pitfalls of Depth First Search getting stuck in infinite paths by limiting the depth at each iteration.

Heuristic search algorithms have broad applications across domains, including artificial intelligence, operations research, and robotics. However, these algorithms' efficacy largely depends on the fitness function's quality and the search space's structure. The following section discusses specific algorithms applied in the search for the best subset from the ensembles.

2.7.3 Hill Climbing Algorithms

A Hill Climbing algorithm is a local search algorithm because it focuses on exploring the immediate neighbourhood of the current solution. The algorithm optimises problems to find a solution within a given search space. It starts with an initial solution and iteratively explores neighbouring solutions by making incremental changes in each iteration. The algorithm selects the nearest solution that improves the objective function or evaluation metric

and moves to that solution. This process continues until a locally optimal solution is reached, where no further improvements can be made by exploring the immediate neighbourhood.

Formally, let β be the search space of possible solutions, and f(β) be the objective function or evaluation metric that quantifies the quality or fitness of a solution β. The Hill Climbing algorithm can be described as follows:

1. select an initial solution β_0.
2. Repeat the following steps until a stopping criterion is met:
    - Generate the set $s(\beta)$ of neighbouring solutions of current solution β.
    - Evaluate the objective function for each neighbour s in $s(\beta)$, i.e., calculate $f(s)$.
    - Select the neighbour $s^{'}$ from $s(\beta)$ that maximises or minimises the objective function $f(s)$, depending on whether the problem is maximising or minimising.
    - If $f(s^{'})$ is better than $f(\beta)$, set β = $s^{'}$ (move to the neighbour $s^{'}$).
    - If $f(s^{'})$ is not better than $f(\beta)$, terminate the algorithm or restart if applicable.

This algorithm may get trapped in local optima, where the current solution is the best in its neighbourhood but suboptimal globally. Finding optimal solutions is not always guaranteed, and a hill climbing algorithm can sometimes become trapped in a state with a dead end. Hill climbing algorithm is also called a generate and test algorithm because it generates and tests each state as it moves towards the goal state. In summary, there are three significant steps in a hill climbing algorithm. First, all possible solutions are generated; second, the solutions are tested to

ensure each is part of the expected solutions. The previous two steps are repeated until there is no more improvement to the current solution or the search reaches a convergence point. Heuristic algorithms may appear to be intelligent, but they are not. They are only more efficient because they take advantage of feedback from the data to direct the search path. However, it has been proven to be highly efficient for specific problems [118]. The diagram in Figure. 2.3 shows the possible regions in a Hill Climbing algorithm. Several alternatives and variations have been proposed to overcome this limitation, such as simulated annealing, random restart Hill Climbing, and genetic algorithms. These variations aim to escape from local maxima (or minima) and reach the global optimum.

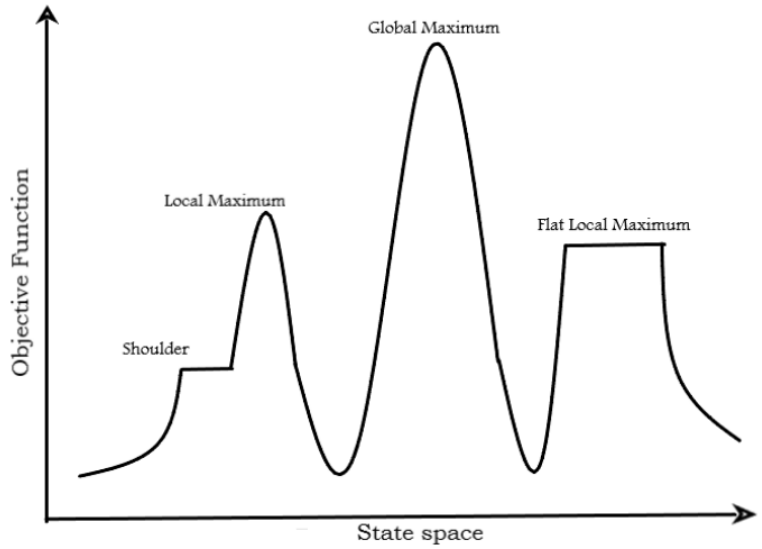


Figure 2.3: State Space Landscape of a Hill Climbing Algorithm. [1]

.

Random Mutation Hill Climbing (RMHC) is a variant of the Hill Climbing algorithm that incorporates random mutations to escape local optima and explore the search space more effectively. It aims to overcome the limitation

Algorithm 2.9 Random Mutation Hill Climbing (RMHC)

Input: ITER (# of iterations)
1: Let θ be a random point in the search space,
2: let f be its fitness
3: **for** $i \leftarrow 1$ to $ITER$ **do**
4: Let $\hat{\theta}$ be a random point close to θ
5: Let $\hat{f}$ be its fitness
6: **if** $\hat{f} > f$ **then**
7: $\theta \leftarrow \hat{\theta}$
8: $f \leftarrow \hat{f}$
9: **end if**
10: **end for**
Output: θ - a solution

of traditional Hill Climbing (getting stuck in local optima), earlier discussed in 2.7.3. In RMHC, the algorithm starts with an initial solution θ in the search space. It iteratively explores neighbouring solutions by making incremental changes $\hat{\theta}$, similar to the standard hill climbing algorithm. However, besides evaluating the best neighbouring solution, RMHC incorporates random mutations by making random changes to the current solution. These random mutations allow the algorithm to explore solutions that may not be directly reachable through incremental changes.

2.7.4 Fitness in a Search Space

Fitness refers to measuring or evaluating how well a particular solution or candidate meets the objectives or criteria of the problem being solved. It serves as a quantitative measure indicating the quality or suitability of a solution within the search space. A fitness function assigns a numerical value to each solution based on its performance or desirability with respect to the problem's objectives. Critical aspects of fitness in a search space include:

- Evaluation Metric: The fitness function defines an evaluation metric based on accuracy, cost, or robustness, measuring how well a solution satisfies the problem's criteria.
- Solution Comparison: Fitness enables comparing and ranking solutions within the search space, with higher fitness values indicating more better solutions.
- Search Space Exploration: The fitness function guides the search algorithm in exploring the search space by evaluating the fitness of each candidate solution. Promising solutions with higher fitness values are prioritised for further exploration, leading to an efficient search process.
- Iterative Improvement: The search algorithm aims to iteratively improve the fitness of solutions through operations like mutation, crossover, or local search. These operations modify the solutions to explore the search space and identify solutions with higher fitness values.
- Convergence: The search algorithm strives to converge towards solutions with high fitness values. As the search progresses, it focuses on promising regions of the search space, leading to the discovery of better solutions.
- Fitness Function Design: An appropriate fitness function is critical for successful search algorithms and optimisation processes. It should accurately capture the problem's objectives and provide a meaningful measure to guide the search towards desirable solutions.

In summary, fitness in a search space involves evaluating and quantifying how well a solution meets the problem's objectives. It is fundamental in guiding the search process, comparing solutions, and driving exploration

towards optimal or desirable solutions. Examples of its application and the technical description can be accessed in this tutorial on properties of fitness functions and search landscapes [122], and more detail on the specific fitness application in this research can be accessed in Chapter 4.

2.7.5 Convergence in a Search Space

Convergence refers to the state or condition where a search algorithm has reached a point where further exploration or optimisation is no longer possible or beneficial. It indicates that the search algorithm has found a satisfactory solution or has reached an end where no better solutions can be obtained within the given search space and constraints.

In optimisation algorithms, convergence implies that the algorithm has successfully approached an optimal solution. The specific criteria for convergence depend on the problem and the algorithm used. For example, in this thesis, convergence is defined based on the objective function value and the difference between consecutive iterations; in other cases, it might be determined based on reaching a predefined threshold of improvement.

When a search algorithm converges, it typically implies that it has found the best or near-best solution available within the explored search space. Convergence is desirable as it indicates that the algorithm has achieved the desired outcome and can be terminated to save computational resources. However, it's important to note that convergence does not guarantee that the obtained solution is the global optimum or the best. In some cases, the algorithm may converge to a local optimum as described in the case of Hill Climbing in Section 2.7.3, which is the best solution within a limited search space region but not the globally optimal solution. Different algorithms and techniques are employed to mitigate the risk of converging to local optima, for example, the Random Mutation Hill Climbing alternative for traditional

Hill Climbing.

2.7.6 Simulated Annealing

Simulated annealing is a metaheuristic algorithm used to find the global optimum of a function that may have multiple local optima. It is a probabilistic algorithm based on slowly cooling a material to decrease its energy and increase its stability. Kirkpatrick et al [123] first introduced a simulated annealing application to optimisation problem; they were inspired by the physical process of annealing in metallurgy, where a material is heated and then cooled slowly to reduce its energy and increase its stability. They demonstrated how the algorithm could solve optimisation problems by iteratively modifying a candidate solution and accepting it with a probability that depends on the difference between the new and old solutions and a temperature parameter that decreases over time. Even in complex search spaces, they demonstrated that the algorithm could find near-optimal solutions to classical computing problems, such as the travelling salesman.

Ingber [124] proposes a modification to the simulated annealing algorithm that improves its efficiency and speed using a numerical method called "minimisation by random search" to quickly find the best values for the temperature parameter and cooling rate, which are critical to the performance of simulated annealing. The modified algorithm, "very fast simulated annealing"(VFSA), uses these optimal values to perform a more focused search of the solution space. Ingber shows that VFSA outperforms traditional simulated annealing on various optimisation problems, including the travelling salesman problem and the quadratic assignment problem. Simulated annealing has been applied to a wide variety of applications some of which could be found in the following references [125, 126]. The Simulated Annealing algorithm is shown in Algorithm 2.10. The PR used in line 7 of Algo-

Algorithm 2.10 Simulated Annealing

Input: T_o (Starting Temp), Iter (Number of Iterations), λ (The cooling rate)

Let s = a random solution
for $i = 0$ to $Iter - 1$ **do**
    Let $f =$ fitness of s
    Make a small change to s to make s'
    Let $f' =$ fitness of new point s'
    if f' is worse than f **then**
        Let $p = PR(f', f, T_i)$
        if $p < UR(0, 1)$ **then**
            Reject change (Keep s and f)
        else
            Accept change (Keep s' and f')
        end if
    else
        Let $f = f'$
    end if
    Let $T_{i+1} = \lambda T_i$
end for

Output: The solution s

rithm 2.10 computes the acceptance probability of a new fitness f' based on the temperature T_i and the cost difference between f' and f. The difference between current fitness f and the new fitness f', referred to as $\Delta f'$, is defined below. Also, the PR and the fitness difference are defined as follows:

$$PR\left(f', f, T_i\right) = exp\left(\frac{-\Delta f}{T_i}\right)$$

where

$$\Delta f = |f - f'|$$

The function $UR(0, 1)$ generates a random number between 0 and 1, inclusive. The number generated is compared against the acceptance probability p to determine whether a change is accepted or rejected.

2.7.7 Genetic Algorithms

Genetic Algorithms (GAs) are a class of search and optimisation algorithms inspired by natural selection and genetics principles [127]. They are widely used to solve optimisation problems where traditional search methods may be inefficient or impractical. The central concept behind Genetic Algorithms is to simulate the process of natural evolution to search for an optimal solution within a population of candidate solutions [128]. The algorithm maintains a population of candidate solutions represented as chromosomes, typically binary strings. Each bit in a chromosome represents a gene, and the population as a whole represents a subset of the search space containing potential solutions. The GA iteratively evolves the population over generations using genetic operators such as selection, crossover, and mutation. Selection involves choosing individuals with higher fitness to serve as parents for producing offspring. Crossover (recombination) combines genetic material from parents to create new individuals, mimicking genetic recombination in biological reproduction. Mutation introduces random changes in the chromosomes to maintain diversity in the population. The fitness of each chromosome is evaluated based on its suitability to solve the problem. Through successive generations, the GA promotes the survival of fitter individuals, gradually improving the population's overall fitness. The fittest individual in the final population is considered the solution to the problem. The performance of a GA heavily depends on parameter settings, such as population size, selection criteria, crossover and mutation rates, and termination conditions. Fine-tuning these parameters is crucial for achieving good results. A simplified genetic algorithm is shown in Algorithm 2.11.

Algorithm 2.11 The Genetic Algorithm

Input: Fitness function, POPULATIONSIZE, NBITS, GENERATIONS, CROSSOVERRATE, MUTATIONRATE
Output: The fittest individual of the last population
1: Generate POPULATIONSIZE Valid Chromosomes of size NBITS bits
2: **for** $i = 1$ to GENERATIONS **do**
3: Crossover the Population to Create Children
4: Mutate the Population
5: Remove Invalid chromosomes
6: Apply Survival of the Fittest to the Population
7: **end for**

2.8 Evolutionary Approaches to Clustering

Evolutionary approaches have been widely used in clustering, applying genetic algorithms and other related techniques to optimise clustering solutions. Genetic Algorithms (GAs) are a prominent example of such approaches, drawing inspiration from natural selection and genetics principles to search for optimal solutions within a population of candidate solutions [129].

In the context of clustering, evolutionary techniques leverage stochastic methods to solve optimisation problems. Evolutionary algorithms employ a set of operators, including selection, recombination, and mutation, to evolve a population of clustering structures; clustering is treated as an optimisation problem. These operators work on encoded representations of candidate clusterings known as chromosomes. Each chromosome is evaluated using a fitness function that quantifies its quality, representing how well it fits the clustering objective. The evolutionary process continues iteratively, generating new generations of clusterings until convergence.

Genetic Algorithms are one of the most widely used evolutionary techniques for clustering [130, 131]. Other evolutionary approaches used in clustering include Particle Swarm Optimisation [132], Evolutionary Programming (EP)

[133], Evolutionary Strategies (ES) [134], and Estimation of Distribution Algorithms (EDA) [135].

Evolutionary approaches offer several advantages in clustering. They provide a computationally efficient search space exploration compared to traditional randomised methods such as multiple runs of the k-means algorithm. The inherent, implicit parallelism in genetic algorithms enables them to evaluate multiple candidate solutions simultaneously. However, it is crucial to carefully select and fine-tune parameters like crossover, mutation, and recombination operators to ensure their effectiveness in achieving optimal clustering solutions [8].

2.9 Related Studies

Numerous studies have addressed the challenge of consolidating multiple clustering results to enhance stability and robustness, aligning with the objectives of this research. Ensemble and Consensus Clustering techniques have emerged as notable solutions, aiming to unify diverse clustering outcomes into a cohesive output, overcoming limitations in individual clustering approaches, and providing more reliable and accurate results.

This study extends the work of Samy et al. [12] - an investigation into various selection techniques, encompassing manual methods and heuristic searches. Experimental evaluations demonstrate the effectiveness of these selection methods in improving the efficiency of Ensemble and Consensus Clustering. Similar to the current research, the study incorporates diverse datasets from various categories, emphasising practical applicability by highlighting real-world data from repositories. The comprehensive dataset collection, comprising 198 datasets with varied attributes and instances, undergoes rigorous data cleansing for accuracy and proper for-

matting. The authors enhance the reliability of the dataset collection by providing known expected clustering arrangements. Furthermore, the present study utilises some of these datasets while incorporating additional ones to broaden the research scope, specifically including datasets with outliers, as discussed further in Chapter 3.

The concept of Seeding used in this study is partly motivated by earlier work on software modularisation [136]; the study addresses the complex task of modularising sequential source code software check-ins to evaluate major changes using the Munch software clustering tool. The study employs a search-based software engineering technique; the tool automatically decomposes software systems into meaningful subsystems, facilitating the understanding and maintenance of large, evolving systems. Introducing a seeding approach based on prior modularisations accelerates the modularisation process and reduces runtime. Through experiments on extensive real-world datasets, the study validates the efficacy of the seeding strategy, extending previous work by introducing a time-series dataset and applying seeding to modularise sequential source code software versions.

The next chapter presents a concise discussion of the selection process of datasets, accompanied by an overview of their variability and characteristics. Careful consideration was given to ensure that these datasets are representative and suitable for the experiments conducted in this study.

Chapter 3

Description of Datasets

3.1 Introduction

This chapter outlines the datasets, the techniques employed to generate the clusterings, and the pre-determined parameters for selecting the dataset applied in later chapters. A thorough examination of the datasets was carried out, since they were sourced from various repositories and comprised both artificial and established standard clustering benchmark datasets.

3.2 Dataset Description

The datasets used for this experiment were selected from various data repositories used by the machine learning community to analyse different algorithms empirically. Within this thesis many yardsticks were used to choose the dataset, emphasising both the benchmark and the real-world dataset. The dataset collated was from a wide range of data categories ranging from bio-medical and statistical to weather-related datasets. Some of the datasets are available in UCI Machine Learning Repository [137], Kaggle Repository [138], StatLib [139], Time Series Library [140], Univer-

sity of Finland's Clustering Basic Benchmark [141] and the Outlier Detection Datasets [142], among others in various formats and dimensions.

The database currently holds two hundred and eighteen (218) datasets from the above sources. The datasets were organised into a format that contains different attributes and the cluster for the row. Through data scrubbing, errors were identified, and duplicates and incorrect or inconsistent values were removed, as well as general errors in the data. The accuracy and consistency of all datasets were validated by comparing the data across multiple sources, repositories and benchmarks. Where needed, variable transformation and new variables were created to transform the datasets into a suitable format for further analysis. Finally, the dataset was inputted into different algorithms to generate ensembles. The dataset attributes, such as the number of columns, range from 3 to 200, and instances ranges up to 5000. All datasets underwent a data cleansing process to ensure accuracy and correct formatting before running them on the clustering methods. This research reports the known number of clusters as the gold standard or the expected clustering arrangements for each dataset.

3.3 Data Collection and Pre-Processing

The data cleaning stage includes handling missing data and outliers; combining the data is data integration, especially if it is obtained from multiple sources. Data transformation involves standardising the numerical features to bring them to a similar scale, preventing any particular feature from dominating the analysis. The data reduction process can include dimensionality reduction, i.e. reducing the number of features in the dataset while preserving its essential information. Data Normalisation, however, is adjusting the data distribution to have a specific range or distribution. The process of ensuring the quality and integrity of the data is data validation, i.e. checking for

inconsistencies, errors, or anomalies that may have been introduced during the preprocessing steps. Each stage is essential and may be omitted or repeated several times depending on the requirement. Figure 3.1 shows the different data preprocessing stages. The total number of datasets initially collected for analysis are two-hundred and eighteen (218), some of which were eliminated because of the above or a combination of the followings reasons:

- Some datasets exhibit a discrepancy between the clusters obtained through clustering methods and the expected number of clusters based on a predefined gold standard.
- The data size is less than 100 instances (too small).
- Missing values are a common occurrence in datasets, and they often pose a challenge for many clustering methods that are not equipped to handle such missing values effectively.

A total of twenty-seven (27) datasets, as presented in Table 3.1, successfully satisfied the aforementioned criteria. These datasets were sourced from diverse origins, including bio-medical, ecological, statistical, and time series. The attributes within these datasets range from 3 to 100, while the instances reached up to 3000.

It is important to note that the process of ensuring dataset consistency and integrity of the published dataset characteristics varied significantly across repositories. Hence, in our final analysis, we took measures to include only those datasets that were verified across multiple repositories for consistency and correctness.

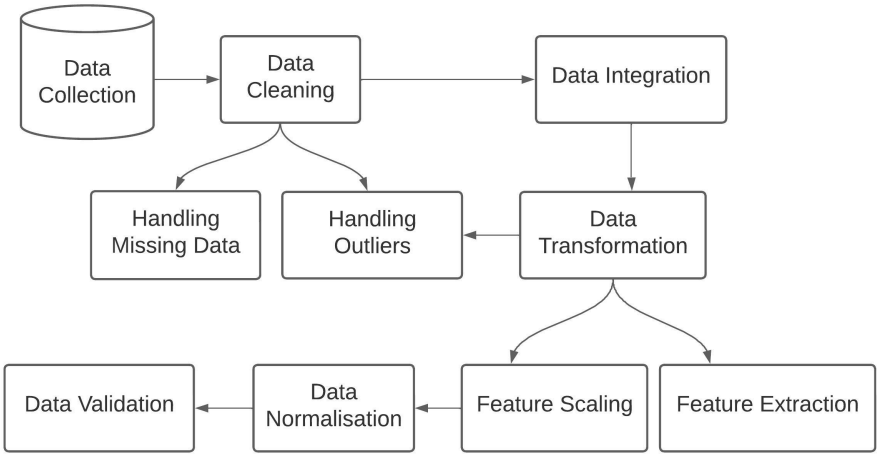


Figure 3.1: Stages in Data Preprocessing

3.4 Dataset Characteristics and Features

This section provides a comprehensive overview of the datasets used in the study. The section sheds light on important features of some of the datasets, such as the origin of the datasets, the range of attributes they encompass, the number of instances available, and the measures taken to ensure data consistency and integrity. Simple graphs of the clustering characteristics are provided where available.

Understanding the characteristics of the datasets is crucial as it allows researchers to gain insights into the nature of the data they are working with. Furthermore, the section highlights the range of attributes present in the datasets, providing insights into the dimensionality and complexity of the data. The number of instances available in each dataset indicates the scale and volume of the data being analysed. It is noteworthy that the number of clusters provided in Table 3.1 corresponds to the numbers provided in the repository from where the data were downloaded. The column indicating

Table 3.1: Dataset by rows, columns and number of clusters

Datasets	#Rows	#Columns	#Clusters
aml28	804	3	5
atom	800	4	2
bezdekIris	150	5	3
Blobs	300	3	3
cassini	1000	3	3
compound	399	3	6
curves1	1000	3	2
gaussian500	3000	3	5
glass	214	11	6
hepta	212	4	7
longsquare	900	3	6
lsun	400	3	3
pearl	266	3	3
pmf	649	4	5
shapes	1000	3	4
size1	1000	3	4
size2	1000	3	4
spherical_5_2	250	3	5
square2	1000	3	4
synthetic_control	600	62	6
tetra	400	4	4
tetragonular_bee	236	16	9
ThreeMC	400	3	3
triangle1	1000	3	4
vehicle	846	19	4
veronica	206	586	7
zelnik3	266	3	3

the number of clusters is based on metadata from the original source and was not derived independently in this study.

Acute Myeloid Leukemia (aml28) Dataset

The dataset consists of sequenced primary tumour and relapse genomes from eight acute myeloid leukaemia - AML patients. Most patients with AML die from progressive disease after relapse, associated with clonal evolution at different levels. The dataset was meant to determine the mutational spectrum associated with the relapse of hundreds of somatic mutations us-

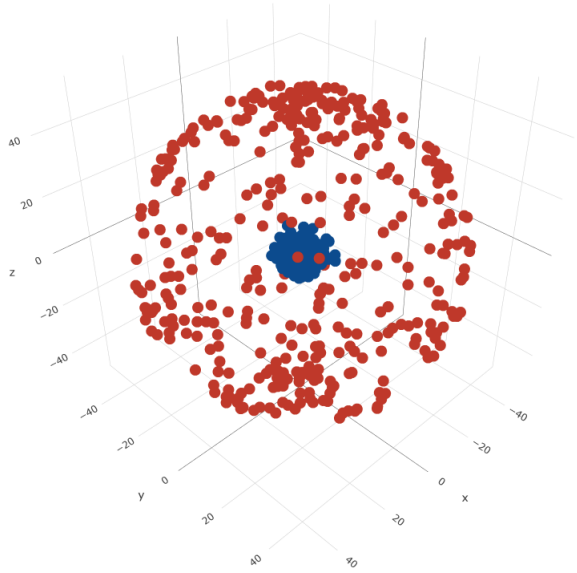


Figure 3.2: Visualisation of Atom Dataset

ing deep sequencing. The mutational spectrum enabled the definition of clonality and precise identification of clonal evolution patterns at relapse. More detail about the dataset can be accessed in the paper by Li Ding et al [143].

The Atom Dataset

The Atom dataset consists of two three-dimensional clusters with a completely overlapping convex hull. Therefore, by definition [144], the Atom dataset is linearly non-separable because the first cluster entirely encloses the second one. Although both consist of 400 data points, the density at the centre is more than the outer part, as shown in Figure 3.2.

The Hepta Dataset

The Hepta dataset has seven clusters, with two additional points added to the central cluster. Thus, the density of the central cluster as described in

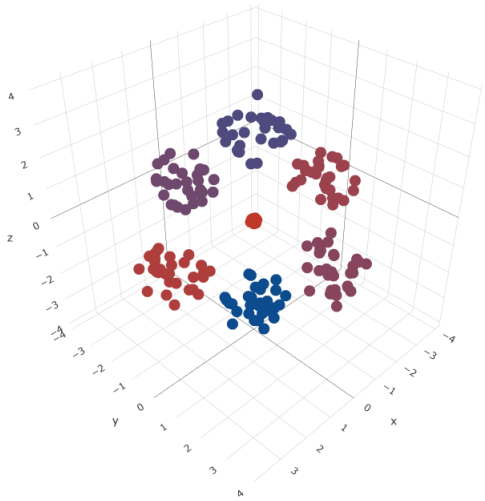


Figure 3.3: Visualisation of Hapta Dataset

[145] is twice the size of the other six clusters Figure 3.3.

The Lsun Dataset

The dataset was initially published as a two-dimensional version in [146]; the challenge is the that the dataset has some special features such as non-overlapping convex hulls with varying geometric shapes, elongated clusters and pockets of outliers. Furthermore, the dataset contains 400 data points, with two clusters containing 100 points and the third having 200 points. Thus, Lsun is often problematic for clustering algorithms because of the differences in the shapes of the clusters and the variance between the inner clusters and the slight cluster separations.

The Tetra Dataset

The dataset comprises 400 data points separated into four spherical clusters in [147]. The main challenge of the Tetra dataset is the large intra-cluster distances and low inter-cluster distances, with the clusters nearly touching one another as shown in Figure 3.5.

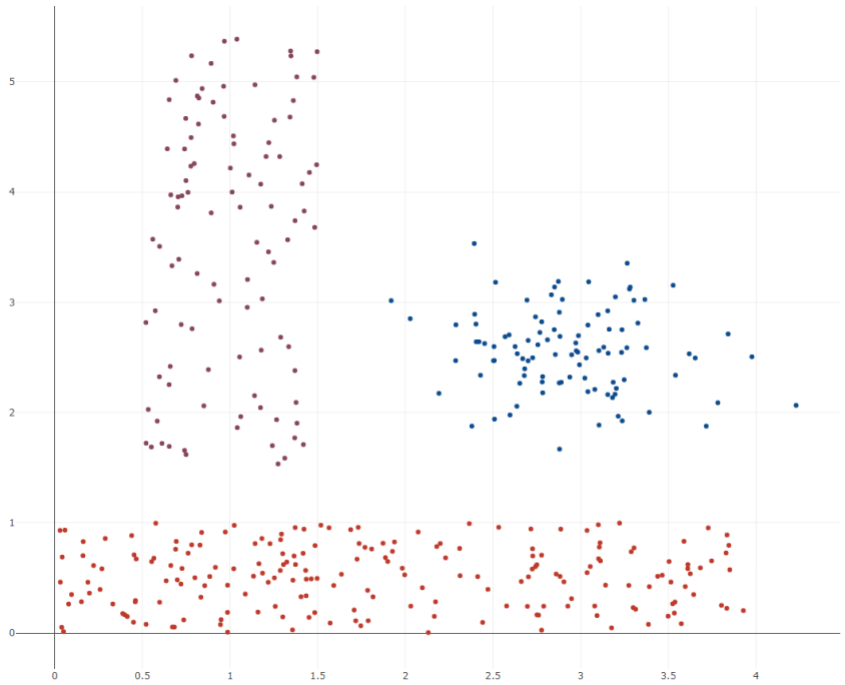


Figure 3.4: Visualisation of Lsun Dataset

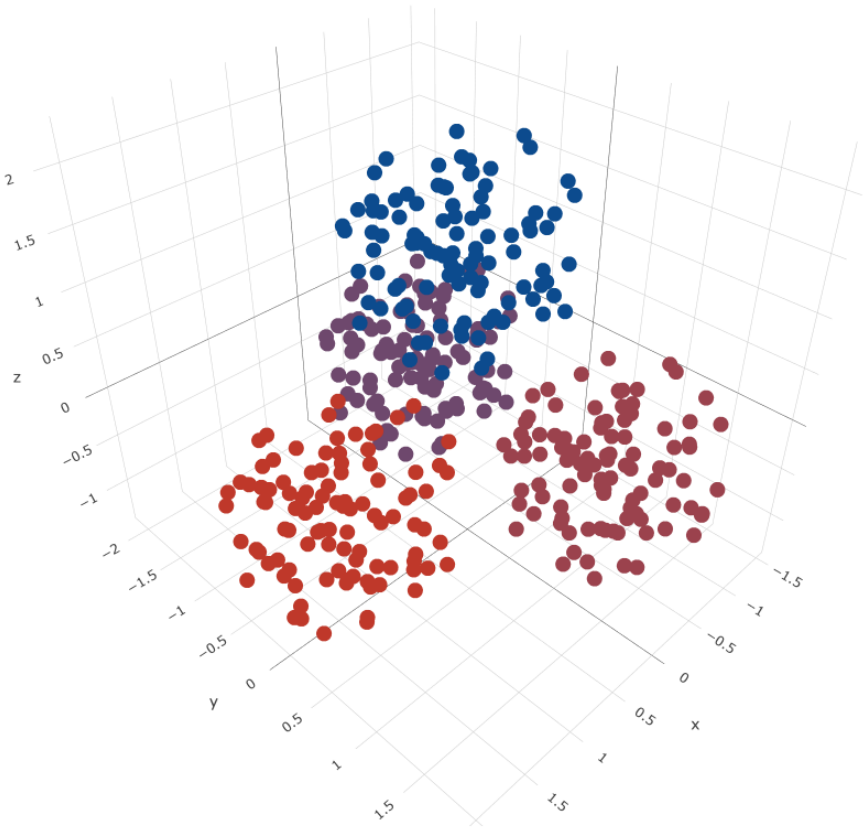


Figure 3.5: Visualisation of Tetra Dataset

Tetragonular Bee Dataset

The tetragonular bee dataset is a genetic dataset for 236 tetragonular bees from Australia and Southeast Asia, with location and species information provided in [144]. The dataset delimitates clusters using the number of bee species in the dataset and the bees that belongs to a species. Species are defined by interbreeding, which means that a much more significant genetic similarity is expected between species. Therefore, the distance matrix and the geographical origin of the species are important to cluster the dataset. The challenge lies in the smooth transition between clusters and outliers because clusters should have smaller intra-cluster than inter-cluster distances while remaining coherent with the geographic origins. The raw data is available in the R package prabclus on CRAN [148].

Glass Dataset

The dataset was based on a study that seeks to classify the types of crime based on the glass types found at the incident scene [149]. The glasses at the scene of the crime were classified into seven categories as listed below:

- building windows float processed
- building windows non-float processed
- vehicle windows float processed
- vehicle windows non-float processed
- containers
- tableware and headlamps

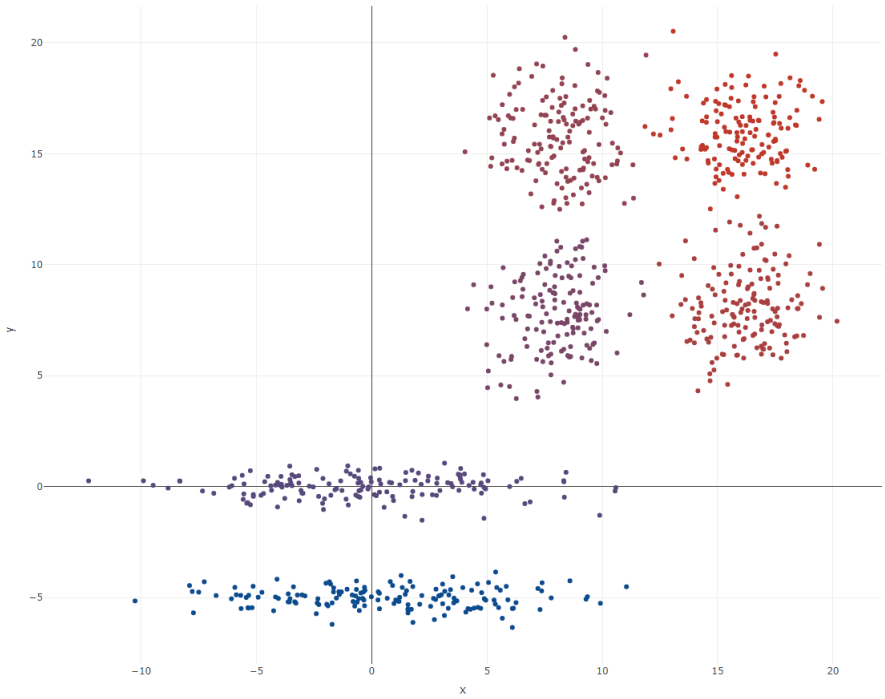


Figure 3.6: Visualisation of LongSquare Dataset

LongSquare Dataset

The LongSquare (Figure 3.6) dataset is a synthetic dataset that combines square1 and Long1 datasets. Although both datasets have 1000 observations, LongSquare has 900 observations [150].

Shapes Dataset

This dataset is synthetic and it can be generated using the R package 'mlbench'. It has four distinct shapes as shown in Figure 3.7.

Vehicle Dataset

This dataset is from the Turing Institute, Glasgow, Scotland. The purpose was to find an appropriate method to distinguish 3D objects within a 2D image using an ensemble of shape feature extractors. They used a rule

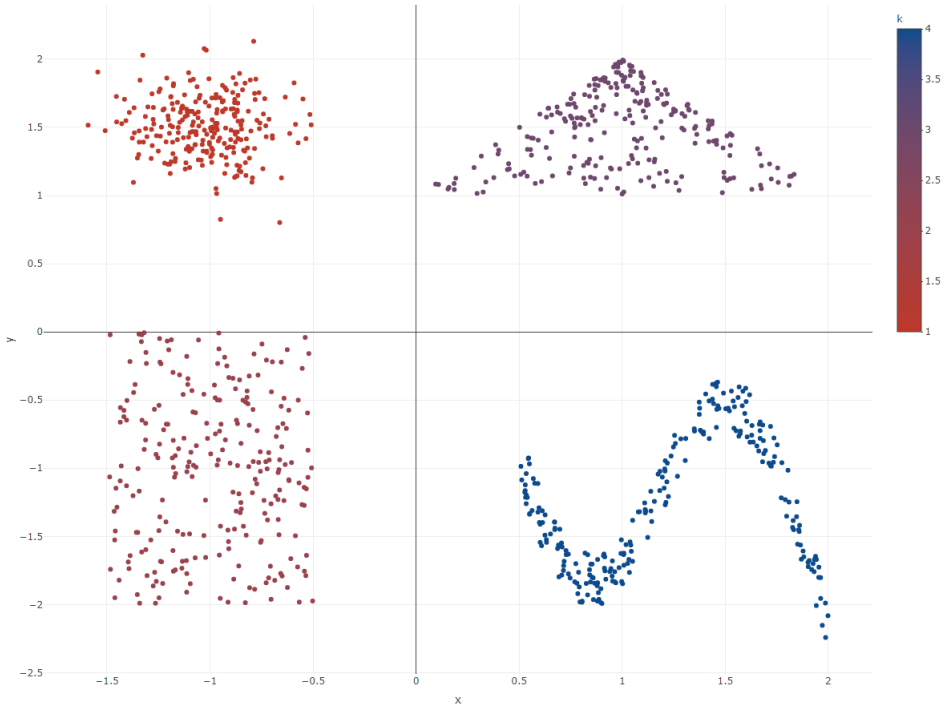


Figure 3.7: Shapes Dataset Visualisation

tree classification method. It compares favourably with other methods, such as MDC (Minimum Distance Classifier) and KNN (k-Nearest Neighbour), to successfully discriminate between model cars silhouettes, vans and buses when viewed from a certain angle of rotation. The number of clusters in the dataset is 4.

3.5 Clustering Ensemble Generation

In the initial phase, forty clustering algorithms were considered for the experiment. However, after careful evaluation and selection, the number of clustering algorithms was reduced to thirty majorly because some of the algorithms' performed poorly across all datasets used for the experiment. For example, algorithms removed from the ensemble have average weighted kappa values less or equal to 0.1, which in the current interpretation meant

that there were no clusters even when clear groupings were within the dataset. On the other hand, the thirty methods selected for the rest of the experiment have, on average weighted kappa values above 0.1 across the datasets tested. A total of thirty clustering algorithms, referred to as input methods, were employed in this study. Including a diverse set of methods to create the ensemble ensures the reliability of the techniques and facilitates the exploration of various combinations derived from them. The clusterings were generated using the R language implementation of the algorithms. Table 3.2 summarises the input methods chosen for this research and the number of variations implemented for each technique. The next section describes the R packages used in clustering the datasets.

3.5.1 R and The List of Packages

R stands out as a robust and highly adopted statistical environment, valued by researchers, data scientists, and statisticians due to its versatility and rich collection of packages. As an open-source platform, R offers various tools for statistical analysis, data visualization, and machine learning. Its syntax is designed to simplify data manipulation and analysis, making it well-suited for various statistical tasks.

For this thesis, R was chosen as the statistical environment, and this decision was influenced by the diverse packages available that enhance functionality and streamline specific analyses. The selection of packages is tailored to the project's requirements, and the following list itemizes some of the critical packages utilized in this thesis:

1. **dplyr:** This package is fundamental for data manipulation and transformation. It provides functions that make it easy to filter, arrange, and summarize data.

2. **ggplot2:** Widely recognized for its elegant and flexible plotting capabilities, ggplot2 is instrumental in creating informative and visually appealing data visualizations.

3. **stats:** This is the core package for basic statistical functions, providing essential tools for hypothesis testing, probability distributions, and summary statistics.

4. **caret:** Commonly used for machine learning tasks, caret streamlines the process of building and evaluating predictive models, facilitating model training and performance assessment.

5. **tidyr:** A package that complements dplyr, tidyr assists in reshaping and tidying data, making it easier to work within the context of various analyses.

6. **glmnet:** Useful for fitting generalized linear models, mainly when dealing with high-dimensional data or when regularization techniques are necessary.

7. **forecast:** Designed for time series analysis and forecasting, this package offers a range of tools for modeling and predicting future values in time-dependent datasets.

8. **tidyverse:** While not a single package, the tidyverse collection, including dplyr, ggplot2, tidyr, and others, promotes a consistent and efficient data analysis and visualization workflow.

Table 3.2: Description of methods used in clustering the dataset

Clustering Methods	Description	Variants
K-means	The following algorithms: Forgy, Lloyd, Macqueen and Hartigan were implemented in R “stats” package.	4
Hierarchical Clustering	The following agglomerative methods: Ward, Single, Complete, Average, Mcquitty, Median and Centroid using both Euclidean and distance correlation methods	14
Model-based clustering	Model based clustering is implemented using ”mclust” with five flavours	5
Affinity Propagation (AP) [103]	”apclust” a package in ’R’ was computed using the following similarity metric: negDistMat, expSimMat and linSimMat	3
Partitioning Around Medoids (PAM)	Two similarity methods:Euclidean and Correlation were, it is available in ’cluster’ package	2
Clara	Clara is available in the ’cluster package’	1
X-means	X-means recursively partition data into two disjoint sets available in ’clusternor’ package	1
DBSCAN	It is available in ’R’ package ’dbscan’	1

The following section discusses the datasets, features, and distinctive clustering characteristics.

3.6 Variability by Method and Dataset

The degree of variability reported here is two-fold: variability by methods and by datasets. In this sub-section, a comparison was made regarding the degree of variability in each dataset with the rest of the data after manually removing those that did not pass the initial tests for size, missing values, and those with an unknown number of clusters. In data analysis, variability refers to the extent to which data points or measurements differ. It measures how much the data points or measurements vary or deviate from the

average or expected value. Variability is a critical concept in data analysis, as it can significantly impact the results and conclusions drawn from analysing a dataset. High variability can indicate that the data points are spread widely, while low variability suggests that the data points are clustered more closely. Understanding the variability of a dataset is crucial for making accurate statistical inferences and identifying trends and patterns in the data. It can be measured using a variety of statistical measures, such as standard deviation, range, or interquartile range.

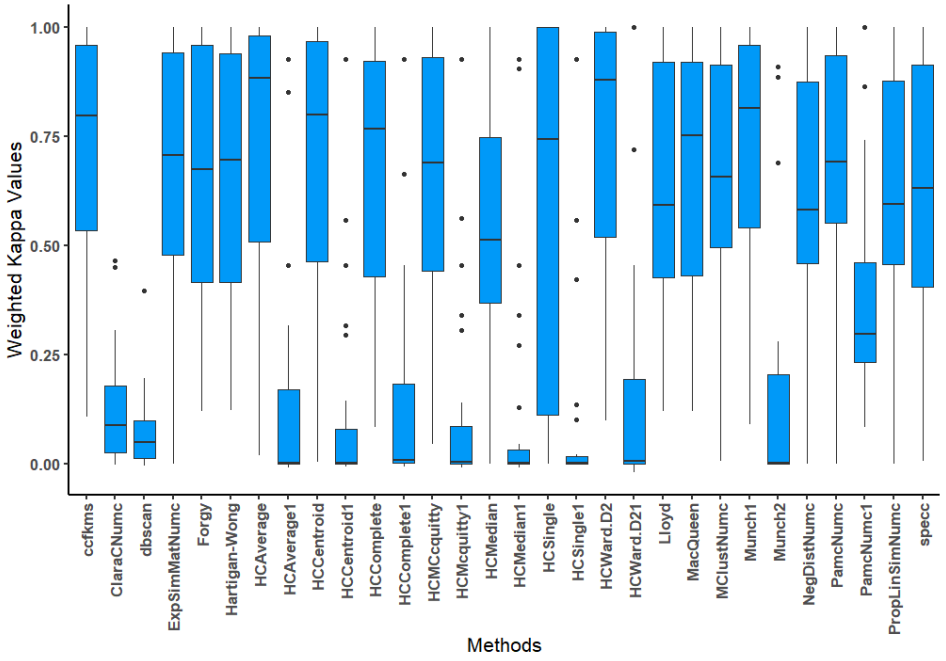


Figure 3.8: Variability of Methods based on Weighted Kappa Values

The variability by dataset and methods is based on the average Weighted Kappa values from the twenty-seven methods and the thirty clustering techniques used as inputs. Figures 3.8, 3.9] show the average Weighted Kappa values, the standard deviation and the range of values (interquartile range) in each of the twenty-seven datasets reported. There is medium variability across all the twenty-seven datasets, with values ranging from 0.01 to 1. The reduction in variability is due to the initial cleaning of the dataset by re-

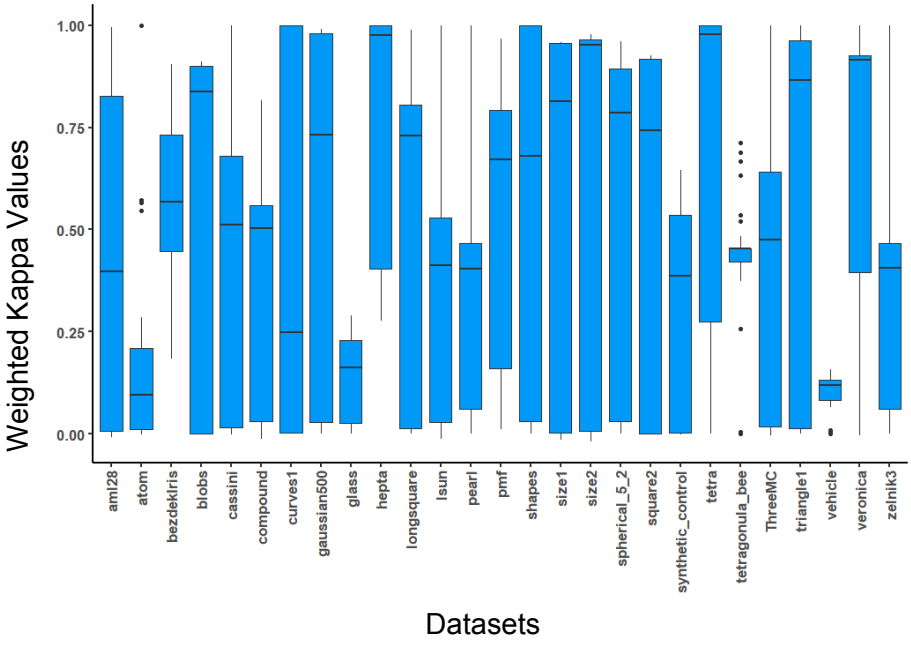


Figure 3.9: Variability of Datasets based on their Weighted Kappa values

moving the dataset with missing values and through dataset normalisation. The variability is detailed in Figures [3.8, 3.9].

In conclusion, a considerable number of clustering methods, including ccfkms, HCSingle, HCAverage, HCCentroid, HCComplete, and specc, exhibit higher variability compared to methods such as HCCentroid1, ClaraC-Numc, and HCSingle1, which show lower variability. However, the latter set of methods tends to produce outliers in the output. Similarly, a certain degree of skewness is observed in HCCentroid1, HCSingle1, HCMedian1, and other methods. Based on the above observations, significant variability exists among the results obtained from the different clustering methods applied to the datasets. When examining the variability across datasets, the length of the box reveals that approximately 80% (22 out of 27) of the datasets exhibit noticeable variability, as indicated by the spread of the middle 50% of the data. However, three datasets: atom, tetragonular_bee, and vehicle, displayed outliers in their output, confirming the earlier findings re-

garding the variability observed across different methods.

In conclusion, the variability analysis carried out in this chapter is vital for several reasons. Firstly, it plays a crucial role in assessing data quality and reliability. Inconsistencies and errors that may affect the analysis results and outliers can be identified and eliminated during the initial data analysis process, ensuring that conclusions drawn from the dataset are based on reliable and representative information. Secondly, variability analysis aids in feature selection and determining feature importance, i.e. the most informative and significant features can be selected, allowing the exclusion of irrelevant or low-variance features, resulting in more efficient and accurate models. Furthermore, variability analysis provides insights into the dataset's values, patterns, and distribution range. This understanding of data variability helps uncover underlying relationships, trends, and clusters, contributing to a better understanding of the data patterns and facilitating informed decision-making. Variability analysis is essential for data quality assessment, feature selection, and understanding data patterns.

In the next chapter (Chapter 4), we introduce the ensemble method built upon the framework discussed in Chapter 2; this approach employs a sequence of transformations to uncover the optimal solution. In the framework, each potential solution is represented as a subset and meticulously evaluated to ascertain its suitability for determining the ideal number of clusters within datasets. Chapter 4 explores the ensemble framework and its use in estimating the number of clusters.

Chapter 4

ESTIMATING NUMBER OF CLUSTERS USING THE ENSEMBLE FRAMEWORK

4.1 Introduction

This chapter focuses on the ensemble framework, which provides a way to determine the number of clusters in a dataset. As noted in earlier chapters, estimating the number of clusters in cluster analysis is often complicated by the absence of a clear and universally accepted definition of what constitutes a “cluster.” In addition to exploring strategies for ensemble-based cluster analysis, this chapter also examines techniques for combining ensembles to achieve maximum diversity. The discussion draws from a research paper presented at the International Conference on Data Science, Technology and Applications, entitled ”Estimating the Optimal Number of Clusters from Subsets of Ensembles” [16].

4.2 Background

Clustering, as earlier defined, is assigning similar data points to the same cluster and dissimilar points to different clusters without prior knowledge of the members' labels [5]. It often involves determining the number of clusters by learning from similarity or dissimilarity between objects or points in the dataset to be clustered. This process is a subtle way of unravelling the pattern or the underlying structure in the dataset from where other analyses can commence.

This chapter explores ensemble methods for estimating the number of clusters in datasets. The concept of ensemble clustering was initially introduced by Strehl and Ghosh in 2002 [96]. This technique has enhanced clustering performance by generating multiple dataset partitions and merging them to create a comprehensive summary clustering solution. In this thesis, the goal is to extend the application and model of ensemble clustering to the domain of unsupervised learning.

The utilisation of clustering ensembles raises two fundamental questions that need addressing:

- What is the optimal way to generate clusterings and combine them into representative solutions while maintaining diversity and promoting accuracy?
- How can the best solution be optimally identified from the pool of representative solutions (subsets)?

To address these questions, an exploration was conducted, which involved creating all possible subsets of ensembles. Subsequently, an agreement matrix was constructed, capturing cluster similarity across various clustering algorithms for each value of the number of clusters (k). Section 2.5 provides a detailed description of the agreement matrix. The subset exhibiting

the highest agreement, as determined by a quality function, is considered the best, and its index represents the estimated number of clusters in the dataset. However, conducting an exhaustive search for the best subset can be computationally demanding, particularly as the number of clusterings increases.

A mapping technique that transforms the outputs from the agreement matrix into Gray code subsets is utilised to overcome this challenge. For detailed information regarding the description of Gray code subsetting, refer to Section 4.2.1. This mapping process facilitates the generation of subsets that are subsequently evaluated by the quality function. Gray code representations are advantageous because they have only a single-digit difference between successive members. In Section 4.6, the runtime of the quality function and the update quality function from Gray code implementation show the algorithm's efficiency is significantly improved.

This study emphasises the proposed approach's effectiveness in achieving accurate outcomes across diverse datasets, distributions, and datasets with outliers. The performance of the approach consistently surpasses that of similar methods. Unlike certain approaches that heavily rely on specific data distributions, such as the Gaussian distribution, the current approach overcomes this limitation and mitigates the risk of overfitting. This adaptability and resilience to various data distribution characteristics contribute to the robustness and reliability of the proposed approach. An important factor contributing to these advantages is using the Weighted Kappa agreement matrix, derived from multiple clusterings, rather than solely relying on the individual algorithms' produced clusterings. This careful consideration enhances the quality and accuracy of the estimated clusters.

In a prior related study conducted by Samy Ayed [12], it was shown that a substantial correlation exists between the average Weighted Kappa (w)

of pairs of inputs and the average Weighted Kappa (w) of each input when compared to the expected number of clusters (gold standard). This finding implies that selecting the best subset, which corresponds to the subset with the highest average Weighted Kappa, should strongly align with the gold standard or the number of clusters in a dataset, even without prior knowledge. In other words, it can serve as a reliable proxy for the gold standard. Apart from this key insight, the research also presents the following contributions:

- Development of a selection scheme that systematically generates a solution space by creating subsets of ensembles from the clusterings.
- Formulation of an objective function that assesses the quality of a subset and identifies the subset that optimises the objective function, thereby maximising quality.
- Establishment of a mathematical framework for quality metrics. These metrics enable larger subsets to be evaluated on an equal footing with smaller subsets using a threshold value.

4.2.1 Why Gray Codes?

The Gray code, introduced by Frank Gray [151], is a type of code that exhibits two important properties. Firstly, it is a single-distance code, meaning that neighbouring code words differ by only one digit position. Secondly, it is cyclic. The cyclic property of Gray code refers to the ability of the code sequence to loop back to its initial value after reaching the maximum value. In other words, the code sequence is designed to form a closed loop. This property ensures that adjacent code words in the sequence differ by only one digit position, even when transitioning from the last code word to the first one. For example, consider a binary Gray code sequence for 3-bit numbers: 000, 001, 011, 010, 110, 111, 101, 100. If we continue the sequence after

100, the next code word will be 000, completing the cycle.

This property is handy in various applications, such as minimising errors in analogue-to-digital converters, reducing noise in communication systems, and generating subsets or permutations of elements in combinatorial problems. It ensures smooth transitions between code words and minimises the potential for errors or abrupt changes, which is particularly useful in our current representation of subsets.

Leveraging the single-digit difference between successive subsets allows subsequent quality values to be calculated from the previous quality using the update quality function (Equation 5.2). With ten input algorithms, a total of $1,013$ Gray codes are generated, each consisting of at least two members (subsets). The matrix below illustrates an example:

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \end{bmatrix}$$

Consecutive members of the Gray code sample (matrix) below differ by a single digit.

4.2.2 How Does a Cluster Estimator Work?

A typical estimator seeks to optimise an objective function, for example, the sum of the square distance between each point in a dataset and its closest or assigned centre, as shown in Figure 4.1. The figure describes the "peak of a curve" based on different values of k (number of clusters) from 2 ... $\sqrt{n}$ (n = number of objects to be clustered); see Section 5.3.3 for the motivation for $\sqrt{n}$. The peak is a cutoff point commonly used in heuristic and

mathematical optimisation to determine where adding another cluster only results in breaking the cluster into clusters within identified cluster rather than any appreciable difference between clusters. The graph's peak, as shown in Figure 4.1, corresponds to the optimal value of the objective function; adding another cluster to it from that point does not model the dataset better; instead, it may result in over-fitting.

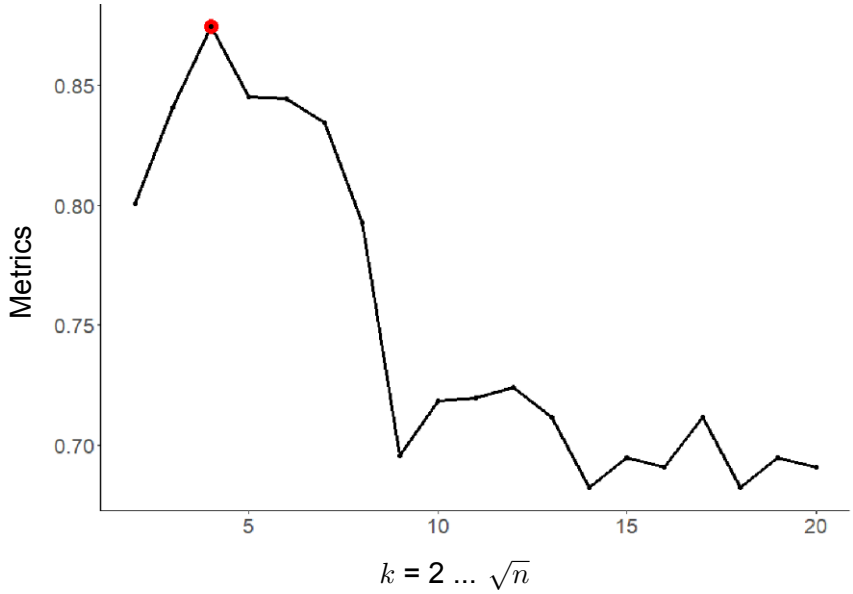


Figure 4.1: Representation of Optimal Number of Clusters in a Dataset.

4.3 Estimating the Number of Clusters

There are numerous methods available for determining the number of clusters in datasets, although many of them are no longer considered up-to-date or commonly used. A range of standard methods and indices were selected to be compared with methods designed in this research to demonstrate their performance. The first categories of techniques used are those that cluster datasets and report the number of clusters as part of the output; essentially, they are clustering algorithms. Three methods were considered in this cate-

gory. First, x-means [53] for example, provides a framework for estimating the number of clusters in datasets using k with the best Bayesian Information Criterion [152] score. However, x-means assumes that the width of the covariances is identical and spherical, thus limiting the method to specific data distribution. x-means is one of the methods used in generating the initial clusterings.

Harmer and Etkan [153] proposed the G-means algorithm. The algorithm grows the value of k starting with a small number of centres and tests if the data is from a Gaussian distribution using a statistical test. Those not from Gaussian distribution are split into two repeatedly until all assume Gaussian distribution. Although G-means work well, if the data is well separated, it can encounter difficulty with overlapping data. In this category, Expectation-Maximisation (EM) algorithm was considered; unlike distance-based and hard clustering algorithms such as k-means, EM constructs statistical models of the data and accommodates categorical and continuous data fields with varying degrees of data membership in multiple clusters.

The second set of methods evaluated in comparison to our ensemble techniques consists of thirty classical methods from R's NbClust package [2], which are detailed in Table 4.1. Some of these methods will be explained, with a focus on the top five based on their outputs/errors when compared to the actual number of clusters in the dataset. One of the highlighted methods is Gap statistics, discussed in Chapter 2, Section 2.4.3. This method assesses the total intra-cluster variation for various cluster values and compares them to their expected values under specific distributions, such as the null reference distribution. Although it is good at identifying well-separated clusters, it can sometimes overestimate the number of clusters [154] for certain distributions, e.g. exponential distributions [155]. The methods were also compared with the Silhouettes technique described in Chapter 2. The

Table 4.1: A summary of the indices implemented in NBClust [2]

S/N	Method	Authors
1	"ch"	Calinski and Harabasz 1974
2	"duda"	Duda and Hart 1973
3	"pseudot2"	Duda and Hart 1973
4	"cindex"	Hubert and Levin 1976
5	"gamma"	Baker and Hubert 1975
6	"beale"	Beale 1969
7	"ccc"	Sarle 1983
8	"ptbiserial"	Milligan 1980, 1981
9	"gplus"	Rohlf 1974; Milligan 1981
10	"db"	Davies and Bouldin 1979
11	"frey"	Frey and Van Groenewoud 1972
12	"hartigan"	Hartigan 1975
13	"tau"	Rohlf 1974; Milligan 1981
14	"ratkowsky"	Ratkowsky and Lance 1978
15	"scott"	Scott and Symons 1971
16	"marriot"	Marriot 1971
17	"ball"	Ball and Hall 1965
18	"trcovw"	Milligan and Cooper 1985
19	"tracew"	Milligan and Cooper 1985
20	"friedman"	Friedman and Rubin 1967
21	"mcclain"	McClain and Rao 1975
22	"rubin"	Friedman and Rubin 1967
23	"kl"	Krzanowski and Lai 1988
24	"silhouette"	Rousseeuw 1987
25	"gap"	Tibshirani et al. 2001
26	"dindex"	(Lebart et al. 2000)
27	"dunn"	Dunn 1974
28	"hubert"	Hubert and Arabie 1985
29	"sdindex"	Halkidi et al. 2000
30	"sdbw"	Halkidi and Vazirgiannis 2001

Silhouettes [80] technique uses partitions from the clustering and the collection of proximities between the objects to construct the Silhouette plot. Lastly, included in the list are Calinski Harabasz [156] (CH) index, Ball [86], Ratkwosky [83], Krzanowski Lai [78], and Milligan [157] as implemented in R's NbClust package [2]. The Calinski index, like all the other indices, maximises the CH index and is computed as shown in Equation 4.3. k is the number of clusters, n is the number of data points, B_k is the between-cluster sum of squares, and C_k is the within-cluster sum. The rest of the methods in this category seek to maximise a value or an index. The top five methods, as well as other standard methods such as Silhouette, Gap, and Expectation Maximization (EM), were selected. Their results, along with a comparison to the ensemble technique, are presented in the Results and Discussion section (Section 4.7).

$$CH(k) = \frac{B_k/(k-1)}{C_k/(n-k)}$$

In summary, the common theme of the above methods is that they are all based on a single input method, such as k-means or hierarchical clustering, for generating the clusterings. The current method explores multiple clusterings to increase diversity, thus encouraging inputs from both strong and weak clusterings for the optimal estimate. However, it is important to note that neither approach can guarantee optimal performance in all scenarios. They are subject to over-fitting, under-fitting, or high computational complexity. Nonetheless, an optimistic outlook is maintained, and it is believed that the current method can effectively mitigate the problem of over-fitting. Section 4.4 provides a detailed description of the ensemble framework, illustrated in Figure 4.2. This framework is the foundation for the proposed method and will be further elaborated, highlighting its key components and

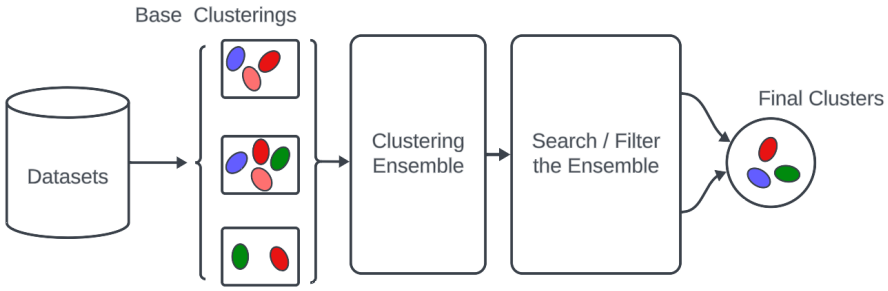


Figure 4.2: Ensemble Framework

functionality.

4.4 The Ensembles Framework

While the framework presented is general to most ensemble clustering in that the ensemble technique reconciles clustering information [94] or derives a single set of clusters from several clustering methods [13], the focus of the revised framework is on two crucial processes: the pre-processing and optimisation steps. The pre-processing step uses clusterings from the input methods to construct the agreement matrix, while the optimisation determines the optimal clustering arrangement using an objective function applied to the agreement matrix. More theoretical framework for clustering ensemble can be found in the following references [96, 13, 7, 97]. With this adjustment, the current ensemble design consists of four key stages:

- Generation of the base clusterings.
- Constructing an agreement matrix from the input clusterings.
- Creating all possible subsets from the agreement matrix.
- Determining the subset that optimises the objective function.

The primary motivation for creating subsets in this model is twofold. First,

create the potential solutions from subsets' similarity matrix with maximum diversity. Second, search for the best subset from the pool of possible solutions, thus reducing complexity. The objective is designed to search all subsets in the current implementation. Figure 4.2 summarises the different stages in the current implementation. To reduce the complexity associated with an exhaustive search of the solution space, especially as the input clustering increases and for large datasets, a mathematical framework and an improved version of the search process are provided in the update quality as shown in Section 4.5.2. The update quality function runs in linear time in terms of the subset size with simulated data (simulated datasets were used to assess the speed gain, not the accuracy).

4.4.1 Subsets Generation

Different approaches exist in the literature for producing the initial partitions, including generating clusterings for different values from a single clustering method or using multiple clustering methods to generate clusterings. The current approach combines both. In generating the subsets, sets of input clustering arrangements ranging from k = 2 to k = $\sqrt{n}$. $\sqrt{n}$ is the commonly suggested maximum number of clusters when the number is unknown [158]), where n is the number of observations. Thirty variants of different clustering algorithms were ranked to select the clustering (m) for input, and the top ten were selected. The reason for choosing the top ten is based on the algorithms' performance against the gold standard (expected number of clusters). Algorithms that performed poorly for the two hundred and eighteen datasets initially selected for the experiment, for example, cases where the Weighted Kappa is below 0.1 [12], were removed from the clustering generation. The methods are listed below, more details about the methods can be found in Chapter 3:

- Three versions of k-means in the R 'stats' package (Macqueen, Hartigan-Wong, and Lloyd)
- Two Hierarchical agglomerative methods (Complete and Average methods)
- Partition Around Medoids - Robust version of k-means (R 'cluster' package)
- Clara deals with larger datasets effectively
- x-means - Partitions data into two disjoint sets as described in the 'clusternor' package.
- DBSAN (Density-Based Clustering of Applications with Noise (DBSCAN) and Related Algorithms)
- ccfkms - k-means based on conjugate convex functions which use sparse data structures for centring found in the 'cba' package in R. More detail about the methods can be found under the literature review in Chapter 2

At the outset, it is crucial to ascertain the potential maximum number of clusters, denoted as k, for each dataset since this information is often not readily available. As previously mentioned, a widely recommended estimate for k is $\sqrt{n}$ [158], where n corresponds to the number of instances in the dataset. Next, the clustering values of k from the input clusterings are compared and evaluated using the Weighted Kappa metric described earlier in Chapter 2. The degree of agreement between the clusterings establishes the agreement matrix. The adjacent values of the input clusterings are compared and assigned ratings to construct the agreement matrix.

The subsets are created using the agreement matrix for each k, the number of clusters. In the current experiment, ten clustering algorithms were

used to generate the ensembles. The variable r represents the number of input algorithms, corresponding to the number of clustering inputs utilised in creating the ensemble. The binary codes of size 2^r determine the total number of clusterings generated. Each subset contains a minimum of two members. These subsets are formed by mapping the binary values of the strings to the corresponding columns in the agreement matrix. For instance, let's consider a string with binary value 1000110110. This string forms a subset that includes columns 1, 5, 6, 8, and 9 selected from the Weighted Kappa agreement matrix. The next step involves finding the best subset for the estimation. This process is explained in Equation 4.1, which also includes the derivation of the quality metric.

Algorithm 4.12 Gray Code Implementation of Exhaustive Search Algorithm

Input: $m \times m$ agreement matrix from clustering algorithms

for i = 0 to 2^{m-1} **do**
  g = binary(i) ▷ generate the i^{th} binary Gray code
  if $nbits(g) > 1$ **then** ▷ test if the subset size > 1
    s = $subset(g)$ ▷ create subsets
    $count = 0$
    q = $Q(w, s, \theta)$ ▷ as per Equation 4.1
    if ($q > bestQ$) **then**
      $bestSS$ = s
      $bestQ$ = q
    end if
  end if
end for

Output: Subset with the best quality

4.4.2 Determining The Best Subset

The process of determining the best subset is outlined in Algorithm 4.12. In this algorithm, the variable "binary" represents either the regular 2^m combinations or the binary codes representation from Gray code. For this particular experiment, threshold values of 0.4 and 0.6 are used, as described in Section 2.2. These threshold values correspond to the categories of fair

and moderate agreement. Additionally, two other measures of central tendency, namely average and median, are employed and denoted as θ in the quality function. The algorithm defines the best quality as $bestQ$ and the best subset as $bestSS$. The subsets are generated using the $subset$ function, which implements the mapping from the agreement matrix to the Gray code.

4.5 Quality Function Description

The quality function, also known as the objective function or fitness function, is a mathematical function used to evaluate the quality or fitness of a solution or model. It quantifies how well a solution or model performs with respect to a specific criterion or goal. The quality function takes input from the solution or model space and assigns a numerical value that reflects the desirability of the solution or model. The higher the value, the better the quality. In the current case, as in most optimisation problems, the quality function helps guide the search for an optimal solution by measuring how well each candidate solution performs. It guides the search towards better solutions, allowing algorithms to iteratively explore the solution space and converge towards optimal or near-optimal solutions. This section describes the mathematical framework for quality and the update quality functions used to determine the best subset.

4.5.1 The Quality Function

The quality function (Q) is utilized to assess the accuracy of a subset (s) in estimating the number of clusters in a dataset. It measures the sum of agreements derived from the Weighted Kappa values of adjacent inputs, considering a threshold value (θ). The quality of a subset is summarized by Equations (4.1) and (4.2).

The average Weighted Kappa value is employed to plot the elbow-like graph, which indicates the peak average value (A_v) that corresponds to the estimated number of clusters in the dataset. However, if the average Weighted Kappa value were used alone to determine the quality, it would select a two-variable subset with the best pair from the subset (w). Instead, the average (A_v) is used for plotting purposes, as shown in an earlier example in Section 4.1. The function described in Equation 4.1 determines a subset's quality and the average is in Equation 4.2.

$$Q = \sum_{i=1}^{|s|-1} \sum_{j=i+1}^{|s|} [w(s_i, s_j) - \theta] \tag{4.1}$$

The average is

$$A_v = \sum_{i=1}^{|s|-1} \sum_{j=i+1}^{|s|} \frac{[w(s_i,s_j)]}{\frac{|s|(|s|-1)}{2}} \tag{4.2}$$

where

$$\hat{s} = \frac{|s|(|s|-1)}{2}$$
$$Q = \hat{s}(A_v - \theta)$$
$$\frac{Q}{\hat{s}} + \theta = A_v$$

4.5.2 The Update Quality Function ($\hat{Q}$)

On the other hand, the update quality function is used in iterative optimisation or learning algorithms to update the quality estimate of a solution. The update quality function is often designed to incorporate new information or adjust the quality estimate based on the feedback obtained during the opti-

misation or learning process. It allows the algorithm to adapt and refine the quality estimates over time, improving the accuracy and reliability of the optimisation or learning process. The current description of the update quality uses the previous value of the subset as input for the next subset quality, reducing the number of iterations significantly.

As represented by Equation (4.1), the quality function exhibits a quadratic runtime complexity as the number of input clusterings increases because each quality value is calculated at every iteration. Hence the need for an updated version of the quality function built to leverage single-digit differences between consecutive Gray codes. By exploring this property of Gray codes, the updated quality function reduces computational overhead through incremental changes between consecutive Gray codes rather than recalculating the entire quality values for each subset.

The next quality is calculated from the previous quality value depending on the bit difference between the Gray code; if the difference from the previous is a 0, then the column's difference in the agreement matrix is added; otherwise, it is subtracted. The Gray code version of the quality function dramatically speeds up the search process and avoids recomputing the quality values of subsets on every iteration. A mathematical derivation of the updated quality function is described in Equations 4.3, 4.4, and 5.2. The quality function Q is used to find the best quality from all possible subsets of inputs, and the average quality is used to plot the elbow-like curve that depicts the best subset in the diagram in Figure 4.1.The optimal subset corresponds to the best quality and is determined based on the dataset's clusters. The index of this optimal subset is indicative of the number of clusters present in the dataset. Consider the quality function denoted by Q, as expressed in Equation 4.3:

$$Q = \sum_{i=1}^{|s|} \sum_{j=1}^{|s|} [w(s_i, s_j)) - \theta] \tag{4.3}$$

This function quantifies the quality of a given subset s by summing the pairwise difference between its elements using the weight function $w(\cdot, \cdot)$, adjusted by the threshold θ.

The update quality enhances the computation by calculating the next quality from the previous, as represented below:

$$\begin{aligned} \hat{Q} = \sum_{i=1}^{|\hat{s}|-1} \sum_{j=1}^{|\hat{s}|-1} [w(s_i, s_j) - \theta] \\ +2 \sum_{j=1}^{|\hat{s}|-1} w(s_i, x) - \theta \end{aligned} \tag{4.4}$$

Furthermore, Equation 5.2 introduces a more concise representation of the update quality function:

$$\hat{Q} = \begin{cases} Q + 2\sum_{i=1}^{|\hat{s}|-1} w(s_i, x) - \theta & \text{positive,} \\ Q - 2\sum_{i=1}^{|\hat{s}|-1} w(s_i, x) - \theta & \text{otherwise.} \end{cases} \tag{4.5}$$

Lastly, the Weighted Kappa guideline [13] is employed to select two thresholds around the mid-point ($0.4 \equiv fair$, $0.6 \equiv good$); the two thresholds were used to ensure a combination of good subsets (high threshold) and fair subsets in the cluster estimates. Also, an examination was conducted on the choice of various threshold values and their impact on the estimated number of clusters. This exploration enables the algorithm to consider all possible solutions for identifying the best subset. Two standard statistical measures - average and median Weighted Kappa were also included as part of the measure. Intuitively the average Weighted Kappa was the best option in the results, as shown in Table 4.3.

Table 4.2: Errors from the Eight Methods and the Average Ensemble

Datasets	EM	CH	Gap	Silhouette	PtBiserial	Ratkwosky	Ball	KL	Ensemble
Aml28	0.800	0.400	0.400	0.200	0.200	0.400	0.400	0.400	0.400
Atom	3.500	3.000	0.500	3.000	3.000	1.500	0.500	0.000	3.000
BezdekIris	0.333	0.000	1.000	0.333	0.333	0.333	0.000	0.333	0.000
Blobs	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.333	0.000
Cassini	1.333	0.667	0.333	0.333	0.333	0.667	0.000	0.667	0.000
Compound	0.800	0.600	0.400	0.600	0.600	0.400	0.400	0.200	0.400
Curves1	2.000	1.667	2.333	0.333	0.333	0.333	0.000	0.333	0.000
Gaussian500	0.000	0.000	0.000	0.000	0.400	0.200	0.400	0.200	0.000
Glass	0.429	0.143	0.429	0.714	0.714	0.571	0.571	0.714	0.429
Hepta	0.000	0.000	0.000	0.000	0.000	0.429	0.571	0.000	0.571
Longsquare	0.167	0.333	0.000	0.667	0.667	0.667	0.500	0.667	0.000
Lsun	0.667	1.000	1.000	0.667	0.333	0.000	0.000	1.000	1.000
Pearl	1.000	1.667	1.333	1.667	1.000	0.000	0.000	0.333	0.667
PMF	0.000	0.000	0.600	0.200	0.200	0.600	0.400	0.000	0.200
Shapes	1.250	0.500	0.500	0.000	0.000	0.250	0.250	0.500	0.000
Size1	0.000	0.000	0.000	0.000	0.000	0.250	0.250	0.000	0.000
Size2	0.000	0.000	0.000	0.000	0.000	0.250	0.250	0.000	0.000
Spherical_5_2	0.000	0.000	0.000	0.000	0.200	0.400	0.400	0.600	0.000
Square2	0.000	0.000	0.000	0.000	0.000	0.000	0.250	0.000	0.000
Synthetic_control	0.667	0.000	0.000	0.167	0.667	0.167	0.500	0.667	0.000
Tetra	0.000	0.000	0.000	0.000	0.000	0.000	0.250	0.000	0.250
Tetragonular_bee	0.111	0.111	0.111	0.111	0.111	0.667	0.667	0.667	0.333
ThreeMC	1.000	1.667	0.667	0.667	0.000	0.000	0.000	0.667	0.333
Triangle1	0.000	0.500	0.250	0.000	0.000	0.250	0.250	0.250	0.000
Vehicle	0.500	0.500	1.000	0.500	0.500	0.500	0.250	0.500	0.000
Veronica	0.143	0.000	0.429	0.000	0.000	1.143	0.571	0.000	0.000
Zelnik3	1.000	1.667	1.333	1.667	1.000	0.000	0.000	0.333	0.000
Average Errors	0.596	0.524	0.460	0.431	0.387	0.374	0.281	0.379	**0.271**
Correct Estimates	**10**	**12**	**10**	**11**	**10**	**7**	**8**	**8**	**17**

4.6 Results and Discussions

The methods in this research estimate the number of clusters in datasets using subsets from the input selected from binary and the Gray code clusterings. The results of four experiments were presented using different thresholds of Weighted Kappa values: average, fair, moderate, and median. To determine which of the four values most accurately predicts the number of clusters in the datasets, cases were recorded where the predictions deviated, noting how far the predicted results differed from the actual number

of clusters. The cumulative errors of the twenty-seven datasets are shown for each case in Table 4.2. The speed difference between the update and quality was also measured, and the results are reported below.

Table 4.3: Errors for the Ensembles(Fair, Moderate, Median and Average)

Datasets	Fair (0.4)	Moderate (0.6)	Median	Average
Aml28	0.400	0.600	0.000	0.400
Atom	2.000	2.500	3.000	3.000
BezdekIris	0.000	0.000	0.000	0.000
Blobs	0.000	0.000	0.000	0.000
Cassini	0.333	0.333	0.333	0.000
Compound	0.200	0.600	0.400	0.400
Curves1	1.000	1.333	0.000	0.000
Gaussian500	1.000	0.600	0.400	0.000
Glass	0.571	0.143	0.429	0.429
Hepta	0.143	0.000	0.571	0.571
Longsquare	0.167	0.333	0.667	0.000
Lsun	0.000	0.000	1.000	1.000
Pearl	1.333	1.333	0.667	0.667
Pmf	0.600	0.000	0.200	0.200
Shapes	0.000	0.000	0.000	0.000
Size1	0.000	0.000	0.000	0.000
Size2	0.000	0.000	0.000	0.000
Spherical_5_2	0.000	0.200	0.000	0.000
Square2	0.000	0.000	0.000	0.000
Synthetic_ control	0.000	0.000	0.167	0.000
Tetra	0.000	0.000	0.250	0.250
Tetragonular_bee	0.333	0.333	0.333	0.333
ThreeMC	0.000	0.000	0.333	0.333
Triangle1	0.000	0.000	0.000	0.000
Vehicle	0.000	0.000	0.000	0.000
Veronica	0.000	0.143	0.000	0.000
Zelnik3	1.667	0.000	0.000	0.000
Average Errors	0.366	0.302	0.313	**0.271**
Correct Estimates	10	16	10	**17**

4.6.1 Estimated Errors

The cumulative errors calculated using Equation 4.6 for the twenty-seven datasets shown in Section 4.2 is the average difference between the predicted values for each method and the number of clusters. The absolute value of the difference is normalised, and the results are compared. There are several reasons for normalising the errors in this context; here are a few key reasons:

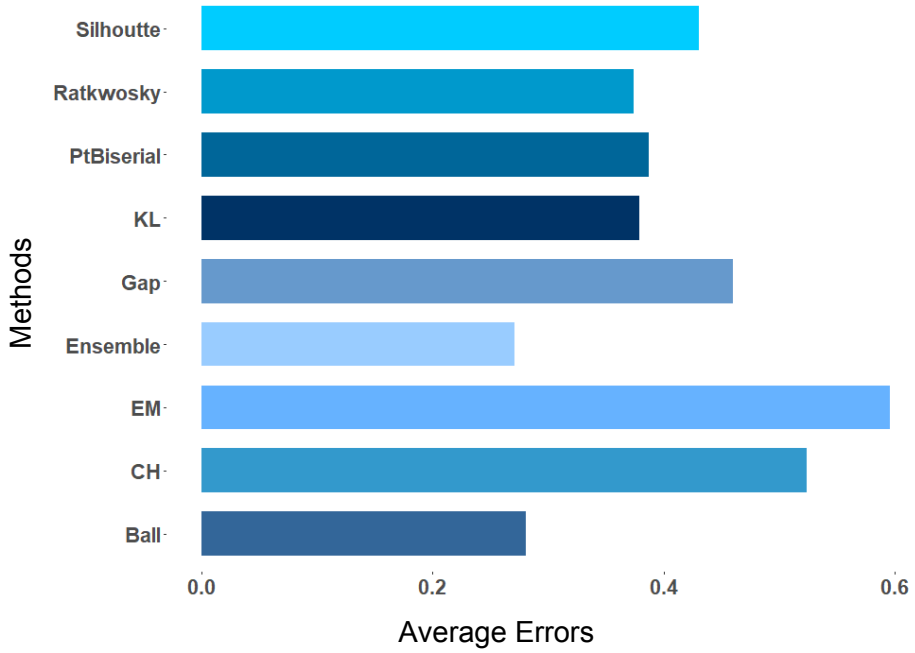


Figure 4.3: Normalised Average Errors on the Twenty-Seven Datasets

- Scaling error to a common range makes it easier to compare the magnitude of errors across different quality values which may sometimes produce values that are significantly different from one another
- Normalising the errors makes them more interpretable and meaningful. The significance or impact of the errors can be easily inferred by scaling errors to a standard range or relative to the magnitude of the data.
- Normalising errors can facilitate statistical analysis and inference. Many statistical techniques and tests assume certain distributional properties or require variables to be on a similar scale. Normalising errors ensure that these assumptions are met, enabling the use of appropriate statistical methods.

$$Error = \frac{|Estimate - \#Clusters|}{\#Clusters} \tag{4.6}$$

Table 4.2 presents the error values for the eight methods compared to the ensemble (average). Across the different thresholds employed by the ensembles, the average threshold has the highest performance, correctly predicting seventeen datasets, followed by sixteen for the moderate threshold and ten each for the fair and median thresholds. Thus, the average ensemble serves as the benchmark for comparing other methods. In comparing the results with the best-performing ensemble (average), the Calinski Index (CH) method achieved the highest number of correct estimates with twelve. However, its error value of 0.524 is twice that of the ensemble technique (0.271), despite both methods correctly predicting the same number of clusters. Similarly, although the Ball method had the best error estimate of 0.281, it only accurately predicted eight clusters compared to the ensemble's seventeen.

Further analysis focuses on datasets where the ensemble exhibited higher errors than other methods; the following datasets: Atom, Compound, Glass, Lsun, and Tetragonular_bee, were examined. Among these datasets, the following methods: Ball, KL, CH, Ptbiserial, and Expectation maximisation, outperformed the ensemble. Notably, the Atom dataset presents unique clustering characteristics, with two clusters overlapping a convex hull, rendering it linearly non-separable. While the ensemble had an error of 3.00 in predicting the number of clusters, the KL method provided accurate estimates. However, on average, the KL method predicted correctly only eight clusters, while the ensemble achieved seventeen. Detailed examination of the Atom dataset's agreement matrix will shed light on whether it's shape or unique characteristics contributed to the observed error margin.

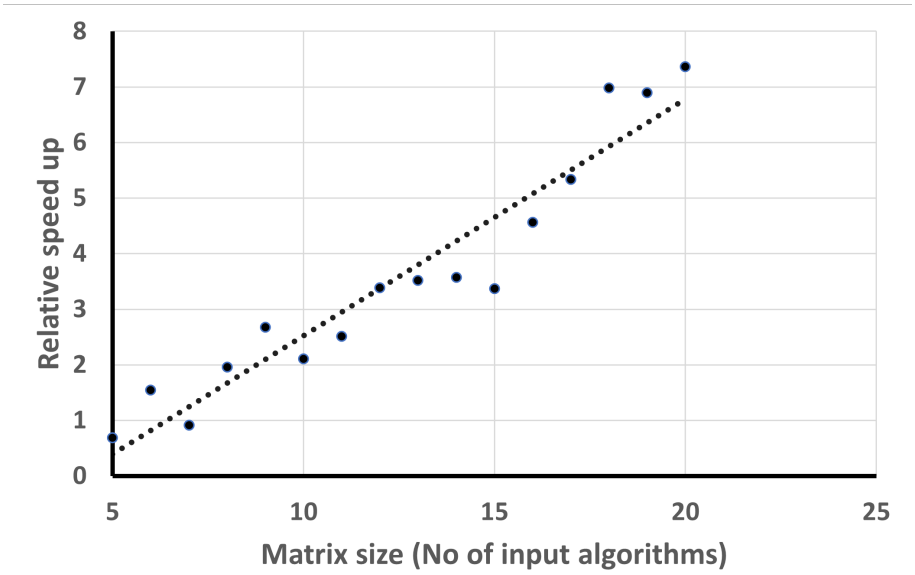


Figure 4.4: The speed gain for Exhaustive

Likewise, the Lsun dataset, as initially published in [146], presents challenges due to non-overlapping convex hulls, varying geometric shapes, and pockets of outliers. Investigating factors such as cluster shapes, variances between inner clusters, and cluster separations will help elucidate their influence on the ensemble's higher error estimate for this dataset.

In conclusion, the average error of the four ensemble techniques is lower across all datasets. Moreover, the clustering ensemble demonstrates a higher number of datasets correctly estimated compared to the other methods, providing further evidence for its superiority compared to the methods considered.

4.6.2 Quality Vs Update Quality

The update quality uses the previous subset quality value to calculate the next quality. It compares the pairs of the subset (the unique pairs) for difference, and depending on the difference in combination, the next subset

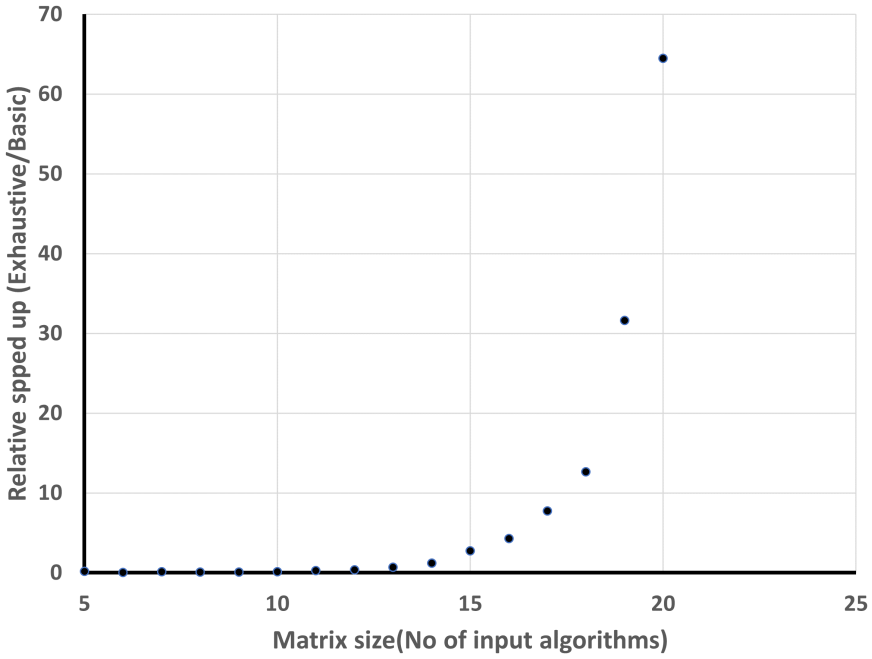


Figure 4.5: The speed gain of Exhaustive Vs $\hat{Q}$

quality is calculated as an update $\hat{Q}$ by adding or subtracting based on the difference as shown in Equation 5.2. Using previous values of the quality in calculating the next quality improves the time taken to calculate the subsequent subset's quality value and reduces the number of iterations. The quality and update quality performance were evaluated using simulated inputs. Simulated data was chosen to prioritise accuracy and assess runtime improvements. The simulated data consisted of a symmetric matrix generated using the random uniform distribution in the R programming language. The values in the matrix ranged from 0.1 to 0.97 (which could be adjusted to any range). The size of the symmetric matrix corresponded to the number of input clusterings, ranging from 5 to 20.

As anticipated, the empirical results revealed a superior performance of the Update quality implementation compared to the original quality function. This enhancement is visually represented in the graph showcasing the ratio

of quality (Q) to update quality ($\hat{Q}$), as depicted in Figure 4.5. These empirical findings substantiate the earlier theoretical anticipation [159] of improved speed and performance, as expressed in Equations 4.7 and 4.8.

The original quality function (Q) is defined by the equation:

$$\sum_{k=1}^{n} k^2 \binom{n}{k} = 2^{n-2} n(n+1) \qquad \text{Quality } (Q) \tag{4.7}$$

On the other hand, the update quality function ($\hat{Q}$) is expressed as:

$$\sum_{k=1}^{n} k \binom{n}{k} = 2^{n-1} n \qquad \text{Update Quality } (\hat{Q}) \tag{4.8}$$

Consequently, the speed gain achieved through the update quality function is quantified by the ratio $Q/\hat{Q}$, given by:

$$\frac{\text{Performance } (Q)}{\text{Performance}(\hat{Q})} = \frac{2^{n-2} n(n+1)}{2^{n-1} n} \tag{4.9}$$

$$= \frac{n+1}{2} \tag{4.10}$$

These equations and their interpretations affirm the theoretical predictions, highlighting the observed improvement in computational efficiency introduced by the update quality function.

4.7 Recommendations

This chapter introduces a novel ensemble technique, employing subsets of ensembles to estimate the number of clusters in datasets. Compared to similar methods, our approach shows promising performance in predicting the number of clusters and associated prediction errors, as demonstrated by the output from the ten input methods. While foreseeing a potential speed

challenge with an increase in the number of datasets and input methods, it's notable that the Gray code implementation addresses this concern, reducing the speed gain from quadratic to linear time as the dataset size increases. It would be interesting to explore whether a heuristic search approach could be applied to further speed up the speed in future implementations.

In conclusion, this chapter introduced a search-based ensemble method that builds upon the framework discussed in Chapter 4. The primary goal of this method is to improve the speed of execution while maintaining the same level of accuracy as the exhaustive search. As datasets become larger and the number of possible subsets from the ensemble increases, performance becomes an issue.

Moving forward, Chapter 5 will focus on the detailed exploration and evaluation of the heuristic search algorithm, an improved version of the Gray code implementation used as the update quality in this chapter. The chapter will delve into the design and implementation of the heuristic algorithms, assessing their effectiveness in achieving the desired speed enhancements to the current update quality. Additionally, experimental results and analysis will be presented to validate the performance improvements achieved by the heuristic search-based ensemble.

Chapter 5

HEURISTIC SEARCH BASED CLUSTERING ENSEMBLE

5.1 Introduction

This chapter presents a heuristic search-based ensemble approach that builds upon the framework introduced in Chapter 4. The objective is to enhance the execution speed while maintaining the same level of accuracy as the exhaustive search method. As the dataset size grows and the number of potential ensembles increases, there is a growing need for improved performance. To address this, we utilise algorithmic techniques such as Hill Climbing, previously introduced in Chapter 2.

The content of this chapter is based on a research paper titled “Using Clustering Ensembles and Heuristic Search to Estimate the Number of Clusters in Datasets.” This paper will be presented at the Intelligent System Conference (IntelliSys) scheduled between September 7th and 9th, 2023.

5.2 Background to The Study

A heuristic search-based approach is employed to estimate the number of clusters within a dataset, utilising an ensemble of clustering methods. In contemporary data analysis, a significant portion of available datasets is unlabeled. Examples include data from social media sources, such as billions of Facebook posts and text messages, or healthcare industry data generated from automated record transactions in everyday life [4]. The question is whether helpful information can be extracted from unlabeled data. For this purpose, many heuristic algorithms are available [160, 161, 162, 163]. Typical methods include k-means, hierarchical clustering and Partitioning Around Medoids (PAM) [56]. One primary requirement for many popular and effective methods is apriori knowledge regarding how many clusters the data should be arranged into. Getting this wrong or even slightly inaccurate may result in a completely different clustering arrangement than that being sought. More detail on the application of ensemble technique in estimating the number of clusters in a dataset is laid out in Chapter 4, and a survey of ensemble techniques can be found in the following references [10, 164]. This chapter extends earlier work in Chapter 4 on estimating the number of clusters in datasets, where previous finding confirmed the ability of ensemble technique to better estimate the number of clusters in datasets [16]. The motivation of the approach presented in this chapter is to improve the time taken for the current exhaustive search of the solution space by combining a heuristic search with the clustering ensemble. The clustering-based ensemble requires a consensus on generating and combining the ensemble to create a solution, which makes it different from the application of an ensemble in classification. The novelty and contribution of the approach are described below:

- A heuristic-based ensemble that consistently produces the same quality as the exhaustive search.
- A mathematical framework for the fitness function and a regression model that estimates the maximum convergence point for the heuristic search, vastly reducing the already improved runtime.
- Finally, deciding between a heuristic and an exhaustive search can pose a challenge, taking into account the trade-off between accuracy, consistency, and speed. The analysis presented here establishes the optimal point for using the exhaustive search and when to employ the heuristic approach.

In Chapter 2, various techniques for generating and combining ensembles were introduced and the framework for the ensemble was introduced in Chapter 4. This chapter extends the previous result, and is organised as follows: Section 5.3 presents the modified ensemble framework, highlighting the introduction of heuristic approach into clustering ensemble in Section 5.4. Section 5.5 describes the experimental setup and the dataset employed in the study. The results of the experiments are presented in Section 5.6. Finally, Section 5.7 concludes the chapter by summarising the findings and providing recommendations for future research.

5.3 The Ensemble Framework

This section presents the modified ensemble framework, which was previously introduced in Chapter 4, incorporating the addition of a heuristic search. The modified ensemble approach is depicted in Figure 5.1, and it encompasses four main stages:

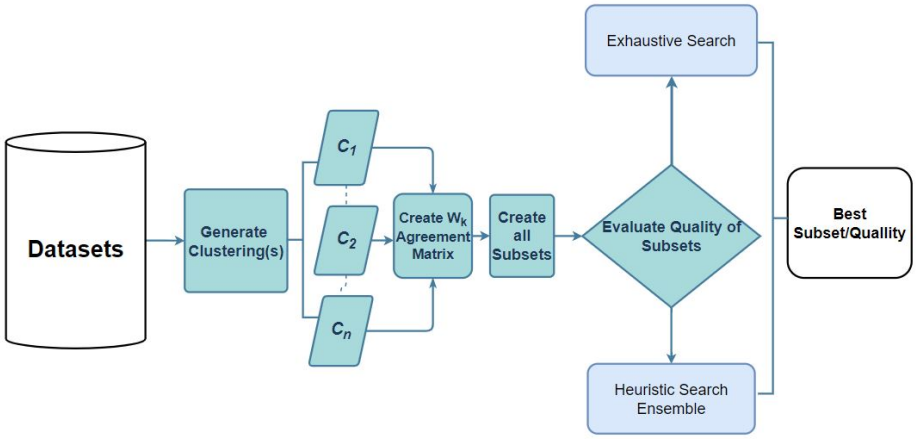


Figure 5.1: The Modified Ensemble Framework

(i) Generation of the base clusterings

(ii) Construction of the agreement matrix from clusterings

(iii) Creation of subsets from the agreement matrix

(iv) Selection of the best subset using the objective function

5.3.1 Generation of the Base Clustering

The initial phase of the ensemble framework is base clustering generation, which involves generating members used for building the ensembles. In addition, the members should be different to each other to achieve diversity. Diversity means the different representative subsets will be captured as subgroups in the ensembles; specifically, it must reflect most variants of the clusterings through a good combination of the ensembles as subsets. Earlier research efforts suggested several options in the clustering context to achieve diversity in generating subsets which includes the followings [96]:

- An iterative algorithm such as k-means can be employed to randomly select the initial number of clusters, resulting in multiple dataset representations for different values of k. These representations are then combined to form ensembles.
- Ensembles can be created by combining the outputs of different algorithms, such as k-means, DB-Scan, and a graph partitioning algorithm to form an ensemble.
- Different dataset representations can be generated by selecting specific features, such as varying pixels or image coordinates in an image dataset.

In this research, the number of algorithms has been increased to thirty to generate more ensembles and further test the efficiency of the algorithms as the number of inputs increases. Furthermore, new techniques were proposed for subsetting ensembles to maximise diversity and spread and sometimes to meet algorithmic requirements. These techniques were built upon the methods introduced in Chapter 4.

The framework presented in Figure 5.1 was modified to enhance the ensemble creation process. A heuristic search approach was incorporated in place of the previous Gray code/exhaustive implementation described in Chapter 4. This heuristic-based approach utilised a small change operator based on bit-flipping, which in theory is equivalent to the Gray code implementation because the objective is to minimise the difference between subsets; in essence, it is directly mapped to the Gray code implementation.

Furthermore, a hill-climbing algorithm based on a generate and test approach was introduced for subsequent subsets, deviating from the sequential testing of each subset in the search space used in the Gray code implementation. The small change operator, described in Algorithm 5.13, out-

lined the steps in modifying subsets using bit-flipping. Small change implementation further enhances the ensemble generation, resulting in a more diverse and representative search space.

Small Change Operator

The Smallchange function creates a new subset value by reversing the binary digit; if the value is 1, it changes to 0, and vice versa.

Algorithm 5.13 Smallchange between Subsest

Input: S_{old} - A binary string of length n

Let $S = S_{old}$
Let i be a random integer between 1 and n inclusive
if $s_i = 0$ **then**
    $s_i = 1$
else
    $s_i = 0$
end if

Output: S - a new subset value close to S_{old}

5.3.2 Construction of The W_k Agreement Matrix

A representation is required to determine the level of agreement of members in the subsets generated from the base clustering. The purpose is to measure accuracy and consistency based on the agreement strength between adjacent pairs of partitions and the number of times objects co-exist in the same cluster. The Weighted Kappa is an external validation index used in this research to measure the accuracy of clustering ensembles, and it ranges from -1 to 1. The Kappa metric serves as a measure of agreement between two raters, which, in this case, are the clustering algorithms used in the research. The previous approach of creating a database of clusterings for the dataset is utilised, employing thirty different clustering algorithms as input. The Weighted Kappa (w) metric is used to assess the agreement between the clustering algorithms. The Weighted Kappa assigns weights to

disagreements between the two raters. It provides an agreement strength ranging from poor to very good, as detailed in the Kappa guideline found in Section 2.5.2 of Chapter 2. The values for k and their corresponding classifications in the Kappa guideline can be found in this publication [13]. These guidelines reference interpreting the agreement strength observed in the clustering results.

5.3.3 Sub-setting the W_k Agreement Matrix

The subsetting of the agreement matrix follows a similar approach as the exhaustive experiment conducted in Chapter 4. A binary string of length n represents a subset, where a zero at position i indicates the exclusion of the ith clustering method, and a one represents its inclusion in the subset. This binary representation enables the generation of all possible subsets using a range of input clustering arrangements from 2 to $\sqrt{n}$. The maximum number of clusters suggested for subsetting is based on the square root of n, where n represents the number of objects being clustered [158]. Additional information regarding the subsetting process and the agreement matrix can be found in Chapter 4.

5.3.4 Experimental data

The experiments were conducted in this chapter using the same twenty-seven datasets as in Chapter 4. The reason was to ensure consistency and facilitate progressive comparison between different methods. As mentioned in Chapter 4, these datasets are renowned for their challenging clustering difficulties and diverse domains, making them widely accepted as standard benchmark datasets.

To construct the clustering ensemble database, a combination of thirty clustering algorithms (refer to Table 3.2) was employed along with various pa-

rameter settings. Some datasets failed to cluster effectively, while others had insufficient instances or contained significant missing values.

For consistency and to facilitate performance assessment, the twenty-seven datasets utilised in the experiments are identical to those used in Chapter 4. These datasets now include the number of reported clustering arrangements for each specific dataset, as indicated in Table 5.1. The determination of the maximum number of clusters (k_{max}) for each dataset is based on the square root of the total number of instances. For a comprehensive understanding of the dataset characteristics presented in Table 5.1. Refer to the detailed description provided in Chapter 3.

Table 5.1: Dataset, Attributes and Number of Clusters with k_{max}

#	Datasets	#Clusters	Attributes	#Instances	k_{max}
1	Aml28	5	2	804	28
2	Atom	2	3	800	28
3	BezdekIris	3	4	150	12
4	Blobs	3	2	300	17
5	Cassini	3	2	1000	31
6	Compound	6	2	399	19
7	Curves1	2	2	1000	31
8	Gaussian500	5	2	3000	54
9	Glass	6	9	214	14
10	Hepta	7	3	212	14
11	Longsquare	6	2	900	29
12	Lsun	3	2	400	19
13	Pearl	3	2	266	16
14	Pmf	5	3	649	25
15	Shapes	4	2	1000	31
16	Size1	4	2	1000	31
17	Size2	4	2	1000	31
18	Spherical_5_2	5	2	250	15
19	Square2	4	2	1000	31
20	Synthetic_control	6	60	600	24
21	Tetra	4	3	400	19
22	Tetragonular_bee	9	15	236	15
23	ThreeMC	3	2	400	19
24	Traingle1	4	2	1000	31
25	Vehicle	4	18	846	29
26	Veronica	7	8	206	14
27	Zelnik3	3	2	266	16
Total					**643**

5.3.5 Selection of The Best Subset

In the selection stage, the goal is to assess the quality of subsets using quality metrics to determine the best subset. The evaluation of the quality of subsets is essential but challenging, as there is no universally agreed standard for measuring what constitutes good-quality clusters. Typically, accuracy and consistency are considered the most important aspects when evaluating subsets or the clustering result. This study uses the Weighted Kappa built into the fitness function to identify the best from subsets. The average performance over repeated runs is used to measure consistency with different experimental set-up conditions represented as variance or standard deviation.

5.3.6 The Fitness Function

The evaluation of subset quality is based on its fitness value, which can be visualised as a point in a two-dimensional fitness landscape. In this landscape, the altitude represents the fitness level of each subset. The fitness function used in this chapter estimates the number of clusters by calculating the Weighted Kappa agreement sum of adjacent inputs. The details of this function are provided in Equation 5.1 and denoted as Q.

In contrast to the exhaustive approach discussed in Chapter 4, this chapter introduces a notable difference in the application of the fitness function. Previously, the update function was utilised to compute the fitness value, but now it is used with the small change operator to calculate subsequent fitness values. These values were then compared, and the best fitness will be the new current fitness until convergence. In this chapter, the small change operator plays a crucial role and differs from the pattern-based sequential Gray code implementation. It employs bit flipping to modify the next fitness value at each iteration. By acting as a generate and test operator,

the small change operator facilitates the exploration of alternative fitness values.

However, certain operations, such as the use of threshold values for subsets, remain integral to the quality metric. The reason is that the subsets from the ensemble, which serve as inputs to the fitness function, vary across a range of clustering algorithms. Consequently, the quality of some subsets, as expressed by their Weighted Kappa values, may be poor, necessitating the selection of a threshold value to normalise the metric.

Research conducted by Odebode et al. [16] has demonstrated that the average Weighted Kappa of the agreement matrix, denoted as θ, performs best as a predictor of the number of clusters in a dataset (See Chapter 4). This finding is consistent with similar research by Ayed et al. [12], which supports the need to incorporate thresholds. The core premise of these studies is that a correlation exists between the average Weighted Kappa (w) of input methods and the gold standard (i.e., the number of clusters published for each dataset).

$$Q = \sum_{a=1}^{|s|-1} \sum_{b=a+1}^{|s|} [w(s(a), s(b)) - \theta] \tag{5.1}$$

$$\hat{Q} = \begin{cases} Q + 2 \sum_{i=1}^{|\hat{s}|-1} w(s_i, x) - \theta & \text{positive,} \\ Q - 2 \sum_{i=1}^{|\hat{s}|-1} w(s_i, x) - \theta & \text{negative.} \end{cases} \tag{5.2}$$

In Chapter 4, the search for the optimal subset was significantly reduced by employing a Gray code implementation, which linked subsets to their preceding subsets. This approach allowed for the progressive evaluation of fitness values. However, despite this optimisation, directly applying the

Gray code implementation to the search for the best subset remains exhaustive. In other words, every subset in the search space is explored. As the size of the input increases, this exhaustive approach may become infeasible.

This issue is addressed by introducing a heuristic approach in the algorithm, specifically in the RMHC implementation described in Section 5.4. This heuristic approach mitigates the computational burden by utilising randomised methods and local search techniques to guide the search process rather than evaluating the values of each subset to determine the optimal subsets. The algorithm can efficiently navigate the search space by employing heuristics and identifying promising subsets without exhaustively evaluating every possible combination. This approach allows for more practical and feasible implementation, particularly as the size of the input increases.

5.4 Methods

In this section, two methods will be presented: The exhaustive and the modified RMHC.

5.4.1 The Exhaustive Approach

The exhaustive approach, previously discussed in detail in Chapter 4, will be summarised here as it will be compared with the current approach. The exhaustive approach searches for the best subset to represent the number of clusters in the dataset. It takes advantage of the unique characteristics of Gray code, such as single-digit differences, to create relationships between subsets in evaluating the quality of each subset. This implementation operates on the assumption that there exists a subset value that maximises the quality in relation to the threshold value of the Weighted Kappa. This value

corresponds to the estimated number of clusters in the dataset.

5.4.2 Random Mutation Hill Climbing (RMHC)

A heuristic is a general principle or a set of loose guidelines that can aid in finding a solution to a problem. While the solution obtained through heuristics is not guaranteed to be optimal, they are frequently utilised in Artificial Intelligence to enhance the performance of search methods. A comprehensive overview of heuristic search algorithms and related topics can be found in Section 2.7 of Chapter 2.

One such heuristic search algorithm is the hill climbing algorithm. This algorithm begins at a randomly selected point within the search space and systematically evaluates each potential solution without exploring all possible alternatives. It selects the best successor node (subset) based on the fitness function and commits the search to that particular node. The process continues iteratively until no further improvement can be achieved, reaching a convergence point.

As the name suggests, the Random Mutation Hill Climbing (RMHC) algorithm combines random mutation with the hill climbing approach. It aims to locate points in the search space that lead to an upward slope, maximising the search space. However, RMHC can sometimes converge to a local optimum rather than a global one. Despite this limitation, it has proven to be effective in numerous applications. Furthermore, RMHC is directly applicable to the current fitness function definition. Considering the significant role of the fitness function in a clustering ensemble, enhancing its performance contributes to improving the effectiveness and efficiency of the algorithm.

The RMHC version of the algorithm aims to enhance the Gray code implementation by incorporating progressive evaluation of subsets. This aspect

is crucial in hill climbing algorithms and directly aligns with the principles of the Gray code version, only that the search is narrowed towards the optimum fitness value. However, it is not guaranteed. By incorporating the ability of RMHC to evaluate and compare fitness values with its neighbouring subsets, the search for the best subset can be further improved.

In Algorithm 5.14, specifically in lines 4 and 6, this enhancement is explained. This modification is anticipated to accelerate the search for the optimal subset and significantly improve the search process while maintaining accuracy, particularly as the dataset's volume and dimension increase. In theory, employing more powerful population-based algorithms, such as Genetic Algorithm or Simulated Annealing [165] as applied in Chapter 6, could potentially enhance the speed of the search process. However, the crossover operator utilised in Genetic Algorithm introduces significant changes that would prevent the utilisation of the update fitness value definition for the GA implementation. This restriction can considerably impact the speedup gained since each subset is evaluated independently.

Hence, while alternative algorithms may offer potential speed improvements, the independent evaluation of subsets achieved through RMHC remains advantageous for maintaining accuracy and facilitating the search process. The random mutation hill-climbing (RMHC) algorithm requires a random starting point, which is generated using a random binary string of size r as described in Algorithm 5.14. The quality of this starting subset is calculated using the fitness function, and subsequent subsets' qualities are evaluated based on the fitness function rather than the updated fitness.

In the modified version, similar to the Gray code representation, the random binary string undergoes a random bit flip to generate a new subset $\hat{s}$ close to the previous subset. The original fitness value is only utilised once for the

Algorithm 5.14 Random Mutation Hill Climbing (RMHC)

Input: $iter, w$ Matrix size $r \times r$ ▷ w = Weighted Kappa
1: Let s be a random binary string of size r
2: $F \leftarrow Q(s, w)$ ▷ F = Fitness
3: **for** $i \leftarrow 1$ to $iter$ **do** ▷ iter =Number of iterations
4: $\hat{s} \leftarrow smallChange(s)$ ▷ $\hat{s}$ = a new solution close to s
5: $\hat{F} \leftarrow Q(s, F, \hat{s}, W)$ ▷ $\hat{F}$ = new fitness(Smallchange
6: **if** $\hat{F} > F$ **then**
7: $s \leftarrow \hat{s}$
8: $F \leftarrow \hat{F}$
9: **end if**
10: **end for**
Output: The best solution(s)

initial starting point in the search. The fitness function and the small change operator determine changes made to subsequent subsets.

5.5 Experimental Procedure

The twenty-seven datasets previously described in Section 4 were used in testing the new algorithm's effectiveness against the exhaustive algorithms. Table 5.1 provides information on the dataset attributes, including the reported number of clusters and the maximum value of k, denoted as k_{max}.

The experiments were divided into three categories, as outlined in Table 5.2 (Table of Experiments).

In the first experiment, an exhaustive search was conducted on subsets of the twenty-seven datasets to find the optimal subset using thirty algorithms as inputs. This experiment was similar to the one conducted earlier in Chapter 4, but with increased inputs. The objective was to determine the optimal value of k for each dataset, ranging from 2 to $\sqrt{n}$, where n represents the number of instances in the dataset. The threshold for the search was set as

the average Weighted Kappa of the agreement matrix, and the maximum values reported for k in Table 5.1 were used as k_{max}. A total of 643 datasets were obtained from the subset. The Update quality version of the algorithm was used to expedite the process. This version, described in Chapter 4, is a Gray code implementation that avoids recomputing the quality values of the subset at every iteration, resulting in significantly reduced computation time. Only the result of the average quality was reported, as it provided the best threshold value.

However, due to the increasing input size, the exhaustive approach took an excessively long time to run, as anticipated. As a result, an exploration of the effectiveness of applying a heuristic search using the Random Mutation Hill Climbing (RMHC) algorithm was conducted in the second experiment. The RMHC algorithm was employed with a fixed number of iterations (ten thousand iterations). Additionally, a model-based version of the algorithm was designed to determine the average point of convergence at which the algorithm performed comparably to RMHC. The search process was initiated from multiple regions within the search space, and this procedure was repeated one hundred times. This approach mirrors a previous experimental setup described in this paper [136]. In Table 5.2, this method was referred to as RM10K, while the model-based version was denoted as RMModel (explained below) .

In the third experiment, the RMHC algorithm was applied to different input sizes ranging from 5 to 30 for all datasets. The convergence points for the different input sizes were used to generate a model. Linear regression models were utilised to validate the models using the maximum convergence points. This information was then used to run a reduced version of the RMHC experiments, testing its efficacy further. The results of the RMHC algorithm with 10,000 iterations over one hundred repeats, as well as the

performance of the model for the same number of repeats, were compared to the results of the exhaustive search in Section 5.6.

Table 5.2: Table of Experiments with the Methods, Datasets, Number of Iterations, and Repeats.

Methods	Datasets	# Iterations	# Repeats
Exhaustive	27	1	1
RM10K	27	10,000	100
RMModel	27	Section 5.6	100
Total			**129,243**

5.6 Results and Discussions

The results are presented in the following summaries:

- Exhaustive by dataset and r (size) presented as average quality
- RM10K (hill climbing over ten thousand iteration) vs exhaustive
- RM10K Vs Model Improvement (RMModel)

The exhaustive performance versus the benchmark methods listed in Section 4.6 Chapter four indicate that compared to the top eight methods, the ensemble predicted seventeen of the twenty-seven dataset correctly compared to twelve predicted by the Calinski Index (CH), the best among the methods. Similarly, the best error estimate across all datasets for all methods was 0.281 for Ball. Although relatively close to the ensemble at 0.271, the number of clusters predicted correctly by Ball was just eight (8) compared to the ensemble correctly predicted seventeen (17).

Tables 5.3 and 5.4 provide a comparison of the accuracy between the RMHC-based methods, namely RM10K and RMModel, and the exhaustive search. The performance of these algorithms was evaluated both on a dataset basis and as the input size increased. The error values, measured in terms of the fitness function Q, reflect the consistency of the clustering

Table 5.3: A summary of the Average and Standard Deviation by Dataset

	Exh		RM10K		RMModel		RMModel vs RM10K		RM10K vs Exh	
Dataset	Av	SD[1]	Av	SD	Av	SD	Av	SD	Av	SD
Aml28	18.30	9.59	18.13	9.65	18.15	9.65	0.08%	0.05%	0.90%	0.70%
atom	21.34	11.70	20.44	11.70	20.41	11.71	0.17%	0.09%	4.20%	0.01%
bezdekIris	11.27	10.71	11.17	10.70	11.18	10.71	0.07%	0.07%	0.89%	0.12%
blobs	18.20	9.36	17.97	9.48	17.95	9.47	0.07%	0.10%	1.28%	1.28%
cassini	23.66	10.50	23.55	10.57	23.55	10.57	0.02%	0.03%	0.47%	0.69%
compound	25.93	12.01	25.78	12.11	25.77	12.10	0.04%	0.03%	0.57%	0.85%
curves1	27.22	11.42	27.07	11.52	27.05	11.52	0.05%	0.04%	0.55%	0.84%
gaussian500	19.11	9.80	19.00	9.86	19.01	9.85	0.04%	0.09%	0.56%	0.64%
glass	21.59	11.66	21.58	11.64	21.57	11.64	0.03%	0.02%	0.07%	0.13%
hepta	19.55	11.48	19.24	11.59	19.25	11.58	0.05%	0.05%	1.59%	0.98%
longsquare	25.34	11.58	25.19	11.68	25.21	11.67	0.09%	0.07%	0.61%	0.84%
lsun	24.16	12.06	23.97	12.18	23.97	12.18	0.02%	0.02%	0.80%	1.01%
pearl	29.45	13.24	29.24	13.36	29.24	13.35	0.01%	0.06%	0.72%	0.90%
pmf	22.21	11.76	22.09	11.70	22.08	11.69	0.05%	0.06%	0.52%	0.50%
shapes	26.16	13.23	26.01	13.32	26.01	13.32	0.01%	0.01%	0.56%	0.66%
size1	18.33	9.13	18.13	9.23	18.14	9.24	0.07%	0.02%	1.08%	1.13%
size2	16.38	8.70	16.17	8.78	16.17	8.78	0.02%	0.02%	1.29%	1.00%
spherical_5_2	24.87	11.11	24.64	11.26	24.65	11.25	0.03%	0.10%	0.93%	1.34%
square2	18.85	9.13	18.69	9.23	18.69	9.23	0.05%	0.01%	0.87%	1.01%
synthetic_control	28.78	12.08	28.57	12.24	28.58	12.22	0.05%	0.15%	0.75%	1.25%
tetra	18.34	10.04	17.99	9.91	17.99	9.92	0.03%	0.05%	1.95%	1.24%
tetragonular_bee	31.18	20.13	30.91	20.22	30.94	20.20	0.09%	0.07%	0.87%	0.43%
threeMC	25.30	11.81	25.14	11.92	25.14	11.91	0.03%	0.05%	0.65%	0.90%
Triangle1	24.90	14.29	24.75	14.36	24.74	14.37	0.04%	0.04%	0.60%	0.49%
vehicle	24.43	14.39	24.40	14.39	24.40	14.40	0.01%	0.02%	0.13%	0.01%
veronica	30.39	20.33	30.22	20.39	30.22	20.39	0.03%	0.03%	0.56%	0.27%
zelnik3	29.49	13.22	29.30	13.35	29.29	13.35	0.01%	0.03%	0.65%	0.92%
Percentage Error							0.05%	0.05%	0.91%	0.75%

output obtained from the exhaustive search compared to the RMHC algorithms.

Variability in the standard deviation of clustering algorithms can stem from several factors, such as those observed in the RMHC algorithm and ensemble clustering. In this experiment, the RMHC algorithm is evaluated using RM10K and RMModel. RM10K involves running the RMHC algorithm for 10,000 iterations across 100 repeats. In contrast, RMModel leverages a reduced run based on the maximum runtime determined by a linear regression model over the same number of repeats. These approaches yield summary results that include the average and standard deviation of error values across datasets and input sizes, providing insights into the consistency of RMHC's performance compared to an exhaustive search, as shown in Table 5.3. In ensemble clustering, non-zero standard deviation across multiple

[1]A non-zero SD value for the exhaustive search results might seem unusual, but it is noted that this table is a summary of different experiments $(n = 2..30)$ which have different exhaustive fitness results. The SD values are included for completeness/reference.

runs arises from the inherent challenge of conducting an exhaustive search for the optimal number of clusters in a dataset. This search involves running clustering algorithms multiple times with varying parameters, particularly the number of clusters [96]. This diversity can enhance the robustness and accuracy of the final consensus clustering solution [166]. However, different base clusterings and combination strategies can also introduce variability in the final clustering results, reflecting the process of exploring various clustering parameters and consensus methods. This variability can be beneficial for identifying the optimal number of clusters but may also point to potential issues in the base clusterings or the consensus approach that require further investigation or adjustment [7].

Figure 5.2 presents the accuracy of the linear regression model in terms of the average and maximum convergence points. This figure illustrates the performance of the linear regression model in predicting the maximum convergence points, which serve as the foundation for the RMModel. A consistent pattern in the linear regression of the average and maximum runtimes was observed as the input size increased. The plot of the linear regression for the average and maximum runtimes against the input size resulted in a linear predictor function that was subsequently used to determine the new maximum in the model run. This analysis aimed to assess the performance of the RMHC algorithm compared to the model. The linear equation derived from the model is given by:

$$\text{iter} = 22 \times r - 52 \tag{5.3}$$

Where “iter” represents the number of iterations, and “r” is the input size. From Figure 5.2, it can be observed that the maximum runtime provides a more accurate representation of the model compared to the average runtime. The error between the RMModel and the actual RMHC runtime, as

shown in Table 5.4, was negligible (0.49%).

Table 5.4: A Summary of the Average and Standard Deviation by Input Size

	Exh		RM10K		RMModel		RMModel vs RM10K		RM10K vs Exh	
Size	Av	SD	Av	SD	Av	SD	Av	SD	Av	SD
5	4.16	1.25	4.16	1.25	4.16	1.25	0.01%	0.07%	0.07%	0.38%
6	4.89	1.69	4.89	1.71	4.89	1.71	0.01%	0.09%	0.16%	0.80%
7	6.88	2.32	6.87	2.34	6.87	2.34	0.01%	0.03%	0.15%	0.78%
8	9.33	3.20	9.32	3.22	9.32	3.22	0.01%	0.04%	0.12%	0.55%
9	12.15	4.01	12.14	4.03	12.14	4.03	0.00%	0.01%	0.10%	0.46%
10	14.99	4.97	14.98	4.98	14.98	4.99	0.02%	0.07%	0.08%	0.32%
11	18.56	6.34	18.54	6.36	18.54	6.36	0.00%	0.01%	0.08%	0.32%
12	18.83	6.13	18.81	6.15	18.81	6.16	0.02%	0.07%	0.09%	0.42%
13	18.98	6.07	18.94	6.13	18.94	6.13	0.00%	0.05%	0.21%	0.97%
14	19.30	5.97	19.23	6.08	19.24	6.07	0.02%	0.11%	0.35%	1.71%
15	19.71	5.97	19.60	6.13	19.60	6.13	0.01%	0.05%	0.55%	2.64%
16	20.16	6.21	20.01	6.43	20.01	6.43	0.00%	0.05%	0.78%	3.52%
17	20.62	6.58	20.37	6.91	20.39	6.88	0.10%	0.40%	1.23%	4.99%
18	21.29	7.22	20.90	7.64	20.90	7.64	0.01%	0.05%	1.84%	5.83%
19	24.79	7.49	24.46	7.87	24.48	7.85	0.07%	0.30%	1.32%	5.13%
20	25.04	7.40	24.72	7.80	24.72	7.77	0.03%	0.36%	1.28%	5.35%
21	25.07	7.40	24.71	7.82	24.73	7.81	0.08%	0.19%	1.42%	5.74%
22	25.10	7.41	24.72	7.88	24.73	7.86	0.05%	0.16%	1.53%	6.27%
23	29.42	8.23	29.14	8.59	29.12	8.63	0.09%	0.36%	0.94%	4.45%
24	30.08	8.75	29.71	9.21	29.71	9.21	0.02%	0.06%	1.25%	5.34%
25	34.77	9.77	34.46	10.16	34.46	10.17	0.01%	0.10%	0.88%	3.98%
26	35.65	10.01	35.34	10.41	35.33	10.44	0.03%	0.21%	0.87%	4.01%
27	35.66	10.00	35.30	10.49	35.30	10.49	0.01%	0.01%	1.01%	4.85%
28	35.75	9.97	35.35	10.50	35.35	10.50	0.01%	0.03%	1.10%	5.35%
29	38.42	10.70	38.07	11.16	38.08	11.14	0.03%	0.15%	0.90%	4.28%
30	43.27	11.75	43.02	12.10	43.00	12.12	0.03%	0.13%	0.59%	2.98%
Error(%)							**0.33**	**(0.33)**	**0.68**	**(0.49)**

An interesting observation was made regarding the linear model in Figure 5.2, which led to the derivation of a general equation. This equation was then used to calculate results similar to those obtained using the RMHC method. The results are presented in Tables 5.3 and 5.4, organized by dataset and input size, respectively. The table displays the model's results as RMModel, while the initial RM10K approach is used for comparison.

The results based on the input size, which corresponds to the input algorithms, showed that the RMModel was approximately 30 times faster than RM10K while maintaining the same level of quality. The errors observed in the dataset comparison indicated that the exhaustive approach had less than one per cent error overall when compared to RM10K. On the other hand, the RMModel exhibited an average error of less than 0.05 percent, producing similar results 99.95 percent of the time.

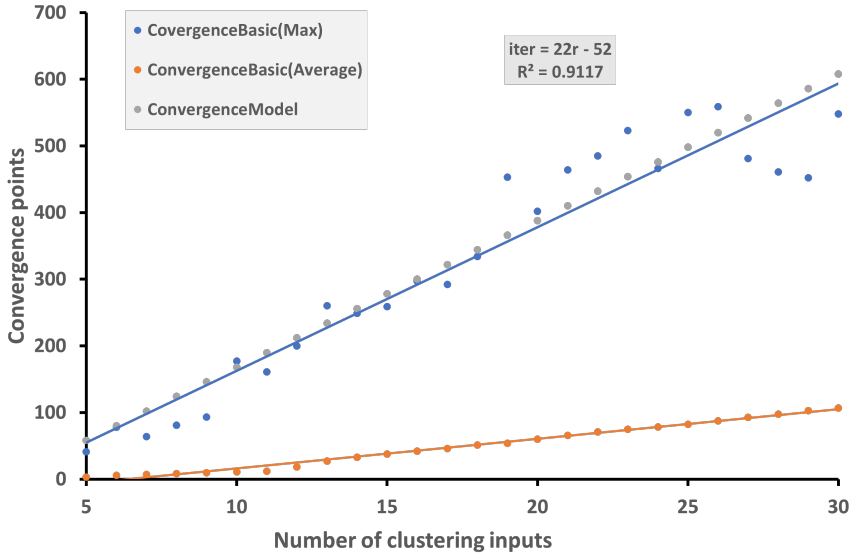


Figure 5.2: A plot Showing the Linear Regression Model by Average and Maximum Convergence Points.

Lastly, Figure 5.3 illustrates the convergence point's growth as the input size extends beyond seven. At this stage, combining a RMHC or a similar heuristic approach may be advantageous to reduce the runtime significantly.

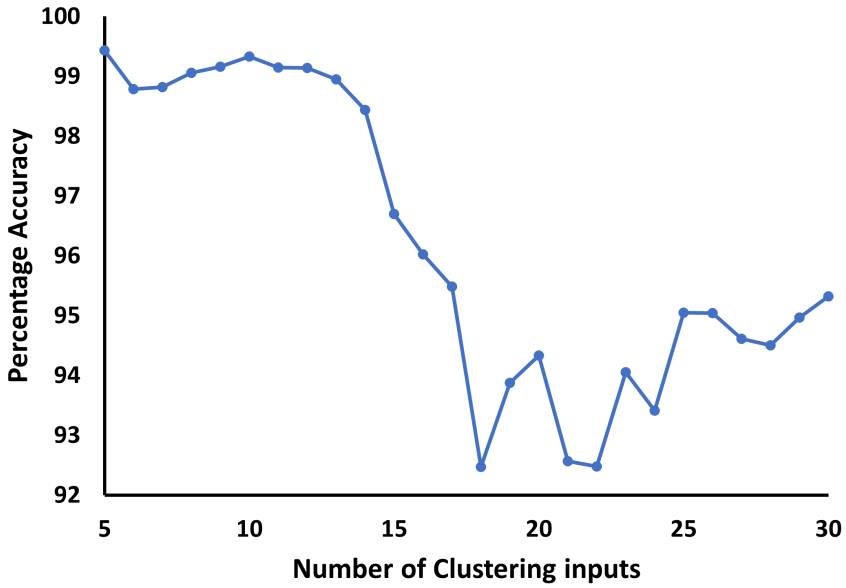


Figure 5.4: The Accuracy of RMHC Compared with Exhaustive

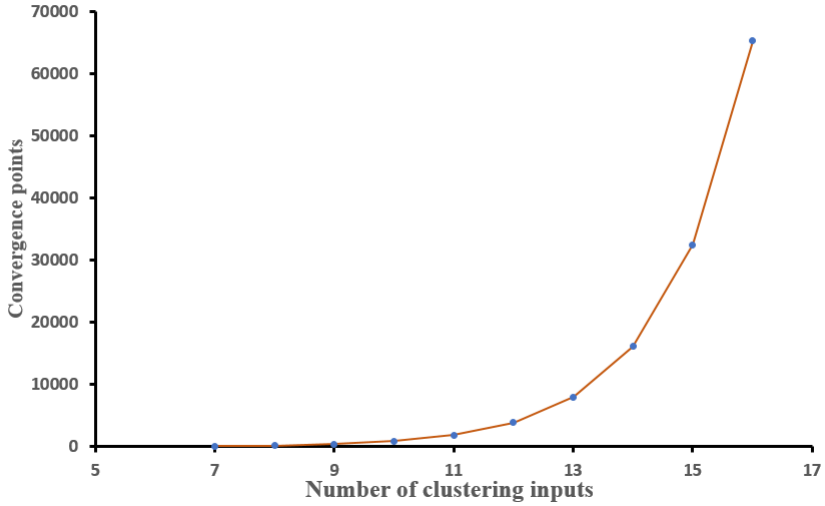


Figure 5.3: RMModel vs Exhaustive Convergence point

From the results there is no correlation between the values of k, the number of clusters and the error, whereas there is a positive correlation between the input size and the error; this may be a result of the quality values (Q)

not being normalised. Figure 5.3 graphically shows the cutoff (number of clustering ensembles used) between using the exhaustive search and the RMModel-based approach and the trend as the input clustering algorithm increases. The cut-off point is around input of size seven (7) from which point the run time increases exponentially. Finally, the accuracy of RM10K ensemble compared to the exhaustive shown in Figure 5.4 shows an average minimum of ninety-two per cent (92%) accuracy with most estimates equal to the exhaustive ninety-five per cent (95%) and above of the time.

5.7 Recommendations

This chapter presents a novel approach for estimating the number of clusters in datasets by combining a clustering ensemble with a heuristic search using the RMHC method. The key contribution of this approach is the introduction of a heuristic search ensemble, which allows for a progressive evaluation of subset quality while minimising the convergence time of the algorithm. The introduced RMHC approach demonstrates comparable results to the exhaustive search but with significantly faster execution. It achieves 95% accuracy on average and is 30 times faster than the convergence point of the RMModel when applied to the same datasets. The results highlight the potential advantage of incorporating heuristic search into clustering ensemble methods to reduce runtime, particularly in scenarios involving large-scale datasets. It opens up possibilities for addressing big data clustering problems more efficiently.

Chapter 6 will explore and investigate the starting point in the search process using the Fiedler vector decomposition, as the starting point can sometimes lead to a local optimal in a hill climbing implementation. Therefore, more advanced local search techniques, such as Simulated Annealing-based approaches, will be investigated to improve efficiency further. Al-

though this work serves as proof of concept, it provides a promising direction for future research in the field.

Chapter 6

Seeding Using The Fiedler Vector Decomposition for Clustering Ensemble Search

6.1 Introduction

In this chapter, we introduce additional concepts that complement the framework established in Chapter 5. Our goal is to enhance the Random Mutation Hill Climbing (RMHC) technique presented in Chapter 5 by incorporating various techniques such as Fiedler vector decomposition, Laplacian of a graph, and Minimum Spanning Tree (MST). These techniques are utilised to develop a guided search approach as an initial starting point within the search space, acting as a seed for the exploration process.

The main objective of this guided search approach is to improve the efficiency of searching for the optimal subset from the clustering ensemble

and enhance the accuracy in determining the number of clusters within the dataset. This approach provides a systematic exploration strategy focusing on regions likely to contain high-quality solutions. Experimental evaluations will be conducted to assess how well the approach enhances the efficiency and accuracy of subset selection from the clustering ensemble.

The following section will introduce technical terms and definitions of concepts used in constructing the graph decomposition, which forms the basis for the guided search approach.

Definition 6.1.1 (Graph)**.** A graph G is a pair (V, E) where V is a nonempty set of vertices and E is a set of edges. Each edge in E is an unordered pair of distinct vertices from V. The edges represent connections between the vertices, and there is no restriction on the nature of these connections [167].

Definition 6.1.2 (Laplacian of a Graph)**.** The Laplacian matrix of a graph is a matrix that encodes the local connectivity structure of a graph. It is a symmetric matrix defined as the difference between the graph's degree matrix and adjacency matrix.

The Laplacian matrix of a graph is often denoted by L and can be defined in multiple ways. One common definition is the unnormalised Laplacian:

$$L = Z - A$$

where Z is the degree matrix and A is the adjacency matrix.

Another common form is the normalised Laplacian, defined as:

$$L_{\text{norm}} = I - Z^{-1/2} A Z^{-1/2}$$

where I is the identity matrix and $Z^{-1/2}$ is the square root of the inverse

of the degree matrix. This research focuses solely on the unnormalised Laplacian.

Definition 6.1.3 (Adjacency Matrix)**.** If G is a graph on n vertices, its adjacency matrix $A(G)$ is then an $n \times n$ symmetric matrix defined by $a_{ij} = 1$ if $\{i, j\}$ is an edge of G and 0 otherwise.

Definition 6.1.4 (Degree Matrix)**.** Given a graph $G = (V, E)$ and $|V| = n$, the Degree matrix Z is a $n \times n$ diagonal matrix defined as [168]:

$$Z_{ij} = \begin{cases} deg(v_i) & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$$

Where the degree $deg(v_i)$ is the number of times an edge terminates at the vertex.

6.2 Minimum Spanning Tree

A minimum spanning tree (MST) is a subgraph of a connected, weighted graph that connects all vertices with the minimum possible total edge weight. In other words, it is a tree that spans all vertices of the graph and has the minimum possible sum of edge weights. Minimum spanning trees are commonly used in network design, where the goal is to construct a network with the minimum cost [108]. MSTs can provide an efficient way to traverse the graphs created. A minimum spanning tree of a graph is a tree that connects all nodes in the graph and has the minimum possible total edge weight. MST makes it possible to break down a complex graph into simpler subgraphs that can be more easily analysed. The most important edges in a graph can also be identified using MST as shown in Figure 6.1.

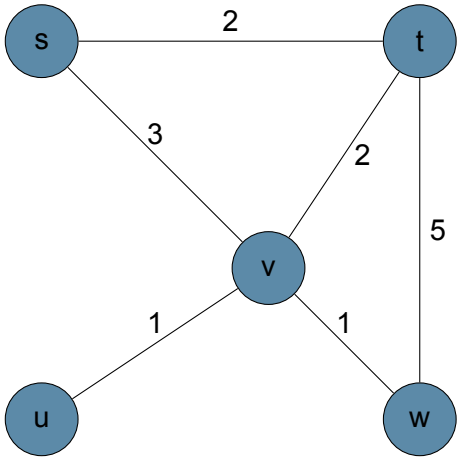


Figure 6.1: A Simple Graph with Five Nodes

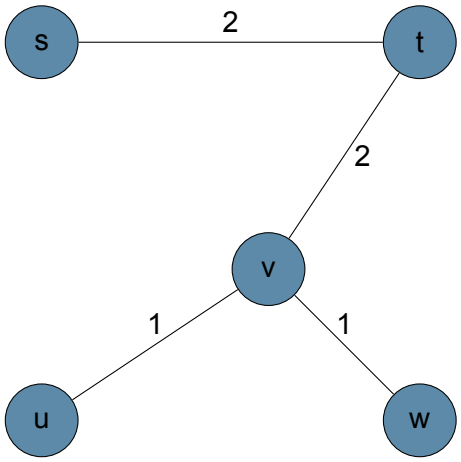


Figure 6.2: The Minimum Spanning Tree of The Graph

6.3 Fiedler Vector Decomposition

The Fiedler Vector Decomposition (FVD) is a technique used in graph theory to analyse the structural properties of a graph by breaking it down into a sum of subgraphs. Miroslav Fiedler introduced it in his work on algebraic connectivity [169, 170]. Each dataset can be viewed as a graph in this method, where each point represents a node. The graph is then de-

Algorithm 6.15 Prims Minimum Spanning Tree (MST)

Input: G- a weighted graph (U- vertices, E- Edges)

Let x be a random node from U
Let $Unew = \{x\}$
Let $Enew = \{\}$
while $Unew \neq U$ **do**
    Choose an edge (z, g) with minimal weight,
    such that $z \in Unew$ and $g \notin Unew$
    $Unew$ = $Unew \cup \{g\}$
    $Enew$ = $Enew \cup \{(z, g)\}$
end while

Output: $Gnew = (Unew, Enew)$

composed into subgraphs based on the spectral properties of its adjacency matrix.

The adjacency matrix of a graph is a square matrix that shows the connections between its vertices [171]. FVD uses the eigenvalues and eigenvectors of the adjacency matrix to partition the graph into two or more subgraphs. Specifically, it involves finding the eigenvectors corresponding to the second smallest eigenvalue of the adjacency matrix, which is then used to identify the subgraphs as shown in Figure 6.3.

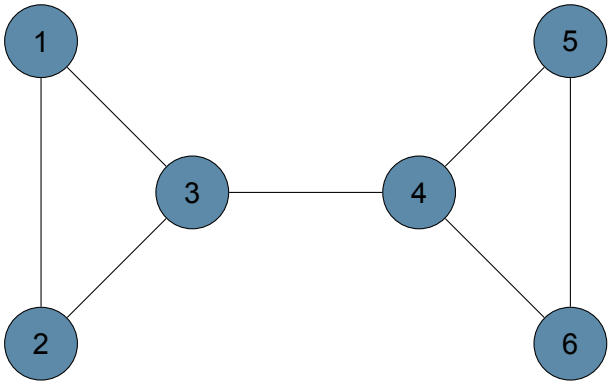


Figure 6.3: Fiedler vector decomposition

The FVD method finds practical applications in various domains, such as network analysis, graph partitioning, and community detection [172, 173].

FVD is particularly useful for analysing the structure of complex networks or clusters by identifying subgraphs that exhibit similar connectivity patterns. These subgraphs can be regarded as distinct groups of nodes within the graph. To better understand FVD, defining key terms such as a graph, subgraph, Laplacian matrix, eigenvalue, and eigenvector is essential. For a more comprehensive example on FVD, please refer to Appendix 7.4.

6.4 Motivation for Seeding

The main objective behind seeding the RMHC algorithm is to provide it with an initial position closer to the optimal solution. When dealing with optimisation problems, the search space can often be vast, making it challenging for the algorithm to locate an appropriate solution without guidance. By introducing a seed, the algorithm can initiate its search from a region in the search space that is more likely to contain a good solution. There are few publications in the literature that have introduced seeding in the clustering process [174, 175].

Apart from providing a better starting point, another reason for seeding the RMHC algorithm is to enhance its convergence speed. A well-chosen seed can help the algorithm converge quickly to a good solution, avoiding unnecessary computation and significantly reducing the overall running time when the matrix size or the number of inputs in the ensemble increases. Seeding a Hill Climbing algorithm can also help improve the quality of the solutions it generates. As demonstrated in the result section, starting the algorithm from a good seed makes it more likely to find high-quality solutions closer to the optimal solution.

The process involves converting each dataset's Weighted Kappa matrix into a graph and finding the graph's MST. The Weighted Kappa matrix consists

of values ranging from -1 to 1. Each Weighted Kappa value entry in the matrix is subtracted from 2 to convert its values to a weighted adjacency matrix or an edge list representing a graph. Positive values are converted to 1 or slightly higher, while negative values become 2 or more. Each cell in the resulting matrix represents the weight of the edge between the corresponding vertices in a graph.

The process begins by inputting the graph into a Minimum Spanning Tree (MST) algorithm, as outlined in Algorithm 6.15. A minimum spanning tree is a tree that connects all vertices of a graph with the least possible total edge weight. The outcome of this algorithm is an adjacency matrix representing a subgraph of the original graph. This subgraph includes all vertices and ensures connectivity. Next, the degree matrix is derived from this adjacency matrix. The Laplacian matrix is then determined using Equation 6.1. Figure 6.1 presents a graph consisting of five nodes, while its corresponding Minimum Spanning Tree (MST) is illustrated in Figure 6.2. In Section 6.2, some key features of a Minimum Spanning Tree are briefly discussed.

The laplacian matrix from MST is then decomposed using the Fiedler vector decomposition to obtain two subgraphs, as shown in Figure 6.3. The Fiedler vector corresponds to the second smallest eigenvalue of the Laplacian matrix or the adjacency matrix and captures essential information about the graph's connectivity.

The Fiedler vector, a concept derived from the Laplacian matrix of a graph, is explained in detail in Section 6.3. The Laplacian matrix represents the connectivity structure of the graph. The Fiedler vector can be obtained by computing the eigenvectors and eigenvalues of the Laplacian matrix. This vector plays a crucial role in partitioning the graph into two subgraphs based on the sign of its values, as discussed in Appendix C (see 7.4). Once the graph is partitioned, the quality of each resulting subgraph is evaluated us-

ing the fitness function described in Chapters 4 and 5. Further details on the experiments can be found in Section 6.9.

6.5 Arnoldi Iteration

Within this thesis, the Arnoldi iteration [176], an iterative technique, was applied to find approximate values of a specified number of eigenvalues and eigenvectors for a large sparse matrix. Arnoldi iteration begins with an initial vector and iteratively constructs an orthonormal basis for the Krylov subspace [177]. This basis is then utilised to build an approximation to the matrix using a smaller, easily diagonalisable Hessenberg matrix [178]. The primary reason for applying the Arnoldi iteration is its capability to target a specific number of eigenvalues and their corresponding eigenvectors, allowing a focused computation rather than obtaining the entire spectrum of the matrix ($n = 2..30$ in this chapter). This attribute proves valuable in computing the second smallest eigenvalue for determining the seed in the calculation of FVD.

6.6 Average Quality of the Search Space

This study's evaluation of subset quality is based on a fitness function, as discussed in Section 4.5.1. This function assesses the accuracy of each subset in estimating the number of clusters in a dataset. It calculates the sum of agreements derived from the Weighted Kappa of adjacent inputs, considering a threshold value (θ). For a more detailed understanding of the quality and update quality functions, refer to Section 4.5.1.

A previous study outlined in Chapter 4 discovered that the average fitness yielded the best results compared to other scales within the Weighted Kappa guideline. This discovery prompted an investigation into the average

fitness across various ranges of matrix sizes, corresponding to the number of inputs to the ensemble. Surprisingly, it was observed that the average fitness approximates zero within the search space. This intriguing result prompted further examination, utilising empirical and theoretical proofs to establish the average value.

6.6.1 Empirical Proof of the Average Quality

Rigorous experiments were conducted on simulated datasets from pseudo-random numbers ranging from 0.1 to a maximum value of 0.9. The selection of 0.1 as the lower bound was motivated by the possibility of datasets with no clusters, where a value of 0 would indicate no clusters at all. Therefore, a slightly higher value was chosen as the cut-off point for the initial selection of algorithms used to generate the numbers. On the other hand, a value of 0.9 represents the highest level of agreement between the clusterings, indicating the presence of well-defined clusters and their corresponding membership. Symmetric matrices of datasets of various sizes, ranging from 5 to 30, were simulated, and the average quality of each subset was calculated. Interestingly, for each matrix size, the average quality was zero. For instance, Figure 6.4 shows two matrices of size ten and eleven, corresponding to ten and eleven input algorithms, respectively, represented as ensembles with one thousand and thirteen (1013) and two thousand and thirty-seven (2037) subsets. The graph is a scatterplot of quality values (the pseudo-random numbers), illustrating the values for positive and negative axes, and the average quality converges to zero. This finding carries significant implications. Values greater than zero in the search space can be considered higher than the average, making them favourable starting points or a good seed for exploration. Conversely, values lower than zero are assumed to be random. Therefore, obtaining a value higher than the average represents a promising starting point within the search space.

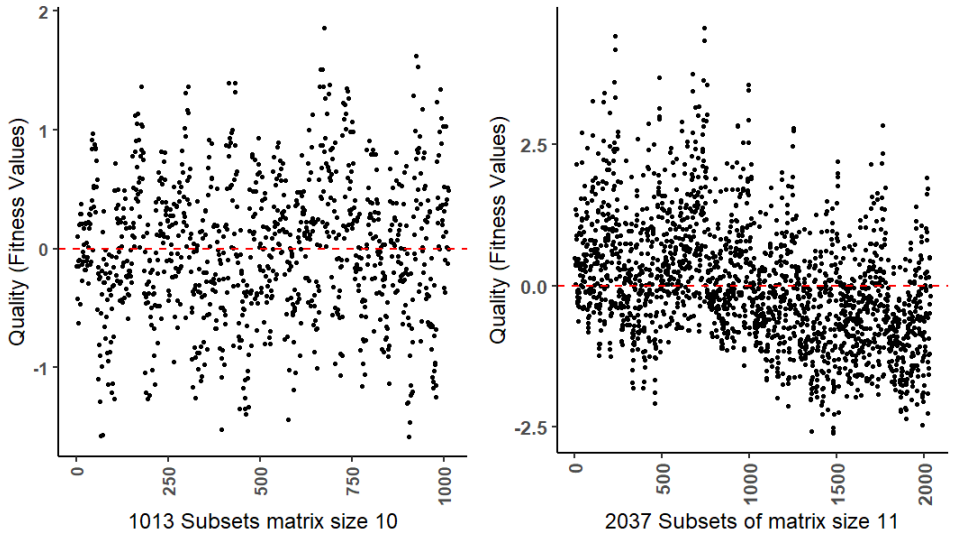


Figure 6.4: The Variability as Average Search Space Converges to Zero (Simulated Dataset for Matrix Size 10 and 11)

6.6.2 Mathematical proof of Average Quality

This section presents the proof for the average fitness of the search space, applicable to matrices of any size corresponding to the number of inputs selected for the ensembles. This proof is of utmost importance as it allows the Fiedler vector selected seed to initiate the search from a point higher than the average, ultimately striving towards an exhaustive outcome, which is the desired result of the search. By establishing the search average and providing solid proof, any starting point surpassing the predetermined average can be confidently regarded as a promising seed.

Proof. Consider a weight matrix W with elements w_{ij} belonging to the set $\mathbb{R}^{n\times n}$ such that $-1 \leq w_{ij} \leq 1$ and $w_{ij} = w_{ji}$. Define the weight function $w(x, y) = w_{xy}$. Let $L = [s]$ be a list containing all the elements of set s, and $s = \{L\}$ be a set containing all the elements of the list.

Note the relationship $L = [\{L\}]$ and $s = \{[s]\}$. Define the set of indices

$V = 1, 2, ..., n$, such that $s \subseteq V$ and $|s| > 1$. Also, let $s_i = [s]_i$.

$$\theta = \frac{2}{n(n-1)} \sum_{i=1}^{|s|-1} \sum_{j=j+1}^{|s|} w_{ij} \tag{6.1}$$

We can define Q as follows:

$$Q(s, W, \theta) = \sum_{i=1}^{|s|-1} \sum_{j=j+1}^{|s|} (w([s]_i, [s]_j) - \theta) \tag{6.2}$$

Let

$$\bar{W} = \begin{bmatrix} \theta & w_{12} & \cdots & \cdots & w_{1n} \\ w_{12} & \theta & \cdots & w_{ij} & \cdots \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \vdots & w_{ji} & \vdots & \theta & \vdots \\ w_{1n} & \cdots & \cdots & \cdots & \theta \end{bmatrix} \tag{6.3}$$

We can then infer that:

$$\sum_{i=1}^{n} \sum_{j=1}^{n} \bar{w}(i, j) = 2\left(\frac{(n(n-1)\theta)}{2}\right) + n\theta = n^2\theta \tag{6.4}$$

Equation (6.4) establishes the average quality of the search space based on the defined weight matrix and threshold parameter. It indicates that the sum of the modified weights in matrix $\bar{W}$ is proportional to the square of the matrix size n. □

Definition 6.6.1. The new quality function $\hat{Q}$ is defined as follows:

Given that $s_i = [s]_i$ then
$\bar{Q}$ is defined as

$$\bar{Q}(s, \bar{W}, \theta) = \sum_{i=1}^{|s|} \sum_{j=1}^{|s|} (\bar{w}(s_i, s_j) - \theta)$$

we can infer that:

$$\bar{Q}(s, \bar{W}, \theta) = 2Q(s, W, \theta)$$

$E(V)$ is defined as the set of all of the possible solutions, in this case, the set of all subsets in the ensemble and $P(V)$ is the power set of V.

Definition 6.6.2. The power set of a set V, denoted by $P(V)$, is the set of all possible subsets of V, including the empty set and the set itself. In other words, it is the collection of all subsets that can be formed by taking zero or more elements from the original set V.

For a set V with n elements, the power set $P(V)$ will have 2^n elements. This is because each element in V can be included or excluded from a subset, leading to a total of 2 choices for each element.

Suppose $V = \{a, b, c\}$, then the power set $P(V)$ would be:

$$P(V) = \{\emptyset, \{a\}, \{b\}, \{c\}, \{a, b\}, \{a, c\}, \{b, c\}, \{a, b, c\}\}$$

Here, $\emptyset$ represents the empty set, and each of the other subsets contains elements from the original set V.

Theorem 6.6.1. *The average fitness over the whole space, $E(V)$, is zero*.

Lemma 6.6.2. The number of times that any pair of elements from V appears together in one of the subsets of $P(V)$ is 2^{n-2}.
Consider a set with V elements. The number of ways to choose a pair of elements from this set is given by the binomial coefficient $C(V, 2)$, which is calculated as:

$$\binom{|V|}{2} == \frac{|V|!}{2!\,(|V|-2)!} \tag{6.5}$$

The above represents the possible combinations of 2 elements out of V. However, each of these combinations can appear in different subsets of the powerset. Generally, each pair of elements will appear in the powerset's $2^{(n-2)}$ subsets for a set with V elements.

Definition 6.6.3 (Set Subtraction)**.** Let r and s be two sets. The set subtraction operation, denoted as $r \setminus s$, generates a new set that includes all the elements belonging to r but not belonging to s. In other words, $r \setminus s$ contains the elements from set r that are not present in set s.

Theorem 6.6.3. *Given that the powerset of the set $V \setminus \{r, s\}$ has a cardinality of 2^{n-2}. This theorem establishes that when the elements r and s are removed from the set V, the resulting set has a powerset (set of all possible subsets without r and s) of size 2^{n-2}.*

Proof. Consider $P(\{r, s\})$, the power set of the set containing r and s. Each element in this set can be combined (unioned) with every element of $P(V \setminus \{r, s\})$. Consequently, there are $2^{(n-2)}$ sets that do not contain either r or s, $2^{(n-2)}$ sets that only contain r, $2^{(n-2)}$ sets that only contain s, and $2^{(n-2)}$ sets that contain both r and s.

From definition 6.5, the sum of $\bar{Q}$ over all subsets of $P(V)$ can be formulated as: Let $t_a = [P(V)]_a$

$$\sum_{a=1}^{|P(V)|} \bar{Q}(t_a, \bar{W}, \theta) = \sum_{i=1}^{n} \sum_{j=1}^{n} 2^{n-2}(\bar{w}(i, j) - \theta) \tag{6.6}$$

which can be expressed as

$$2^{n-2} \sum_{i=1}^{n} \sum_{j=1}^{n} (\bar{w}(i,j) - \theta) \tag{6.7}$$

Equation 6.7 shows that the sum involves 2^{n-2} times the sum of the differences between the modified weights in $\bar{W}$ and the threshold θ. Simplifying further:

$$\begin{aligned} \sum_{a=1}^{|P(V)|} \bar{Q}(t_a, \bar{W}, \theta) = 2^{n-2} \sum_{i=1}^{n} \sum_{j=1}^{n} (\bar{w}(i,j)) - \\ 2^{n-2} \sum_{i=1}^{n} \sum_{j=1}^{n} \theta = 0 \end{aligned} \tag{6.8}$$

Recall that:

$$P(V) = E(V) \cup \{1\} \cup \{2\} \cdots \cup \{n\} \cup \phi \tag{6.9}$$

It is important to note that $\bar{Q}(\{x\}, \bar{W}, \theta)$ for any $x \in V$ is equal to $\bar{w}(x,x) - \theta$, which is zero. Similarly, $\bar{Q}(\phi, \bar{W}, \theta)$ (the empty set) is also zero.

Thus, the sum of $\bar{Q}$ over all subsets of $E(V)$ is also zero, implying that the average quality (Q) is zero. □

The next section describes the dataset used to run the experiment.

6.7 Experimental Data

The dataset used in this chapter is based on the earlier chapters, and the processes leading to adopting the dataset for this experiment are the same as before. Continuing with the thirty clustering algorithms, a database of the clustering ensemble was generated using the same parameter settings. The best ensemble subset for each dataset was determined by evaluating the quality value for each k; the index corresponding to k with the highest quality is the number of clusters in the dataset. Further details can be found in Chapter 4. Although the process of finding the best subset value is significantly the same, the search for the best subset has been modified to speed up the process through a guided search approach with the introduction of seeding to the search process or starting point for the $RMHC$, see Chapter 5 for more detail on the $RMHC$.

In this study, the same quality metrics used in Chapters 4 and 5 are employed, but with the implementation of seeded $RMHC$ to accelerate the search process while maintaining comparable accuracy to the exhaustive approach in determining the best subset. While there is no universally agreed-upon definition of what constitutes good-quality clusters, accuracy and consistency are generally considered essential criteria. The experimental results presented in Section 6.10 demonstrate a significant improvement in convergence while yielding similar outcomes to the exhaustive approach. The fitness function utilised in this study builds upon the concept discussed in Chapter 5. Consistency is evaluated by assessing the average perfor-

mance across multiple runs.

6.8 Methods

This section introduces three additional methods derived from the FVD: QBestHC, QWorstHC, and RRHC2. It also describes the implementation of SA. The results obtained from applying these techniques will be compared with those from the exhaustive and RMModel discussed in Chapter 5.

The FVD obtains two starting points for the search space partitioning: QBest and QWorst. QBest represents the maximum quality value result from the partitioning, while QWorst represents the minimum. These quality values serve as the starting points for the search, and the corresponding RMHC values obtained from the search starting from QBest are referred to as QBestHC; similarly, for QWorst, it is QWorstHC. The QBestHC result is obtained by executing RMHC from QBest over a hundred repeats to assess the consistency of the results. The experiment section provides further details on these approaches and their implementation.

6.9 Experiments

The experiments are broken into three, as described in the table of experiments in Table 6.1. The three main experiments are:

- RMHC with QBest from Fiedler over ten thousand iterations
- RMHC with QWorst from Fiedler over ten thousand iterations
- Twice RMHC, five thousand from QBest and QWorst, return the Highest QBestHC and QWorstHC from two sub-HC

The first experiment runs a RMHC algorithm using the QBest from the Fiedler vector over a thousand iterations and for hundred repeats. The second experiment is similar but from the QWorst, and the last experiment is the average starting from QBest and QWorst but for half the number of iterations $5,000$ from both starting points.

Recall that in the earlier experiment in Chapter 5, we ran an exhaustive search on the twenty-seven datasets for the best subset, the same as the earlier experiment in Chapter 4, each for different algorithms but similar implementation. The purpose was to find the optimal value of k for each dataset, ranging from 2 to $\sqrt{n}$, where n is the number of instances in the dataset. The total number of datasets from the subset is 643. Each dataset is a $n \times n$ matrix corresponding to the number of inputs used to create the ensembles. The update quality version of the quality function is described in Chapter 5. The exhaustive search is the benchmark for the expected output for the other algorithms. The result from the second experiment contains RMModel described in the table of experiments in Table 6.1 in Chapter 5. The last set of experiment repeats the above for a range of input sizes from 5 to 30. Each sub-matrix is derived from the dataset as a symmetric sub-matrix from the 30×30 matrix of each dataset, similar to the experiment in Chapter 5 for the RMHC. The convergence points of different input sizes and the quality result by dataset and input sizes are reported in the result section in Section 6.10.

Table 6.1: FVD table of experiments: Showing the methods, datasets, number of iterations, and repeats.

Methods	Datasets	# Iterations	# Repeats
Exhaustive	27	1	1
$RMModel$	27	see Section 5.6	100
$QBestHC$	27	$10,000$	100
$QWorstHC$	27	$10,000$	100
$RRHC2$	27	$10,000$	100
SA	27	$10,000$	100
Total			**4,008,759**

6.10 Results and Discussions

The results is presented as follows:

1. Results by Quality
2. Results by Convergence points
3. Results by state

6.10.1 Results by Quality

This subsection presents the quality values for dataset sizes ranging from 5 to 30. The benchmark result obtained from the exhaustive method is denoted as Ex, and the results for the RMModel, QBestHC, QWorstHC, RRHC2, and (SA) implementations are presented in Table 6.2. This table provides a comprehensive overview of the quality values for each method.

The results clearly demonstrate the effectiveness of using Fiedler Vector seeding to guide the search for the best subset. Starting from QWorst yielded consistently poor results compared to the benchmark - the exhaustive. Similarly, the average performance of the QBest from two search points in the search space, denoted as RRHC2 for 5000 iteration, was close to the exhaustive method and even outperformed QBestHC. However, on

Table 6.2: Table showing the quality values for dataset size 5 to 30 based on the methods

Dataset size	Ex	$RMModel$	$QBestHC$	$QWorstHC$	$RRHC2$	SA
5	0.324	0.278	0.293	0.143	0.295	0.324
6	1.223	0.986	1.048	0.255	1.053	1.223
7	1.512	1.282	1.492	0.771	1.498	1.512
8	1.865	1.628	1.808	0.966	1.821	1.865
9	2.271	2.034	2.202	1.214	2.218	2.271
10	2.736	2.512	2.644	1.563	2.668	2.736
11	3.297	3.063	3.147	1.922	3.192	3.297
12	6.162	5.795	6.124	3.413	6.147	6.162
13	9.053	8.680	9.037	2.794	9.047	9.053
14	11.293	10.956	11.246	3.289	11.259	11.293
15	13.098	12.775	13.014	3.983	13.028	13.098
16	14.443	14.077	14.411	5.049	14.427	14.443
17	15.470	15.004	15.419	5.703	15.455	15.468
18	16.240	15.583	16.183	6.417	16.248	16.237
19	18.639	18.035	18.583	6.989	18.635	18.636
20	20.003	19.462	19.912	7.896	19.966	19.998
21	21.600	21.076	21.489	8.237	21.525	21.593
22	23.157	22 .658	22.973	8.286	23.025	23.139
23	26.403	25.931	26.237	8.990	26.282	26.395
24	27.468	26.973	27.210	9.822	27.282	27.441
25	30.969	30.517	30.752	10.625	30.813	30.943
26	33.296	32.894	33.149	14.389	33.199	33.269
27	35.012	34.655	34.888	14.380	34.930	35.005
28	36.647	36.291	36.500	15.125	36.557	36.637
29	39.290	38.969	39.161	16.962	39.214	39.287
30	43.272	42.995	43.160	18.830	43.213	43.263

average, QBestHC was 33% better than the RMModel, particularly as the dataset size increased, as shown in the graph in Figure 6.5. Surprisingly, the SA implementation performed the best on the estimate but the convergence was very poor compared to other methods. Aside from slower convergence, SA may not be suitable in the current context due to some of the following reasons:

Sensitivity to temperature parameters: The performance of SA is highly dependent on the selection of temperature parameters, such as the initial temperature and the cooling schedule and choosing inappropriate temperature values or a suboptimal cooling schedule can result in poor convergence and suboptimal solutions.

Difficulty in tuning parameters: SA involves several parameters that must

be appropriately tuned for optimal performance. Determining suitable values for parameters such as the initial temperature, cooling rate, and acceptance criteria can be challenging and time-consuming, requiring careful experimentation and analysis.

Inefficiency for large-scale problems: SA can become computationally expensive and inefficient for large-scale optimisation problems. The algorithm sequentially explores the search space, and as the problem size grows, the time required for exploration increases significantly. This scalability issue can limit the applicability of SA to large and complex problem domains.

Inability to seed SA: Unlike the current technique (FVD), SA does not lend itself well to seeding because at the beginning of the optimisation process, SA often accepts worse solutions to encourage exploration. As a result, it tends to "run away" from the seed point rather than converging towards it. This lack of control over the starting point can make it challenging to target specific areas of the search space or guide the algorithm towards known promising regions.

In conclusion, it is crucial to carefully assess these disadvantages concerning the search space in the clustering ensemble and consider the ensemble size, solution requirements, time constraints, and available domain knowledge before choosing a suitable technique.

6.10.2 Results by Convergence Points

The convergence point of FVD is generally expected to increase as the size of the input dataset grows due to the computational complexities associated with the method. However, the results obtained for the convergence points differ from this assumption, especially for lower data size, and several reasons may contribute to this discrepancy. The results for the con-

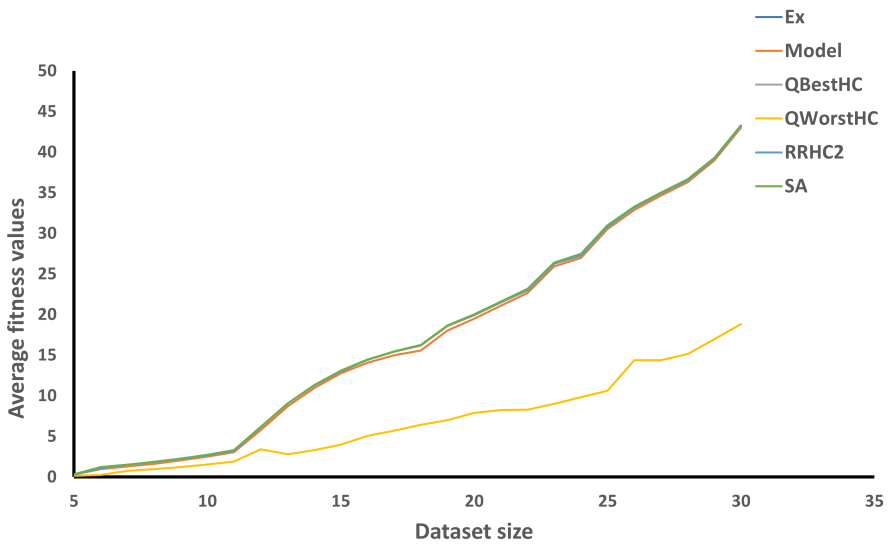


Figure 6.5: Quality Value for Methods: Shows the Average Quality Values for Each Method.

vergence points of dataset sizes greater than 13 are in Figure 6.6, showing an increasing trend which aligns with the expected behaviour. Notably, the RRHC2 method, utilises two different starting points in the search space, and exhibits significantly higher convergence points than the other methods. However, QWorstHC, which employs the lower quality values as the starting point, performs poorly in terms of convergence.

6.10.3 Results by State

In data analysis, the term “state” refers to the condition or configuration of a system at a specific moment in time. Gaining insight into the state of data analysis is crucial for researchers and analysts as it provides an understanding of the current condition of the data and its behaviour over time. This understanding enables the identification of patterns, trends, and changes, which can be leveraged for making predictions and drawing data-driven conclusions. In this research, three states are identified based on

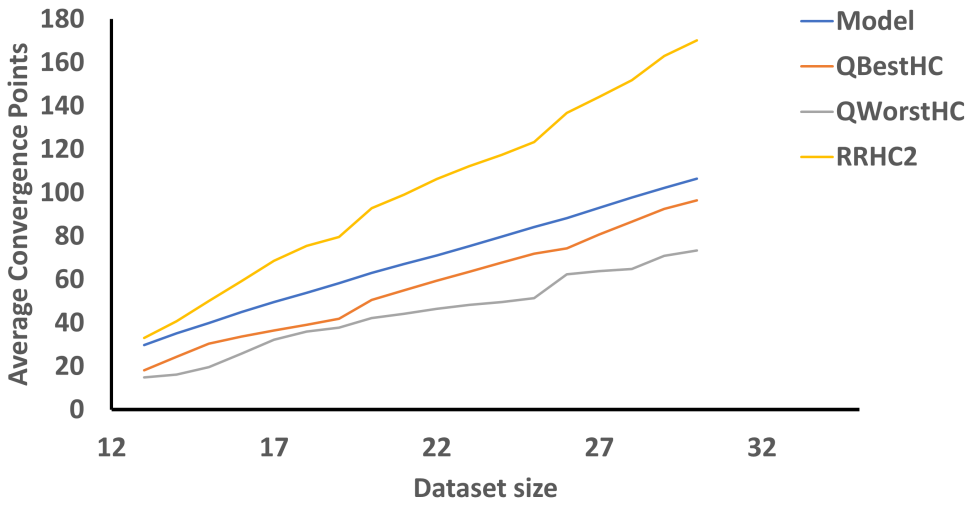


Figure 6.6: Convergence Values Methods: The figure shows the convergence values for each method in the legend on average from data size 13 to 30

the run of the experiment for different FVD implementations. The states are:

1. FVD failed – No results/outcomes
2. FVD ≤ 0 – The results are equal or below the average search space of zero.
3. FVD > 0 – the results are greater than the average search space of zero.

The underlying assumption in this thesis is that any value greater than zero, which corresponds to the average value in the search space, can serve as a promising starting point for finding the optimal subset within the search space. The implementation of this approach is evaluated based on the quality values and convergence points obtained from the experiments. The table below shows the state distribution based on the above description:

The plot in Figure 6.7 illustrates values greater than 0 for the Fiedler Vector

Table 6.3: State Distributions

State Name	State	Count	Percentage
FVD Failed	1	7	0.03%
FVD ≤ 0	2	756	4.52%
FVD > 0	3	15955	95.43%

(FVD) across datasets ranging from 5 to 30. Notably, a consistent upward trend in the number of successful datasets indicates an outcome from the Fiedler vector from a point towards the optimal solution from the corresponding quality value. However, a slight dip is observed around the dataset size of 11. The diagram shows a predominantly increasing pattern, further confirming significant poor outcomes for lower values of the dataset between 5 and 12.

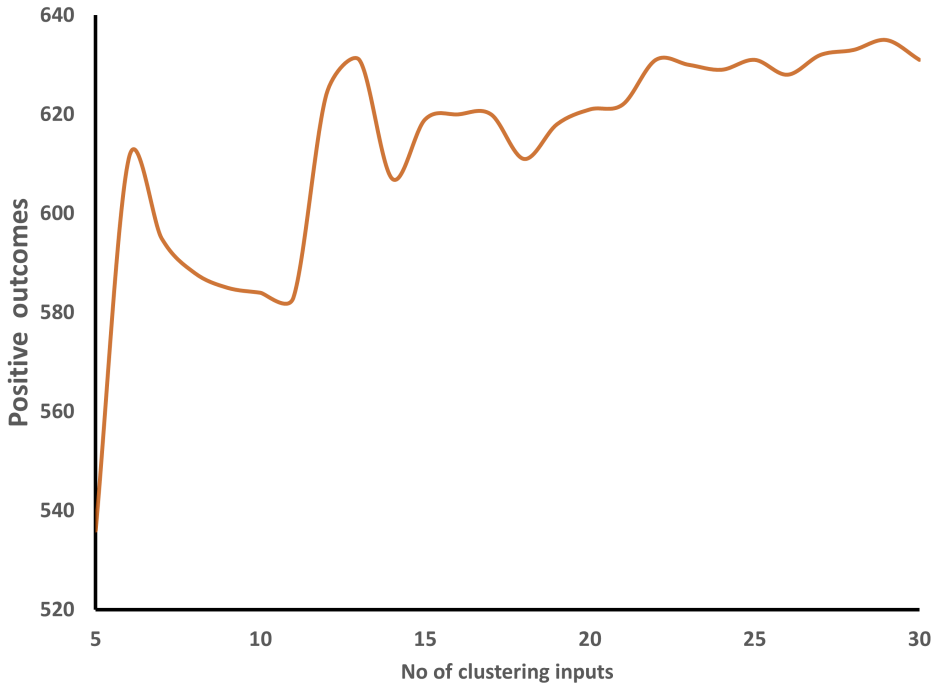


Figure 6.7: The figure shows the number of outcomes for State 3 of the FVD for all datasets

The efficiency of the FVD seeding method has been evaluated across various dataset sizes ranging from 5 to 30. The results showed that the av-

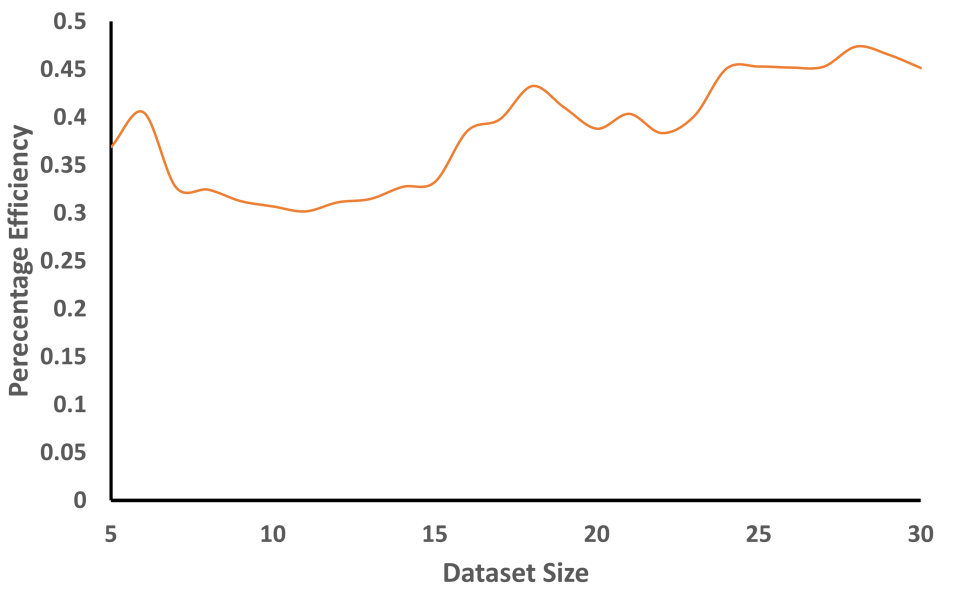


Figure 6.8: The Figure Shows the Efficiency by Dataset Size

erage efficiency across all dataset sizes was 38.7%, indicating a significant improvement. Figure 6.8 provides a visual representation of the specific efficiency values for each dataset size, and those with positive outcomes and the corresponding numbers are shown in Figure 6.7.

This analysis offers valuable insights into the effectiveness of the FVD seeding approach for estimating the number of clusters in datasets and conducting cluster analysis. The average efficiency values comprehensively measure the method's overall performance across the dataset sizes.

Based on the observed trend in the graph, it is evident that the FVD seeding approach is not well-suited for small dataset sizes. In such cases, it is advisable to opt for the model alternative described in Chapter 5 (RMModel), as indicated in the graph in Figure 6.7. However, as the dataset size increases, employing a guided search alternative like Fiedler vector decomposition becomes increasingly beneficial. This approach harnesses the Fiedler Vector's capability to decompose the graph and establish a meaningful starting

point within the search space. Leveraging this technique increases the efficiency and accuracy of subset evaluation from the clustering ensemble, improving results, particularly for larger dataset sizes.

6.11 Recommendations

This chapter introduced seeding into the RMHC implementation, utilising graph-based techniques and Fiedler vector decomposition. The utilisation of seeding has been shown to enhance the search process by starting from a seed point in the search space that is likely to contain the optimal solution. However, introducing the Fiedler vector decomposition is computationally intensive, rendering it unsuitable for lower input values. Surprisingly, this leads to prolonged search processes, potentially resulting in increased convergence time or failure, as evidenced by the observed results on the convergence points. Two plausible reasons may contribute to these outcomes:

Firstly, seeding with the Fiedler vector decomposition is sensitive to the initial conditions of the data, particularly the graph structure or adjacency matrix. Even minor variations in these initial conditions can yield significantly divergent results, making the method sensitive to data inaccuracies. This sensitivity can potentially impact convergence time, as the quality of the initial seed directly affects the efficiency of the search process; smaller datasets ranging from 5 to 12 exhibit a substantial number of missing results from the search, potentially due to these factors.

Secondly, the static nature of the seed, determined based on the graph structure or adjacency matrix at the onset of the search process, may contribute to the observed high convergence points for smaller dataset sizes. Further investigation is required to confirm the above observations on the

convergence points.

The next chapter provides a comprehensive discussion of the major contributions made throughout the research, summarises the key findings obtained, acknowledges the limitations encountered, and outlines potential research directions for future work.

Chapter 7

Conclusions

In conclusion, this thesis highlights the importance of cluster analysis and the limitations of single clustering algorithms. It addresses the challenging task of determining the optimal number of clusters in datasets and proposes novel methods that leverage ensemble techniques. These methods overcome single clustering algorithms' limitations, such as sensitivity to initialisation, difficulties in handling complex data structures, and scalability issues. As presented in this thesis, applying ensemble methods in cluster analysis enhances clustering results' accuracy, stability, and interpretability. The developed methods offer innovative approaches for navigating the search space and provide more reliable and insightful data analysis outcomes. The rest of this chapter summarises and draws conclusions on earlier findings:

Chapter 1 serves as the introductory chapter of this thesis. It introduces the research topic, establishes the study's motivation, and lays the foundation for heuristic optimisation in clustering ensembles. Furthermore, it provides a general overview of the research, giving readers a comprehensive introduction to the subject and setting the stage for the subsequent chapters that address the specifics of the study.

Chapter 2 presents a comprehensive literature review, focusing on the key concepts relevant to the research. It introduces different clustering types and discusses their distinctive characteristics, with a primary emphasis on single clustering algorithms. The chapter underlines the limitations of single clustering and underscores the importance of constructing ensembles using multiple algorithms as an alternative approach. Furthermore, the chapter explores heuristic search algorithms and strategies, which play a critical role in the optimisation process of clustering ensembles. It also introduces various cluster indices that serve as quantitative measures for evaluating clustering solutions' quality and performance. Moreover, the chapter thoroughly describes the fitness of a search space and the convergence properties, particularly in the context of hill climbing and Simulated Annealing algorithms applied in the research. These concepts are essential for understanding the optimisation techniques used in the study.

Chapter 3 of the research describes the datasets used in the experiments and explains the data collection and pre-processing. The pre-processing stages involved handling missing data, outliers, data integration (if obtained from multiple sources), data transformation (standardisation), data reduction (dimensionality reduction), and data normalisation; after these steps, a total of 27 datasets were chosen for further analysis. Chapter 3 concludes by discussing the characteristics and specific dataset features, highlighting their unique properties and clustering challenges. In conclusion, this chapter presents a comprehensive variability analysis of the selected datasets. The variability analysis allowed valuable insight into the hidden structures and patterns within the datasets. This information is crucial for understanding the characteristics and nature of the data. It also enables the assessment of the consistency of clustering outcomes across different runs or variations in the input parameters.

Chapter 4 focuses on estimating the number of clusters in a dataset using the ensemble framework. It highlights the importance of determining the number of clusters and the potential of ensemble clustering to enhance the performance and accuracy of the estimated number of clusters. Chapter 4 addresses two fundamental issues in clustering ensembles: the optimal way to generate and combine clusterings while maintaining diversity and accuracy and identifying the best solution from subsets of ensembles in the search space. It was shown in the chapter that the ensemble technique consistently gives lower average errors across all datasets and correctly estimates the number of clusters compared to similar methods. The section recommends adopting the ensemble clustering method as a preferred alternative. However, acknowledging the potential speed challenges as the dataset and input method sizes increase is crucial. Further application of heuristic search approaches to enhance the method's efficiency is also suggested for future work, as detailed in Chapter 5.

Chapter 5 introduces a heuristic search-based clustering ensemble approach to improve the speed of execution of the earlier work in Chapter 4, precisely the random mutation hill climbing. The goal is to address the speed requirements as the dataset size and possible number of subsets from the ensemble increase. The novelty and contributions of the approach are outlined, including the development of a heuristic-based ensemble that achieves the same quality as the exhaustive search, a mathematical framework for the fitness function, and a regression model to estimate the maximum convergence point for the heuristic search were presented. The study also addresses the trade-off between accuracy, consistency, and speed when choosing between heuristic and exhaustive search methods.

Chapter 6 address the challenges associated with starting points and outlines strategies to overcome the limitations of traditional hill climbing algo-

rithms identified in Chapter 5. The Fiedler vector decomposition allows for a guided initialisation of the initial starting point in the search space. These strategies enhance the exploration capability of the algorithm and increase the likelihood of converging to global optima rather than getting trapped in local optima. Addressing the starting point issue and proposing practical solutions has improved the overall performance and reliability of the search process, ultimately leading to more robust and accurate results in the search for the optimal number of clusters in datasets.

This thesis serves as a proof of concept and sets a promising direction for future research. It contributes to cluster analysis by introducing novel ensemble-based methods for determining the number of clusters and addressing the limitations of single clustering algorithms. The presented methodologies offer promising avenues for future research, including applying heuristic search approaches to enhance efficiency and further exploring the proposed concepts.

7.1 Contributions

This section is an overview of the main results presented in this thesis.

7.1.1 Gray Code Subsetting

The main goal of Gray Code sub-setting is to generate potential solutions systematically by constructing subsets of ensembles with minimal variations between subsets. This approach aims to accurately represent the search space while ensuring a diverse range of representative solutions. Gray code sub-setting enhances the interdependence between subsets and is leveraged to compute fitness values, thereby improving efficiency in the search process. This method converged to a higher average at a faster convergence rate than some of the standard techniques used to compare, and

it compares favourably to the exhaustive search method in the number of clusters estimated correctly.

7.1.2 Assessing Subsets Based on Quality

Another significant outcome of this research is the introduction of an objective function that enables the evaluation and optimisation of each subset's quality, resulting in improved accuracy compared to conventional techniques. Unlike many methods used for determining the number of clusters that rely on statistical formulas that can be data or distribution dependent, the proposed approach overcomes these limitations by utilising a weighted kappa agreement matrix generated during the initial base clustering process. This matrix ensures a robust and reliable assessment of subset quality, enhancing the overall accuracy of the analysis.

7.1.3 Quality Metric Framework

The research introduced a quality metric to assess the suitability of each subset as a predictor of the number of clusters in the datasets. This metric allows evaluating each subset among its pairs as a potential solution. The results of the different metrics outlined in Chapter 4 guided the selection of a suitable threshold for the datasets. They served as the basis of the search towards the optimal subset. In summary, the research presented a mathematical framework for quality metrics that facilitates the scoring of subsets, considering both weaker and stronger subsets. Furthermore, incorporating threshold values into the metric enhances the evaluation of subsets based on their relative strength.

7.1.4 Seeded RMHC

The implementation of $RMHC$ may be inefficient in exploring the search space, as it is limited to making small random changes to the current solution. This limitation can lead to the oversight of superior solutions that exist further away. Additionally, RMHC requires a substantial amount of time to converge to an optimal solution, especially in complex search spaces, and the initial starting point can influence its performance. Hill climbing algorithms are also prone to get trapped in local optima, i.e. Once the algorithm reaches a peak in the search space, it may be unable to explore other potentially better solutions that are further away. These challenges have been addressed in Chapter 6 of this thesis using seeded RMHC. The research proposes using the Fiedler vector decomposition to provide a guided or good initial starting point in the search space. A rigorous set of experiments in Chapter 6 shows the approach as a promising alternative to earlier implementation in Chapter 5. The implementation results demonstrate a significant enhancement in the search for the best subset, indicating the effectiveness of this approach in overcoming the limitations of traditional RMHC.

7.1.5 Proof of the Search Space Fitness Average

The search space fitness average has been established by conducting extensive experimentation and empirical analysis. This evaluation has enabled the assessment of the seeding process as viable starting points within the search space. Determining the search space fitness average involves rigorous testing and analysis to measure the performance and effectiveness of the seeding process. Empirical proof has demonstrated that the seeding process provides satisfactory initial configurations for the search space. The evaluation of the search space fitness average is a valuable

indicator of the quality and reliability of the seeding process. It provides confidence in the effectiveness of the chosen starting points, allowing for more efficient and accurate exploration of the search space. Researchers and practitioners can make informed decisions regarding selecting and using the seeding process by utilising the established search space fitness average. This proof optimises the overall search process, ensuring it starts from a favourable position within the search space.

7.2 Summary

Much of the research on estimating the number of clusters in datasets still relies on either clustering-based or statistical methods, as discussed in this thesis. However, these approaches fail to consider the potential benefits of incorporating diverse views of the data and exploring a range of search possibilities to obtain accurate estimates of the number of clusters. The proposed methodology of clustering ensemble offers a promising solution by leveraging multiple clustering perspectives and searching for the best subset that captures the underlying characteristics of the dataset. This approach improves accuracy and holds potential for application in various domains beyond those explored in this thesis, including real-world datasets. In summary, the methodology for estimating the number of clusters in an ensemble, considering accuracy and effectiveness, will depend on the size and selection of the candidate solutions in the ensemble. Here are some recommendations:

1. **Exhaustive method**: When the number of clustering inputs is less than or equal to seven (7), it is advisable to use the exhaustive method. The exhaustive applies all possible combinations of clustering inputs to find the optimal solution, and this approach ensures a thorough exploration of the solution space.

2. **Model Method**: If the number of clustering algorithms falls between eight 8 and twelve 12, it is recommended to use a model-based approach such as $RMModel$. They can handle a moderate number of algorithms effectively.

3. **FVD Seeding**: When dealing with more input algorithms (≥ 13) in the ensemble, it is beneficial to employ Fiedler vector seeding. This technique helps direct the search for the optimal number of clusters and reduces the time required to converge further with similar accuracy to the exhaustive. Therefore, considering the number of clustering algorithms and employing the appropriate method: exhaustive, model-based, or Fiedler vector seeding can enhance the efficiency and effectiveness of the clustering process, leading to better results in finding the optimal number of clusters and minimising convergence time.

In addition, the FVD approach proves particularly valuable as it provides a guided starting point in the search space for clustering. Establishing the search space average, supported theoretically and empirically, enhances the effectiveness of the starting point. Further exploration can focus on identifying even better starting points in the search process to improve the prediction of the number of clusters in datasets. This avenue holds potential for future research and development.

7.3 Limitations

Like any novel ideas, the methods proposed in this thesis are not without their limitations. The following are the identified limitations of these methods.

7.3.1 Generating Ensembles

One of the challenges associated with generating ensembles is the increased computational complexity and resource requirements. Ensembles typically involve combining multiple models or algorithms, significantly increasing the computational load compared to using a single model, especially when dealing with large datasets or complex algorithms. Furthermore, generating ensembles requires diverse and representative individual clusterings. Obtaining diverse clustering solutions can be challenging, especially when using complex algorithms or when there is limited availability of different clusterings to combine. It may require generating clusterings with different parameter settings, using different algorithms, or collecting data from different sources to achieve the desired diversity. Another issue is the potential for disagreement or conflicting predictions among the individual subsets in the ensemble. While diversity is desired, conflicting predictions can pose a challenge in decision-making or result interpretation. Resolving conflicts and aggregating the predictions of different subsets in a meaningful and effective way is a non-trivial task.

7.3.2 Seeded RMHC

Two notable limitations of the proposed methodology are identified. Firstly, the sensitivity of seeding with the Fiedler vector to initial data conditions, particularly the graph structure or adjacency matrix, poses a challenge. Even minor variations in these conditions can lead to significant differences in the results, making the method susceptible to data inaccuracies. This sensitivity may also affect the convergence time, as the initial seed's quality directly impacts the search process's efficiency. Secondly, the static nature of the seed, determined based on the graph structure or adjacency matrix at the beginning of the search process, may contribute to the observed high convergence points for smaller dataset sizes.

7.4 Further Work

The thesis has introduced a novel approach for estimating the number of clusters in datasets through clustering ensembles and a heuristic search algorithm. This approach has demonstrated improved accuracy compared to traditional methods by incorporating diverse data views and evaluating subset's quality. Future work in the research area could focus on several aspects to further improve and expand the existing methods. Here are some potential avenues for future investigation:

MST threshold for graph creation: An exciting avenue would be investigating different threshold values for constructing the graph from the Weighted Kappa (w) matrix. Researchers can gain insights into the impact on the resulting graph structure by exploring various threshold values. This analysis can lead to the identification of optimal threshold selection for different datasets, thereby enhancing the accuracy of the clustering process.

More efficient local search methods: It would be beneficial to apply and evaluate more efficient local search methods to optimise the search process. Advanced optimisation techniques can be investigated, such as metaheuristic algorithms (e.g., particle swarm optimisation [179]) or advanced local search strategies like tabu search [180] or iterated local search [181].

Suitability of the method for different datasets: Investigate and analyse the characteristics of datasets for which the proposed method is more suited and identify the types of data distributions, sizes, and structures where the method exhibits superior performance. This analysis can provide valuable insights into the strengths and limitations of the approach and guide its application in various domains.

Incorporation into outlier detection: Exploring the integration of the proposed method into outlier detection techniques would be valuable. Evaluating the method's performance in identifying outliers within datasets and assessing its effectiveness in outlier detection tasks. This integration can provide a more comprehensive approach to data analysis.

Real-valued matrix versions of FVD: Currently, the FVD method primarily deals with binary data. Extending the technique to handle continuous or real-valued data would expand its applicability, enable its usage on a broader range of domains and datasets, and open up new possibilities.

Alternative methods to Arnoldi: Exploring alternative approaches to eigenvalue computation, specifically for the Arnoldi method, is worth investigating. Full-spectrum exact methods can be explored for low-dimensional cases where computational cost is less of a concern. Comparing the performance and accuracy of these alternative methods with the Arnoldi method in different scenarios will enable researchers to assess their suitability for various problem sizes and dimensions. Such analysis will provide insights into the trade-offs between accuracy and computational cost, given the differences in time complexity between the Arnoldi method $O(n^2)$ and full-spectrum exact methods $O(n^3)$.

Appendices

Appendix A

This appendix describes the results from the simulated datasets. The average fitness values are extremely close to zero, with at least eleven zeros after the decimal point before the first non-zero digit. This indicates that the values are almost indistinguishable from zero for most practical purposes.

Table showing the quality values for inputs 5 to 30 based and their average fitness values

Matrix Size	Range	No of Simulation	Average Fitness
5	32	26	-4.604×10^{-13}
6	64	57	-5.604×10^{-14}
7	128	120	-2.704×10^{-12}
8	256	247	-4.304×10^{-12}
9	512	502	5.901×10^{-10}
10	1024	1013	-4.699×10^{-10}
11	2048	2036	-5.894×10^{-12}
12	4096	4083	4.507×10^{-11}
13	8192	8178	-8.939×10^{-11}
14	16384	16369	-7.313×10^{-11}
15	32768	32752	3.752×10^{-11}
16	65536	65519	-6.800×10^{-15}
17	131072	131054	-6.800×10^{-15}
18	262144	262125	-7.351×10^{-16}
19	524288	524268	-4.013×10^{-15}
20	1048576	1048555	-2.920×10^{-14}
21	2097152	2097130	6.604×10^{-11}
22	4194304	4194281	-3.604×10^{-13}
23	8388608	8388584	-4.604×10^{-15}
24	16777216	16777191	-7.604×10^{-15}
25	33554432	33554406	-1.235×10^{-11}
26	67108864	67108837	6.457×10^{-10}
27	134217728	134217700	-2.125×10^{-10}
28	268435456	268435427	-5.231×10^{-12}
29	536870912	536870882	-2.604×10^{-13}
30	1073741824	1073741793	-3.321×10^{-13}
Total Simulation	$2^n - n$	**1610612675**	

Appendix B

This appendix describes the datasets and provide a short description on the number of columns and rows as well as sources where they were collected.

Table showing the dataset and their description

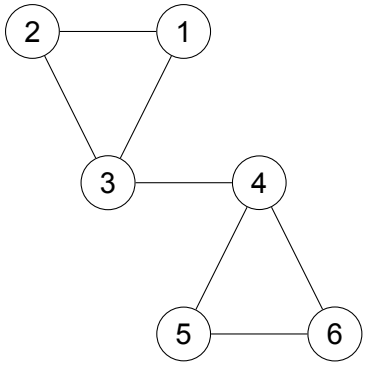
#	Dataset names	#rows	#columns	Source	DOI	Metadata	Fair-Compliant
1	aml28.csv	3	804	https://github.com/milaan9/Clustering-Datasets/blob/master/02.%20Synthetic/aml28.arff	N	Y	Y
2	atom.csv	4	800	https://cran.r-project.org/web/packages/FCPS/index.html	10.1016/j.dib.2020.105501	Y	Y
3	bezdekIris.csv	5	150	https://archive.ics.uci.edu/dataset/53/iris	10.24432/C56C76	Y	Y
4	blobs.csv	3	300	Data generated using Python package `sklearn` (sklearn.datasets.samples_generator)	N	Y	Y
5	cassini.csv	3	1000	Data generated using R package `mlbench`	10.1198/106186006X59243	Y	Y
6	compound.csv	3	399	https://cs.joensuu.fi/sipu/datasets/Compound.txt	10.1109/T-C.1971.223083	Y	Y
7	curves1.csv	3	1000	https://github.com/milaan9/Clustering-Datasets/blob/master/02.%20Synthetic/curves1.csv	N	Y	Y
8	gaussian500.csv	3	3000	generated two Gaussian clouds (R lang)	N	Y	Y
9	glass.csv	11	214	http://odds.cs.stonybrook.edu/glass-data/	10.1145/2830544.2830549	Y	Y
10	hepta.csv	4	212	https://cran.r-project.org/web/packages/FCPS/index.html	10.1002/9780470316801	Y	Y
11	longsquare.csv	3	900	https://github.com/milaan9/Clustering-Datasets/blob/master/02.%20Synthetic/longsquare.arff	10.1145/1276958.1277126	Y	Y
12	lsun.csv	3	400	https://github.com/milaan9/Clustering-Datasets/blob/master/02.%20Synthetic/lsun.arff	N	Y	Y
13	pearl.csv	3	266	https://github.com/milaan9/Clustering-Datasets/blob/master/02.%20Synthetic/pmf.arff	N	Y	Y
14	pmf.csv	4	649	https://github.com/milaan9/Clustering-Datasets/blob/master/02.%20Synthetic/pmf.arff	N	Y	Y
15	shapes.csv	3	1000	Data generated using R package `mlbench`	N	Y	Y
16	size1.csv	3	1000	https://github.com/milaan9/Clustering-Datasets/blob/master/02.%20Synthetic/sizes1.arff	10.1109/cec.2005.1554742	Y	Y
17	size2.csv	3	1000	https://github.com/milaan9/Clustering-Datasets/blob/master/02.%20Synthetic/sizes2.arff	10.1109/cec.2005.1554742	Y	Y
18	spherical_5_2.csv	3	250	https://github.com/milaan9/Clustering-Datasets/blob/master/02.%20Synthetic/spherical_5_2.arff	10.1016/s0031-3203(01)00108-x	Y	Y
19	square2.csv	3	1000	https://github.com/milaan9/Clustering-Datasets/blob/master/02.%20Synthetic/square2.arff	N	Y	Y
20	synthetic_control.csv	62	600	https://zenodo.org/record/4566804	N	Y	Y
21	tetra.csv	4	400	https://cran.r-project.org/web/packages/FCPS/index.html	10.1016/j.dib.2020.105501	Y	Y
22	tetragonula_bee.csv	16	236	https://cran.r-project.org/web/packages/FCPS/index.html	10.1016/j.dib.2020.105501	Y	Y
23	ThreeMC.csv	3	400	https://rdrr.io/github/mrc-ide/threemc/src/R/data.R	10.48550/arXiv.2108.09142	Y	Y
24	triangle1.csv	3	1000	https://github.com/milaan9/Clustering-Datasets/blob/master/02.%20Synthetic/triangle1.arff	N	Y	Y
25	vehicle.csv	19	846	https://archive.ics.uci.edu/ml/datasets/Statlog+%28Vehicle+Silhouettes%29	10.24432/C5HG6N	Y	Y
26	veronica.csv	586	206	https://ifcs.boku.ac.at/repository/data/veronica/index.html	10.1600/036364404245107 1	Y	Y
27	zelnik3.csv	3	266	https://github.com/milaan9/Clustering-Datasets/blob/master/02.%20Synthetic/zelink3.mat	N	Y	Y

Appendix C

This section of the appendix describes the steps involved in Fiedler Vector Decomposition (FVD) for an unweighted graph. Fiedler Vector Decomposition is a fundamental technique in graph theory utilised to partition graphs into distinct clusters.

Step 1:

Consider the unweighted graph below:



C.1 Example of an unweighted graph

The corresponding adjacency matrix A is shown below:

$$A = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 \end{bmatrix}$$

Step 2:

The Laplacian matrix is a fundamental concept in graph theory, offering valuable insights into a graph's connectivity and structure. It is constructed from the adjacency matrix and is a tool for analysing various graph properties. Specifically, the Laplacian matrix is obtained by subtracting the adjacency matrix from the degree matrix, thereby providing a means to examine the graph's connectivity and relationships between its nodes.

To calculate the Laplacian matrix L, we first compute the degree matrix Z, which represents the degree (number of edges connected to a node) of each node along the diagonal:

$$Z = \begin{bmatrix} 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 \end{bmatrix}$$

Then, we subtract the adjacency matrix A from the degree matrix Z to obtain the Laplacian matrix L:

$$L = Z - A$$

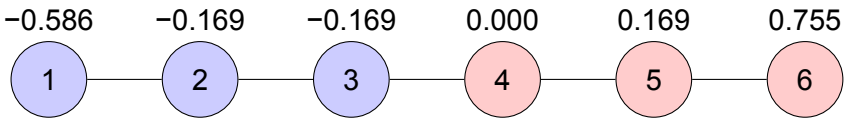
$$L = \begin{bmatrix} 2 & -1 & -1 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 & 0 \\ -1 & -1 & 3 & -1 & 0 & 0 \\ 0 & 0 & -1 & 3 & -1 & -1 \\ 0 & 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & -1 & -1 & 2 \end{bmatrix}$$

Step 3:

The calculated eigenvalues of L sorted in ascending, order are:

$-0.586, -0.169, -0.169, 0.000, 0.169, 0.755$.

The **second smallest eigenvalue**, the Fiedler value, is crucial for graph partitioning. The corresponding eigenvector, called the Fiedler vector, provides information about the optimal partitioning of the graph into two groups. In this example, the Fiedler value is -0.169, and its corresponding eigenvector represents the optimal partitioning of the graph. Vertices with negative values in the Fiedler vector are assigned to one group, while vertices with positive values are assigned to the other group. Based on the Fiedler vector, the graph is partitioned into two groups as follows:



Bibliography

[1] mygreatlearning, “An introduction to hill climbing algorithm,” 2023, accessed March 07, 2023. [Online]. Available: https://www.mygreatlearning.com/blog/an-introduction-to-hill-climbing-algorithm/

[2] M. Charrad, N. Ghazzali, V. Boiteau, and A. Niknafs, “Determining the number of clusters using nbclust package,” *MSDM*, vol. 2014, p. 1, 2014.

[3] A. K. Jain, “Data clustering: 50 years beyond k-means,” *Pattern recognition letters*, vol. 31, no. 8, pp. 651–666, 2010.

[4] C. C. Aggarwal, S. Y. Philip, J. Han, and J. Wang, “A framework for clustering evolving data streams,” in *Proceedings 2003 VLDB conference*. Elsevier, 2003, pp. 81–92.

[5] A. K. Jain, M. N. Murty, and P. J. Flynn, “Data clustering: A review,” *ACM computing surveys (CSUR)*, vol. 31, no. 3, pp. 264–323, 1999.

[6] D. Jiang, C. Tang, and A. Zhang, “Cluster analysis for gene expression data: a survey,” *IEEE Transactions on knowledge and data engineering*, vol. 16, no. 11, pp. 1370–1386, 2004.

[7] X. Z. Fern and C. E. Brodley, “Solving cluster ensemble problems by bipartite graph partitioning,” *Proceedings of the 21st international conference on Machine learning*, p. 36, 2004.

[8] L. Rokach and O. Maimon, “Clustering methods,” in *Data mining and knowledge discovery handbook*. Springer, 2005, pp. 321–352.

[9] C. Lin, C. Chen, H. Lee, and J. Liao, “Expert Systems with Applications Fast K-means algorithm based on a level histogram for image retrieval,” *Expert Systems With Applications*, vol. 41, no. 7, pp. 3276–3283, 2014. [Online]. Available: http://dx.doi.org/10.1016/j.eswa.2013.11.017

[10] A. Topchy, A. K. Jain, and W. Punch, “Clustering ensembles: Models of consensus and weak partitions,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 27, no. 12, pp. 1866–1881, 2005.

[11] A. A. Odebode, Q. Ye, S. Sampalli, and S. Dey, “Kd1: A sampling-based clustering scheme for large data sets,” in *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*. IEEE, 2018, pp. 822–827.

[12] S. Ayed, M. Arzoky, S. Swift, S. Counsell, and A. Tucker, “An exploratory study of the inputs for ensemble clustering technique as a subset selection problem,” in *Proceedings of SAI Intelligent Systems Conference*. Springer, 2018, pp. 1041–1055.

[13] A. Swift, H. Huang, and P. Green, “Consensus and stability in large networks,” *Physical Review E*, vol. 69, no. 6, p. 065102, 2004.

[14] J. R. Sampson, “Adaptation in natural and artificial systems,” 1976.

[15] M. Mitchell, *An introduction to genetic algorithms*. MIT press, 1998.

[16] A. Odebode, A. Tucker, M. Arzoky, and S. Swift, “Estimating the optimal number of clusters from subsets of ensembles,” *Proceedings of the 11th International Conference on Data Science, Technology and Applications*, 2022.

[17] C. Fraley and A. E. Raftery, “How many clusters? which clustering method? answers via model-based cluster analysis,” *The computer journal*, vol. 41, no. 8, pp. 578–588, 1998.

[18] S. Swift, A. Tucker, J. Crampton, and D. Garway-Heath, “An improved restricted growth function genetic algorithm for the consensus clustering of retinal nerve fibre data,” in *Proceedings of the 9th annual conference on Genetic and evolutionary computation*. ACM, 2007, pp. 2174–2181.

[19] R. C. Dubes, “How many clusters are best?-an experiment,” *Pattern Recognition*, vol. 20, no. 6, pp. 645–663, 1987.

[20] C. Hennig, “What are the true clusters?” *Pattern Recognition Letters*, vol. 64, pp. 53–62, 2015.

[21] R. Xu and D. Wunsch, “Survey of clustering algorithms,” *IEEE Transactions on neural networks*, vol. 16, no. 3, pp. 645–678, 2005.

[22] C. M. Koch, S. F. Chiu, M. Akbarpour, A. Bharat, K. M. Ridge, E. T. Bartom, and D. R. Winter, “A beginner’s guide to analysis of rna sequencing data,” *American journal of respiratory cell and molecular biology*, vol. 59, no. 2, pp. 145–157, 2018.

[23] M. A. Ginos, G. P. Page, B. S. Michalowicz, K. J. Patel, S. E. Volker, S. E. Pambuccian, F. G. Ondrey, G. L. Adams, and P. M. Gaffney, “Identification of a gene expression signature associated with recurrent disease in squamous cell carcinoma of the head and neck,” *Cancer research*, vol. 64, no. 1, pp. 55–63, 2004.

[24] H. Ng, S. Ong, K. Foong, P. Goh, and W. Nowinski, “Medical image segmentation using k-means clustering and improved watershed algorithm,” in *2006 IEEE Southwest Symposium on Image Analysis and Interpretation*. IEEE, 2006, pp. 61–65.

[25] M. Arzoky, “Munch: an efficient modularisation strategy on sequential source code check-ins,” Ph.D. dissertation, Brunel University London, 2015.

[26] M. Saeed, O. Maqbool, H. A. Babri, S. Z. Hassan, and S. M. Sarwar, “Software clustering techniques and the use of combined algorithm,” in *Seventh European Conference onSoftware Maintenance and Reengineering, 2003. Proceedings.* IEEE, 2003, pp. 301–306.

[27] M. Harman, S. Swift, and K. Mahdavi, “An empirical study of the robustness of two module clustering fitness functions,” in *Proceedings of the 7th annual conference on Genetic and evolutionary computation*, 2005, pp. 1029–1036.

[28] B. M. Sarwar, G. Karypis, J. Konstan, and J. Riedl, “Recommender systems for large-scale e-commerce: Scalable neighborhood formation using clustering,” in *Proceedings of the fifth international conference on computer and information technology*, vol. 1, 2002, pp. 291–324.

[29] Z. Cui, X. Xu, X. Fei, X. Cai, Y. Cao, W. Zhang, and J. Chen, “Personalized recommendation system based on collaborative filtering for iot scenarios,” *IEEE Transactions on Services Computing*, vol. 13, no. 4, pp. 685–695, 2020.

[30] H. Sun, Z. Liu, and L. Kong, “A document clustering method based on hierarchical algorithm with model clustering,” in *22nd International Conference on Advanced Information Networking and Applications-Workshops (aina workshops 2008)*. IEEE, 2008, pp. 1229–1233.

[31] A. Huang *et al.*, “Similarity measures for text document clustering,” in *Proceedings of the sixth new zealand computer science research student conference (NZCSRSC2008), Christchurch, New Zealand*, vol. 4, 2008, pp. 9–56.

[32] J. Lv, Z. Kang, X. Lu, and Z. Xu, “Pseudo-supervised deep subspace clustering,” *IEEE Transactions on Image Processing*, vol. 30, pp. 5252–5263, 2021.

[33] J. Li, H. Izakian, W. Pedrycz, and I. Jamal, “Clustering-based anomaly detection in multivariate time series data,” *Applied Soft Computing*, vol. 100, p. 106919, 2021.

[34] A. Saxena, M. Prasad, A. Gupta, N. Bharill, O. P. Patel, A. Tiwari, M. J. Er, W. Ding, and C.-T. Lin, “A review of clustering techniques and developments,” *Neurocomputing*, vol. 267, pp. 664–681, 2017.

[35] D. P. Huttenlocher, G. A. Klanderman, and W. J. Rucklidge, “Comparing images using the hausdorff distance,” *IEEE Transactions on pattern analysis and machine intelligence*, vol. 15, no. 9, pp. 850–863, 1993.

[36] M. Dubuisson and A. K. Jain, “A modified hausdorff distance for object matching,” in *Proceedings of 12th international conference on pattern recognition*, vol. 1. IEEE, 1994, pp. 566–568.

[37] T. Oates, L. Firoiu, and P. R. Cohen, “Using dynamic time warping to bootstrap hmm-based clustering of time series,” in *Sequence Learning: Paradigms, Algorithms, and Applications*. Springer, 2001, pp. 35–52.

[38] R. N. Aufmann, V. C. Barker, and R. D. Nation, *College trigonometry*. Cengage Learning, 2007.

[39] F. Szabo, *The linear algebra survival guide: illustrated with Mathematica*. Academic Press, 2015.

[40] M. Norouzi, D. J. Fleet, and R. R. Salakhutdinov, “Hamming distance metric learning,” *Advances in neural information processing systems*, vol. 25, 2012.

[41] I. Cohen, Y. Huang, J. Chen, J. Benesty, J. Benesty, J. Chen, Y. Huang, and I. Cohen, “Pearson correlation coefficient,” *Noise reduction in speech processing*, pp. 1–4, 2009.

[42] S. Santini and R. Jain, “Similarity measures,” *IEEE Transactions on pattern analysis and machine Intelligence*, vol. 21, no. 9, pp. 871–883, 1999.

[43] G. J. Székely, M. L. Rizzo, and N. K. Bakirov, “Measuring and testing dependence by correlation of distances,” *The Annals of Statistics*, vol. 35, No 6, 2769-2794, 2007.

[44] G. J. Székely and M. L. Rizzo, “The distance correlation t-test of independence in high dimension,” *Journal of Multivariate Analysis*, vol. 117, pp. 193–213, 2013.

[45] J. Macqueen, “Some methods for classification and analysis of multivariate observations,” *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, vol. 1, no. 233, pp. 281–297, 1967.

[46] S. Lloyd, “Least squares quantization in pcm,” *IEEE transactions on information theory*, vol. 28, no. 2, pp. 129–137, 1982.

[47] J. MacQuuen, “Some methods for classification and analysis of multivariate observation,” in *Proceedings of the 5th Berkley Symposium on Mathematical Statistics and Probability*, 1967, pp. 281–297.

[48] J. A. Hartigan, M. A. Wong *et al.*, “A k-means clustering algorithm,” *Applied statistics*, vol. 28, no. 1, pp. 100–108, 1979.

[49] L. Morissette and S. Chartier, “The k-means clustering technique: General considerations and implementation in mathematica,” *Tutorials in Quantitative Methods for Psychology*, vol. 9, no. 1, pp. 15–24, 2013.

[50] L. Rdusseeun and P. Kaufman, “Clustering by means of medoids,” in *Proceedings of the statistical data analysis based on the L1 norm conference, neuchatel, switzerland*, vol. 31, 1987.

[51] P. Berkhin, “Survey of clustering data mining techniques, 2002,” *Accrue Software: San Jose, CA*, 2004.

[52] M. Van der Laan, K. Pollard, and J. Bryan, “A new partitioning around medoids algorithm,” *Journal of Statistical Computation and Simulation*, vol. 73, no. 8, pp. 575–584, 2003.

[53] D. Pelleg, A. W. Moore *et al.*, “X-means: Extending k-means with efficient estimation of the number of clusters.” in *Icml*, vol. 1, 2000, pp. 727–734.

[54] A. A. Neath and J. E. Cavanaugh, “The bayesian information criterion: background, derivation, and applications,” *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 4, no. 2, pp. 199–203, 2012.

[55] L. Kaufman, P. K. Hopke, and P. J. Rousseeuw, *Using a parallel computer system for statistical resampling methods*. Technische Universiteit Delft, 1987.

[56] L. Kaufman and P. J. Rousseeuw, “Partitioning around medoids (program pam),” *Finding groups in data: an introduction to cluster analysis*, vol. 344, pp. 68–125, 1990.

[57] E. H. Ruspini, J. C. Bezdek, and J. M. Keller, “Fuzzy clustering: A historical perspective,” *IEEE Computational Intelligence Magazine*, vol. 14, no. 1, pp. 45–55, 2019.

[58] J. Nayak, B. Naik, and H. Behera, “Fuzzy c-means (fcm) clustering algorithm: a decade review from 2000 to 2014,” in *Computational Intelligence in Data Mining-Volume 2: Proceedings of the International Conference on CIDM, 20-21 December 2014*. Springer, 2015, pp. 133–149.

[59] S. Kotsiantis and P. Pintelas, “Recent advances in clustering: A brief survey,” *WSEAS Transactions on Information Science and Applications*, vol. 1, no. 1, pp. 73–81, 2004.

[60] T. Zhang, R. Ramakrishnan, and M. Livny, “Birch: an efficient data clustering method for very large databases,” *ACM sigmod record*, vol. 25, no. 2, pp. 103–114, 1996.

[61] S. Guha, R. Rastogi, and K. Shim, “Cure: An efficient clustering algorithm for large databases,” *ACM Sigmod record*, vol. 27, no. 2, pp. 73–84, 1998.

[62] G. Karypis, E.-H. Han, and V. Kumar, “Chameleon: Hierarchical clustering using dynamic modeling,” *Computer*, vol. 32, no. 8, pp. 68–75, 1999.

[63] J. Han, J. Pei, and H. Tong, *Data mining: concepts and techniques*. Morgan kaufmann, 2022.

[64] E. Schubert, J. Sander, M. Ester, H. P. Kriegel, and X. Xu, “Dbscan revisited, revisited: why and how you should (still) use dbscan,” *ACM Transactions on Database Systems (TODS)*, vol. 42, no. 3, pp. 1–21, 2017.

[65] K. Pötzelberger and H. Strasser, “Data compression by unsupervised classification,” *Department of Statistics, Vienna University of Economics and Business*, 1997.

[66] M. Charrad, N. Ghazzali, V. Boiteau, and A. Niknafs, “Nbclust: an r package for determining the relevant number of clusters in a data set,” *Journal of statistical software*, vol. 61, pp. 1–36, 2014.

[67] S. Theodoridis, A. Pikrakis, K. Koutroumbas, and D. Cavouras, *Introduction to pattern recognition: a matlab approach*. Academic Press, 2010.

[68] O. Arbelaitz, I. Gurrutxaga, J. Muguerza, J. M. Pérez, and I. Perona, “An extensive comparative study of cluster validity indices,” *Pattern recognition*, vol. 46, no. 1, pp. 243–256, 2013.

[69] J. C. Dunn, “A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters,” *Journal of Cybernetics*, 1973.

[70] D. L. Davies and D. W. Bouldin, “A cluster separation measure,” *IEEE transactions on pattern analysis and machine intelligence*, pp. 224–227, 1979.

[71] T. Calinski, “A dendrite method for cluster analysis,” in *Biometrics*, vol. 24. International Biometric Society 1441 I ST, NW, Suite 700, Washington, DC 20005-2210, 1968, p. 207.

[72] M. Halkidi, M. Vazirgiannis, and Y. Batistakis, “Quality scheme assessment in the clustering process,” in *PKDD*, vol. 1910. Citeseer, 2000, pp. 265–276.

[73] M. Halkidi, “On Clustering Validation Techniques - Springer,” pp. 107–145, 2001. [Online]. Available: https://link.springer.com/content/pdf/10.1023{\%}2FA{\%}3A1012801612483.pdfhttp://link.springer.com/article/10.1023/A:1012801612483

[74] R. Tibshirani, G. Walther, and T. Hastie, “Estimating the number of clusters in a data set via the gap statistic,” *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 63, no. 2, pp. 411–423, 2001.

[75] T. Caliński and J. Harabasz, “A dendrite method for cluster analysis,” *Communications in Statistics-theory and Methods*, vol. 3, no. 1, pp. 1–27, 1974.

[76] W. S. Sarle, “Sas technical report a-108, cubic clustering criterion,” *Cary, NC: SAS Institute Inc*, p. 56, 1983.

[77] P. Duda and E. Hart, “Pattern classification and scene analysis,” *John Wiley, New York, NY*, 1973.

[78] W. J. Krzanowski and Y. Lai, “A criterion for determining the number of groups in a data set using sum-of-squares clustering,” *Biometrics*, pp. 23–34, 1988.

[79] A. Dudek, “Cluster quality indexes for symbolic classification—an examination,” in *Advances in Data Analysis: Proceedings of the 30th Annual Conference of the Gesellschaft für Klassifikation eV, Freie Universität Berlin, March 8–10, 2006*. Springer, 2007, pp. 31–38.

[80] P. J. Rousseeuw, “Silhouettes: a graphical aid to the interpretation and validation of cluster analysis,” *Journal of computational and applied mathematics*, vol. 20, pp. 53–65, 1987.

[81] J. A. Hartigan, *Clustering algorithms*. John Wiley & Sons, Inc., 1975.

[82] L. J. Hubert and J. R. Levin, “A general statistical framework for assessing categorical clustering in free recall.” *Psychological bulletin*, vol. 83, no. 6, p. 1072, 1976.

[83] D. Ratkowsky and G. Lance, “Criterion for determining the number of groups in a classification,” *Australian Computer Journal*, 1978.

[84] A. J. Scott and M. J. Symons, “Clustering methods based on likelihood ratio criteria,” *Biometrics*, pp. 387–397, 1971.

[85] F. H. C. Marriott, “Practical problems in a method of cluster analysis,” *Biometrics*, pp. 501–514, 1971.

[86] G. H. Ball and D. J. Hall, “Isodata, a novel method of data analysis and pattern classification,” Stanford research inst Menlo Park CA, Tech. Rep., 1965.

[87] G. W. Milligan and M. C. Cooper, “An examination of procedures for determining the number of clusters in a data set,” *Psychometrika*, vol. 50, pp. 159–179, 1985.

[88] H. P. Friedman and J. Rubin, “On some invariant criteria for grouping data,” *Journal of the American Statistical Association*, vol. 62, no. 320, pp. 1159–1178, 1967.

[89] E. Beale, “Cluster analysis scientific control system,” 1969.

[90] L. E. Perez Perez, “Análisis de estabilidad de convertidores de segundo orden con la metodología de optimización de suma de polinomios cuadráticos,” *Transaction on Energy Systems and Applications*, 2020.

[91] F. J. Rohlf, “Methods of comparing classifications,” *Annual Review of Ecology and Systematics*, vol. 5, no. 1, pp. 101–113, 1974.

[92] J. O. McClain and V. R. Rao, “Clustisz: A program to test for the quality of clustering of a set of objects,” *Journal of Marketing Research*, pp. 456–460, 1975.

[93] L. Lebart, A. Morineau, and M. Piron, “Statistique exploratoire multidimensionnelle, dunod, paris, france,” 2000.

[94] L. Hubert and P. Arabie, “Comparing partitions,” *Journal of classification*, vol. 2, no. 1, pp. 193–218, 1985.

[95] H. Alqurashi and K. Barker, “Clustering ensemble algorithm based on k-medoids optimization,” *Computers, Materials & Continua*, vol. 58, no. 3, pp. 849–863, 2019.

[96] A. Strehl and J. Ghosh, “Cluster ensembles—a knowledge reuse framework for combining multiple partitions,” *Journal of machine learning research*, vol. 3, no. Dec, pp. 583–617, 2002.

[97] T. Li and C. Ding, “Solving consensus problems in cluster ensembles by argmax transformation,” *Computer Vision and Image Understanding*, vol. 106, no. 2-3, pp. 162–175, 2007.

[98] M. Meila and J. Shi, “Comparing clusterings—an information based distance,” *Journal of multivariate analysis*, vol. 98, no. 5, pp. 873–895, 2007.

[99] J. Ghosh and A. Acharya, “Cluster ensembles,” *Wiley interdisciplinary reviews: Data mining and knowledge discovery*, vol. 1, no. 4, pp. 305–315, 2011.

[100] X. Zhang and L. Wu, “Ensemble clustering in bioinformatics,” *Briefings in bioinformatics*, vol. 19, no. 2, pp. 298–314, 2018.

[101] Y. Huang and D. Zhang, “Image segmentation using local variation and global similarity optimization,” *Pattern Recognition*, vol. 44, no. 10-11, pp. 2502–2511, 2011.

[102] X. Sevillano, G. Cobo, F. Alías, and J. C. Socoró, “Feature diversity in cluster ensembles for robust document clustering,” in *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, 2006, pp. 697–698.

[103] B. J. Frey and D. Dueck, “Clustering by Passing Messages Between Data Points,” *Science*, vol. 315, no. 5814, pp. 972–976, feb 2007. [Online]. Available: http://www.sciencemag.org/cgi/doi/10.1126/science.1136800

[104] M. Meilă, “Comparing clusterings—an information based distance,” *Journal of multivariate analysis*, vol. 98, no. 5, pp. 873–895, 2007.

[105] J. Cohen, “Weighted kappa: nominal scale agreement provision for scaled disagreement or partial credit.” *Psychological bulletin*, vol. 70, no. 4, p. 213, 1968.

[106] A. J. Viera and J. M. Garrett, “Understanding interobserver agreement: the kappa statistic,” *Family Medicine*, vol. 37, no. 5, pp. 360–363, 2005.

[107] J. Li, S. Swift, and X. Liu, “The effect of cooling functions on ensemble clustering using simulated annealing,” *Intelligent Data Analysis*, vol. 14, no. 6, pp. 701–730, 2010.

[108] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to algorithms*. MIT press, 2022.

[109] T. M. Mitchell, “Machine learning,” 1997.

[110] M. Kubat and J. Kubat, *An introduction to machine learning*. Springer, 2017, vol. 2.

[111] J. Bergstra and Y. Bengio, “Random search for hyper-parameter optimization,” *Journal of Machine Learning Research*, vol. 13, pp. 281–305, 2012.

[112] J. Snoek, H. Larochelle, and R. P. Adams, “Practical bayesian optimization of machine learning algorithms,” *Advances in neural information processing systems*, vol. 25, 2012.

[113] J. H. Holland, *Adaptation in Natural and Artificial Systems*. University of Michigan Press, 1975.

[114] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[115] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. De Freitas, “Taking the human out of the loop: A review of bayesian optimization,” *Proceedings of the IEEE*, vol. 104, no. 1, pp. 148–175, 2016.

[116] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, ..., and D. Hassabis, “Human-level control through deep reinforcement learning,” *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[117] J. Nievergelt, “Exhaustive search, combinatorial optimization and enumeration: Exploring the potential of raw computing power,” in *Sofsem*. Springer, 2000, pp. 18–35.

[118] S. Edelkamp and S. Schrodl, *Heuristic search: theory and applications*. Elsevier, 2011.

[119] S. J. Russell, *Artificial intelligence a modern approach*. Pearson Education, Inc., 2010.

[120] C. Tillmann and H. Ney, “Word reordering and a dynamic programming beam search algorithm for statistical machine translation,” *Computational linguistics*, vol. 29, no. 1, pp. 97–133, 2003.

[121] R. E. Korf, “Depth-first iterative-deepening: An optimal admissible tree search,” *Artificial intelligence*, vol. 27, no. 1, pp. 97–109, 1985.

[122] L. Kallel, B. Naudts, and C. R. Reeves, “Properties of fitness functions and search landscapes,” *Theoretical aspects of evolutionary computing*, pp. 175–206, 2001.

[123] S. Kirkpatrick, C. D. Gelatt Jr, and M. P. Vecchi, “Optimization by simulated annealing,” *science*, vol. 220, no. 4598, pp. 671–680, 1983.

[124] L. Ingber, “Very fast simulated re-annealing,” *Mathematical and computer modelling*, vol. 12, no. 8, pp. 967–973, 1989.

[125] S. Bandyopadhyay, S. Saha, U. Maulik, and K. Deb, “A simulated annealing-based multiobjective optimization algorithm: Amosa,” *IEEE transactions on evolutionary computation*, vol. 12, no. 3, pp. 269–283, 2008.

[126] J. Cho and Y.-D. Kim, “A simulated annealing algorithm for resource constrained project scheduling problems,” *Journal of the Operational Research Society*, vol. 48, no. 7, pp. 736–744, 1997.

[127] A. Lambora, K. Gupta, and K. Chopra, “Genetic algorithm-a literature review,” in *2019 international conference on machine learning, big data, cloud and parallel computing (COMITCon)*. IEEE, 2019, pp. 380–384.

[128] H. John, “Adaptation in natural and artificial systems,” *Ann Arbor*, 1975.

[129] J. H. Holland, “Genetic algorithms,” *Scientific american*, vol. 267, no. 1, pp. 66–73, 1992.

[130] L. Davis, *Handbook of Genetic Algorithms*. Santa Fe Institute, USA, 1991.

[131] D. E. Goldberg, *Genetic algorithms*. pearson education India, 2013.

[132] J. Kennedy and R. Eberhart, “Particle swarm optimization,” in *Proceedings of ICNN’95-international conference on neural networks*, vol. 4. IEEE, 1995, pp. 1942–1948.

[133] X. Yao, Y. Liu, and G. Lin, “Evolutionary programming made faster,” *IEEE Transactions on Evolutionary computation*, vol. 3, no. 2, pp. 82–102, 1999.

[134] H.-G. Beyer and H.-P. Schwefel, “Evolution strategies–a comprehensive introduction,” *Natural computing*, vol. 1, pp. 3–52, 2002.

[135] P. Larrañaga and J. A. Lozano, *Estimation of distribution algorithms: A new tool for evolutionary computation*. Springer Science & Business Media, 2001, vol. 2.

[136] M. Arzoky, S. Swift, A. Tucker, and J. Cain, "A seeded search for the modularisation of sequential software versions." *J. Object Technol.*, vol. 11, no. 2, pp. 6–1, 2012.

[137] D. Dua and C. Graff, *UCI Machine Learning Repository*. University of California, Irvine, 2017. [Online]. Available: http://archive.ics.uci.edu/ml

[138] P. J. Anjula, "K-means, k-mediods clustering on uci seed dataset," https://www.kaggle.com/code/praanj/k-means-k-mediods-clustering-on-uci-seed-dataset, 2022, accessed: 2020-09-30.

[139] K. Charles, "Statlib - datasets archive, carnegie mellon university," 1989, accessed: 2019-09-30. [Online]. Available: http://lib.stat.cmu.edu/datasets/

[140] R. Hyndman, "Time series library," 2021, accessed: 2021-09-30. [Online]. Available: https://datamarket.com/data/list/?q=provider:tsdl

[141] P. Fränti and S. Sieranoja, "K-means properties on six clustering benchmark datasets," pp. 4743–4759, 2018. [Online]. Available: http://cs.uef.fi/sipu/datasets/

[142] S. Rayana, "ODDS library," 2016. [Online]. Available: http://odds.cs.stonybrook.edu

[143] L. Ding, T. J. Ley, D. E. Larson, C. A. Miller, D. C. Koboldt, J. S. Welch, J. K. Ritchey, M. A. Young, T. Lamprecht, M. D. McLellan *et al.*, "Clonal evolution in relapsed acute myeloid leukaemia revealed by whole-genome sequencing," *Nature*, vol. 481, no. 7382, pp. 506–510, 2012.

[144] A. Ultsch, "Strategies for an artificial life system to cluster high dimensional data," in *The Logic of Artificial Life: Abstracting and Synthesizing the Principles of Living Systems: Proceedings of the 6th German Workshop on Artificial Life*, 2004.

[145] Ultsch, "Maps for the visualization of high-dimensional data spaces," in *Proc. Workshop on Self organizing Maps*, 2003, pp. 225–230.

[146] M. C. Thrun and A. Ultsch, "Swarm intelligence for self-organized clustering," *Artificial Intelligence*, vol. 290, p. 103237, 2021.

[147] A. Ultsch, "Self-organizing neural networks for visualisation and classification," in *Information and classification*. Springer, 1993, pp. 307–313.

[148] C. Hennig and B. Hausdorf, "Prabclus: functions for clustering and testing of presence-absence, abundance and multilocus genetic data," 2020.

[149] I. W. Evett and J. S. Ernest, "Rule induction in forensic science. central research establishment. home office forensic science service. aldermaston," *Reading, Berkshire RG7 4PN*, 1987.

[150] N. Matake, T. Hiroyasu, M. Miki, and T. Senda, "Multiobjective clustering with automatic k-determination for large-scale data," in *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, 2007, pp. 861–868.

[151] R. W. Doran, “The gray code.” *Journal of Universal Computer Science.*, vol. 13, no. 11, pp. 1573–1597, 2007.

[152] R. E. Kass and L. Wasserman, “A reference bayesian test for nested hypotheses and its relationship to the schwarz criterion,” *Journal of the american statistical association*, vol. 90, no. 431, pp. 928–934, 1995.

[153] G. Hamerly and C. Elkan, “Learning the k in k-means,” *Advances in neural information processing systems*, vol. 16, 2003.

[154] S. Dudoit and J. Fridlyand, “A prediction-based resampling method for estimating the number of clusters in a dataset,” *Genome biology*, vol. 3, no. 7, pp. research0036–1, 2002.

[155] C. A. Sugar and G. M. James, “Finding the number of clusters in a dataset: An information-theoretic approach,” *Journal of the American Statistical Association*, vol. 98, no. 463, pp. 750–763, 2003.

[156] R. Calinski and G. Harabasz, “A dendrite method for cluster analysis,” *Communications in Statistics*, pp. 1–27, 1974.

[157] G. W. Milligan, “A monte carlo study of thirty internal criterion measures for cluster analysis,” *Psychometrika*, vol. 46, no. 2, pp. 187–199, 1981.

[158] J. Kent, J. Bibby, and K. Mardia, “Multivariate analysis (probability and mathematical statistics),” 2006.

[159] W. R. Inc., “Mathematica, Version 13.3,” champaign, IL, 2023. [Online]. Available: https://www.wolfram.com/mathematica

[160] N. Arica and F. T. Yarman-Vural, “An overview of character recognition focused on off-line handwriting,” *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 31, no. 2, pp. 216–233, 2001.

[161] D. J. Higham, G. Kalna, and M. Kibble, “Spectral clustering and its use in bioinformatics,” *Journal of computational and applied mathematics*, vol. 204, no. 1, pp. 25–37, 2007.

[162] G. Vishnuvarthanan, M. P. Rajasekaran, P. Subbaraj, and A. Vishnuvarthanan, “An unsupervised learning method with a clustering approach for tumor identification and tissue segmentation in magnetic resonance brain images,” *Applied Soft Computing*, vol. 38, pp. 190–212, 2016.

[163] Y. Zhang and Y. Zhao, “Automated clustering algorithms for classification of astronomical objects,” *Astronomy & Astrophysics*, vol. 422, no. 3, pp. 1113–1121, 2004.

[164] S. Vega-Pons and J. Ruiz-Shulcloper, “A survey of clustering ensemble algorithms,” *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 25, no. 03, pp. 337–372, 2011.

[165] D. Bertsimas and J. Tsitsiklis, “Simulated annealing,” *Statistical science*, vol. 8, no. 1, pp. 10–15, 1993.

[166] H. G. Ayad and M. S. Kamel, “Cumulative voting consensus method for partitions with variable number of clusters,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 30, no. 1, pp. 160–173, 2007.

[167] D. B. West *et al.*, *Introduction to graph theory*. Prentice hall Upper Saddle River, 2001, vol. 2.

[168] L. W. Beineke, R. J. Wilson, P. J. Cameron *et al.*, *Topics in algebraic graph theory*. Cambridge University Press, 2004, vol. 102.

[169] M. Fiedler, “Algebraic connectivity of graphs,” *Czechoslovak mathematical journal*, vol. 23, no. 2, pp. 298–305, 1973.

[170] Fiedler, “A property of eigenvectors of nonnegative symmetric matrices and its application to graph theory,” *Czechoslovak mathematical journal*, vol. 25, no. 4, pp. 619–633, 1975.

[171] R. J. Wilson, *Introduction to graph theory*. Pearson Education India, 1979.

[172] M. Wang, C. Wang, J. X. Yu, and J. Zhang, “Community detection in social networks: an in-depth benchmarking study with a procedure-oriented framework,” *Proceedings of the VLDB Endowment*, vol. 8, no. 10, pp. 998–1009, 2015.

[173] M. A. Porter, J.-P. Onnela, P. J. Mucha *et al.*, “Communities in networks,” *Notices of the AMS*, vol. 56, no. 9, pp. 1082–1097, 2009.

[174] L. Suresh, J. B. Simha, and R. Velur, “Seeding cluster centers of k-means clustering through median projection,” in *2010 International Conference on Complex, Intelligent and Software Intensive Systems*. IEEE, 2010, pp. 217–222.

[175] M. Arzoky, S. Swift, A. Tucker, and J. Cain, “Munch: An efficient modularisation strategy to assess the degree of refactoring on sequential source code checkings,” in *2011 IEEE Fourth International Conference on Software Testing, Verification and Validation Workshops*. IEEE, 2011, pp. 422–429.

[176] W. E. Arnoldi, “The principle of minimized iterations in the solution of the matrix eigenvalue problem,” *Quarterly of applied mathematics*, vol. 9, no. 1, pp. 17–29, 1951.

[177] J. Liesen and Z. Strakos, *Krylov subspace methods: principles and analysis*. Oxford University Press, 2013.

[178] R. A. Horn and C. R. Johnson, *Matrix analysis*. Cambridge university press, 2012.

[179] R. Poli, “Analysis of the publications on the applications of particle swarm optimisation,” *Journal of Artificial Evolution and Applications*, vol. 2008, pp. 1–10, 2008.

[180] F. Glover, “Tabu search—part i,” *ORSA Journal on computing*, vol. 1, no. 3, pp. 190–206, 1989.

[181] H. R. Lourenço, O. C. Martin, and T. Stützle, *Iterated local search*. Springer, 2003.