

Blending Multiple Algorithmic Granular Components: A Recipe for Clustering

by

Olayinka Idowu Oduntan

A thesis submitted to
The Faculty of Graduate Studies of
The University of Manitoba
in partial fulfillment of the requirements
of the degree of

Doctor of Philosophy

Department of Computer Science
The University of Manitoba
Winnipeg, Manitoba, Canada
December 2020

© Copyright 2020 by Olayinka Idowu Oduntan

Thesis advisor

Author

Parimala Thulasiraman

Olayinka Idowu Oduntan

Blending Multiple Algorithmic Granular Components: A Recipe for Clustering

Abstract

Trends in algorithm design have shown that hybrid algorithms, which combine or merge multiple algorithms, can create synergies to overcome the inherent limitations of the underlying individual algorithms. There are two broad types of hybridization: *collaborative* - individual algorithms tackle an instance of the problem sequentially or in parallel and exchange information accordingly while solving the problem; *integrative* - individual algorithms are dedicated to tackling different aspect(s) of the problem-solving process.

In this thesis, we propose a schema for an enhanced form of integrative hybridization that blends granular algorithmic components from multiple algorithms to derive a new singular clustering algorithm. As a case study for the proposed hybridization technique, we examine ant clustering algorithm (a swarm intelligence algorithm that is based on the natural phenomenon of brood sorting in some species of ants); highlight the strengths and weaknesses of the algorithm; and present a blend of algorithmic components from tabu search into the algorithm to improve its solution quality.

Empirical results from applying the blended algorithm to clustering benchmark datasets show improved clustering validation measures for the proposed blended hy-

brid algorithm compared to other forms of hybridization of the same underlying individual algorithms. Besides, the quality of clusters uncovered by this hybrid algorithm competes favorably with those uncovered using popular clustering algorithms such as DBSCAN and mean shift.

Finally, we show the feasibility and viability of the blended algorithm when used in a novel application of clustering to the estimation of the cost of claims for group insurance benefits.

Contents

Abstract	ii
Table of Contents	vi
List of Figures	vii
List of Tables	viii
Acknowledgments	ix
Dedication	x
1 Introduction	1
1.1 Research contributions	4
2 Literature Review	7
2.1 Clustering Problem	7
2.2 Types of Clustering Algorithms	9
2.3 Traditional vs. Modern Clustering Algorithms	14
2.3.1 Traditional Clustering Algorithms	14
2.3.1.1 Hierarchical clustering algorithms	15
2.3.1.2 Partitional clustering algorithms	16
2.3.1.3 Model-based clustering algorithms	18
2.3.1.4 Density-based clustering algorithms	18
2.3.2 Modern/Metaheuristic-based Clustering Algorithms	19
2.3.2.1 Evolutionary algorithm based clustering	21
2.3.2.2 Simulated annealing based clustering	21
2.3.2.3 Tabu search based clustering	22
2.3.2.4 Ant brood sorting based clustering (ant clustering algorithm - ACA)	23
2.4 Hybrid Clustering Algorithms	25
2.4.1 Collaborative hybrid clustering algorithm (CHCA)	26
2.4.2 Integrative hybrid clustering algorithms - (IHCA)	29
3 Blended Integrative Hybrid Algorithm (BIHA)	32
3.1 Blended Integrative Hybridization (BIH)	33

3.1.1	Identification phase	34
3.1.2	Selection phase	35
3.1.3	Blending phase	35
3.1.3.1	Blending by replacement	36
3.1.3.2	Blending by fusion	37
3.1.4	BIH for N metaheuristic algorithms	38
3.2	Schema for Blending Algorithmic Components	38
4	Blending Algorithmic Components for Clustering	41
4.1	Steps for blending clustering algorithms	41
4.2	ACA (Ant Clustering Algorithm)	44
4.2.1	Motivation for using ACA	50
4.2.2	Limitations and solutions	51
4.2.3	Proposed blending of ACA algorithmic components	54
4.3	Tabu Search Algorithm	55
4.4	Blended Integrative Hybrid Clustering Algorithm - ACA and Tabu Search	59
4.5	Collaborative Hybrid Clustering Algorithm - ACA and Tabu Search	65
5	Experiments and Results	68
5.1	Benchmark Datasets	69
5.2	Measuring Data Similarity	70
5.3	Cluster Validation Measures	72
5.4	Experiment Setup	75
5.5	Results Discussion	78
6	Industry Application - Clustering Group Insurance Dataset	82
6.1	Pricing Insurance Products (Ratemaking)	85
6.1.1	The pure premium method	87
6.1.2	The loss ratio method	89
6.1.3	The experience rating method (using credibility theory)	90
6.2	Group Insurance Ratemaking	91
6.2.1	Ratemaking for pooled group benefits	93
6.2.2	Ratemaking for experience-rated group benefits	94
6.3	Enhanced Experience-Rated Benefits Ratemaking	95
6.4	Clustering Group Insurance Datasets	100
6.4.1	Group insurance datasets	101
6.4.2	Experimental setup (clustering group insurance dataset)	102
6.4.3	Results and analysis (clustering group insurance dataset)	103

7	Discussion, Conclusion and Future Work	107
7.1	Discussion and Conclusion	107
7.2	Future Work	110
	Bibliography	127

List of Figures

2.1	Overview of clustering algorithm	10
2.2	Categories of optimization algorithms [Das et al., 2009]	20
2.3	Depiction of collaborative hybrid clustering algorithm (CHCA)	27
2.4	Depiction of integrative hybrid clustering algorithm (IHCA)	30
3.1	Depiction of blended integrative hybridization (BIH) strategy	34
3.2	Depiction of blending by replacement in BIH	37
3.3	Depiction of blending by fusion in BIH	38
3.4	Depiction of BIH for blending N metaheuristics (M_A, M_B, \dots, M_N)	39
4.1	Depiction of blended integrative hybrid clustering algorithm (BIHCA)	43
4.2	Depiction of 2D grid ($m \times m$) in ACA	45
4.3	Depiction of 2D grid before and after ACA clustering	47
4.4	Flowchart of basic ant clustering algorithm	51
4.5	Flowchart of basic tabu search clustering algorithm	58
4.6	Depiction of blended ACA and tabu search clustering algorithm	63
4.7	Flowchart of blended ACA and tabu search clustering algorithm	65
4.8	Depiction of collaborative ACA and tabu search clustering algorithm	66
5.1	Average similarity measures for iris data	72
5.2	Average similarity measures for wheat seed data	73
5.3	Average similarity measures for wine data	73

List of Tables

5.1	Clustering results for Iris dataset	76
5.2	Clustering results for Wheat Seed dataset	77
5.3	Clustering results for Wine dataset	77
5.5	Comparing BIHCA, Mean shift and DBSCAN (Iris dataset)	77
5.4	Average number of random walk steps by ants	78
5.6	Comparing BIHCA, Mean shift and DBSCAN (Wheat Seed dataset)	78
5.7	Comparing BIHCA, Mean shift and DBSCAN (Wine dataset)	78
6.1	Clustering result on group insurance dataset	104
6.2	Average estimate of claims for clusters (DBSCAN)	105
6.3	Average estimate of claims for clusters (mean shift)	105
6.4	Average estimate of claims for clusters (BIHCA)	106

Acknowledgments

I am thankful for all the diverse support and help I received in the course of undertaking this research and thesis. I am particularly grateful to those who have directly or indirectly played a major role in the completion of this thesis.

I am particularly grateful to my advisor, Dr. Parimala Thulasiraman, for her invaluable guidance, support, coaching and encouragement, especially during the challenging times. I am also thankful to my research committee members (Dr. Peter Graham and Dr. Ken Ferens) for their constructive feedback and suggestions during this research. I greatly appreciate the administrative help I received from Lynne Hermiston (Computer Science) and Rosemary Visevic (Graduate Studies).

I am also using this opportunity to express gratitude to the following extended and adopted family and friends for believing in me and lending a hand when most needed: my siblings and their families - Femi, Timothy, Bamidele, Lanre and Adebowale Oduntan; family friends - Dr. and Mrs Makinde, Pastor and Mrs. Okunnu; and my bosses at Wawanesa Insurance - Guy Delaurier and Demetri Karavas.

Special thanks to my wonderful wife (Elizabeth Oduntan) and my incredible sons (Opemipo and Ayomipo Oduntan) for being a great source of strength and courage to me all through this academic journey.

My utmost gratitude is to God - the giver of life, good health and a sound mind.

This thesis is dedicated to my parents and parents-in-law: late father (Emmanuel Oduntan), late mother (Sarah Oduntan), late mother-in-law (Esther Aderinto) and father-in-law (Amos Aderinto)

Chapter 1

Introduction

The availability of data in large volumes and varying complexities is growing rapidly. Researchers in data science and related disciplines are innovating tools, techniques and concepts for deriving intelligence from data. This entails capturing, maintaining, processing, analyzing and communicating data [Kelleher and Tierney, 2018]. Clustering is one of the unsupervised machine learning techniques for processing data for knowledge discovery purposes.

The basic conceptual process of grouping data objects within a collection into categories based on underlying similarity or dissimilarity traits is known as clustering. Clustering aims to find smaller, more homogeneous groups from a large heterogeneous collection of data objects [Aldenderfer et al., 1984] [Ng, 2003]. The grouping is such that data objects within the same groups are more similar to one another than data objects in other groups. The smaller groups, that result from clustering a given collection, are known as *clusters*. Clusters can be referred to as either *hard* or *fuzzy* [Duda et al., 1973]. Hard clusters consist of data objects that belong exclusively to

only one cluster; and fuzzy clusters consist of data objects that belong to multiple clusters with varying degree of belongingness. In this research, the focus is primarily on hard clusters.

Clustering has rooted practical applications in science, technology and business: species taxonomy in biology; astrological grouping of planets; market segmentation in business and financial analysis; image analysis in computer vision; recommendation system in on-line search (e.g. Google), commerce (e.g. amazon), entertainment (e.g. Netflix) and social network (e.g. Facebook); etc. This broad application spectrum, as well as a growing availability of data in large volumes and in different variations, continue to make clustering relevant for active research.

The broad practical applications of clustering has also resulted in the creation of diverse algorithms to tackle the computational problem. These algorithms include traditional clustering algorithms that can be categorized into groups such as partitional, hierarchical, density-based, model-based and grid-based [Aldenderfer et al., 1984; Manning et al., 2008; Rokach and Maimon, 2005; Xu and Tian, 2015]. The applicability of traditional clustering algorithms often depend largely on the nature of the data collection to be clustered and the underlying rationale for clustering. Besides, many of these algorithms are not practically applicable for clustering large data collections - clustering has been proven to be NP-hard [Welch, 1982]. Hasan and Ramakrishnan [2011] summarize the shortcomings of traditional clustering algorithms to include: dependency on initial state, convergence to local optima, global solutions of large problem instances cannot be found with reasonable amount of computation effort, etc.

Popular metaheuristics and nature-inspired algorithms used in solving difficult optimization problems have been adapted by many researchers to tackle clustering. Examples of such include simulated annealing [Selim and Alsultan, 1991; Lukashin and Fuchs, 2001; Duczmal and Assunção, 2004], particle swarm optimization [der Merwe and Engelbrecht, 2003; Cohen and de Castro, 2006], genetic algorithm [Murthy and Chowdhury, 1996; Krishna and Murty, 1999; Maulik and Bandyopadhyay, 2000; Chiou and Lan, 2001; Lu et al., 2004], tabu search [Al-Sultan, 1995; Al-Sultan and Fedjki, 1997; Sung and Jin, 2000; Ng and Wong, 2002] and ant clustering algorithm (ACA) [Lumer and Faieta, 1994; Handl et al., 2003; Oduntan et al., 2014; Liu et al., 2016; Qasem et al., 2018], among others. However, these algorithms are also known to have established strengths and weaknesses that make no single algorithm suitable for all the varied instances of the clustering problem.

Emerging trends in algorithm design for such techniques have shown that hybrid algorithms, which combine or merge multiple algorithms, can create synergies to overcome the inherent limitations of the underlying individual algorithms [Hasan and Ramakrishnan, 2011]. Traditional hybridization can be categorized into collaborative and integrative techniques [Blum et al., 2011]. For the former, individual algorithms tackle an instance of a given computational problem sequentially or in parallel and exchange information accordingly during the process. And for the latter, individual algorithms are mutually dedicated to tackling different aspects of the problem-solving process. A review of existing literature shows a large collection of hybrid clustering algorithms that are designed based on these hybrid approaches [Hasan and Ramakrishnan, 2011].

While these traditional hybrid approaches have potential to leverage the high-level strengths of many popular metaheuristics, they often do not fully harness the strengths of low-level tactics or techniques (algorithmic components) of these individual algorithms. In this research, we explore a different approach to hybridizing algorithms to tackle the clustering problem. We propose an in-depth form of integrative hybridization that blends granular algorithmic components (i.e low-level aspects of an algorithm that fulfil specific task, strategy or concept) from multiple individual algorithms to derive a new singular algorithm to tackle the clustering problem. We submit that a low-level algorithmic hybridization can enable the design of enhanced hybrid algorithms that blend tactical strengths of the individual algorithms to enable improved clustering results.

1.1 Research contributions

This dissertation presents the following core contributions to existing knowledge in the design of hybrid clustering algorithms.

1. **Schema for blended hybrid clustering algorithm.** We present a schema for blending granular algorithmic components from multiple individual algorithms to create a hybrid clustering algorithm. The schema presents a structured method on how to *blend* two or more metaheuristic algorithms into a single algorithm to tackle clustering problem. The proposed blending method can also be used to design hybrid algorithms for tackling other practical optimization problems, thereby opening up new research paradigms in the design of hybrid algorithms.

2. **Case study - blending of ant clustering algorithm (ACA) and tabu search (TS).** As a case study for the proposed blended hybridization schema, this research also presents a hybrid clustering algorithm that blends algorithmic components from ant based clustering algorithm (ACA) and tabu search (TS). This research also establishes the viability of the proposed hybrid clustering algorithm using empirical inferences derived from using the algorithm to uncover clusters in benchmark datasets - the Iris, Wine and Wheat datasets. The clusters derived by the blended algorithm show better quality when compared with those derived by a collaborative hybrid version with the same underlying algorithms.
3. **Similarity judgement varies with dataset.** Clustering algorithms have an underlying mechanism that defines a measure of similarity between pairs of data objects or the fitness of a given partition of data objects based on a measure of the objects' similarity. We investigated using different similarity measures (distance-based and kernel-based) for the proposed blended algorithm and the impact on the clustering results. The proposed algorithm performs better with kernel-based similarity measures.
4. **Application to insurance industry dataset.** The use of knowledge discovery techniques, such as clustering, is gaining attention in the insurance industry. Although, many of the industry rating mechanisms (i.e. the underlying technique for costing an insured risk) are still based on statistical models and industry estimates, we applied the proposed clustering algorithm to uncover implicit knowledge in group insurance datasets that can be used in making more

informed decisions when predicting rating values for new clients without prior insurance claim history.

Chapter 2

Literature Review

This chapter presents an overview of existing research work on clustering. The first section describes a formulation of clustering as a computational problem; section 2.2 presents an overview of what a clustering algorithm entails and the underlying strategies that differentiate clustering algorithms; section 2.3 describes some popular categories of clustering algorithms; and lastly, section 2.4 describes existing research efforts on the hybridization of algorithms to tackle clustering. This last section further highlights the limitations of traditional hybridization strategies in the design of hybrid clustering algorithms as the basis for the newly proposed approach in this research.

2.1 Clustering Problem

Clustering is a classic computational problem that is still relevant in modern research. Despite the enormous amount of work that have been done by different research communities on clustering, there still appears to be no single universal formal

definition of the clustering problem [Everitt et al., 2011], since there can be varying context and areas of interest in resolving the problem. This lack of a universal formulation is part of what makes clustering a challenging and difficult computational problem [Das et al., 2009]. The clustering problem can be formulated differently depending on the technique being used to uncover the clusters.

Clustering generally involves categorizing a given collection of n data objects into $k \ll n$ groups such that a measure of similarity between data objects $S(o_q, o_r)$ is significantly greater for data objects within the same group, than for data objects that belong to different groups. Consider a collection O , of n data objects such that $O = \{o_1, o_2, \dots, o_n\}$; a general goal of hard clustering is to create or find a collection of subsets $C = \{c_1, c_2, c_3, \dots, c_k\}$ of O such that:

- $\bigcup_{i=1}^k c_i = O$
- $c_i \neq \emptyset; \forall c_i \in C$
- $c_i \cap c_j = \emptyset; \forall i \neq j$
- $k \in \mathbb{Z}^+; 1 < k \ll n$

The data objects within the same subset c_i are as similar as possible, while data objects within any two different subsets (c_i, c_j) are as dissimilar as possible - i.e. $S(o_x^{c_i}, o_y^{c_i}) \gg S(o_x^{c_i}, o_z^{c_j})$. The subsets $c_1, c_2, c_3, \dots, c_k$ are the non-overlapping clusters of the data collection O . The underlying measure of similarity $S(o_x, o_y)$ can vary depending on the nature of the data objects and the technique used to uncover the clusters.

A review of existing literature indicates that there are two broad instances of the clustering problem based on whether the value of k is known *a priori* or not. For the first instance, the value of k is a required input parameter and algorithms that are designed to tackle this instance assume k as a part of the required inputs or derive k as a precursor to the clustering process.

For the second instance, the value of k is part of the outcome of the clustering process - the clusters are first uncovered and the value of k is typically implied from the cardinality of the final set of clusters uncovered. This instance is applicable to practical clustering problems where obtaining *a priori* number of clusters k is either impractical or too expensive. In this research, we investigate tackling this category of clustering problem instance using hybridization strategy that harnesses the low-level strengths of the underlying individual clustering algorithms.

2.2 Types of Clustering Algorithms

A clustering algorithm typically takes, as input, a heterogeneous collection of data objects and produces, as output, a set of clusters each consisting of subset of the original collection that are significantly homogeneous. Some clustering algorithms require additional meta information (e.g. the number of clusters k) about the heterogeneous collection as part of the input. A clustering algorithm, using an underlying similarity judgement as a decision guide, ultimately aims to directly or indirectly search all possible combinations of data objects from the collection to find the set of clusters that most fit the rationale for clustering. Figure 2.1 depicts an overview of a clustering algorithm.

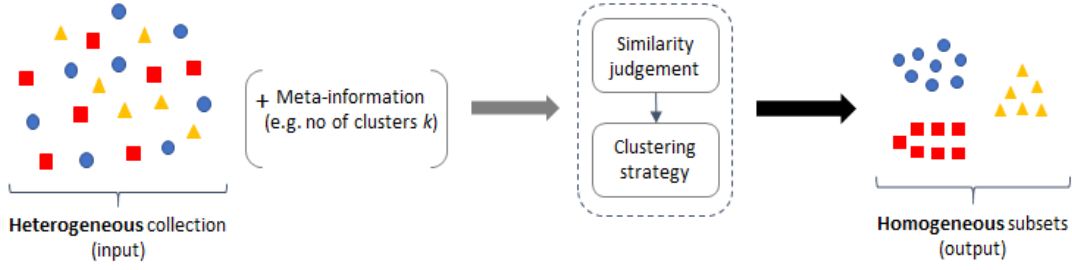


Figure 2.1: Overview of clustering algorithm

Similar to the depiction in Figure 2.1, Jain et al. [1999] identifies three aspects of a clustering algorithm to include: representation of the input data objects; definition of a similarity measure; and a clustering strategy for uncovering the clusters. A representation of the input data objects involves using a uniform data structure to capture information about each data object. Typically, an input data object is represented as a data point in a multidimensional space or as a feature vector where each feature denotes an observable or measurable attribute of the data object. Given a set of data objects as input $O = \{o_1, o_2, \dots, o_n\}$, each data object o_i can be represented as an attribute vector $\mathbf{o}_i = [o_{i1} \ o_{i2} \ \dots \ o_{iq}]$ in \mathbb{R}^q where q is the number of attributes for each attribute vector or the dimension of each vector representing a data object.

Similarity judgement refers to a measure of the underlying trait between the data objects or partitions of the data objects collection that is used as the basis of deciding which data objects belong to the same cluster and vice versa. This can be based on a measure of distance between pair of data objects $D(o_i, o_j)$ which translates to dissimilarity between the objects or based on a direct measure of similarity between data objects $S(o_i, o_j)$, where a pair of data objects o_i and o_j are considered similar if $S(o_i, o_j)$ is significantly large or if $D(o_i, o_j)$ is significantly small. For data objects rep-

resented using numerical attribute data, examples of popular dissimilarity measures used in clustering algorithms include the following [Shirkhorshidi et al., 2015]:

1. Euclidean Distance

$$d_{euc}(o_i, o_j) = \sqrt{\sum_{v=1}^q (o_{iv} - o_{jv})^2}$$

2. Average Distance

$$d_{avg}(o_i, o_j) = \sqrt{\frac{1}{q} \sum_{v=1}^q (o_{iv} - o_{jv})^2}$$

3. Manhattan Distance

$$d_{mh}(o_i, o_j) = \sum_{v=1}^q |o_{iv} - o_{jv}|$$

4. Minkowski Distance

$$d_{mk}(o_i, o_j) = \left(\sum_{v=1}^q |o_{iv} - o_{jv}|^p \right)^{1/p}$$

where p , an integer, is the order of the distance. Manhattan and Euclidean distances are specific instances of the Minkowski distance for when p is 1 and 2 respectively.

Cosine similarity and its variants (notably, angular cosine similarity) are popular similarity measures used in clustering categorical data objects [Kotu and Deshpande,

2019]. Different kernel-based similarity measures have also been used in clustering algorithms [Filippone et al., 2008]. Examples of popular kernel functions used in estimating similarity measures include Gaussian, Laplacian, Sigmoid and Chi-Squared. The definition of these similarity measures are as follows [Hofmann et al., 2008]:

1. Cosine similarity

$$S_{cs}(o_i, o_j) = \frac{\sum_{v=1}^q o_{iv} o_{jv}}{\sqrt{\sum_{v=1}^q (o_{iv})^2} \sqrt{\sum_{v=1}^q (o_{jv})^2}}$$

2. Gaussian Kernel

$$S_g(o_i, o_j) = \exp\left(-\frac{\|o_i - o_j\|^2}{2\sigma^2}\right)$$

3. Laplacian kernel

$$S_l(o_i, o_j) = \exp\left(-\frac{\|o_i - o_j\|}{\sigma}\right)$$

4. Sigmoid kernel (hyperbolic tangent kernel)

$$S_s(o_i, o_j) = \tanh(\gamma o_i^T o_j + c)$$

where c is the intercept and γ refers to the slope

5. Chi-Squared kernel

$$S_{cq}(o_i, o_j) = 1 - \sum_{v=1}^q \frac{(o_{iv} - o_{jv})^2}{\frac{1}{2}(o_{iv} + o_{jv})}$$

The crux of a clustering algorithm is the strategy through which the algorithm uncovers (constructs or searches for) the clusters. Clustering algorithms are usually categorized based on the strategy being used to uncover clusters. According to Hartigan [1975], clustering algorithms use strategies such as switching, joining, splitting, adding and searching to uncover clusters.

Algorithms that are based on switching start with an initial set of clusters, and new partitions are derived by switching data object(s) from one cluster to another on the basis that such switching results in better clusters. These algorithms typically terminate when switching data objects between the clusters no longer result into better clusters.

Algorithms that use the joining strategy typically start with an initial set of clusters each consisting of a single data object. Subsequently, clusters that contain data objects that are similar are merged together progressively (i.e. joined) until a single cluster containing all the data objects is obtained. A level within the hierarchy of coarsened clusters is eventually set as the output of the clustering. Algorithms that use a splitting strategy start with a single cluster consisting of the entire data collection and progressively divide into sub-clusters until there is one data object per cluster. Similarly, a level within the hierarchy of coarsened clusters is eventually set as the output of the clustering.

The adding strategy in clustering algorithms involves having a starting structure

(a partition or a tree) and adding each data object to the structure in turn.

Clustering algorithms that are based on a search strategy adapt optimization techniques to find clusters. These algorithms typically partition the given input data collection into k subsets such that each data object belongs to exactly one subset. Each possible partition is associated with an error function (cost function) and the goal of the clustering algorithm is to find the partition that minimizes the error function. This clustering strategy results in the adaptation of many popular optimization algorithms for clustering purposes.

2.3 Traditional vs. Modern Clustering Algorithms

Xu and Tian [2015] group clustering algorithms as traditional or modern. Algorithms that belong to classic categories such as hierarchical, partitional, density-based, model-based and distribution-based are considered traditional. Algorithms that are based on metaheuristics (optimization techniques) such as swarm intelligence, hybrid algorithms, and so on are considered modern. It is important to note that these two broad groups are not mutually exclusive as some algorithms can fit into either of the categories. The remaining sections in this chapter describe popular algorithms in these two groups.

2.3.1 Traditional Clustering Algorithms

Traditional clustering algorithms can be broadly grouped into hierarchical, partitional, model-based, density-based and grid-based [Aldenderfer et al., 1984; Rokach and Maimon, 2005; Manning et al., 2008; Xu and Tian, 2015].

2.3.1.1 Hierarchical clustering algorithms

Hierarchical clustering algorithms generally group a collection of data objects into clusters by recursively sub-dividing or coarsening instances of the input data collection in a top-down or bottom-up manner, respectively. These algorithms are either agglomerative or divisive. Agglomerative algorithms perform bottom-up recursive coarsening - i.e. start with each data object as a cluster and at each step merge the closest pair of clusters into one. Divisive algorithms perform top-down recursive partitioning - i.e. start with the entire collection of data objects as a single cluster and at each step split a cluster until arriving at only singleton clusters of individual data objects.

Two general strengths of hierarchical algorithms are: the number of clusters to be uncovered is not a required input *a priori*; and the quality of the clustering process is not dependent on an initial solution [Jain et al., 1999]. One of the general shortcomings of hierarchical algorithms is the computational expense - runtime complexity is $O(n^2 \log n)$ and the space complexity is $O(n^2)$ [Das et al., 2009]. Consequently, these algorithms, in their original forms, do not scale well over very large input data collections. Further, hierarchical clustering methods generally use a static coarsening or partitioning mechanism [Das et al., 2009] - the algorithms typically lack inherent ability to back-track after a partitioning step has been executed. This implies that these algorithms cannot move a data object that has been assigned to one cluster to another during the clustering process.

Popular examples of hierarchical methods include BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies) [Zhang et al., 1996], ROCK (RObust Clus-

tering using linKs) [Guha et al., 2000] and CHAMELON [Karypis et al., 1999].

2.3.1.2 Partitional clustering algorithms

Given S (a collection of n data objects), k (the predefined number of clusters) and a starting initial solution (i.e. an initial set of k clusters), a partitional clustering algorithm iteratively partitions S into k clusters by moving data objects from one cluster to another using an exploratory strategy while optimizing the objective partitioning criterion or criteria. An objective criterion typically minimizes a measure of dissimilarity between objects within the same cluster while maximizing the same measure between objects within different clusters.

Partitional clustering algorithms generally begin with an initial solution (a set of k clusters) and iteratively explore the set of possible k clusters (i.e. the solution space) to find the set that ‘best’ fits the objective criterion. The best set of clusters can be found by examining all possible k clusters, which is generally not practical. Consequently, partitioning algorithms typically adopt greedy mechanisms to find the ‘best’ k clusters by iteratively shifting data objects from one cluster to another during the clustering process.

The k -means [MacQueen et al., 1967] and k -medoids [Kaufman and Rousseeuw, 2009] algorithms are popular classical partitioning algorithms for clustering data objects. The k -means algorithm (also referred to as centroid-based technique [Tan et al., 2006]) starts with an initial random or predetermined set of k data objects as the cluster centers or means; and the remaining data objects are distributed into clusters by assigning each to the cluster with the nearest center. Subsequently, the means

of the current clusters are recalculated and the data objects are redistributed based on their proximity to the latest cluster means. The recalculation and redistribution processes continue iteratively until a convergence condition is satisfied. Common convergence conditions include: reaching a local optimum (i.e. when recalculation of cluster means and redistributing data objects no longer results in better clustering objective criterion value); or reaching a predefined number of iterations.

The ease of implementing and interpreting k -means algorithm are reasons why the algorithm is a popular option for clustering data objects [Tan et al., 2006]. Also, the k -means algorithm has linear complexity over the number of input data objects [Tan et al., 2006], which makes it a popular option for clustering large datasets.

Despite the popularity of the k -means algorithm, it has some inherent limitations and deficiencies stated as follows:

1. The value of k is a required input and has to be known *a priori*.
2. The set of final clusters derived by the algorithm is sensitive to the initial cluster centers selected at the beginning of the clustering process.
3. The algorithm is typically only applicable for clustering data objects having numeric attributes.
4. The underlying structure of the clusters have to be convex in nature (i.e. a line between any two points within a cluster lies within the cluster), as the algorithm, in its basic form, cannot uncover clusters with arbitrary shapes.
5. The algorithm is sensitive to the presence of outliers in the data collection.

The k -medoids algorithm (also known as Partitioning Around Medoids - PAM [Brusco and Köhn, 2008]) is similar in concept to the k -means except that it represents each cluster by the most centric data object within the cluster instead of an implicit mean value of the cluster. Also, k -medoid algorithms are typically applicable to categorical statistical data object collections.

Although many different variations of the k -means and k -medoids algorithms have been created to mitigate some of the deficiencies mentioned above, some of the core issues remain as they are fundamental to this category of algorithms generally. Particularly, the requirement for a pre-defined value of k and the intolerance to outliers remain as impediments to their general use.

2.3.1.3 Model-based clustering algorithms

Model-based clustering algorithms aim to optimize how well the given data object collection fits some mathematical model. Besides uncovering the underlying clusters in the data, these algorithms provide additional information that describe each cluster. Decision tree clustering methods and neural network clustering methods are popular examples of model-based clustering algorithms [Rokach and Maimon, 2005].

2.3.1.4 Density-based clustering algorithms

Density-based clustering algorithms are based on the assumption that data objects in a given dataset are derived from an underlying density function or a mix of density functions (probability distributions) [Banfield and Raftery, 1993]. The aim of these algorithms is to determine the underlying density function(s) of the data objects and use the same, as well as mechanisms designed around the density function(s), as a

basis for grouping the data objects into clusters.

These algorithms are typically applicable for clustering data collections consisting of arbitrarily shaped clusters and they can also handle datasets with noise [Rokach and Maimon, 2005]. A common drawback of these algorithms is their dependence on initial parameters Das et al. [2009] which are typically challenging to determine. Further, for high-dimensional or large number of data objects, the runtime complexity for the basic forms of these algorithms is quadratic [Jang and Jiang, 2019]. Two popular examples of density-based clustering algorithms are DBSCAN (Density-Based Spatial Clustering for Application with Noise) [Ester et al., 1996b] and mean shift [Hyrien and Baran, 2016]. DBSCAN, originally proposed by Ester et al. [1996a], is based on the idea that given a data space of data objects, the density of points within a cluster is considerably higher than outside of a cluster; and clusters are separated by regions of considerably low density of points (*noise*). The algorithm aims to identify the clusters (high density regions) and the arrears of noise (low density regions) that separate the clusters from one another. Mean shift is a non-parametric clustering algorithm that defines clusters based on the modal regions of a density function

Other examples of density-based clustering algorithms are AUTOCLASS [Cheeseman et al., 1988], MCLUST [Fraley and Raftery, 1998], SNOB [Wallace and Dowe, 1994] and DENCLUE [Hinneburg and Gabriel, 2007].

2.3.2 Modern/Metaheuristic-based Clustering Algorithms

Heuristic algorithms tackle difficult optimization problems by proffering *good* solution within practical computational time in place of a guaranteed *best* solution

that can require non-practical amount of computation resource to find. Sörensen and Glover [2013] describe a metaheuristic as a high-level set of guidelines for developing heuristic optimization algorithms. Metaheuristic-based clustering algorithms, also referred to as optimization-based clustering algorithms, consists of metaheuristic algorithms designed to tackle clustering as a computational problem [Das et al., 2009]. These algorithms aim to find the *best* suited set of clusters that satisfies an objective rationale for clustering.

Considering that many optimization techniques in the literature, Das et al. [2009] present a structured grouping of these techniques. Figure 2.2 depicts a representation of different categories of optimization algorithms.

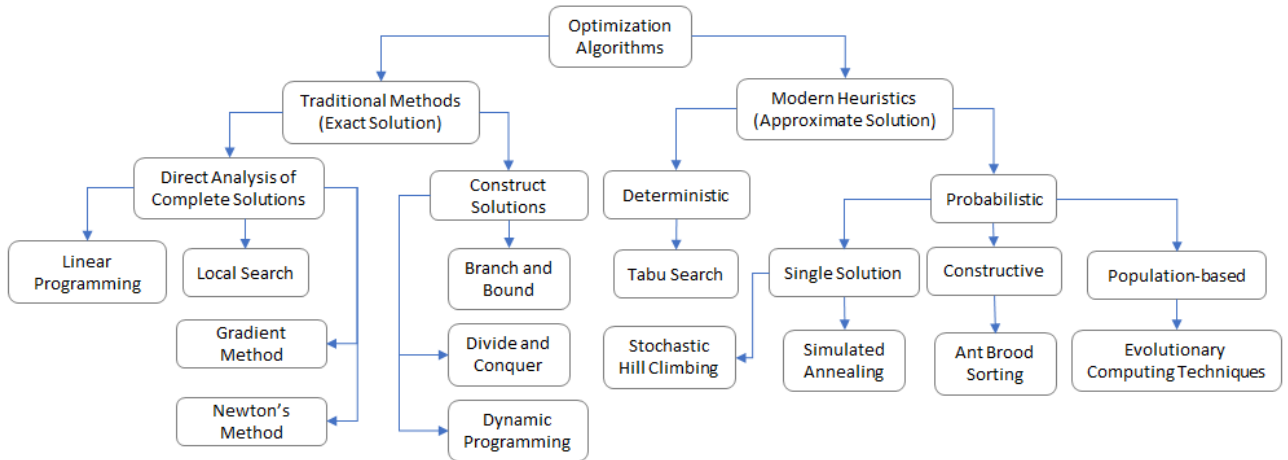


Figure 2.2: Categories of optimization algorithms [Das et al., 2009]

A survey of the existing literature reveals that clustering algorithms that are based on optimization algorithms have been created mostly using techniques or a combination of techniques that are grouped under the ‘*modern heuristics*’ category [Das et al., 2009]. These techniques have been applied to many practical optimization

problems thereby making them candidates for the clustering problem. The following sub-sections describe some clustering algorithms that are based on popular optimization methods.

2.3.2.1 Evolutionary algorithm based clustering

Evolutionary algorithms are inherently parallel stochastic techniques that adapt Darwinian evolution's concepts of natural selection to solving computational optimization problems. They typically involve iterative steps that select feasible solutions from a population (set of encoded solutions) using a combination of operations (such as crossover and mutation) and fitness measure(s). Examples of evolutionary algorithms used in clustering include genetic algorithm [Raghavan and Birchard, 1979], evolution strategy [Lee and Antonsson, 2000] and evolutionary programming [Sarkar et al., 1997].

2.3.2.2 Simulated annealing based clustering

Simulated annealing is a probabilistic global optimization technique that is based on the process of annealing in metal works (i.e. a process that alters the physical properties of metal by heating and gradually cooling the temperature to change the internal structure). This optimization technique seeks a global optimum of an optimization problem by mimicking the gradual cooling process of annealing. The technique starts with an initial random solution and an initial high system temperature and seeks a new solution within a neighborhood by looping over a set of feasible solutions until a good enough solution is found or a stopping criteria is reached. The technique decides on accepting the neighborhood solution based on a combination of

the quality difference between the current solution and the new neighborhood solution and the temperature set for the iteration. Selim and Al-Sultan [Selim and Alsultan, 1991] describe a clustering algorithm that is based on simulated annealing.

2.3.2.3 Tabu search based clustering

Tabu search, created by Fred Glover [Glover and Laguna, 2013], is a metaheuristic search method that guides local or neighborhood search heuristic beyond local optimum using a dynamic neighborhood structure and a combination of exploratory strategies in a deterministic manner.

A basic local neighborhood search heuristic seeks the optimal solution to an optimization problem by starting with an initial solution x_i (pre-determined or randomly chosen) as the current solution; constructs a neighborhood $N(x_i)$ of x_i (i.e. a set consisting of solutions that are similar to x_i except for a few attributes or solution components); searches for a solution x'_i within $N(x_i)$ having the best fit for the objective function and assigning x'_i as the new current solution; and then proceeds iteratively until no better solution can be reached or a stopping criteria has been satisfied. This local search technique easily gets stuck within a local optimum as its final solution when exploring *plateaued* solution spaces or solution spaces having multiple local optima.

Tabu search complements the local search heuristic with a dynamic neighborhood structure and an adaptive memory system, both inspired by the concept of *tabu* (maintaining a dynamic list of solutions or solution components that are forbidden from being examined during neighborhood exploration used as a means of escaping

local optimum traps).

Tabu search, in its basic and enhanced forms, has been applied to different optimization problems such as the popular traveling sales man problem. Tabu search has also been applied to the clustering problem. Al-Sultan [1995] created a tabu search based clustering algorithm and compared the performance of the algorithm with simulated annealing clustering technique and the popular k -means clustering technique. The findings show that tabu search is a promising algorithm for clustering nominal datasets and that using tabu search is a more promising technique for clustering in comparison to techniques that are based on simulated annealing and the classic k -means technique.

2.3.2.4 Ant brood sorting based clustering (ant clustering algorithm - ACA)

Brood sorting is a natural phenomenon exhibited by some ant species (*Leptothorax Unifasciatus*) whereby the working ants sort brood items (eggs, microlarvae, larvae, prepupae, pupae) according to their sizes and brooding needs using a distributed positive feedback mechanism. Some other ant species demonstrate similar ability in organizing corpses in their nest into piles to clean up their nest - this is termed cemetery organization [Bonabeau et al., 1999].

Nature-inspired computing models have been developed to emulate this distributed positive feedback mechanism for solving complex real-life problems. Deneubourg et al. [1991] proposed two related models (a clustering model and a brood sorting model) that capture the corpse clustering and brood sorting phenomena in ants. The core

idea of the models entails ants (i.e. agents) wandering randomly within a definite spatial terrain, picking up isolated items and dropping them off at other locations where more related or similar items are present. Although the systematic pick-up/drop-off actions of the individual ants are primarily based on intelligence local to each ant, collectively, these actions translate into a distributed system-level intelligence that can be applied to difficult real-life problems such as clustering. To apply the ant brooding concept to data analysis, Lumer and Faieta [1994] proposed projecting the attribute space of the data objects onto a lower dimension (typically 2) such that the attribute distance can be used as a measure of the underlying common traits amongst the data objects. This nature-inspired clustering technique is commonly referred to, in literature, as ant clustering algorithm (ACA).

Different clustering techniques have been developed, based on the *ant brood sorting* phenomenon, and applied to solving practical clustering problems. Oduntan et al. [2014] applied a simple *ant brood sorting* technique to cluster financial time series of stock prices to achieve diversification in portfolio management. Liu et al. [2016] also developed a parallel version of the *ant brood sorting* technique for clustering financial time series data on the cloud. Qasem et al. [2018], Qasem and Thulasiraman [2019], Qasem [2018] developed and evaluated a semi-supervised constrained ant brood clustering algorithm and tested the feasibility of this algorithm against other well-known machine learning techniques for the constrained sentiment analysis problem [Agarwal and Mittal, 2013]. The results of these applications show *ant brood sorting* as a promising technique for clustering data collections with intrinsic natural correlation amongst the data objects.

Notable strengths of *ant brood sorting* are: it is one of the few metaheuristics-based clustering techniques that does not require the number of clusters k to be known *a priori*; inherently distributive with potential for parallelization; and can be applied to clustering in multiple problem domains. However, common deficiencies of this technique are: getting trapped in a local optimum; highly probabilistic; and convergence can be slow. But some of these deficiencies can be addressed by finding a means of combining the capabilities of some other optimization techniques (e.g. tabu search, simulated annealing) that can find global optimal solution more 'quickly'. One of the goals of this research is to explore combining the *ant brood sorting* technique with tabu search using a hybrid mechanism that harnesses the strengths of the individual algorithms to tackle clustering.

2.4 Hybrid Clustering Algorithms

Hybrid metaheuristic algorithms (or hybrid algorithms, for short) combine different metaheuristic techniques with other optimization techniques to better resolve difficult optimization problems [Blum et al., 2011]. This type of algorithm, when applied to clustering problems, is referred to as hybrid clustering algorithms. Trends in algorithm design have shown that hybrid algorithms, which combine or merge multiple algorithms, can create synergies to overcome the inherent limitations of the underlying individual algorithms [Hasan and Ramakrishnan, 2011].

Blum et al. [2011] describe two broad categories of hybrid algorithms: *collaborative* and *integrative*. Collaborative hybrids combine multiple instances of standalone algorithms running sequentially (or in parallel) on the same problem instance and

exchanging information about states, sub-problems or solutions, while doing so. In this form of hybridization, the component standalone algorithms only exchange information, but are not integral parts of each other. The Integrative hybrids, on the other hand, combine multiple algorithms such that one algorithm is integrated into another, typically as a subordinate.

The following sub-sections highlight existing hybrid clustering algorithms and group them into two high-level categories based on the underlying hybridization techniques.

2.4.1 Collaborative hybrid clustering algorithm (CHCA)

This category of hybrid clustering algorithms combine multiple standalone algorithms together such that each algorithm tackles the same clustering problem instance sequentially or in parallel, partially or as a whole, while exchanging information about the problem, the solution (partial or complete) and/or environment. The underlying standalone algorithms are typically loosely combined and connected primarily for the purpose of exchanging information.

CHCA typically consists of multiple phases with a different standalone algorithm tackling the given problem instance within each phase sequentially or in parallel. Usually, a CHCA has at least two phases - a starting phase and a final phase; and there can also be intermediary phase(s). In a sequential paradigm, the starting phase algorithm takes as input the given data collection ($S = \{o_1, o_2, \dots, o_n\}$) and optionally the value of k (if known *a priori*); and generates a starting solution ($C'_s = \{c'_{s1}, c'_{s2}, \dots, c'_{sk}\}$) that can be partial or sub-optimal. In the subsequent intermediate phase (where

applicable), the standalone algorithm consumes the original inputs and any additionally derived information pieces from the previous phase (e.g. $C'_s = \{c'_{s1}, c'_{s2}, \dots, c'_{sk}\}$, k); and generates new intermediary solution $C'_i = \{c'_{i1}, c'_{i2}, \dots, c'_{ik}\}$. Eventually, the final phase follows a similar process to generate the final solution $C = \{c_1, c_2, \dots, c_k\}$. The parallel paradigm typically involves more complicated cooperative interactions between the standalone algorithms within the phases. Figure 2.3 depicts a high-level representation of CHCA.

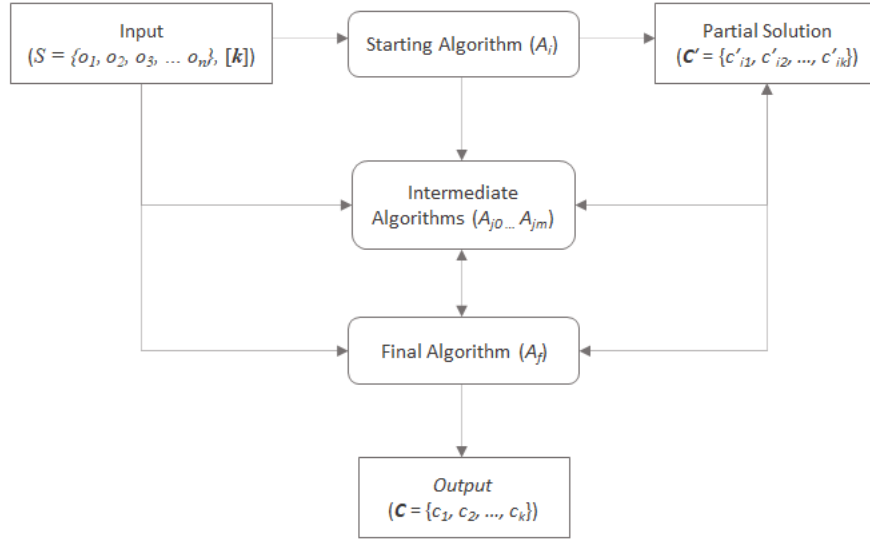


Figure 2.3: Depiction of collaborative hybrid clustering algorithm (CHCA)

Monmarché et al. [1999] proposed a pioneering CHCA that combines ant clustering with k-means. The ant clustering algorithm (which is the starting algorithm) generates an initial set of clusters and derives the value of k . The k-means algorithm (which is the final algorithm) uses the derived k as part of the required input and refines the initial solution by reducing the classification error, thereby improving the homogeneity of the clusters. CHCAs with similar concepts have been proposed -

these hybrids typically use a starting algorithm to generate an initial solution and a derived value for k as intermediary outputs; and use a final algorithm that consumes the intermediary outputs as input and generate a refined final solution. Kanade and Hall [2007] and Wang et al. [2009] proposed a CHCAs that combine ant clustering algorithm and fuzzy c-means. The ant clustering algorithm generates an initial set of clusters and a derived k as outputs; while the fuzzy c-means uses these outputs to find a global *optimal* set of clusters as a final solution. Similarly, Chiu and Lin [2007] proposed a CHCA that uncovers an initial set of clusters and the value of k using ant clustering algorithm; and use an immune system optimization algorithm to find a final global set of clusters afterwards. He and Tan [2012] propose a two-stage hybrid clustering algorithm that combines ant clustering with a genetic algorithm. In the first stage, ant clustering finds an initial solution and the value of k ; and the genetic algorithm then finds a globally optimal set of clusters as the final solution in the second stage. Rahman and Islam [2014] also proposed a genetic algorithm (GA) based CHCA that uses a novel GA technique as a starting algorithm that finds the value of k and an initial set of clusters that is subsequently used as an initial solution by the final k-means algorithm.

For the CHCA implementation in this research (Chapter 4), ACA is the starting phase algorithm and tabu search is the final phase algorithm. The ACA first tackles the clustering problem and produces as output the value of k and an initial set of clusters that is further refined using tabu search to reduce the classification error and produce a final set of clusters as solution. The intent is to have ACA partially solve the clustering problem by producing a sub-optimal set of clusters and an implied

value for k . These outputs are then used as part of the input for the tabu search, which seeks to find a global *optimal* set of clusters.

A notable advantage of CHCA is the simplicity of the design and implementation, since the component standalone algorithms typically retain their original form and only exchange solution and information at different phases. However, retaining the originality of the standalone algorithms equally implies that the inherent limitation(s) of the standalone algorithms are retained in the resulting CHCA algorithm.

2.4.2 Integrative hybrid clustering algorithms - (IHCA)

For this category of the hybrid algorithms, one or more subordinate algorithms are embedded within a main algorithm to tackle the clustering problem. Typically, there is a primary algorithm (i.e. the main algorithm) that handles the overall construction or overall iterative search for the global clustering solution; and there are one or more secondary algorithms (subordinate algorithm(s)) that handle dedicated tasks and return the outcome of such tasks to the primary algorithm. Figure 2.4 depicts a high-level representation of IHCA. The primary algorithm takes on a single instance of the given problem, delegates some sub-processes to the subordinate algorithm(s) and coordinates the outcome of those sub-processes to uncover the final set of clusters.

For IHCA wherein the primary algorithm is based on iterative search methods, the primary algorithm typically handles the input/output processing, performs strategic exploration processes that uncover regions of the search space with potentially good solutions and diversifies the search process to prevent getting stuck in a local optimum. The secondary algorithm(s), on the other hand, typically perform sub-processes that

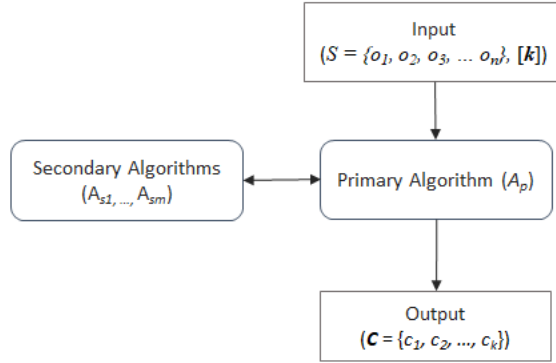


Figure 2.4: Depiction of integrative hybrid clustering algorithm (IHCA)

exploit good solution regions to uncover the best known solution for the given problem in that region and pass on such solutions onto the primary algorithm for subsequent processing. The secondary algorithm serves as an underlying local optimizer that intensifies search effort in the regions identified by the primary algorithm.

Güngör and Ünler [2007] proposed an IHCA that integrates k-harmonic into simulated annealing (SA). In their hybrid algorithm, SA serves as the primary global search algorithm and k-harmonic is used as the local optimizer. In [Güngör and Ünler, 2008], these authors also proposed integrating k-harmonic into tabu search. Similarly, tabu search is the primary global search algorithm and k-harmonic is used as the local optimizer.

Niknam et al. [2009] proposed integration of particle swarm optimization (PSO) and simulated annealing to tackle clustering. The latter is used to diversify the search for the optimal set of clusters thereby preventing the PSO from getting trapped in a local optimum. Esmin and Matwin [2012] also proposed a hybrid clustering algorithm that integrates PSO and genetic algorithm (GA). Similarly, GA mutation techniques are used to enable the PSO search to escape from local optima.

Tvrđik and Křivý [2015] also proposed an IHCA. In their proposed hybrid clustering algorithm, they integrate k-means as a local optimizer within a differential evolution algorithm. Jiang and Xie [2018] also propose memetic hybrid clustering algorithm that integrates simulated annealing as the local search technique (secondary algorithm) within a differential evolutionary clustering algorithm.

Many of the hybrid clustering algorithms in the literature fall under the IHCA category, since the hybridization strategy follows a popular optimization paradigm - population-based metaheuristic as the primary algorithm for finding the global optimal solution; and a single-solution metaheuristic as the secondary algorithm for finding local optimal solution. However, for this category of optimization clustering algorithms, having a predefined value of k (the number of clusters) is typically a requirement. This limits the applicability of these algorithms to real-life clustering problems where the value of k is typically not known *a priori*.

For the case study in this research, we investigate integrating ACA and tabu search with the former as the primary algorithm and the latter as the secondary algorithm. However, we take a deeper look into the hybridization strategy of these algorithms by considering the *low-level algorithmic components* of the algorithms to harness the individual low-level strengths as much as possible. The proposed algorithm *does not* follow the typical IHCA approach wherein the primary algorithm coordinates the global search process while the secondary algorithm acts as a local optimizer. Instead, we propose a blend of the low-level algorithmic components of both algorithms such that the strength of one component is used to address the limitations of another. We present the details of the proposed hybridization in the following section.

Chapter 3

Blended Integrative Hybrid

Algorithm (BIHA)

Considering the inherent limitations of collaborative and integrative hybridization strategies for clustering algorithms as described in Section 2.4, creating an enhanced integrative hybridization strategy that blends low-level algorithmic components from multiple algorithms should provide a basis for designing hybrid metaheuristic algorithms with synergies that can address inherent limitations.

A review of many metaheuristic algorithms shows that these algorithms typically consist of low-level algorithmic components that are pieced together to make full-fledged algorithms. For instance, genetic algorithms consist of components such as crossover, mutation and a strategy for selecting the fittest offspring [Yang, 2013]. Ant brood sorting consists of components such as the pick-up and drop-off operations, a spatial grid for the artificial ants to perform these operations, and a stochastic neighborhood exploration technique [Lumer and Faieta, 1994]. Tabu search consists

of components such as an adaptive memory mechanism (*tabu list*), dynamic neighborhood structure, diversification and intensification strategies [Glover and Laguna, 2013]. The strengths of these metaheuristic algorithms can be attributable to the underlying algorithmic components as well as how these components are pieced together. Ironically, the limitations of these algorithms can also be attributable to the inherent limitations of some of the algorithmic components.

This thesis presents a deeper integrative hybridization strategy that provides a basis for *blending low-level algorithmic components from multiple algorithms* to create more effective hybrid algorithms. The newly proposed hybridization strategy is referred to as *blended integrative hybridization* (BIH). A clustering algorithm designed based on this proposed strategy is referred to as a blended integrative hybrid clustering algorithm (BIHCA).

3.1 Blended Integrative Hybridization (BIH)

This section describes the proposed blended integrative hybridization strategy. To hybridize two or more metaheuristic algorithms (M_A, M_B, \dots) that can be used to tackle a given computational problem (such as clustering), the BIH strategy involves:

- *Identification phase*: identifying the underlying low-level algorithmic components of the individual algorithms $((m_{a1}, m_{a2}, \dots), (m_{b1}, m_{b2}, \dots), \dots)$
- *Selection phase*: selecting the algorithmic components to be blended together
- *Blending phase*: blending the selected algorithmic components to create a hybrid algorithm such that the inherent limitations of the individual algorithms

are reduced or eliminated.

Considering the simplified case of blending two metaheuristic algorithms (M_A, M_B), Figure 3.1 depicts the different constituent phases of the BIH strategy: *identification*, *selection* and *blending*.

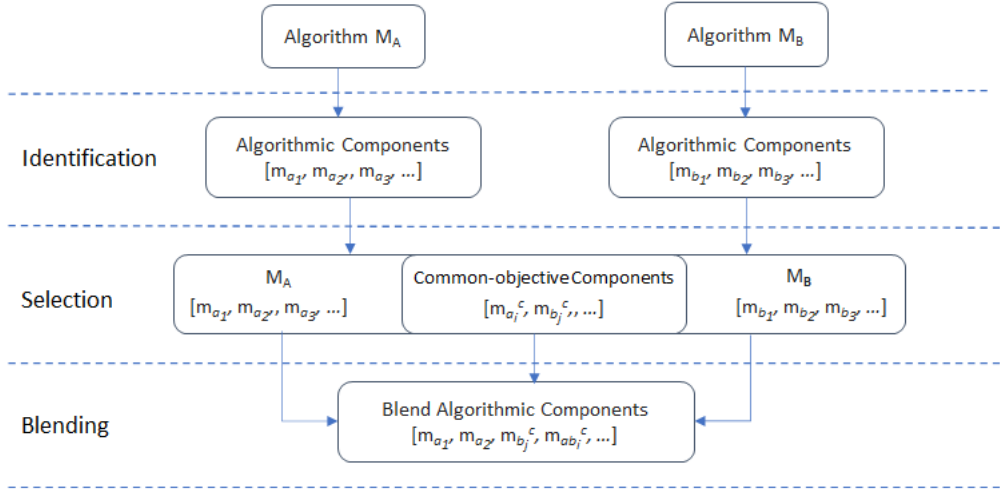


Figure 3.1: Depiction of blended integrative hybridization (BIH) strategy

3.1.1 Identification phase

Given two or more metaheuristic algorithms (M_A, M_B, \dots), the goal of the identification phase is to uncover the low-level algorithmic components of each algorithm. Considering that many metaheuristic algorithms consist of an ensemble of tactical and/or strategic pieces, this phase involves mapping these tactical pieces to corresponding low-level algorithmic components for each individual algorithm. The result of this phase is a pool of algorithmic components $((m_{a1}, m_{a2}, \dots), (m_{b1}, m_{b2}, \dots), \dots)$ derived from the individual standalone algorithms.

3.1.2 Selection phase

In the selection phase, the algorithmic components uncovered from the previous phase $((m_{a1}, m_{a2}, \dots), (m_{b1}, m_{b2}, \dots), \dots)$ are examined in the context of the underlying rationale for hybridization. One of the goals of this phase is to uncover the algorithmic components that are attributable to inherent limitations in an individual algorithm, as well as algorithmic components (with similar objectives) from other individual algorithm(s) that can be leveraged to overcome such limitations. For instance, a highly effective mutation operator from a differential evolution algorithm can be selected for leverage in place of a basic mutation operator in a genetic algorithm. We refer to algorithmic components in this category as *common-objective* components. In Figure 3.1, algorithmic components selected for having similar objectives are denoted as $(m_{ai}^c, m_{bj}^c, \dots)$.

3.1.3 Blending phase

The blending phase defines the overall structure of the resulting hybrid algorithm and how the algorithmic components can be fused together to create a new hybrid algorithm. The concept of primary and secondary algorithms described in section 2.4.2 in the basic integrative hybridization can be applicable in this phase. In this case, the primary algorithm still retains the overall structure of the resulting hybrid algorithm, while the secondary algorithm, instead of functioning as a local optimizer, provides low-level algorithmic components that can be blended with *common-objective* components in the primary algorithm. The choice of which algorithm is the primary algorithm or the secondary algorithm is based on the nature of the problem being

tackled and the overall objective of the hybrid algorithm. For instance, to blend two or more metaheuristic algorithm to tackle the clustering problem instance where the value of k (number of clusters) is not known *a priori*, the choice of the primary algorithm would have to not require k as an input parameter.

The algorithmic components that have been selected for having a common objective will either replace one another (blending by replacement) or be fused into a composite component (blending by fusion) to achieve a target synergy in the resulting hybrid algorithm. The choice of replacing or fusing is also dependent on the context of the algorithmic components under consideration. The replacement and fusion blending options are described further in the following subsections.

3.1.3.1 Blending by replacement

Blending by replacement implies using *common-objective* algorithmic components from one algorithm to replace equivalent algorithmic components in another algorithm. This form of blending is applicable when the components from one algorithm can be replaced exactly with components from another algorithm - i.e., when the components from the algorithms to be blended perform exactly the same task or sub-task. For an example that involves blending differential evolutionary algorithm and genetic algorithm, a component shared by both techniques is "mutation". Therefore, as suggested earlier, we can blend an enhanced mutation operator from the differential evolutionary algorithm to replace the basic mutation operator in the genetic algorithm.

Figure 3.2 depicts blending of algorithmic components by replacement - the algo-

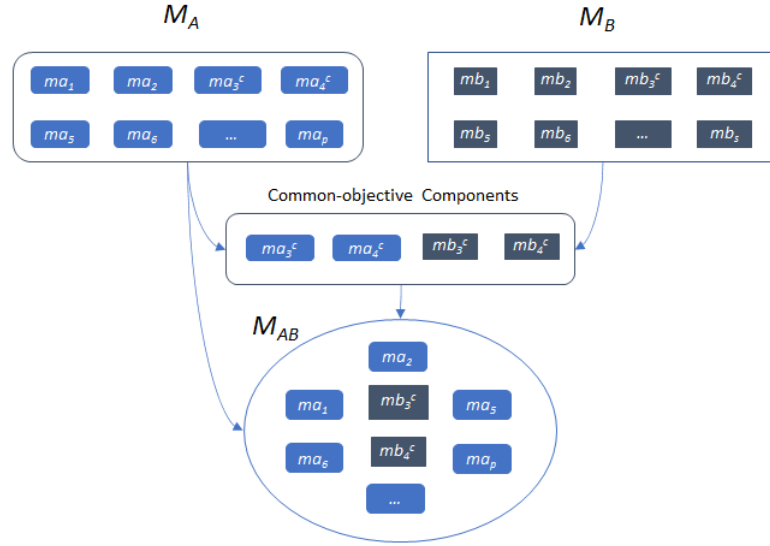


Figure 3.2: Depiction of blending by replacement in BIH

rithmic components m_{b3}^c and m_{b4}^c are used to replace corresponding components m_{a3}^c and m_{a4}^c in the resulting hybrid algorithm.

3.1.3.2 Blending by fusion

For some other form of blending scenarios, the *common-objective* algorithmic components are not outright replaceable; blending by fusion is applicable in such scenarios. The corresponding *common-objective* components are fused together to form composite algorithmic components in the resulting hybrid algorithm.

As depicted in Figure 3.3, the algorithmic components m_{b3}^c and m_{b4}^c are fused together with corresponding components m_{a3}^c and m_{a4}^c to create composite components m_{ab3}^c and m_{ab4}^c in the resulting hybrid algorithm.

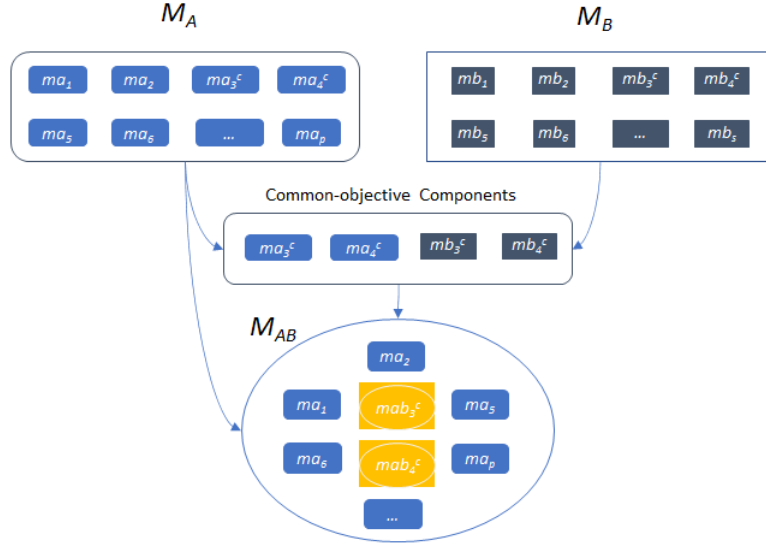


Figure 3.3: Depiction of blending by fusion in BIH

3.1.4 BIH for N metaheuristic algorithms

For the general case of blending N metaheuristics (M_A, M_B, \dots, M_N , the same 3-step processes described above would apply. In particular, the selection and blending phases will be broader with more common-objective components to select and blend together across the multiple algorithms. Figure 3.4 depicts a high-level representation of the 3-phase processes for blending N metaheuristic algorithms.

3.2 Schema for Blending Algorithmic Components

To blend algorithmic components from multiple metaheuristic algorithms together effectively, we observe that it is imperative to properly understand the computational problem to be solved and to have an in-depth understanding of the workings of the individual algorithms. Having specific rules for all possible blends of algorithms is

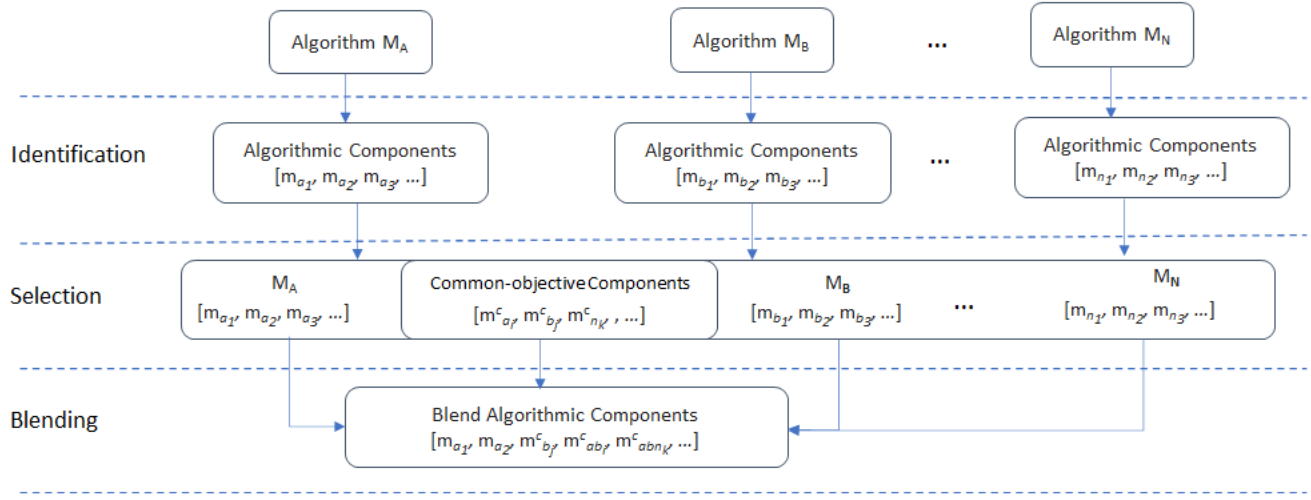


Figure 3.4: Depiction of BIH for blending N metaheuristics (M_A, M_B, \dots, M_N)

equally not practical.

However, we propose the following set of generic guidelines (constituting a sort of schema) for blending algorithmic components from multiple standalone algorithms to create blended hybrid versions:

1. Uncover the concrete algorithmic components that make up the individual algorithms. This involves reviewing each algorithm and identifying portions with concrete or coherent tactic, strategy or concept. This guide connotes the identification phase described in the BIH strategy above (Section 3.1.4).
2. Select the *common-objective* algorithmic components from the individual algorithms and determine the appropriate blending option(s) based on the target synergy goal.
3. Depending on the hybridization goal, determine which of the individual algorithms is suitable as the *primary* algorithm and which is suitable as the *sec-*

ondary algorithm in the design of the resulting hybrid algorithm.

4. Design and create an ensemble that consists of the algorithmic components from the standalone algorithms with particular attention to leveraging the effective version of the *common-objective* algorithmic components and the hybridization goal.

To retain the flexibility of the blended hybrid algorithm that can be designed based on this set of guidelines, the schema does not restrict how the algorithmic components are to be blended together to produce the resulting hybrid algorithm. However, we identify two criteria that are required to blend two or more metaheuristic algorithms together:

- Each algorithm to be blended has to be divisible into concrete low-level algorithmic components, and many metaheuristics satisfy this criterion.
- The algorithms have to share one or more *common-objective* algorithmic components and an in-depth understanding of the individual algorithms is needed to establish this criterion.

The next chapter describes the viability of the blending technique to the clustering problem.

Chapter 4

Blending Algorithmic Components for Clustering

In this chapter, we present an adaptation of the blended integration hybridization (BIH) strategy to tackle the clustering problem. We follow the steps proposed in the schema for blending algorithmic components (Section 3.2) to create a technique for blending two clustering algorithms: ant clustering algorithm and tabu search. For comparative purposes, towards the end of this chapter, we also present a collaborative hybrid clustering algorithm that also combines ACA and tabu search.

4.1 Steps for blending clustering algorithms

1. The first step, being the *identification phase*, involves uncovering discrete tactics, strategies and/or concepts within the standalone algorithms that can be mapped to algorithmic components. These components fulfil specific tasks or

sub-tasks within the overall clustering process (Section 2.2) in the standalone version of the algorithms. Examples of these tasks or sub-tasks include representation of the data objects; finding the value of k , if it is not known *a priori*; a definition of the solution search space for sets of feasible clusters; a search strategy for exploring and exploiting the solution search space; and evaluation of the uncovered clusters to gauge the quality of the clusters and to make decisions on which set of clusters to choose as the final solution.

2. The second step is to *select the common algorithmic components* that perform similar clustering tasks or sub-tasks for each standalone algorithm. For instance, many metaheuristic clustering algorithms have a search strategy that defines how the algorithm explores and exploits the solution space in search of the best set of clusters. The goal of this step is to select the common algorithmic components that can be harnessed to create a more effective hybrid clustering algorithm.
3. The third step involves *examining the choice of primary versus secondary algorithm* within the resulting hybrid algorithm. The choice is based on the underlying rationale for clustering and the formulation of the clustering problem being tackled. For instance, if the problem formulation is such that the number of clusters (k) is a required input, a standalone clustering algorithm that requires the value to be predefined can be used as a primary algorithm in such context - the contrary scenario applies as well.
4. The last step is to *create a hybrid clustering algorithm* from an ensemble of

the algorithmic components using the primary algorithm as the foundation. This involves blending the common algorithmic components by either replacing inherently limiting components with more effective equivalents or fusing the components to create a composite version. Also, the choice of primary versus secondary algorithm within the resulting hybrid algorithm is based on the underlying rationale for clustering and the formulation of the clustering problem being tackled.

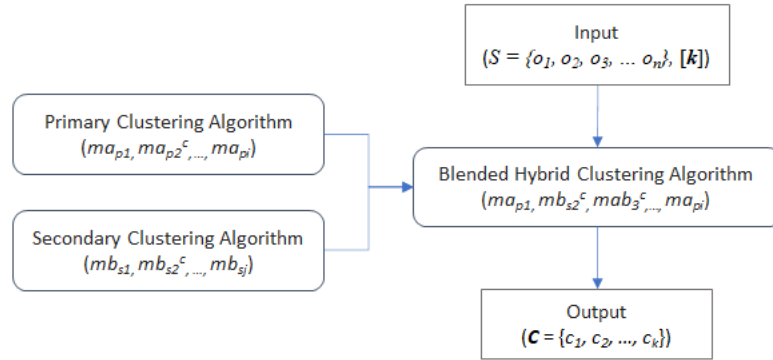


Figure 4.1: Depiction of blended integrative hybrid clustering algorithm (BIHCA)

Figure 4.1 depicts a representation of the design process to create a hybrid clustering algorithm using the BIH strategy.

In the remainder of this chapter, we propose a hybrid clustering algorithm that blends two standalone algorithms used in the literature for clustering: *ant brood sorting* (also referred to as ant clustering algorithm - ACA) [Lumer and Faieta, 1994; Boryczka, 2008] and *tabu search* [Al-Sultan and Fedjki, 1997; Glover and Laguna, 2013]. The resulting blended hybrid algorithm is used to showcase the *viability* of the proposed blending schema (Section 3.2). For comparative analysis purposes, we

also present a Collaborative Hybrid Clustering Algorithm (CHCA) (Section 2.4.1) version that combines the same underlying standalone algorithms and compare the performance of both versions in the subsequent chapter.

One of the rationales for choosing these algorithms is: ACA is one of the few meta-heuristic clustering algorithm that can tackle a clustering problem instance wherein the value of k is not known *a priori*; but the algorithm also has inherent limitations that can be overcome by leveraging strategies and tactics that are native to tabu search. Besides, as the analysis in the sections below reveal, tabu search shares some common-objective algorithmic components with ACA. So, in the following sections, we present an overview of each of the standalone clustering algorithms and the hybrid versions afterwards.

4.2 ACA (Ant Clustering Algorithm)

Deneubourg et al. [1991] started the pioneering work on using artificial ants to cluster objects. In [Deneubourg et al., 1991], commonly referred to as the basic model, a population of robots (artificial ants) are able to stack objects together by imitating natural heuristics exhibited by some ant species (*Leptothorax Unifasciatus*) in sorting their broods according to sizes and brooding needs within the ants' colony. The robots move randomly within a definite spatial terrain, picking up and dropping off objects based on the number of similar surrounding objects.

Lumer and Faieta [1994] extended and applied the basic model to tackle the problem of clustering numerical data and this is often referred to as the ant clustering algorithm (ACA). Using Lumer and Faieta's technique, a given collection of data

objects $S = (o_1, o_2, o_3, \dots, o_n)$ are scattered randomly onto a \mathbb{Z}^2 dimensional space (typically a 2D grid $m \times m$); a set of artificial ants are also randomly placed on the grid; and the artificial ants move around within the grid, picking up and dropping off data objects based on the density of similar objects within an ant's locality (neighborhood).

The 2D grid depicts the spatial terrain of the ants' colony (Figure 4.2). Each point on the grid, referred to as a site (r_{xy}), indicates a coordinate where an artificial ant or data object can be located at any given point in time. Each site r_{xy} has a predefined neighborhood $N(r_{xy})_{s \times s}$ that aids the ant on the site in assessing the similarity of a given data object within the ant's locality. The value of the dimension for the grid (m) varies in proportion to the number of data objects to be clustered. Boryczka [2008] recommends $m = \sqrt{10 * n}$, where n is the number of data objects to be clustered.

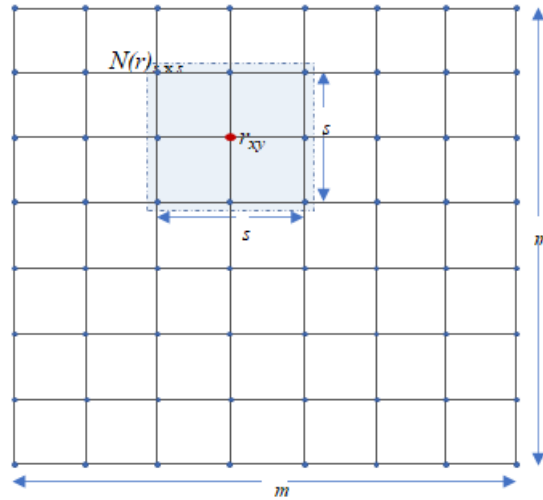


Figure 4.2: Depiction of 2D grid ($m \times m$) in ACA

An artificial ant (a_w) on the 2D grid corresponds to an ant within the terrain of its colony; and the data objects, with underlying similarity measures, correspond

to the broods that have inherently similar brooding needs. Within the grid, the artificial ants perform decentralized *pick-up* and *drop-off* operations that are based on intelligence local to the ants.

A *pick-up* operation involves an *unladen* ant a_w^r (i.e an ant, at site r , that is without an assigned data object at the given point in time) getting assigned to a data object (o_i) from the site r where the data object is currently located on the grid. This operation results in a site r_{xy}^i that is loaded with data object o_i becoming a free site r_{xy} , and an *unladen* ant becoming *laden* a_w^{ri} (i.e an ant with assigned data object o_i at the given point in time).

Conversely, a *drop-off* operation involves a *laden* ant offloading an assigned data object onto a free site. The *drop-off* operation results in a *laden* ant becoming *unladen* and a free site becoming a loaded site. Over many iterations, the pick-up and drop-off operations of the artificial ants evolve into a decentralized feedback mechanism that eventually results in having similar data objects assigned to contiguous sites on the grid (i.e. clusters), separated by sites that are free of data objects.

Figure 4.3 depicts the state of the 2D grid before and after the artificial ants create clusters of the data objects on the grid.

As the ants move around within the grid, the decision to pick up and/or drop off a data object o_i is based on computed *pick-up probability* P_p and *drop-off probability* P_d respectively. The probabilities are derived from a similarity measure $f(o_i)$ that determines the extent of similarity or dissimilarity of o_i to the other data objects within its neighborhood. Lumer and Faieta [1994] define the value of P_p and P_d as follows:

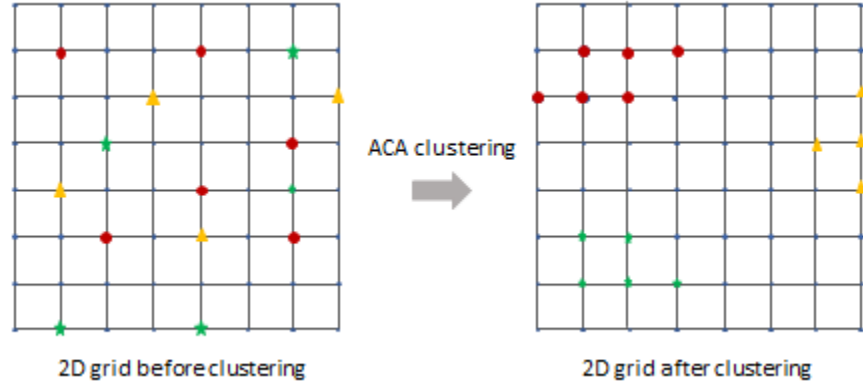


Figure 4.3: Depiction of 2D grid before and after ACA clustering

$$P_p(o_i) = \left(\frac{k_1}{k_1 + f(o_i)} \right)^2 \quad (4.1)$$

$$P_d(o_i) = \begin{cases} 2f(o_i), & \text{if } f(o_i) < k_2 \\ 1, & \text{if } f(o_i) \geq k_2 \end{cases} \quad (4.2)$$

$f(o_i)$ is a measure of the similarity between o_i and the rest of the data objects within its neighborhood; and k_1 and k_2 are constants. The definition of f and the choice of values of k_1 and k_2 need to be such that when $f \ll k_1$, P_p is close to 1. This implies that the chances of picking an isolated data object is high as the value of P_p gets closer to 1, since the value of f is closer to 0. Conversely, when $f \gg k_1$, P_p is close to 0. This implies that the chances of picking a data object at a dense location (i.e. a cluster) is low. Similarly, when $f \ll k_2$, P_d gets close to 0, which implies that the chances of dropping a data object in an isolated neighborhood is low. Also, when $f \gg k_2$, P_d gets close to 1, which implies that the chances of dropping a data object

at a dense location (i.e. a cluster) is high.

For a given data object o_i at site r with a neighborhood denoted as $N_{s \times s}(r)$, Lumer and Faieta [1994] define the value of f as follows:

$$f(o_i) = \begin{cases} \frac{1}{s^2} \sum_{o_j \in N_{s \times s}(r)} [1 - \frac{d(o_i, o_j)}{\alpha}], & \text{if } f \geq 0 \\ 0, & \text{otherwise} \end{cases} \quad (4.3)$$

where $d(o_i, o_j)$ is a measure of the dissimilarity distance between o_i and another object (o_j) within its neighborhood; $\frac{1}{s^2}$ is a neighborhood scaling parameter; and α is a parameter used in scaling the dissimilarities within the neighborhood.

The definition of f , which depends largely on the value of $d(o_i, o_j)$, can vary depending on the nature of the data objects, as well as the rationale for clustering. Lumer and Faieta [1994] use the Euclidean distance (vector attribute distance) as a measure of the dissimilarity distance between numeric data objects (o_i, o_j). Similarly, Kuntz et al. [1997] measure dissimilarity distance using weighted symmetric difference for graph data objects. These dissimilarity distance measures are defined as follows:

For numerical data objects where each data object is represented as a q -dimensional attribute vector o_i , Euclidean distance or vector attributes distance is given as:

$$d(o_i, o_j) = \sqrt{\sum_{v=1}^q |o_{iv} - o_{jv}|^2} \quad (4.4)$$

For graph data, weighted symmetric difference for graph vertices is given as:

[Kuntz et al., 1997]):

$$d(v_i, v_j) = \frac{|D(\rho(v_i), \rho(v_j))|}{|\rho(v_i)| + |\rho(v_j)|} \quad (4.5)$$

where $\rho(v_i)$ denotes the set of vertices adjacent to v_i , including v_i ; and $D(\rho(v_i), \rho(v_j))$ is the symmetric difference of the two sets $\rho(v_i)$ and $\rho(v_j)$ (i.e. $D(A, B) = (A \cup B) - (A \cap B)$).

Besides dissimilarity distance measures, the similarity of objects can also be measured based on kernel or affinity models [Shawe-Taylor et al., 2004; Zhang et al., 2007] that estimate similarity using the underlying distribution of the data objects. Depending on the characteristics of the data objects, the definitions of similarity measures presented in Section 2.2 can also be used to estimate similarities amongst the data objects. In Section 5.2, we show the results of a preliminary experiment that depicts the impact of different definitions of similarity measures on the separation of a given dataset.

In the remainder of this thesis, ant brood sorting and ant clustering algorithm (ACA) are used interchangeably to refer to the same algorithm. The following pseudocode and flowchart depicts a basic ant clustering algorithm (Algorithm 1, Figure 4.4) [Lumer and Faieta, 1994]:

Algorithm 1 Basic ant brood sorting algorithm for clustering

```

1: Input:  $S = \{o_1, o_2, \dots, o_n\}$  - data objects
2: Parameters:  $tmax$  - no. of iterations,  $w$  - no of ants,  $l$  - dimension of grid,  $s$  - dimension of static neighborhood
3: procedure ANTbroodSORTING( $S$ )
4:   Initialization:
5:   for each data object  $o_i$  do
6:     place  $o_i$  randomly onto site  $r$  on the grid
7:   end for
8:   for each ant  $a_w$  do
9:     place  $a_w$  randomly onto site  $r$  on the grid
10:  end for
11:  Main loop:
12:  while  $t \leq tmax$  do
13:    for all ants  $a_w$  do
14:      if (ant  $a_w$  is unladen) and ( $r$  occupied by  $o_i$ ) then
15:        compute  $f(o_i)$  and  $P_p(o_i)$ 
16:        select random real number  $R$  between 0 and 1
17:        if  $R \leq P_p(o_i)$  then
18:          pick up object  $o_i$ 
19:        end if
20:      else if (ant  $a_w$  laden with  $o_i$ ) and (site  $r$  empty) then
21:        compute  $f(o_i)$  and  $P_d(o_i)$ 
22:        draw random real number  $R$  between 0 and 1
23:        if  $R \leq P_d(o_i)$  then
24:          drop off object  $o_i$ 
25:        end if
26:      end if
27:      move ant  $a_w$  to randomly selected neighboring site  $r'$  not occupied by any other ant
28:    end for
29:  end while
30: end procedure
31: Output
32: Display final state of exploratory grid
33: Extracts clusters from grid to derive final clusters  $C = c_1, c_2, \dots, c_k$ 
34: Derive value of  $k = |C|$ 

```

4.2.1 Motivation for using ACA

The motivation for using ant brood sorting to tackle clustering includes the following [Monmarché et al., 1999; Handl et al., 2006]:

1. Many of the metaheuristic clustering algorithms require the value of k (i.e. the number of clusters) by design and are therefore not applicable for clustering problem instances wherein the value of k is not known a priori. For ant brood sorting, on the other hand, the value of k is not required; instead k is implied from the number of clusters uncovered by the algorithm.

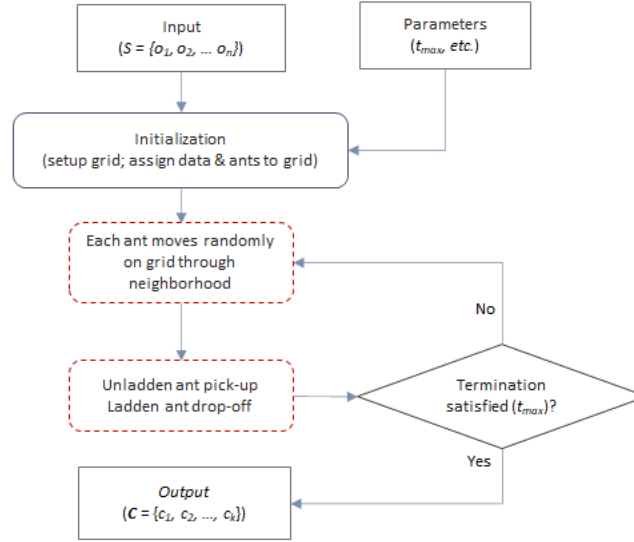


Figure 4.4: Flowchart of basic ant clustering algorithm

2. The algorithm's artificial ant model (derived from natural *Leptothorax Unifasciatus* ants' behavior) is a close natural representation of the clustering problem. The end goal of natural ant's behavior is similar in principle to the end goal of clustering - objects with underlying similarity or similar needs are grouped together while clearly separating those that are dissimilar.
3. There is no constraint on the shape of the clusters that can be uncovered by the algorithm, provided there is a suitable measure of the similarity or dissimilarity between data objects within the clusters.

4.2.2 Limitations and solutions

However, a notable inherent limitation of the ant clustering algorithm is the intensive computational effort needed for the algorithm to converge successfully even for small-sized problem instances [Dorigo et al., 1999]. This is attributable to the

high number of random walks that the ants have to perform during the pick-up and drop-off operations to complete the clustering process. The underlying causes of the high volume of random walks can be traceable to the following:

1. A rigid memory structure. A memory structure in the ants can be used to keep track of historical pick-up or drop-off decisions. Incorporating a memory structure in the ants can enable the ants to take informed decisions when traversing the grid. Existing designs of memory structure in the artificial ants focus primarily on enabling the ants to remember locations on the grid that have a high concentration of data objects similar to a given data object in context. This can implicitly enable the ants to uncover clusters quicker, but does not directly prevent the ants from *cycling* (i.e. ants moving back and forth between historically visited sites within a neighborhood on the grid during exploration) when traversing grid. However, incorporating an *adaptive* memory structure within the ants can enable them to avoid neighborhood sites that should not be revisited in the near future when traversing the grid.
2. A rigid neighborhood structure used by the ants to guide local exploration during the pick-up and drop-off operations. With a static neighborhood structure, the moves by the ants are susceptible to *cycling*. By reducing back and forth movement when exploring a neighborhood, the ants can traverse the neighborhood more efficiently and consequently uncover the clusters more efficiently.
3. In existing designs of the algorithm, the final decisions of the ants to pick-up and drop-off are based on a random variable component. This can result in

more occasionally wrong decisions than necessary, thereby causing prolonged exploration steps for the ants, if appropriate deterministic thresholds are not set to guide these decisions.

Over the years, additional research efforts have been expended on alleviating some of these limitations. Lumer and Faieta [1994] added short-term memory to the ants to help the ants remember where similar data objects were previously dropped. Chen et al. [2004] also proposed giving ants memory banks to guide traversing in the 2D grid and speed up convergence. Boryczka [2008] proposed an enhanced short-term memory version to improve convergence. Pereira et al. [2005] made a series of improvements to Lumer and Faieta's model [Lumer and Faieta, 1994] to enable the ants to uncover more robust clustering solutions. [Handl et al., 2006] created an enhanced version of the ant clustering algorithm that includes using heterogeneous ants, varying ant actions over time, and a technique for transforming the 2D representation of the algorithm's outcome into explicit clusters. Yang and Kamel [2006] introduced the concept of multiple ant colonies wherein each colony uses ants with different attributes such as moving speed, pick-up/drop-off probabilities, etc. Qu and Liu [2007] presented another enhanced version of the ant clustering algorithm - quick and effective ant clustering (QAC) algorithm - wherein each ant represents a data object with a little local information to make clustering happen quickly.

Furthermore, Li et al. [2008] introduced the use of kernel function in the underlying measure of similarity by the ants. Weili [2009] adapts the concept of information entropy in modelling the behavior of ants in the ant clustering algorithm for improved cluster quality. Qasem et al. [2018] also proposed using a kernel density based function

in the computation of the pick-up/drop-off probabilities in Lumer and Faieta's model [Lumer and Faieta, 1994] and implemented a parallel version of the algorithm on a multi-core architecture.

There have also been additional research efforts aimed at enhancing ant clustering algorithms by creating hybrid versions of the algorithm. Monmarché [1999] combined the ACA with k-means algorithm to reduce any mis-classification of the data objects in the clusters derived by the ACA. Hamdi et al. in [Hamdi et al., 2008] and [Hamdi et al., 2016] proposed hybridizing the ACA with other swarm techniques from birds and spiders to guide the exploration and decision making process of the ants and make the uncovering of clusters faster.

4.2.3 Proposed blending of ACA algorithmic components

In this research, without further complex modifications to the features of the artificial ants, we reexamined the inherent limitations of the ACA algorithm. To show the viability of our blending strategy, we propose a hybrid version of the algorithm that *blends* strategies from another algorithm (tabu search) to overcome these limitations.

Contrary to the white-listing memory strategy adopted in existing enhancements to the ACA algorithm, tabu search has a blacklisting memory strategy (*tabu list*) that can be used to guide the ants away from *cycling* when exploring a neighborhood. Tabu search also has an adaptive neighborhood mechanism (incorporates the *tabu list*) that can help the ants explore a neighborhood more efficiently by preventing *cycling*.

We submit that tabu search has algorithmic components that can naturally be blended with ACA to overcome the inherent limitations of ACA. A basic tabu search

metaheuristic is described in the following subsection.

4.3 Tabu Search Algorithm

Tabu search (TS) is a meta-heuristic search method that uses intelligent problem-solving concepts of adaptive memory and responsive exploration to tackle difficult optimization problems [Martí et al., 2007]. To find a global *optimal* solution, tabu search guides a local or neighborhood search heuristic beyond local optima using a combination of strategies to explore the solution space [Glover and Laguna, 2013].

The strategies include:

- An adaptive memory structure (*tabu list*) that keeps track of historic solutions or actions related to historic solutions that are forbidden to prevent back and forth movement between solutions already examined; a dynamic neighborhood structure influenced by the adaptive memory structure to implicitly diversify searching the solution space.
- Restricting and/or freeing up previous exploration paths (i.e. *aspiration criteria*).
- Explicitly exploring diverse regions of the solution space (diversification).

To apply tabu search to the clustering problem, consider a given collection of data objects $S = \{o_1, o_2, \dots, o_n\}$ to be assigned to $C = \{c_1, c_2, \dots, c_k\}$ clusters, the goal is to find the *optimal* solution C_t^* (a combination of clusters) amongst each feasible solution C_t that minimizes the cost function $F(C_t)$. The cost function is usually defined based on a measure of the dissimilarity among data objects that belong to the

same cluster for each cluster $c_{k(t)}$ in C_t . Similar to what applies with other clustering algorithms, the definition of the dissimilarity cost function $F(C_t)$ can vary depending on the nature of the data objects and/or the rationale for clustering. For numeric data objects, a common definition of $F(C_t)$ is the sum of squared errors between each data object o_i and the corresponding centroid z_k derived for each cluster $c_{k(t)}$ in C_t . A popular definition of $F(C_t)$ in the literature is given as follows [Al-Sultan, 1995]:

$$F(C_t) = \sum_{i=1}^n \sum_{j=1}^k w_{ij} \left\| o_i - z_j \right\|^2 \quad (4.6)$$

where w_{ij} is a measure of the belongingness of data object o_i to cluster c_j . The value of $w_{ij} = 0$ or 1 for $i = 1, 2, \dots, n; j = 1, 2, \dots, k$; and for non-overlapping clusters where each data object o_i belongs to only one cluster $\sum_{j=1}^k w_{ij} = 1, i = 1, 2, \dots, n$. z_j is the derived centroid for each cluster $c_{j(t)}$ in solution C_t .

Tabu search clustering starts with an initial solution $C_0 = \{c_{1(0)}, c_{2(0)}, \dots, c_{k(0)}\}$ consisting of clusters having data objects from S assigned to them. The initial solution can be randomly generated by randomly assigning each data objects to exactly one cluster; or obtained using formal means such as the Ward's algorithm in [Turkensteen and Andersen, 2009]. The current solution C_t^c and the best solution C_t^b are then set to the initial solution $C_t^c = C_0, C_t^b = C_0$ (the current solution C_t^c is the solution in context at iteration t ; the best solution C_t^b holds the overall best solution at any given instance and eventually becomes the global optimal solution). The object function for the current solution $F(C_t^c)$ is then evaluated.

In the main iteration loop in Algorithm 2 (at iteration t), the current solution C_t^c is varied slightly as a result of a *move* to derive new sets of feasible solutions

$(C_{1(t)}^c, C_{1(t)}^c, \dots, C_{N(t)}^c)$ that make up the neighborhood set $N(C_t^c)$. An example of a *move* is reassigning the cluster membership of a data object in each of the k clusters that make up C_t^c . The reassignment of cluster membership can be based on probability [Al-Sultan and Fedjki, 1997] or deterministic strategies [Turkensteen and Andersen, 2009]. A subset $N^*(C_t^c)$ of the neighborhood set is then selected - a basic selection mechanism is finding the difference between the neighborhood set and the *tabu list* ($N^*(C_t^c) = N(C_t^c) - TL$). The value of $F(C_t^c)$ is computed for each element in $N^*(C_t^c)$ and the element C_t^{c*} having the best value is set as the new current solution. The new current solution or the *move* that resulted in the current solution is then blacklisted by adding it to the *tabu list* (i.e becomes forbidden) to exclude it from being reassigned as a current solution within the near future iterations. This forbidden restriction creates an inherent *diversification* strategy that enables tabu search to explore beyond a local optimal solution and eventually uncover a global optimal solution. The restriction can be overridden when an *aspiration criterion* is satisfied. An example of an aspiration criterion is when a forbidden solution has a better objective function value than the best solution so far.

The new current solution is assigned as the overall best solution ($C^b = C_t^{c*}$), if $F(C_t^{c*}) < F(C^b)$. The main loop continues iteratively until a stopping criteria is satisfied. Examples of a stopping criteria include: reaching the maximum number of iterations; not finding a better solution after a given number iterations. Algorithm 2 depicts the pseudocode and Figure 4.5 depicts the flowchart of a basic tabu search algorithm for clustering.

Algorithm 2 Basic tabu search algorithm for clustering

```

1: Input:  $S = \{o_1, o_2, \dots, o_n\}$  - data objects,  $k$  - no. of clusters
2: Parameters:  $tmax$  - no. of iterations,  $N$  - neighborhood size,  $tlsize$  - tabu list size

3: procedure TABUSEARCH( $S, k, [C_0]$ )
4:   Initialization:
5:   Initial solution  $C_0 = \{c_{1(0)}, c_{2(0)}, \dots, c_{k(0)}\}$ ; set  $t = 0$ ; set  $C_t^c = C_0$ ; set  $C_t^b = C_0$ ; set  $TL = \emptyset$  (tabu list)
6:   Main loop:
7:   repeat
8:      $t = t + 1$ 
9:     Generate adaptive neighborhood set for current solution  $N^*(C_t^c) = N(C_t^c) - TL$ 
10:    for each element  $C_t^{c*}$  in the derived neighborhood set  $N^*(C_t^c)$  do
11:      Compute  $F(C_t^{c*})$ 
12:      if  $C_t^{c*}$  satisfies aspiration criteria then
13:         $C_t^c = C_t^{c*}$  (set  $C_t^{c*}$  as new current solution)
14:         $C_t^b = C_t^{c*}$  (set  $C_t^{c*}$  as latest best solution)
15:      else if  $C_t^{c*}$  is the best solution in  $N^*(C_t^c)$  then
16:         $C_t^c = C_t^{c*}$  (set  $C_t^{c*}$  as new current solution)
17:      end if
18:      Add  $C_t^{c*}$  to tabu list  $TL = TL \cup \{C_t^{c*}\}$ 
19:    end for
20:  until : a termination condition is satisfied
21: end procedure

22: Output : output  $C_t^b$  as the final solution

```

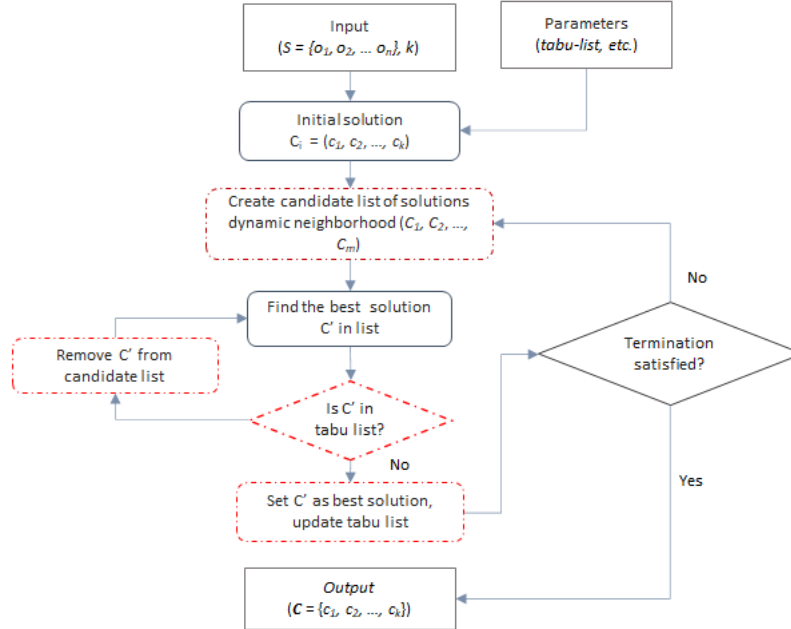


Figure 4.5: Flowchart of basic tabu search clustering algorithm

4.4 Blended Integrative Hybrid Clustering Algorithm - ACA and Tabu Search

The proposed hybrid clustering algorithm blends algorithmic components (strategies and tactics) from ACA and tabu search together into a single clustering algorithm, using the blended integrative hybridization strategy described in Section 3.1 above. We adopt the schema for blending algorithmic components (Section 3.2) in the design of the proposed hybrid algorithm. The aim is to identify, select and blend the algorithmic components from both underlying algorithms by giving preference to the components that contribute more positively towards the overall clustering goal.

We start by identifying the different algorithmic components for both algorithms. The algorithmic components for ACA (as described in Section 4.2) include the following:

1. A setup of the exploratory spatial grid (aca_1) - a 2D grid representation of the spatial terrain wherein the ants move around to perform pick-up and drop-off operations on the data objects.
2. A static neighborhood structure (aca_2) - a predefined list of sites on the grid that is relative to the current site of a given ant wherein the ant examines to gather local intelligence. The neighborhood structure is static with respect to the ant's current location on the grid (depicted as region $s \times s$ with respect to site r in Figure 4.2 above).
3. A white-listing memory structure (aca_3) - used by an ant to bias its movement

towards a potential drop-off site after picking up a data object and becoming laden.

4. A probabilistic pick up decision-making process that is influenced by a random component (aca_4) - the decision to pick up a given data object by an *unladen* ant is ultimately based on the comparison between a similarity measure that is based on the probabilistic distribution of dissimilar data objects around the given data object, and a randomly drawn variable.
5. A probabilistic drop-off decision-making process that is influenced by a random component (aca_5) - the decision to drop-off the data object by a laden ant is ultimately determined by comparing a similarity measure that is based on the probabilistic distribution of similar data objects around the given data object with a randomly drawn variable.
6. Identifying clusters on the spatial grid (aca_6) - a mechanism to automatically identify regions on the spatial grid where clusters exist.

The following are notable algorithmic components of the basic tabu search (TS) algorithm as described in Section 4.3 above:

1. A setup of the initial solution C^* (ts_1) - this is the starting solution for tabu search algorithm and can be generated randomly.
2. A dynamic neighborhood structure (ts_2) - a set of feasible solutions that vary slightly from a current solution. The dynamic behaviour is implicitly obtained by tactically excluding solutions that have been historically selected as potential current solutions from the set.

3. A blacklisting adaptive memory structure (*tabu list*) (ts_3) - keeps track of historical current solutions or *moves* that resulted in historical current solutions to guide the search process away from cycling around a local optimum. Solutions or *moves* in the *tabu list* are forbidden for consideration as current solutions, while they remain on the list.
4. Implicit diversification strategy (ts_4) - a combination of the dynamic neighborhood structure and *tabu list* enables the search process to implicitly explore diverse regions in the solution space when searching for a global *optimal* solution.
5. A deterministic trajectory search path (ts_5) - deterministic decision making process is used to guide the search path of the algorithm while exploring the solution space.
6. An evaluation of the fitness of a feasible solution (ts_6) - an objective function is used to evaluate the fitness of a feasible solution.

Considering the second step of the schema (selection phase in BIH), we examine the algorithmic components identified above for both algorithms ($aca_1, aca_2 \dots aca_6$ for the ACA, and $ts_1, ts_2, \dots ts_6$ for tabu search) to select the common algorithmic components that are appropriate as candidates for blending. For instance, both algorithms have the concept of a local neighborhood structure and a solution selection tactic. The neighborhood structure in ACA consists of a predefined set of sites on the search grid that an ant can explore at a given instance during a *pick-up* or *drop-off* operation. In basic tabu search, a neighborhood structure consists of a dynamic set

of possible solutions that are a *move* away from the current solution and not in the *tabu list*. This dynamic capability can be adapted into the artificial ants in ACA to make the neighborhood search more adaptive in finding good solution candidates with less computation.

Another tactic common to both algorithms is the use of memory. In ACA, each ant uses a white-listing memory tactic that keeps a history of previous drop-off sites and biases subsequent drop-off decisions based on that history. While this white-listing memory structure can enable *laden* ants to locate the right drop-off sites quickly, it does not address potential cycling by *laden* and *unladen* ants. Conversely, the blacklisting memory structure in tabu search can be blended into an artificial ant in ACA to prevent revisiting of neighborhood sites recently examined by the ant during pick-up or drop-off operations. Additionally, the ultimate decision making process of ants in ACA, when picking up or dropping off a data object on the grid, is influenced by a random variable that can skew the clustering results. However, in tabu search, the decision making process that results in uncovering the clusters involves making informed deterministic choices when selecting a solution. This deterministic decision making process in tabu search can be adapted into ACA to temper the influence of the random variable in the pick-up and drop-off decision making processes.

Following the third step of the schema, we examine the choice of primary versus secondary algorithm in the proposed ensemble of the algorithmic components. This choice is influenced by the overall clustering goal and the positive contribution the algorithmic components can make towards such goal. In the proposed hybrid algorithm, we select ACA as the primary algorithm, since ACA can tackle clustering

problem instances wherein k is not predefined; and tabu search is considered as the secondary algorithm.

In the final step of the hybrid algorithm design, we create an ensemble of the algorithmic components with consideration to the design decisions in the previous steps. Using ACA as the primary algorithm, we augment the static neighborhood structure with the dynamic neighborhood strategy of tabu search $[aca_2 (ts_2)]$; enhance the basic memory structure of the ants with the adaptive memory inspired by *tabu list* $[aca_3 (ts_3)]$; and incorporate the deterministic decision making tactic and implicit diversification strategy of tabu search into the pick-up and drop-off decision making process of the ants $[aca_4 (ts_4, ts_5); aca_5 (ts_4, ts_5)]$.

Figure 4.6 depicts the result of integrative blending of ACA and tabu search to create a hybrid clustering algorithm, Algorithm 3 below presents the pseudo code and Figure 4.7 depicts the a flowchart for the blended hybrid clustering algorithm.

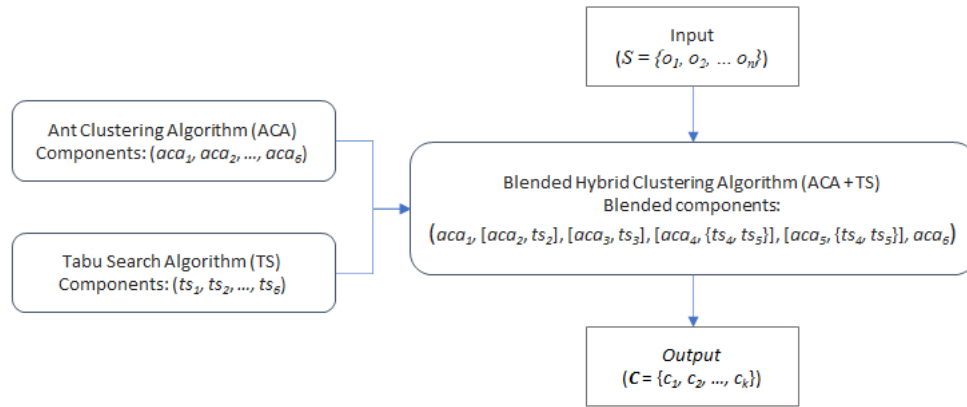


Figure 4.6: Depiction of blended ACA and tabu search clustering algorithm

Algorithm 3 Blended integrative hybrid clustering algorithm
 BIHCA (ACA + Tabu Search)

```

1: Input:  $S = \{o_1, o_2, \dots, o_n\}$  - data objects
2: Parameters:  $tmax$  - no. of iterations,  $tlst$  - tabu list size,  $ppt$  - pickup probability threshold,  $dpt$  - drop-off probability threshold,

3: procedure BLENDEDACATABUSEARCH( $S$ )
4:   Initialization:
5:   for each data object  $o_i$  do
6:     place  $o_i$  randomly onto site  $r$  on the grid
7:   end for
8:   for each ant  $a_w$  do
9:     place  $a_w$  randomly onto site  $r$  on the grid
10:  end for
11:  Main Loop:
12:  while  $t \leq tmax$  do
13:    for all ants  $a_w$  do
14:      if ( $a_w^r$  is unladen) and (site  $r$  is loaded with  $o_i$ ) then
15:        compute  $P_p(o_i)$  wrt neighborhood of site  $r$ 
16:        #Deterministic check
17:        if  $P_p(o_i) < ppt$  then
18:          pick up object  $o_i$ 
19:          ant become laden ( $a_w^{ri}$ )
20:        else
21:          #Randomized check
22:          Generate random real number  $R$  between 0 and 1
23:          if  $R \leq P_p(o_i)$  then
24:            pick up object  $o_i$ 
25:            ant become laden ( $a_w^{ri}$ )
26:          end if
27:        end if
28:        if ( $a_w$  is still unladen) then
29:           $tl'_{a_w} = tl_{a_w} \cup \{r\}$  Add  $r$  to ant tabu list of forbidden sites to NOT go back to shortly
30:        end if
31:      else if ( $a_w^{ri}$  laden with  $o_i$ ) and (site  $r$  empty) then
32:        compute  $P_d(o_i)$  wrt neighborhood of site  $r$ 
33:        #Deterministic check
34:        if  $P_d(o_i) > dpt$  then
35:          drop off object  $o_i$ 
36:          ant become unladen ( $a_w^r$ )
37:        else
38:          #Randomized check
39:          Generate random real number  $R$  between 0 and 1
40:          if  $P_d(o_i) > R$  then
41:            drop off object  $o_i$ 
42:            ant become unladen ( $a_w^r$ )
43:          end if
44:        end if
45:        if ( $a_w$  is still laden) then
46:           $tl'_{a_w} = tl_{a_w} \cup \{r\}$  Add  $r$  to ant tabu list of forbidden sites to NOT go back to shortly
47:        end if
48:      end if
49:      #Create adaptive neighborhood structure
50:       $N'(r_{a_w}) = N(r_{a_w}) - tl_{a_w}$ 
51:      move ant to a site in  $N'(r_{a_w})$  that is not occupied by another ant
52:    end for
53:  end while
54: end procedure

55: Output
56: Display final state of exploratory grid
57: Extracts clusters from grid to derive final clusters  $C = c_1, c_2, \dots, c_k$ 
58: Derive value of  $k = |C|$ 

```

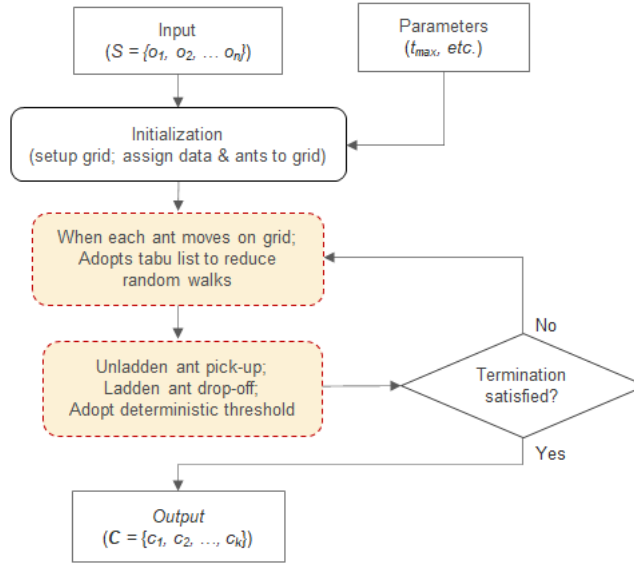


Figure 4.7: Flowchart of blended ACA and tabu search clustering algorithm

4.5 Collaborative Hybrid Clustering Algorithm - ACA and Tabu Search

For comparative analysis purposes, we also propose a collaborative hybrid clustering algorithm that combines ACA and tabu search. This technique adopts the collaborative hybridization strategy described in Section 2.4.1. The algorithm combines standalone versions of ACA and tabu search in sequence, with the ACA as the starting phase algorithm and tabu search as the final phase algorithm. Considering the clustering problem instance in scope for this research, wherein the number of clusters k is not predefined, the hybrid algorithm starts with ACA consuming the given data collection $(S = \{o_1, o_2, \dots, o_n\})$ as input. At the end of the starting phase, the ACA outputs a sub-optimal set of clusters $(C_{sub} = \{c_{s1}, c_{s2}, \dots, c_{sk}\})$ and an implied

value of k . Consequently, the output from the ACA are used as additional input requirements by tabu search in the final phase of the hybrid algorithm.

Subsequently, the sub-optimal solution (C_{sub}) and the derived number of clusters (k), besides the given data collection ($S = \{o_1, o_2, \dots, o_n\}$), are used as the required input for tabu search in the final phase. In the final phase, the tabu search algorithm starts by setting the initial solution to the sub-optimal solution from the ACA algorithm ($C_0 = C_{sub}$). The clustering process continues with the following iterative steps involved in a basic tabu search algorithm (Section 4.3) and the best solution (C^b) from this phase is eventually set as the final solution of the collaborative hybrid clustering algorithm.

Figure 4.8 depicts a collaborative hybrid clustering algorithm that combines ACA and tabu search. Algorithm 4 below represents the pseudo code for the blended hybrid clustering algorithm. As shown in the pseudo code, the collaborative hybrid algorithm combines the procedures of the standalone versions of ACA and tabu search in the right sequence.

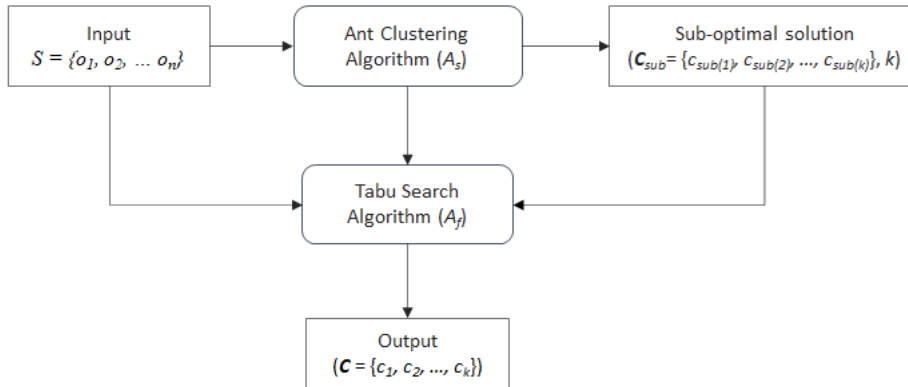


Figure 4.8: Depiction of collaborative ACA and tabu search clustering algorithm

Algorithm 4 Collaborative hybrid clustering algorithm

CHCA (ACA + Tabu Search)

- 1: **Input:** $S = \{o_1, o_2, \dots, o_n\}$ - data objects
 - 2: **Parameters:** $tmax$ - no. of iterations, N - neighborhood size, $tlist$ - tabu list size
 - 3: **Starting phase:**
 - 4: $(C_{sub}, k) = AntBroodSorting(S)$
 - 5: **Final phase:**
 - 6: $C^b = TabuSearch(S, k, C_{sub})$
 - 7: **Output :** output C^b as the final solution
-

The next chapter presents the experimental results of the proposed blended algorithmic strategy on benchmark datasets.

Chapter 5

Experiments and Results

In this chapter, we describe the empirical analyses undertaken to verify the viability of the proposed blended integrative hybridization technique and the BIHCA clustering algorithm designed using this technique. An implementation of the BIHCA algorithm, as described in Section 4.4, that blends ACA and tabu search together is compared with implementations of standalone version of a basic ACA and a CHCA algorithm, also consisting of ACA as the starting algorithm and tabu search as the final algorithm (Section 4.5). The algorithms are implemented and applied to clustering problem instances wherein the value of k (the number of clusters) is not predefined.

The BIHCA is also compared with implementations of mean shift [Comaniciu and Meer, 2002] and DBSCAN (Density-based spatial clustering of applications with noise) [Ester et al., 1996a] clustering algorithms in the scikit.org python package. These are classic clustering algorithms that also tackle instances of clustering problems wherein the number of clusters k is not predefined. The performance of the algorithms are assessed using cluster validation metrics described in the subsequent

sub-sections.

The experiments undertaken in this research to validate the viability of the newly proposed hybrid algorithm use popular benchmark datasets commonly used when assessing clustering algorithms. The following section presents descriptions of these benchmark datasets.

5.1 Benchmark Datasets

Using implementations of the clustering algorithms (ACA, CHCA, BIHCA, DBSCAN and mean shift), we perform experiments to uncover clusters within three real-life benchmark datasets (the iris, wheat seed and wine datasets). The datasets are sourced from the UCI machine learning data repository [Dua and Graff, 2017].

1. *Iris*: Consists of 150 random samples of three species of iris plant (setosa, versicolor, and virginica). The samples are evenly distributed - there are 50 samples of each species. Each sample is represented using 4 numeric attributes (sepal length, sepal width, petal length, and petal width in cm). One of the species is linearly separable from the other two, while the other two species are not.
2. *Wheat Seed*: Consists of 210 random samples of 3 different varieties of wheat (Kama, Rosa and Canadian). The samples are also evenly distributed - there are 70 samples of each wheat seed variety. Each sample has 7 numeric attributes that represent the geometric properties of the kernel for each variety.
3. *Wine*: The wine dataset includes 178 samples representing the result of chemical

analysis of wines derived from 3 types of cultivars grown in the same region in Italy. The samples are unevenly distributed within the three types. Each sample has 13 numeric attributes representing the quantities of 13 chemical constituents found in each of the wine.

For the above datasets used in experiments, each sample is represented as a data object denoted using an attribute vector: $\mathbf{o}_i = (o_{i1} \ o_{i2} \ o_{i3} \ \dots \ o_{id})$ where $i = 1, 2, 3, \dots, n$ is the number of samples in the dataset and d is the number of attributes in each data object.

5.2 Measuring Data Similarity

Similar to other clustering algorithms, the artificial ants in the ant clustering algorithms require an objective way to determine the degree of similarity between the data objects to facilitate the pick-up and drop-off decision processes. A measure of the similarity between data objects can be derived from dissimilarity coefficients (distance based measures) [Shirkhorshidi et al., 2015] or similarity coefficient (correlation and distribution kernel based measures) [Hofmann et al., 2008].

We performed preliminary experiments to identify which similarity measure is most appropriate for the ants to use when clustering the different benchmark datasets examined in this research. The average intra-cluster and inter-cluster similarity measures are computed using different similarity and dissimilarity coefficients for the data objects within the datasets. The similarity coefficients examined are chi-squared, Laplacian, sigmoid and Gaussian kernels; and the dissimilarity coefficients examined are Euclidean, cosine and normalized Euclidean distances. The similarity measures

are computed for the original and normalized versions of the datasets. For each dataset, the normalized version is derived by scaling the attribute values to lie between 0 and 1 for each data object. The scaled attribute value for feature d in data object o_i is derived as follows:

$$o_{id}^* = \frac{(o_{id} - o_d^{min})}{(o_d^{max} - o_d^{min})} \quad (5.1)$$

where o_d^{min} and o_d^{max} are respectively, the minimum and maximum value of the attribute d across the data objects in the given dataset.

The goal is to uncover the similarity measure that demonstrates the widest range between the intra-cluster and inter-cluster similarity measures. We submit that the similarity measure that demonstrates these separation characteristics will be most suitable as the basis for deriving $f(o_i)$ that is used by the artificial ants' during *pick-up* and *drop-off* decision making processes. Using predefined label information about the datasets, we derive the average intra-cluster similarity measure for the data objects within the same cluster, as well as the average inter-cluster similarity measure for the data objects that belong to different clusters.

For the three benchmark datasets (*iris*, *wheat* and *wine*), Fig 5.1, Fig 5.2 and Fig 5.3 respectively show a plot of the average intra-cluster and inter-cluster similarity measures for different similarity and dissimilarity coefficients. The figures show that the range between the average intra-cluster and inter-cluster similarity measures is generally wider for the distance-based coefficients when applied to the normalized version of the datasets; and the reverse is mostly the case for the kernel-based coefficients.

Further, for the original form of the *iris* and *wheat* datasets, the Gaussian kernel average inter-cluster similarity measure is notably low (less than 0.2) and the intra-cluster measure is notably high (0.7). For the *wine* dataset, applying chi-squared kernel to the normalized version of the dataset results in average inter-cluster measure that is about 0.2 and average intra-cluster measure that is just below 0.6. Therefore, for the artificial ants in the subsequent experiments, we use the Gaussian kernel as the basis for computing the value of $f(o_i)$ when clustering the *iris* and *wheat* datasets; and the chi-square kernel is used as the basis for computing the value of $f(o_i)$ when clustering the *wine* dataset. The computation of $f(o_i)$ has a direct influence on the value of the probabilities used by the artificial ants during the *pick-up* and *drop-off* decision making processes.

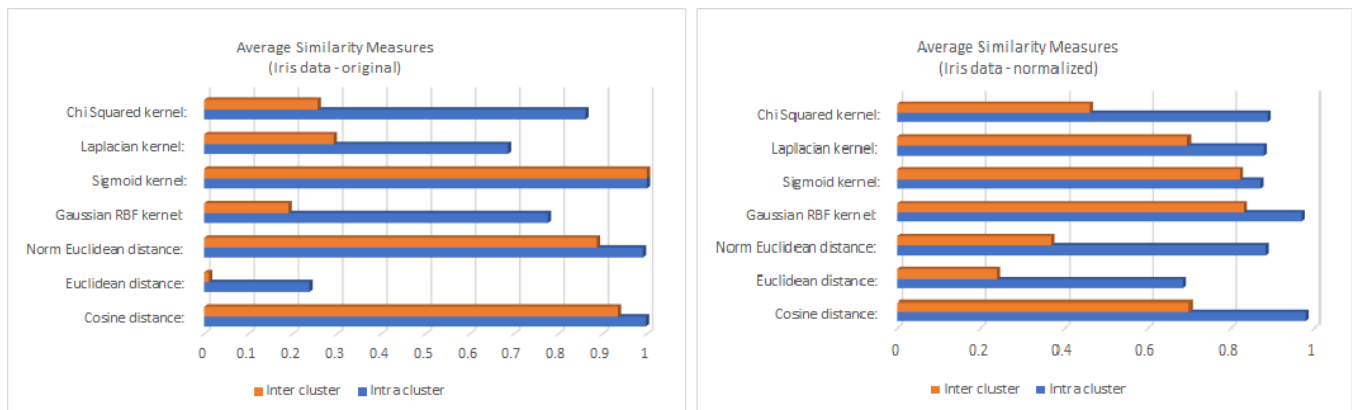


Figure 5.1: Average similarity measures for iris data

5.3 Cluster Validation Measures

There are three broad categories of validation measures used in assessing the quality of clusters uncovered by a clustering algorithm: external cluster validations,

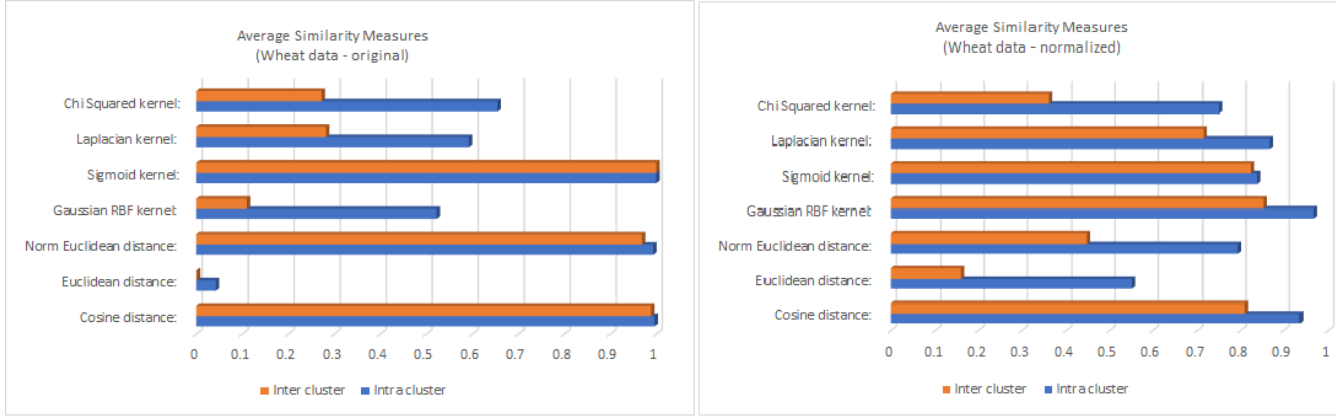


Figure 5.2: Average similarity measures for wheat seed data

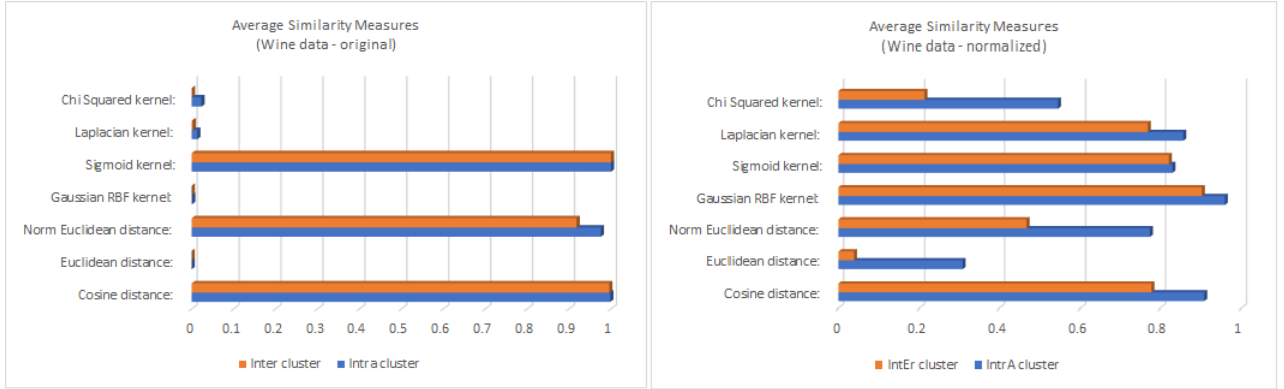


Figure 5.3: Average similarity measures for wine data

internal cluster validations and relative cluster validations [Liu et al., 2010]. External cluster validation measures require information such as *a priori* class labels (i.e. ground truth) of the data objects to validate the derived clusters. Internal cluster validation measures, on the other hand, only use information that is intrinsic to the data objects to assess the quality of the derived clusters.

The quality of clusters uncovered by the algorithms examined in these experiments are assessed using both external and internal validation measures. The external measures used are described in the following.

Homogeneity [Rosenberg and Hirschberg, 2007]: establishes by how much each cluster contains only members of a single class. The value of homogeneity ranges from 0 to 1.0 and the higher the value, the better the homogeneity of the clusters.

Completeness [Rosenberg and Hirschberg, 2007]: measures by how much all members of a given class are assigned to the same class. The value of completeness also ranges from 0 to 1.0 and the higher the value, the better the clustering result.

V-Measure [Rosenberg and Hirschberg, 2007]: is the harmonic average of homogeneity and completeness. The value ranges from 0 to 1.0 and higher values imply better clustering result.

Adjusted Rand Index [Hubert and Arabie, 1985]: measures the similarity between the ground truth class labels and the class labels assigned by the clustering algorithm, while ignoring permutations and normalizing chance. The value can range from -1 to 1, with 1 implying a perfect clustering result.

Adjusted Mutual Information [Vinh et al., 2010]: measures the extent of agreement of the ground truth class labels with the derived clusters while ignoring permutations. The value can range from 0 to 1, with 1 implying a perfect clustering result.

The following internal clustering measure is also used to assess the cluster qualities in the experiments.

Silhouette Coefficient [Rousseeuw, 1987]: is composed of two scores that measure

the inter and intra clustering distances for each data object in the dataset. The final score is derived by taking a mean value for the dataset. The value of this measure ranges from -1 to 1, with -1 implying a poor clustering result. A value around 0 implies that the clusters are overlapping and 1 denotes that the clusters are well separated.

5.4 Experiment Setup

In the experiments, we implemented a standalone basic ACA as described in Section 4.2 with parameters based on recommendations in Handl and Meyer [2007] and Boryczka [2008]. The number of ants is set, based on the size of the dataset, to $\sqrt[3]{n}$ and the dimension of the 2D square grid is set to $(\sqrt{10 \times n}) \times (\sqrt{10 \times n})$. We also implemented a CHCA that combines the ACA algorithm and a basic tabu search clustering algorithm as described in Section 4.5. ACA is used as the starting algorithm and tabu search as the finishing algorithm. The parameters used in the tabu search are based on recommendations in [Glover and Laguna, 2013].

Lastly, we implemented a BIHCA - an ensemble consisting of algorithmic components from ACA and tabu search as described in Section 4.4. For the artificial ants in BIHCA, the deterministic threshold parameter that influences *drop-off* decisions (*dpt*) is set to 0.75, and the deterministic threshold parameter that influences *pick-up* decisions (*ppt*) is set to 0.25. These values are influenced based on the outcomes of the preliminary experiment described in Section 5.2.

The implemented algorithms are used to uncover clusters in the *iris*, *wheat seed* and *wine* datasets. The quality of the clusters are assessed using the validation mea-

asures described in Section 5.3. We performed 10 runs of the clustering process for each algorithm per dataset. The best clustering result of each set of the runs are provided in the result tables (Tables 5.1, 5.3 and 5.2) and discussed in the result section below. In the experiments, equivalent amount of computation steps are allocated to ACA, CHCA and BIHCA. For instance, when the ACA and BIHCA are assigned 5000 iterations, the CHCA is assigned 2500 iterations for the ACA component and an equivalence of 2500 iteration steps for the Tabu Search component.

Further, during the experimental runs for ACA and BIHCA, we tracked and recorded the average number of random walks undertaken by the artificial ants while clustering the datasets (Table 5.4). This is to establish the impact of the adaptive memory structure (*tabu list*) that is incorporated into the artificial ants in BIHCA to prevent cycling while exploring their neighborhood.

We also used the scikit.org package implementation of mean shift and DBSCAN to cluster the three datasets and compare the cluster validation metrics with those derived from the outcome of the newly proposed blended hybrid algorithm.

The algorithms examined in this research are implemented in Python 3.2 on a basic-grade personal computer (HP laptop Intel Core i5 8GB RAM Windows 10).

Table 5.1: Clustering results for Iris dataset

Validation Measures	ACA	CHCA	BIHCA
Homogeneity	0.425	0.633	0.679
Completeness	0.340	0.598	0.690
V-measure	0.377	0.615	0.684
Adjusted Rand Index	0.297	0.571	0.644
Adjusted Mutual Information	0.330	0.591	0.675
Silhouette Coefficient	0.089	0.425	0.583

Table 5.2: Clustering results for Wheat Seed dataset

Validation Measures	ACA	CHCA	BIHCA
Homogeneity	0.597	0.662	0.806
Completeness	0.428	0.480	0.649
V-measure	0.499	0.556	0.719
Adjusted Rand Index	0.374	0.441	0.660
Adjusted Mutual Information	0.360	0.415	0.616
Silhouette Coefficient	0.129	0.299	0.517

Table 5.3: Clustering results for Wine dataset

Validation Measures	ACA	CHCA	BIHCA
Homogeneity	0.494	0.526	0.824
Completeness	0.179	0.183	0.711
V-measure	0.263	0.271	0.763
Adjusted Rand Index	0.064	0.062	0.637
Adjusted Mutual Information	0.106	0.120	0.619
Silhouette Coefficient	-0.509	-0.464	0.162

Table 5.5: Comparing BIHCA, Mean shift and DBSCAN (Iris dataset)

Validation Measures	Mean Shift	DBSCAN	BIHCA
No of clusters	2	10	3
Homogeneity	0.554	0.567	0.679
Completeness	0.949	0.378	0.690
V-measure	0.699	0.454	0.684
Adjusted Rand Index	0.558	0.282	0.644
Adjusted Mutual Information	0.551	0.350	0.675
Silhouette Coefficient	0.849	0.019	0.583

Table 5.4: Average number of random walk steps by ants

Datasets	ACA	BIHCA
Iris Dataset	1.4M	0.9M
Wheat Seed Dataset	3.2M	1.1M
Wine Dataset	5M	1M

Table 5.6: Comparing BIHCA, Mean shift and DBSCAN (Wheat Seed dataset)

Validation Measures	Mean Shift	DBSCAN	BIHCA
No of clusters	2	8	3
Homogeneity	0.432	0.111	0.806
Completeness	0.706	0.219	0.649
V-measure	0.536	0.148	0.719
Adjusted Rand Index	0.468	0.006	0.660
Adjusted Mutual Information	0.430	0.075	0.616
Silhouette Coefficient	0.710	-0.702	0.517

Table 5.7: Comparing BIHCA, Mean shift and DBSCAN (Wine dataset)

Validation Measures	Mean Shift	DBSCAN	BIHCA
No of clusters	3	8	4
Homogeneity	0.431	0.183	0.824
Completeness	0.496	0.257	0.711
V-measure	0.461	0.213	0.763
Adjusted Rand Index	0.397	0.039	0.637
Adjusted Mutual Information	0.425	0.144	0.619
Silhouette Coefficient	0.625	-0.427	0.162

5.5 Results Discussion

This section presents analyses of the results obtained from the experiments described in Section 5.4. Table 5.1 shows the internal and external quality measures of the clusters uncovered for the *iris* dataset using the ACA, CHCA and BIHCA algorithms. The results show consistent improvement in the quality of the derived

clusters from ACA to CHCA and then to BIHCA. The value of the clustering validation measures (homogeneity, completeness, v-measure, adjusted rand index, adjusted mutual information and silhouette coefficient) is lowest for ACA, intermediate for CHCA and highest for BIHCA - this . Table 5.2 and Table 5.3 show similar trends for the *wheat seed* and *wine* datasets, respectively.

We infer that the improved cluster quality derived for CHCA results from using a full-fledged tabu search algorithm to reduce the mis-classification of data objects in the clusters firstly derived by the starting phase algorithm - ACA. The improvement in cluster quality for BIHCA is attributable to the more effective neighborhood exploration capabilities of the enhanced ants. The artificial ants in BIHCA, which have tabu search algorithmic components blended into them, are less susceptible to cycling and therefore expend more effective computation steps on exploring their neighborhood for data objects to pick-up or drop-off. This inference is buttressed by the significantly lower number of average random walks undertaken by ants in BIHCA compared to the ants in ACA as shown in Table 5.4. The average random walk steps recorded for ants in BIHCA is about half of the number of steps recorded for ants in ACA when clustering the three benchmark datasets.

Further, the BIHCA ants are less stochastic in their pick-up and drop-off decision making processes, since the ants are first guided by the deterministic threshold parameters (*ppt* - pick-up probability threshold and *dpt* - drop-off probability threshold) before resorting to the randomly generated number R , afterwards. The guidance from these thresholds enable the ants to make fewer erroneous decisions in moving similar data objects together and in moving dissimilar ones apart quickly on the search grid.

Based on the analyses above, we infer that creating a hybrid algorithm using the BIH strategy offers promising results when compared with standalone version of the algorithms or other hybrid version created using conventional hybridization strategy such as the collaborative hybridization strategy. This inference can be extended to other optimization problems that can be tackled using metaheuristic algorithms that have some common form of tactical or strategic tasks or sub-processes or components.

Furthermore, Tables 5.5, 5.6 and 5.7 respectively, show the cluster quality measures for the clusters uncovered in *iris*, *wheat* and *wine* datasets using the BIHCA, mean shift and DBSCAN algorithms. The values of k (i.e. numbers of clusters) derived by these algorithms are also included in the tables for analyses. For the *iris* dataset, Table 5.5 shows that only the BIHCA derives the correct value of $k = 3$. Mean shift only separates the dataset into 2 clusters (one cluster that is linearly separable from the other two, which are grouped together as a single cluster). DBSCAN derives a significantly higher value $k = 8$.

Besides, the cluster quality measures from BIHCA consistently show better value than those from DBSCAN across the three datasets. Although mean shift derives higher values of V-measure and Silhouette Coefficient for the datasets, this is attributable to the higher value of completeness (0.949 for *iris* and 0.706 for *wheat*) resulting from having incorrect number of clusters based on the ground truth cluster labels (2 clusters instead of 3 for Iris; 2 clusters instead of 3 for Wheat).

Also, for the Wine dataset, the ants in BIHCA used normalized instance of the data when computing similarity coefficient. This is based on the inference from Fig 5.3 which shows that the gap between the intra-cluster and inter-cluster average sim-

ilarity measures are significant for the normalized dataset, but not for the original dataset. In the original dataset, two of the attributes (attribute 5 - measure of Magnesium; attribute 13 - measure of Proline) are much higher than the others and therefore can easily skew the similarity measures when used in the raw form. Although, the mean shift has a higher silhouette coefficient, the internal cluster measures for BIHCA are much higher in comparison. Based on the analyses above, we infer that hybrid clustering algorithm designed using the BIH strategy is a viable and promising candidate for tackling clustering problem instance where the value of k is not known *a priori*.

This chapter has suggested through benchmark dataset results that the proposed blended strategy is a viable approach. To further support the feasibility of the strategy, in the next chapter, we focus on an industrial application - clustering group insurance claims dataset to price new group policies for experience-rated benefits.

Chapter 6

Industry Application - Clustering Group Insurance Dataset

This chapter presents an overview of insurance (a key part of modern financial industry) and the evolving application of clustering techniques, such as the proposed hybrid clustering algorithm, to practical knowledge discovery problems in the insurance industry.

Since time immemorial, humanity is constantly being exposed to different forms of dangers that threaten human's property, health, ability or life as a whole. For instance, the loss of harvest for farmers due to flooding or severe drought; the loss of ship, treasured cargoes and crew due to shipwreck; the loss of home and personal possessions for home owners due to fire; the loss of means of livelihood for athletes due to sport injuries; the loss of vehicles and incurred liabilities due to automobile accident; and ultimately, the loss of life due to death. Insurance is a complex concept that stems from the inherent urge of human to protect self from uncertainties and

calamities that can result in undesirable losses or liabilities and the desire to guard against them [van der Merwe, 1970].

A foundational aspect of insurance involves pooling risk (i.e. a chance that a loss will occur) to alleviate the identifiable losses or consequences of such losses. In ancient times, this involved sharing the risk within social and business communities, primarily for solidarity purposes [Haueter, 2017]. Insurance, as known in modern times, involves laws and legal frameworks, statistical recording of different types of data (institutional and personal information) and the use of formal methods to calculate future likelihoods [Haueter, 2017]. The science of insurance (actuarial science) hinges on the use of probability models to predict future occurrences of the calamities that can result in loss(es) [Haueter, 2017].

Although there is no universally accepted formal definition for insurance, a definition that is popularly used is as follows [Denenberg, 1963]:

“By insurance in trade is understood a written contract between parties. This contract is called a policy. The contracting parties are the insured, who pays a consideration, which is called a premium, and the insurer, who received it. For this premium, the insurer engages to satisfy, and make good to the insured, unless a fraud appears, any loss, damage or accident that may happen; according to the tenor of the contract or policy”

In insurance, the product (i.e. the written contract or *policy*), which is paid for in the present (by the *insured* or *policyholder*) in the form of a *premium*, is a promissory note (issued by the *insurer* or *provider*) to cover a loss in the future should it occur under some stipulated circumstances and within a period of time

in the future. Typically, the insurer undertakes the process of assessing the risk exposure from the insured and the eligibility of the loss being covered (this is known as *underwriting*) [Haueter, 2017]. If a covered loss eventually happens as stated in the policy, the *claimant* issues a demand (*claim*) for the insurer to make good the stated benefits to the beneficiary or beneficiaries.

Examples of insurance include a promise to pay for rebuilding or restoring a house if it is burnt; to pay for the cost of repairing or replacing a car if it is involved in an accident; to pay a beneficiary an amount after the demise of an insured; to pay for medical treatment of an employee that was injured while on the job; and to pay for the vision and dental care of dependents of an employee.

Insurance products vary in types and categories based on the kind of risk or loss being covered, as well as the supporting legal framework which can vary from one jurisdiction to another. Naylor [2017b] categorizes insurance products into the following: home insurance, automobile insurance, personal insurance, travel insurance, business insurance, liability insurance and marine/shipping insurance. Insurance provisions can also be differentiated based on whether the risk of coverage is assessed and provided for an individual as the insured (individual insurance) or for a group of individuals collectively as the insured (group insurance) [Bluhm et al., 2003; Fras, 2019]. In this thesis, the focus is on *group insurance* provisions.

Group insurance plan provides an efficient means to offer multiple insurance products to a collection of people (*plan members*) through a *plan sponsor*, which can be an employer, association or any other entity [Bluhm et al., 2003]. In group insurance, the premium can be paid for wholly by the plan sponsor, wholly by the plan members

or jointly paid for by the plan sponsor and the plan members. An example of a group insurance plan can include a variety of covered losses (often referred to as benefits) such as group term life, dependent term life, accidental death and dismemberment, short term disability, long term disability, hospital and medical expense, medical, ancillary health care (dental, vision, hearing, etc) and critical illness [Attwood, 1980]

6.1 Pricing Insurance Products (Ratemaking)

For many conventional business products (goods and services), the cost of production is known beforehand. However, unlike many conventional business products, the cost of an insurance product is not known before it is sold. Hence, the price of insurance products is mostly based on predictive cost, rather than actual cost. Pricing insurance products usually involves using predictive models that are based on historical data to gauge expected future cost of the insurance protection. Therefore, the process of determining the right price (*premium*) to set for an insurance product, also referred to as *ratemaking*, is complicated [Yao, 2008].

For conventional products and services, the following classic economic pricing model typically applies:

$$price = (cost\ components) + (profit\ components) \quad (6.1)$$

Adapting the classic economic model to insurance gives a pricing model that can be represented as follows [Frees, 2018]:

$$premium = (loss + expense) + (UW profit) \quad (6.2)$$

(*UW* denotes UnderWriting) where *premium* represents the price charged to the policyholder by the insurer for the insurance product (i.e. for underwriting the risk for the possible losses covered in the policy); *loss* represents the amount payable to claimant(s) as claim(s); the *expense* component represents additional cost incurred by the insurer relating directly or indirectly to the policy and its corresponding claim(s). The *expense* can further be categorized into *LA expense* and *UW expense*. *LA expense* (loss adjustment expense) represents additional cost incurred by the insurer to process claim settlement and the *UW expense* (underwriting expense) represents the costs incurred by the insurer in acquiring and servicing of policies. The classic pricing model for insurance can then be represented as:

$$premium = (losses + LA expense + UW expense) + (UW profit) \quad (6.3)$$

UW profit (underwriting profit) represents the profit margin from the premium after all losses and expenses have been deducted, as well as business and regulatory cost requirements taken into consideration. Insurers also passively generate profits from investment income derived by setting aside capital for investment, but that is not usually factored into the pricing model.

The *loss* portion of the equation is where most of the complications happen; this portion can be expanded to describe the different models for assessing the pricing of insurance products. Two common methods used for determining premiums are: pure

premium and loss ratio. These two methods are described below.

6.1.1 The pure premium method

When the occurrence of a loss is purely random, then its cost can be derived using the expected value of cost of claim, which can be approximated to be the average claim payment over many insureds. Assuming an insurer's portfolio of homogeneous n risks (i.e. all policies can be considered to cover the same risk), each with corresponding associated loss, X_1, \dots, X_n that is independent and identically distributed, the approximate expected loss can be represented as follows [Frees, 2018]:

$$E(X) \approx \frac{\sum_{i=1}^n X_i}{n} = \frac{\text{loss}}{\text{exposure}} \quad (6.4)$$

where *loss* refers to the cost of a settled claim and *exposure* refers to the possibility of a loss (i.e. a count of the number of insured risks n , for homogeneous risks). This representation is often referred to as the *pure premium* model and can be expressed in terms of the *frequency* and *severity* of claim losses as follows:

$$\begin{aligned} \text{pure premium} &= \frac{\text{loss}}{\text{exposure}} = \left(\frac{\text{claim count}}{\text{exposure}} \right) \times \left(\frac{\text{loss}}{\text{claim count}} \right) \\ \text{pure premium} &= (\text{frequency}) \times (\text{severity}) \end{aligned} \quad (6.5)$$

where *frequency* refers to a count of the number of anticipated claims for a risk, and *severity* refers to the cost of a claim.

The *expense* component in equation 6.2 above can be divided into a portion that

varies with the premium (e.g. sales commission) - *variable* and a portion that is fixed irrespective of the premium (e.g. cost of operating the business) - *fixed* and the equation can be expressed as follows:

$$premium = (loss + fixed + variable) + (UW profit) \quad (6.6)$$

By expressing variable expenses and *UW profit* as ratio of the premium results in variable expense factor (V) and profit/contingencies factor (Q) respectively:

$$premium = (loss + fixed) + premium \times \left(\frac{variable}{premium} + \frac{UW profit}{premium} \right)$$

$$premium = (loss + fixed) + premium \times (V + Q)$$

And by solving for *premium* in the equation above we get:

$$premium = \frac{loss + fixed}{1 - V - Q}$$

Factoring the premium expression above by exposure gives a standardized unit for expressing premium, known as rate - i.e, a measure of premium per exposure:

$$rate = \frac{premium}{exposure} = \frac{\frac{loss}{exposure} + \frac{fixed}{exposure}}{1 - V - Q}$$

$$rate = \frac{pure\ premium + \frac{fixed}{exposure}}{1 - V - Q} \quad (6.7)$$

6.1.2 The loss ratio method

Loss ratio is a measure of the proportion of the premium amount that is used to pay for losses resulting from claims [Werner et al., 2016]. A representation of the ratio is as follows:

$$loss\ ratio = \frac{loss}{premium} \quad (6.8)$$

Unlike the pure premium rating method that is used mainly to determine actual rates, the loss ratio method is used mainly to determine a target change in rate based on a current or given rate [Frees, 2018]. For a target underwriting profit (Q_t), and excluding the expense component in equation 6.2, a premium change factor cf can be derived as follows:

$$\begin{aligned} Q &= \frac{UW\ profit}{premium} = 1 - \left(\frac{loss}{premium} \right) = 1 - loss\ ratio \\ Q_t &= \frac{UW\ profit_{new}}{premium} = 1 - \left(\frac{loss}{cf \times premium} \right) \\ cf &= \frac{loss}{premium \times (1 - Q_t)} = \frac{loss\ ratio}{1 - Q_t} \end{aligned} \quad (6.9)$$

When the expense component of 6.2 is put into consideration, a premium change factor cf can then be derived as follows:

$$\begin{aligned} Q_t &= 1 - \left(\frac{loss + fixed\ expense}{cf \times premium} \right) - V \\ cf &= \frac{\left(\frac{loss}{premium} + \frac{fixed\ expense}{premium} \right)}{(1 - V - Q_t)} \end{aligned}$$

$$cf = \frac{(\text{loss ratio} + \text{fixed expense ratio})}{(1 - V - Q_t)} \quad (6.10)$$

The derived value of cf determines the change factor that needs to be applied to a given premium amount to achieve the target profit ratio indicated by Q_t .

6.1.3 The experience rating method (using credibility theory)

Pure premiums and loss ratios can also be estimated based on actual past claims experiences for an individual risk or class of risks [Frees, 2018]. Experience rating involves giving consideration to actual past claim experience (frequency and severity) of a policyholder or group of policyholders when calculating the premium to charge the policyholder(s) for the risk(s) being covered [Bluhm et al., 2003]. The experience rating method attempts to complement rating estimates that are based on classic rating methods with actual claim experience data of the insured clients.

Credibility theory provides a framework for estimating the future claim experience for policyholder(s) as a function of the actual past claim experience and the average claim experience for similar policyholders [Margolin, 1971]. The extent to which some given claim experience data is considered credible is usually represented using the credibility weight Z . For a given risk, the following is a representation of the premium rate derived using a credibility factor set over existing claim experience data [Frees, 2018]:

$$\hat{R} = Z\bar{X} + (1 - Z)M \quad (6.11)$$

where R is the credibility weighted premium rate derived for a given risk; \bar{X} is the

average loss from the claim experience data over a specified period of time; and M is the manual rate (i.e. the rate derived based on grouping the insureds into risk classes).

The value of Z is such that $0 \leq Z \leq 1$. For a given risk, when the average loss \bar{X} is stable over a given period of time, the experience data can be considered highly credible and the credibility weight Z set to 1. This implies that the premium rate \hat{R} can be derived mostly from the claim experience data (\bar{X}). Conversely, when the claim experience data shows a wide variation in average loss over a given period of time, the experience data is considered not credible and the credibility weight Z is set to 0. This implies that the claim experience data (\bar{X}) should have no influence on the derived premium rate R .

In this part of the research, we explore the possibility of estimating claims experience data for groups with non-existing or unavailable claims experience data using clustering that summarizes claims data for other similar insured groups.

6.2 Group Insurance Ratemaking

Ratemaking in group insurance also consists of the pricing components represented in the insurance pricing model presented above (equation 6.2). Besides, the classic rating methods (pure premium and loss ratio), experience rating methods are also applicable when determining the price for a group policy. However, since a group insurance policy can consist of multiple products or benefits, the gross premium for a group policy can consist of composite premiums derived based on the different benefits covered by the policy. For instance, for a group insurance policy that covers

the cost of dental care and the loss of life for the plan members, the gross premium of the policy will consist of a premium rate for the life insurance (loss of life) and a separate premium rate for the cost of routine dental care.

There are two broad categories of group insurance benefits based on the characteristics of the claim losses and how the premiums are derived: pooled benefits and experience-rated benefits [Bluhm et al., 2003]. The claim losses of pooled benefits are usually characterized with low frequency and high severity. This implies that the chances that a claim will occur for this category of benefits is low, but whenever a claim occurs, the cost of covering the claim by the insurer is much higher compared to the premium charged to the policyholder. Examples of such benefits include: coverage for the loss of life for a plan member (group life), coverage for the loss of life for the dependant of a plan member (dependant life), long term disability (LTD), critical illness (CI) and accidental death dismemberment (AD&D) [Bluhm et al., 2003].

Experience-rated group insurance benefits have claim losses that are usually characterized with high frequency and low severity. This implies that the chances that a claim will occur for this category of benefits is high, and whenever a claim occurs, the cost of covering the claim by the insurer is not significantly higher compared to the premium charged to the policyholder. Examples of group benefits in this category include: extended health care, dental care, vision, and short-term disability (STD) [Bluhm et al., 2003].

6.2.1 Ratemaking for pooled group benefits

To assess the cost of pooled group benefits, insurers usually examine the group to be insured within the context of a *pool* of several other groups within the insurer's portfolio. Pricing of pooled benefits involves combining the claims experience of a group with those of a broad collection of other groups (typically within the insurer's block of business) in order to average out the claim losses across the larger group. The claim experience of a given group over a pooled benefit is not examined in isolation when calculating what the group will be charged for such a benefit. Consequently, groups with lower than the average claim cost for a pooled benefit end up paying premium rates similar to what other groups with higher than average claim cost have to pay.

Common industrial practice is to use *manual rating* to determine the premium for pooled group benefits. Manual rating involves using identified characteristics (rating factors) that can reliably predict future claim losses for a large pool of insureds and categorizing the insureds into risk classes based on these characteristics. The categorization is such that for the same volume of risk exposure, lower risk groups are charged lower premiums and higher risk groups are charged higher premiums accordingly. Examples of manual rating factors used in assessing pooled benefits include: age, gender, life style choices (such as smoking, drinking, etc) and occupation. The pricing model used for manual rating factors are usually derived based on a classic actuarial modeling methods described in Sections 6.1.1 and 6.1.2 above (pure premium and loss ratio methods). Consequently, for pooled group benefits, the calculation of premium rate for a given group is not impacted by the absence of previous claims

experience data for the group.

6.2.2 Ratemaking for experience-rated group benefits

For experience-rated group insurance benefits, the premium is typically derived using the *experience rating* method, which involves incorporating a group's actual past claims experience data in the pricing of the immediate past (retrospective) or immediate future (prospective) coverage period. When the coverage period in context is the immediate past, the method is referred to as *retrospective experience rating*; when the context is the immediate future, the method is referred to as *prospective experience rating* [Bluhm et al., 2003]. *In this thesis, the focus is on prospective experience rating and subsequent references to experience rating implies prospective experience rating.*

To estimate the claim losses that are used in experience rating for a given group, a common industrial practice is to apply a credibility weight or factor Z (similar to what applies in equation 6.11) to the actual past claim losses of the group, as well as the average expected claim losses derived by pooling a larger collection of groups. Equation 6.12 estimates the claim losses (cl) for a group by applying credibility weight Z ($0 \leq Z \leq 1$) to actual past claim losses (*actual claim loss*) and the expected average claim losses (*avg claim loss*) [Bluhm et al., 2003]:

$$claim\ loss = Z(actual\ claim\ loss) + (1 - Z)(avg\ claim\ loss) \quad (6.12)$$

Equation 6.12 shows that when the credibility factor is highest (i.e. $Z = 1$), the future claim loss can be estimated purely based on past *actual claim loss*. Conversely, when

the credibility factor is lowest (i.e. $Z = 0$), the future claim loss should be estimated purely based on the *avg claim loss* derived from averaging the claim loss over a larger pool of groups or through manual rating.

Expressing equation 6.12 in terms of loss ratio by dividing by premium gives:

$$\text{loss ratio} = Z(\text{actual loss ratio}) + (1 - Z)(\text{average loss ratio}) \quad (6.13)$$

Different methods for calculating the value of the credibility factor Z are available in [Margolin, 1971; Bluhm et al., 2003].

6.3 Enhanced Experience-Rated Benefits Ratemaking

There are practical scenarios wherein the claims experience data for an insurable group may be non-existent or not available to a prospective insurer. For instance, a newly created company seeking group benefit coverage for its new staff - the new staff would not have a prior collective claim experience as a group. Another example is an employer or association seeking experienced-rated benefits from a new group insurance provider - the new insurer would not have direct access to the past collective claim experience data of the group from the previous insurer.

A common industrial practice in group insurance is to use manual rating or some other pooled rating technique to estimate the initial claim losses for a group with non-existent or unavailable claims experience data [Margolin, 1971]. This practice is also applicable to insured groups that are very small in size such that the claims

experience data are not reliably sufficient to estimate future claim loss expectations.

For insurable groups with non-existent claims experience data, equation 6.12 becomes:

$$claim\ loss = Z(0) + (1 - Z)(avg\ claim\ loss) \quad (6.14)$$

and a credibility factor of 0 becomes inevitable as the estimated claim loss is wholly based on the average claim loss.

$$claim\ loss = 0(0) + (1 - 0)(avg\ claim\ loss) = avg\ claim\ loss \quad (6.15)$$

In this research, the goal is to uncover an alternative option for group insurers to estimate claim loss of experience-rated benefits for groups with non-existent or unavailable claim experience data. We investigate the possibility of using machine learning techniques, such as clustering, to uncover underlying summarization or grouping of claims experience data that is based on the inherent pattern within demographic data of group plan members within a larger pool of insurable groups. We propose using this summarization partly as a basis for assessing experience-rated benefits for groups with non-existent or unavailable claims experience data in place of the common industrial practice of using mainly manual rating to estimate claim loss for such groups.

Consider that an insurer has a pool of insured groups (G_1, G_2, \dots, G_m) consisting of N plan members collectively. The insurer also has demographic data (x_1, x_2, \dots, x_N) of the plan members, as well as past claim experience data of the groups for an experience-rated benefit that is covered by the insurer. Also, given an insurable group

G^* consisting of n insurable plan members with demographic data (y_1, y_2, \dots, y_n) and non-existent or unavailable claims experience data. For a credibility factor of Z set for the insurable group G^* , we propose a claim loss estimate given as follows, in place of the claim loss estimate depicted in equation 6.15 above:

$$\text{claim loss} = Z(\text{weighted cluster avg claim loss}) + (1 - Z)(\text{avg claim loss}) \quad (6.16)$$

where *weighted cluster avg claim loss* is the weighted average claim loss derived by averaging actual past claim losses of the different clusters to which plan members of the given group G^* belong to and applying count weights n_i to the cluster averages respectively.

$$\text{weighted cluster avg claim loss} = \frac{\sum_{i=1}^k n_i \bar{X}_{c_i}}{n} \quad (6.17)$$

where \bar{X}_{c_i} is the average claim loss derived from the claims experience data of plan members in the pool (x_1, x_2, \dots, x_N) grouped into cluster c_i and n_i is the number of plan members in the insurable group G^* that are classified to belong to cluster c_i . k is the number of clusters uncovered for the pool of plan members (x_1, x_2, \dots, x_N) . The insurer also has a *classifier* that can assign a plan member y_i from any insurable group G^* to one of the cluster labels c_i (a *classifier* maps a given data object to a label from a predefined set of labels).

To explain the proposed clustering-based claim loss estimation, we present the following hypothetical scenario. Consider that an insurer has a pool of groups consisting of a total of 10 plan members with the demographic data of each plan member denoted as x_1, x_2, \dots, x_{10} and the corresponding past actual claim loss incurred by the

insurer for each member is denoted as X_1, X_2, \dots, X_{10} such that $X_1 = 1, X_2 = 2, X_3 = 3, \dots, X_{10} = 10$. Furthermore, the insurer applies a clustering technique to x_1, x_2, \dots, x_{10} to uncover inherent clusters within the plan members pool and the outcome hypothetically reveals 3 clusters such that the first cluster c_1 consists of (x_1, x_2, x_3) , the second cluster c_2 consists of (x_4, x_5, x_6) and the third cluster c_3 consists of (x_7, x_8, x_9, x_{10}) .

For the above scenario, the respective average claim loss for the three clusters are given below:

$$\text{avg claim loss}(\bar{X}_{c_1}) = \frac{X_1 + X_2 + X_3}{3} = \frac{1 + 2 + 3}{3} = 2.0$$

$$\text{avg claim loss}(\bar{X}_{c_2}) = \frac{X_4 + X_5 + X_6}{3} = \frac{4 + 5 + 6}{3} = 5.0$$

$$\text{avg claim loss}(\bar{X}_{c_3}) = \frac{X_7 + X_8 + X_9 + X_{10}}{4} = \frac{7 + 8 + 9 + 10}{4} = 8.5$$

And the overall average claim loss for the entire pool is:

$$\text{overall avg claim loss} = \frac{X_1 + X_2 + X_3 + \dots + X_{10}}{10} = \frac{1 + 2 + 3 + \dots + 10}{10} = 5.5$$

Consider that the insurer is seeking to estimate the claim cost for two insurable groups G_a and G_b , each having 3 plan members, wherein the demographic data of the insurable plan members are represented as (y_1^a, y_2^a, y_3^a) and (y_1^b, y_2^b, y_3^b) respectively; and the claims experience data for these groups are non-existent or unavailable. Besides, consider that the insurer's *classifier* successfully labels the insurable plan members as follows: $(y_1^a, y_2^a) \rightarrow c_1$; $(y_3^a) \rightarrow c_2$; $(y_1^b) \rightarrow c_2$; $(y_2^b, y_3^b) \rightarrow c_3$.

For this scenario, using the proposed cluster-based experience rating, the weighted

cluster average claim loss for the insurable group G_a can be derived as follows:

$$\text{weighted cluster claim loss } (G_a) = \frac{\sum_{i=1}^3 n_i \bar{X}_{c_i}}{n} = \frac{2 \times (\bar{X}_{c_1}) + 1 \times (\bar{X}_{c_2}) + 0 \times (\bar{X}_{c_3})}{3}$$

$$\text{weighted cluster claim loss } (G_a) = \frac{2 \times (2.0) + 1 \times (5.0) + 0 \times (8.5)}{3} = 3.0$$

Similarly, for insurable group G_b , the weighted cluster average claim loss for the insurable group can be derived as follows:

$$\text{weighted cluster claim loss } (G_b) = \frac{\sum_{i=1}^3 n_i \bar{X}_{c_i}}{n} = \frac{0 \times (\bar{X}_{c_1}) + 1 \times (\bar{X}_{c_2}) + 2 \times (\bar{X}_{c_3})}{3}$$

$$\text{weighted cluster claim loss } (G_b) = \frac{0 \times (2.0) + 1 \times (5.0) + 2 \times (8.5)}{3} = 7.33$$

The above hypothetical scenario shows that with the uncovered clusters and the corresponding average cluster claim losses the insurer is more informed, in comparison to using the overall pool average, to adequately estimate the claim loss for the insurable groups G_a and G_b .

However, creating the right clustering technique(s) to uncover the underlying clusters within a pool of demographic data for group insurance plan members is a challenging problem. Besides, creating a classifier that accurately labels the plan members from insurable groups with the correct uncovered cluster is equally challenging. The former challenge is within the scope of this thesis and is addressed in the following sections.

6.4 Clustering Group Insurance Datasets

The use of machine learning techniques to discover knowledge or uncover underlying intelligence within data is one of the contributors to the evolving technological disruption in the insurance industry [Naylor, 2017a]. Clustering, a knowledge discovery technique, is being used to tackle complex practical problems in different aspects of the insurance industry. Some of the applications of clustering to insurance in the existing literature include customer segmentation in sales and marketing [Abolmakarem et al., 2016; Wang and Keogh, 2008], risk classification and prediction in underwriting risk assessments [Yeo et al., 2001], automatic fraud detection in claims management [Peng et al., 2006; Palacio, 2019], and ratemaking or pricing models in product design and pricing [Yao, 2008; Zou and Diehl, 2010].

The following sub-sections present a novel application of clustering techniques to the demographic data of group insurance plan members being insured for experience-rated group benefits. The clustering problem instance applicable in this case is such that the number of clusters k is not known predefined. Therefore the clustering algorithms suitable for this case include those that are designed to tackle clustering problems wherein the number of clusters are not known a priori. Clustering algorithms such as the newly proposed BIHCA can be applied to this clustering problem instance.

Given a set of N pooled demographic data vectors $X = (x_1, x_2, \dots, x_N)$, the clustering algorithm partitions the set of vectors into subsets $C = \{c_1, c_2, c_3, \dots, c_k\}$ such that data vectors with underlying similarity are grouped into the same subset. The goal is to uncover the clusters that can be used to derive summarized claim

experience data usable as the basis for estimating claim loss for insurable groups with non-existent or unavailable claims experience data. Instead of using the overall average rating of claims experience for these insurable groups, *we propose using a clustering technique* to derive summarization of existing claim experience data of other groups, and use that as the basis of implied claim experience for the insurable groups. To the best of our knowledge, this is the first application of clustering technique to group insurance datasets for such a goal.

6.4.1 Group insurance datasets

Benchmark datasets on group insurance benefits for clustering purposes are not readily available and securing industry participation to have access to industry data is equally challenging. For the experiments described in this research, we use a sample dataset available on kaggle.com (a machine learning and data science online community) for estimating or predicting insurance claim [Eason, 2018]. The dataset consists of demographic data of plan members for a health insurance benefit, as well as charges by the insurer and labels indicating if the member submitted a claim or not and the corresponding claim cost.

The version of the dataset used has 1338 samples and 8 attributes. Four of the attributes are originally numeric and the other four are numeric translations from non-numeric format. The attributes that are numeric include: *age*, *bmi* (body mass index - a ratio of body mass to height), *children* (number of children or dependants) and *charges* (individual amount billed by the insurance provider). The attributes that are originally non-numeric and the corresponding numeric translations include:

sex (female = 0, male = 1), *smoker* (non-smoker = 0, smoker = 1), *region* (residential area of insured in the US - northeast = 0, northwest = 1, southeast = 2, southwest = 3) and *insurance claim* (whether the insured submitted a claim or not - yes = 1, no = 0).

For the clustering process, six of the attributes (ordered as *age*, *bmi*, *children*, *sex*, *smoker* and *region*) are used to represent a data object for each plan member that is used in the clustering process and the corresponding *charges* attribute is used in the analysis of the uncovered clusters. Each data object is represented as an ordered attribute vector $x_i = (x_{i1}, x_{i2}, x_{i3}, x_{i4}, x_{i5}, x_{i6})$ with a corresponding *charges* value y_i . In our experiments, we use the *charges* attribute as a basis for estimating the claim expense since the amount charged by an insurer is typically a direct reflection of expected cost of covering a claim for a given plan member. Consequently, only an internal metric (such as silhouette coefficient) is used to assess the quality of the clusters uncovered by the clustering techniques - external evaluation metrics are not considered. The values of the *insurance claim* attribute in the dataset can be used as predefined labels when examining the dataset for binary clustering analysis (i.e. grouping the dataset into two categories - plan members that submit a claim, and those that did not).

6.4.2 Experimental setup (clustering group insurance dataset)

In the experiments, we applied the BIHCA clustering algorithm described in section 4.4 to the group insurance dataset described above. The goal was to uncover clusters within the group insurance demographic dataset. The parameters used for the

BIHCA algorithm in this experiment are the same as those used in the experiments described in Section 5.4 above.

For comparative analysis purposes, we also applied basic implementations of the DBSCAN and mean shift clustering algorithms (as implemented in the scikit-learn package) to the group insurance dataset - these algorithms are also applicable to the clustering problem in context, since the value of k is not predefined. The quality of the clusters uncovered by the algorithms are compared by evaluating the silhouette coefficient of uncovered clusters and the interpretation that can be inferred for claims estimation purposes. The results of the experiments are described and analyzed in the following subsection.

6.4.3 Results and analysis (clustering group insurance dataset)

In this subsection, we present the results and analysis of applying the three clustering techniques (DBSCAN, mean shift and BIHCA) to uncover clusters within the group insurance dataset. The records in the group insurance dataset are not labelled - the number of clusters k is not known a priori, as well as the cluster membership of the records. The clustering algorithms derive the value of k as an output of the algorithm.

Table 6.1 shows the number of clusters uncovered by each of the clustering algorithms and the corresponding silhouette coefficient value for the clusters. It is important to note that the accurate value for k is not predefined for the group insurance dataset, since the cluster membership of each data object is not known a priori.

The BIHCA demonstrate results that are competitive with the popular DBSCAN and mean shift algorithms. DBSCAN uncovers 4 clusters, but the silhouette coefficient is negative (-0.607), implying that the quality of the clusters is relatively poor as depicted in subsequent analysis below. Mean shift has the highest silhouette coefficient value (0.707), but the number of clusters uncovered is only 2. The number of clusters being 2 implies that there are only binary possibilities for the assignment of plan members in the group insurance dataset, which can be too limiting when classifying new plan member data objects. BIHCA shows competitive silhouette coefficient value of 0.510 and the number of clusters uncovered is 4. This implies that the clusters uncovered by BIHCA have competitive quality compared to those uncovered by the other algorithms and offer a wider spectrum for classifying new group insurance data object.

Table 6.1: Clustering result on group insurance dataset

Clustering Algorithms	Number of Clusters	Silhouette Coefficient
DBSCAN	4	-0.607
Mean shift	2	0.707
BIHCA	4	0.510

To derive the average claim estimate for each uncovered cluster \bar{X}_{c_j} , the corresponding value of the *charges* attribute y_i representing the policy members that belong to the same cluster c_j are summed together and divided by the size of the cluster (i.e. the number of policy members belonging to the cluster). The value of the derived average claim estimate for each cluster uncovered by the clustering algorithms are presented in Table 6.2 (for DBSCAN), Table 6.3 (for mean shift) and Table 6.4 (for BIHCA).

As shown in Table 6.2, the clusters uncovered using DBSCAN have uneven spread. Three out of the four clusters have a size of 3 and the fourth cluster has a size of 1329. Although, the range between the lowest average claim group (*Cluster 1*) and the highest average claim group (*Cluster 4*) is significant, the excessively higher size of *Cluster 4* makes the average claims estimate for that group (13301.50) not far from the overall average claims estimate for the entire pool (13270.42). Consequently, using these clusters as basis for classifying new group plan members does not offer significant advantage over using the overall average claims estimate.

Table 6.2: Average estimate of claims for clusters (DBSCAN)

Clusters	Size of Cluster	Avg Estimate of Claims for Cluster
Cluster 1	3	5798.84
Cluster 2	3	8584.02
Cluster 3	3	11658.07
Cluster 4	1329	13301.50

The two clusters uncovered using mean shift can be used to label new group insurance plan members as belonging to either the low (*Cluster 1*) or high (*Cluster 2*) claim group. The average estimated claims for both groups (9913.88 and 16142.80) are not significantly far from the overall average claim for the entire pool (13270.42).

Table 6.3: Average estimate of claims for clusters (mean shift)

Clusters	Size of Cluster	Avg Estimate of Claims for Cluster
Cluster 1	617	9913.88
Cluster 2	721	16142.80

Lastly, the clusters uncovered using the BIHCA algorithm provide a wider spectrum for classifying new group insurance plan members. The lowest average claim

group (*Cluster 1* - 7705.80) and the highest average claim group (*Cluster 4* - 17733.21) have claim estimates that are significantly lower and higher than the overall average claim (13270.42), respectively. Further, *Cluster 2* and *Cluster 3* have average claim estimates (12234.54 and 13642.86, respectively) that are around the overall average claim (13270.42), but still somewhat separated. These groups can serve as intermediary groups for classifying new group insurance plan members that fit in between the extremes of the low and high average claim groups. The fairly even sizes of the clusters uncovered by this algorithm equally gives a summarization of the original dataset that can be simply interpreted to correlate to high, medium and low claim cost plan members.

Table 6.4: Average estimate of claims for clusters (BIHCA)

Clusters	Size of Cluster	Avg Estimate of Claims for Cluster
Cluster 1	298	7705.80
Cluster 2	239	12234.54
Cluster 3	408	13642.86
Cluster 4	393	17733.21

In summary, compared to the other two algorithms, BIHCA uncovers clusters with competing quality and is feasible and applicable to practical, industry related applications.

Chapter 7

Discussion, Conclusion and Future Work

7.1 Discussion and Conclusion

Clustering is an important data-driven knowledge discovery technique that has been extensively researched and is finding increasing relevance as modern society is evolving into a data driven society. In this thesis, we reviewed metaheuristic algorithms used in clustering and the evolving concept of hybridizing these algorithms for synergy purposes. Existing forms of hybridizing metaheuristic algorithms (*collaborative* and *integrative*) involve combining high-level aspects of the individual algorithms. These hybridization strategies are incapable of exploiting the synergies that can be derived by combining low-level components of the individual algorithms.

In this research, we proposed a new paradigm for hybridizing metaheuristic algorithms - blended integrative hybridization (BIH). BIH enables hybrid algorithms to

leverage low-level strengths of algorithmic components from multiple algorithms to create improved techniques. This thesis presents the schema for BIH as a 3-phase process: identification phase, selection phase and blending phase (replacement and/or fusion). An application of the proposed schema in the design of a blended hybrid clustering algorithm is also provided in this thesis.

To establish the viability of BIH, we designed and implemented a blended integrative hybrid clustering algorithm (BIHCA) with ant clustering algorithm (ACA) and tabu search as the constituent metaheuristic algorithms. For comparative analysis, we also designed and implemented a collaborative hybrid version (collaborative hybrid clustering algorithm - CHCA) that combines the same constituent algorithms, as well as a basic ACA. Further, the proposed hybrid algorithm (BIHCA) is applicable to clustering problem instances where the number of clusters, k , is not known *a priori*. Therefore, we also compared the hybrid algorithm to two other popular clustering algorithms (DBSCAN and mean shift) that can tackle similar instances. These algorithms were applied to three clustering benchmark datasets (Iris, Wheat Seed and Wine) to uncover the clusters therein and compare their results.

The experimental results obtained show overall progress in the quality of the derived clusters from ACA to CHCA and then to BIHCA. Based on the empirical results, we infer that creating a hybrid algorithm using the BIH strategy offers promising results when compared with standalone version of the constituent algorithms or other hybrid version created using conventional hybridization strategy such as the collaborative hybridization strategy. This inference can be extended to other optimization problems that can be tackled using metaheuristic algorithms that have some com-

mon form of tactical or strategic tasks or sub-processes or components. Additional empirical results also show that the BIHCA, even in its basic form, can compete favorably with popular clustering algorithms (DBSCAN and means shift) with respect to finding the correct value of k and the quality of the derived clusters.

Finally, this thesis also presents a novel application of clustering techniques to estimate insurance claims cost for experience-rated group benefits. Compared to the prevailing industrial practice wherein an insurer's portfolio average is used to estimate claim cost for insurable groups with non-existent historical claims data, the proposed approach was able to categorize insured plan members having historical claims data into low, intermediate and high claims groups for a sample dataset. The average claims cost for these summarized representations are subsequently used as bases for estimating the claims for new insurable groups without historical claims data.

BIHCA, DBSCAN and mean shift were applied to a sample group insurance dataset to verify the viability of the novel claim estimation strategy and to establish the suitability of these algorithms in clustering group insurance datasets. In comparison to the other two algorithms, BIHCA uncovers clusters with competing quality and that can be used to derived more informed claim loss estimate for pricing experience-rated group benefits.

In conclusion, the newly proposed BIH is a potentially viable hybridization strategy that can be used in the design of hybrid algorithms that can be used to tackle difficult practical optimization problems. The clustering algorithm created using this strategy (BIHCA) show potentials that clustering algorithms designed using this strategy offers enhanced synergy that better leverages the strength of the constituent

algorithms in comparison to existing hybridization strategies. The ability of BIHCA to leverage low-level strengths of component algorithmic components from multiple algorithms offer design flexibility that can be adapted to suite difficult clustering problems in industry such as insurance.

7.2 Future Work

The newly proposed approach to hybridization (BIH) creates a new paradigm for the design of hybrid metaheuristic algorithms with the possibility of creating new sets of hybrid algorithms that can more effectively leverage low-level strengths of popular individual metaheuristics.

For future work, We will undertake an analytical study of popular algorithms (e.g. simulated annealing, genetic algorithms, particle swarm optimization, cuckoo search) to identify combinations of these algorithms that have common algorithmic components and that can be blended together for better performance. We will also investigate the applicability of the BIH to the design of hybrid algorithms for other difficult optimization problems (e.g. feature selection, classification, portfolio optimization, crew scheduling) with practical industrial applications.

Considering that this research has established the suitability of BIHCA for clustering group insurance datasets, we will also be exploring the creation of classifiers that will be based on cluster labels uncovered using BIHCA. Combining the right classifier with clustering results using BIHCA can be developed into systems that can meet an immediate industrial need that is currently a niche area in group insurance benefit ratemaking.

Noting that the insurance industry is still largely dependent on established complex statistical models for ratemaking, and is only gradually embracing machine learning techniques such as clustering to augment this complex models. This, we walso plan to investigate more classic problems (such as feature selection) in the insurance industry where BIHCA clustering algorithm can be applicable. We will be investigating the applicability of BIHCA to clustering other insurance data (individual life insurance, auto insurance, travel insurance, and property and casualty insurance).

Bibliography

- S. Abolmakarem, F. Abdi, and K. Khalili-Damghani. Insurance customer segmentation using clustering approach. *International Journal of Knowledge Engineering and Data Mining*, 4(1):18–39, 2016.
- B. Agarwal and N. Mittal. Optimal feature selection for sentiment analysis. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 13–24. Springer, 2013.
- K. S. Al-Sultan. A tabu search approach to the clustering problem. *Pattern recognition*, 28(9):1443–1451, 1995.
- K. S. Al-Sultan and C. A. Fedjki. A tabu search-based algorithm for the fuzzy clustering problem. *Pattern Recognition*, 30(12):2023–2030, 1997.
- M. Aldenderfer, R. Blashfield, R. Blashfield, SAGE., and i. Sage Publications. *Cluster Analysis*. Number no. 44 in Cluster Analysis. SAGE Publications, 1984. ISBN 9780803923768. URL <https://books.google.ca/books?id=ZIARBoJQxzcC>.
- J. A. Attwood. *Group Insurance Plans*, pages 665–669. Springer US, Boston, MA,

1980. ISBN 978-1-4684-1449-3. doi: 10.1007/978-1-4684-1449-3_30. URL https://doi.org/10.1007/978-1-4684-1449-3_30.
- J. D. Banfield and A. E. Raftery. Model-based gaussian and non-gaussian clustering. *Biometrics*, pages 803–821, 1993.
- W. F. Bluhm, R. B. Cumming, A. D. Ford, J. E. Lusk, and P. L. Perkins, editors. *Group Insurance (Fourth Edition)*. Actex Publications, Winsted, Connecticut, 2003.
- C. Blum, J. Puchinger, G. R. Raidl, and A. Roli. Hybrid metaheuristics in combinatorial optimization: A survey. *Applied Soft Computing*, 11(6):4135–4151, 2011.
- E. Bonabeau, M. Dorigo, and G. Theraulaz. *Swarm Intelligence : From Natural to Artificial Systems*. Santa Fe Institute Studies on the Sciences of Complexity. Oxford University Press, 1999.
- U. Boryczka. Ant clustering algorithm. In *Intelligent Information Systems*, pages 377–386, 2008.
- M. Brusco and H.-F. Köhn. Optimal partitioning of a data set based on the p-median model. *Psychometrika*, 73:89–105, 2008.
- P. Cheeseman, M. Self, J. Kelly, W. Taylor, D. Freeman, and J. C. Stutz. Bayesian classification. In *AAAI*, volume 88, pages 607–611, 1988.
- Y.-F. Chen, C. A. Fattah, Y.-S. Liu, and G. Yan. Hdacc: A heuristic density-based ant colony clustering algorithm. In *IAT*, pages 397–400. IEEE Computer Society, 2004. ISBN 0-7695-2101-0.

- Y.-C. Chiou and L. W. Lan. Genetic clustering algorithms. *European Journal of Operational Research*, 135(2):413–427, 2001.
- C.-Y. Chiu and C.-H. Lin. Cluster analysis based on artificial immune system and ant algorithm. In J. Lei, J. Yao, and Q. Zhang, editors, *Third International Conference on Natural Computation (ICNC 2007)*, pages 647–650. IEEE Computer Society, 2007.
- S. C. M. Cohen and L. N. de Castro. Data clustering with particle swarms. In *IEEE Congress on Evolutionary Computation*, pages 1792–1798. IEEE, 2006. ISBN 0-7803-9487-9.
- D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(5):603–619, May 2002. ISSN 0162-8828.
- S. Das, A. Abraham, and A. Konar. *Metaheuristic Clustering*, volume 178. Springer Verlag, 2009.
- H. S. Denenberg. The legal definition of insurance: Insurance principles in practice. *The Journal of Insurance*, 30(3):319–343, 1963. ISSN 10473483. URL <http://www.jstor.org/stable/250245>.
- J.-L. Deneubourg, S. Goss, N. Franks, A. Sendova-Franks, C. Detrain, and L. Chrétien. The dynamics of collective sorting robot-like ants and ant-like robots. In *Proceedings of the first international conference on simulation of adaptive behavior on From animals to animats*, pages 356–363, 1991.

- D. V. der Merwe and A. P. Engelbrecht. Data clustering using particle swarm optimization. In *Evolutionary Computation, 2003. CEC'03. The 2003 Congress on*, volume 1, pages 215–220. IEEE, 2003.
- M. Dorigo, G. A. D. Caro, and L. M. Gambardella. Ant algorithms for discrete optimization. *Artificial Life*, 5:137–172, 1999.
- D. Dua and C. Graff. UCI machine learning repository, 2017. URL <http://archive.ics.uci.edu/ml>.
- L. Duczmal and R. M. Assunção. A simulated annealing strategy for the detection of arbitrarily shaped spatial clusters. *Computational Statistics & Data Analysis*, 45(2):269–286, 2004.
- R. O. Duda, P. E. Hart, et al. *Pattern classification and scene analysis*, volume 3. Wiley New York, 1973.
- Eason. Sample insurance claim prediction dataset, May 2018. URL <https://www.kaggle.com/easonlai/sample-insurance-claim-prediction-dataset/version/1>. Accessed: 2020-05-30.
- A. A. A. Esmin and S. Matwin. Data clustering using hybrid particle swarm optimization. In H. Yin, J. A. F. Costa, and G. Barreto, editors, *Intelligent Data Engineering and Automated Learning - IDEAL 2012*, pages 159–166, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.
- M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters a density-based algorithm for discovering clusters in large spa-

- tial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, pages 226–231. AAAI Press, 1996a.
- M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *KDD*, volume 96, pages 226–231, 1996b.
- B. Everitt, S. Landau, M. Leese, and D. Stahl. *Cluster analysis*. Wiley, 5th edition, 2011.
- M. Filippone, F. Camastra, F. Masulli, and S. Rovetta. A survey of kernel and spectral methods for clustering. *Pattern recognition*, 41(1):176–190, 2008.
- C. Fraley and A. E. Raftery. How many clusters? which clustering method? answers via model-based cluster analysis. *The computer journal*, 41(8):578–588, 1998.
- M. Fras. The group insurance contract in private international law. *Netherlands International Law Review*, 11 2019. doi: 10.1007/s40802-019-00146-2.
- E. Frees. Loss data analytics, 2018.
- F. Glover and M. Laguna. *Tabu Search*. Springer, 2013.
- S. Guha, R. Rastogi, and K. Shim. Rock: A robust clustering algorithm for categorical attributes. *Information systems*, 25(5):345–366, 2000.
- Z. Güngör and A. Ünler. K-harmonic means data clustering with simulated annealing heuristic. *Applied Mathematics and Computation*, 184(2):199–209, 2007.

- Z. Güngör and A. Ünler. K-harmonic means data clustering with tabu-search method. *Applied Mathematical Modelling*, 32(6):1115–1125, 2008.
- A. Hamdi, N. Monmarché, M. A. Alimi, and M. Slimane. Swarmclass: A novel data clustering approach by a hybridization of an ant colony with flying insects. In M. Dorigo, M. Birattari, C. Blum, M. Clerc, T. Stützle, and A. F. T. Winfield, editors, *Ant Colony Optimization and Swarm Intelligence*, pages 411–412, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.
- A. Hamdi, M. Slimane, N. Monmarché, and A. M. Alimi. Flyantclass: Intelligent move for ant based clustering algorithm. In *2016 IEEE/ACS 13th International Conference of Computer Systems and Applications (AICCSA)*, pages 1–8, Nov 2016.
- J. Handl and B. Meyer. Ant-based and swarm-based clustering. *Swarm Intelligence*, 1(2):95–113, 2007. ISSN 1935-3812.
- J. Handl, J. D. Knowles, and M. Dorigo. On the performance of ant-based clustering. In A. Abraham, M. Köppen, and K. Franke, editors, *HIS*, volume 105 of *Frontiers in Artificial Intelligence and Applications*, pages 204–213. IOS Press, 2003. ISBN 1-58603-394-8.
- J. Handl, J. Knowles, and M. Dorigo. Ant-based clustering and topographic mapping. *Artificial Life*, 12(1):35–61, 2006.
- J. A. Hartigan. *Clustering algorithms*. Wiley, NY, USA, 1975.
- M. J. A. Hasan and S. Ramakrishnan. A survey: hybrid evolutionary algorithms for cluster analysis. *Artif. Intell. Rev.*, 36(3):179–204, 2011.

- N. V. Haueter. Swiss Reinsurance Company Ltd. "A history of insurance". https://www.swissre.com/dam/jcr:64b0fdca-f4d8-401c-a5bd-9c51614843c0/150Y_Markt_Broschuere_Canada_web.pdf, 2017. Accessed: June 6, 2020.
- H. He and Y. Tan. A two-stage genetic algorithm for automatic clustering. *Neurocomputing*, 81:49–59, 2012.
- A. Hinneburg and H.-H. Gabriel. Denclue 2.0: Fast clustering based on kernel density estimation. In *International symposium on intelligent data analysis*, pages 70–80. Springer, 2007.
- T. Hofmann, B. Schölkopf, and A. J. Smola. Kernel methods in machine learning. *The annals of statistics*, pages 1171–1220, 2008.
- L. Hubert and P. Arabie. Comparing partitions. *Journal of Classification*, 2(1):193–218, Dec 1985. ISSN 1432-1343.
- O. Hyrien and A. Baran. Fast nonparametric density-based clustering of large datasets using a stochastic approximation mean-shift algorithm. *Journal of Computational and Graphical Statistics*, 25(3):899–916, 2016.
- A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM computing surveys (CSUR)*, 31(3):264–323, 1999.
- J. Jang and H. Jiang. DBSCAN++: Towards fast and scalable density clustering. In K. Chaudhuri and R. Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 3019–3029, Long Beach, California, USA, 09–15 Jun 2019. PMLR.

- L. Jiang and D. Xie. An efficient differential memetic algorithm for clustering problem. *IAENG International Journal of Computer Science*, 45(1), 2018.
- P. M. Kanade and L. O. Hall. Fuzzy ants and clustering. *IEEE Trans. Systems, Man, and Cybernetics, Part A*, 37(5):758–769, 2007.
- G. Karypis, E.-H. Han, and V. Kumar. Chameleon: Hierarchical clustering using dynamic modeling. *Computer*, 32(8):68–75, 1999.
- L. Kaufman and P. J. Rousseeuw. *Finding groups in data: an introduction to cluster analysis*, volume 344. Wiley-Interscience, 2009.
- J. D. Kelleher and B. Tierney. *Data Science*. MIT Press, 2018.
- V. Kotu and B. Deshpande. Chapter 13 - anomaly detection. In *Data Science (Second Edition)*, pages 447 – 465. Morgan Kaufmann, second edition edition, 2019. ISBN 978-0-12-814761-0.
- K. Krishna and M. N. Murty. Genetic k-means algorithm. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 29(3):433–439, 1999.
- P. Kuntz, P. Layzell, and D. Snyers. A colony of ant-like agents for partitioning in vlsi technology. In *Proceedings of the Fourth European Conference on Artificial Life*, pages 417–424. MIT Press, Cambridge, MA, 1997.
- C.-Y. Lee and E. Antonsson. Dynamic partitional clustering using evolution strategies. In *Industrial Electronics Society, 2000. IECON 2000. 26th Annual Conference of the IEEE*, volume 4, pages 2716–2721. IEEE, 2000.

- J. Li, H. Fan, D. Yuan, and C. Zhang. Kernel function clustering based on ant colony algorithm. In *Proceedings - 4th International Conference on Natural Computation, ICNC 2008*, volume 7, pages 645 – 649, 11 2008.
- Y. Liu, Z. Li, H. Xiong, X. Gao, and J. Wu. Understanding of internal clustering validation measures. In *2010 IEEE International Conference on Data Mining*, pages 911–916. IEEE, 2010.
- Y. Y. Liu, P. Thulasiraman, and R. K. Thulasiram. Parallelizing active memory ants with mapreduce for clustering financial time series data. In *Proceedings of IEEE International Conferences on Big Data and Cloud Computing (BDCloud), Social Computing and Networking (SocialCom), Sustainable Computing and Communications (SustainCom)(BDCloud-SocialCom-SustainCom)*, pages 137–144. IEEE, 2016.
- Y. Lu, S. Lu, F. Fotouhi, Y. Deng, and S. J. Brown. Fgka: a fast genetic k-means clustering algorithm. In H. Haddad, A. Omicini, R. L. Wainwright, and L. M. Liebrock, editors, *SAC*, pages 622–623. ACM, 2004. ISBN 1-58113-812-1.
- A. V. Lukashin and R. Fuchs. Analysis of temporal gene expression profiles: clustering by simulated annealing and determining the optimal number of clusters. *Bioinformatics*, 17(5):405–414, 2001.
- E. D. Lumer and B. Faieta. Diversity and adaptation in populations of clustering ants. In *Proceedings of the third international conference on Simulation of adaptive behavior: from animals to animats 3: from animals to animats 3*, pages 501–508. MIT Press, 1994.

- J. MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297, 1967.
- C. D. Manning, P. Raghavan, and H. Schütze. *Hierarchical clustering*, page 346–368. Cambridge University Press, 2008. doi: 10.1017/CBO9780511809071.018.
- M. H. Margolin. On the credibility of group insurance claim experience. *Transactions of Society of Actuaries*, page 229, 1971.
- R. Martí, M. Laguna, and F. W. Glover. Principles of tabu search. In *Handbook of Approximation Algorithms and Metaheuristics*, 2007.
- U. Maulik and S. Bandyopadhyay. Genetic algorithm-based clustering technique. *Pattern Recognition*, 33(9):1455–1465, 2000.
- N. Monmarché. On data clustering with artificial ants. In *AAAI-99 & GECCO-99 Workshop on Data Mining with Evolutionary Algorithms: Research Directions*, pages 23–26, 1999.
- N. Monmarché, M. Slimane, and G. Venturini. On improving clustering in numerical databases with artificial ants. In D. Floreano, J.-D. Nicoud, and F. Mondada, editors, *ECAL*, volume 1674 of *Lecture Notes in Computer Science*, pages 626–635. Springer, 1999. ISBN 3-540-66452-1.
- C. A. Murthy and N. Chowdhury. In search of optimal clusters using genetic algorithms. *Pattern Recognition Letters*, 17(8):825–832, 1996.

- M. Naylor. *Key Technological Disruptors*, pages 15–40. Springer International Publishing, Cham, 2017a. ISBN 978-3-319-63835-5. doi: 10.1007/978-3-319-63835-5_2. URL https://doi.org/10.1007/978-3-319-63835-5_2.
- M. Naylor. *Types of Insurance*, pages 41–45. Springer International Publishing, Cham, 2017b. doi: 10.1007/978-3-319-63835-5_3. URL https://doi.org/10.1007/978-3-319-63835-5_3.
- M. Ng. A parallel tabu search heuristic for clustering data sets. In *Parallel Processing Workshops, 2003, International Conference on*. IEEE, 2003.
- M. K. Ng and J. C. Wong. Clustering categorical data sets using tabu search techniques. *Pattern Recognition*, 35(12):2783–2790, 2002.
- T. Niknam, B. Amiri, J. Olamaei, and A. Arefi. An efficient hybrid evolutionary optimization algorithm based on pso and sa for clustering. *Journal of Zhejiang University-SCIENCE A*, 10:512–519, 2009.
- O. I. Oduntan, P. Thulasiraman, and R. Thulasiram. Portfolio diversification using ant brood sorting clustering. In *Sixth World Congress on Nature and Biologically Inspired Computing, NaBIC 2014, Porto, Portugal, July 30 - August 1, 2014*, pages 256–261. IEEE, 2014.
- S. Palacio. Abnormal pattern prediction: Detecting fraudulent insurance property claims with semi-supervised machine-learning. *Data Science Journal*, 18:35, 07 2019.
- Y. Peng, G. Kou, A. Sabatka, Z. Chen, D. Khazanchi, and Y. Shi. Application

- of clustering methods to health insurance fraud detection. In *2006 International Conference on Service Systems and Service Management*, volume 1, pages 116–120, 2006.
- A. L. Pereira, L. De Castro, E. Hruschka, and R. Gudwin. Towards improving clustering ants: An adaptive ant clustering algorithm. *Informatica (Slovenia)*, 29:143–154, 06 2005.
- M. Qasem. *Bio-inspired constrained clustering: A case study on aspect-based sentiment analysis*. PhD thesis, University of Manitoba, 2018.
- M. Qasem and P. Thulasiraman. Evaluation and validation of semi-supervised ant-inspired sentence-level sentiment prediction clustering. In *IEEE CEC*, Wellington, New Zealand, June 2019.
- M. Qasem, Y. Ying, Z. Wang, P. Thulasiraman, and R. Thulasiram. Enhancing ant brood clustering with adaptive radius of perception and non-parametric estimation on multi-core architectures. In *Advances in Intelligent Networking and Collaborative Systems. INCoS 2017*. Springer, 2018.
- J. Qu and X. Liu. A quick ant clustering algorithm. In *Fourth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD 2007)*, volume 1, pages 722–725, Aug 2007.
- V. V. Raghavan and K. Birchard. A clustering strategy based on a formalism of the reproductive process in natural systems. In *ACM SIGIR Forum*, volume 14, pages 10–22. ACM, 1979.

- M. A. Rahman and M. Z. Islam. A hybrid clustering technique combining a novel genetic algorithm with k-means. *Knowledge-Based Systems*, 71:345 – 365, 2014. ISSN 0950-7051.
- L. Rokach and O. Maimon. Clustering methods. In *Data Mining and Knowledge Discovery Handbook*, pages 321–352. Springer, 2005.
- A. Rosenberg and J. Hirschberg. V-measure: A conditional entropy-based external cluster evaluation measure. In *Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning (EMNLP-CoNLL)*, pages 410–420, 2007.
- P. J. Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65, 1987.
- M. Sarkar, B. Yegnanarayana, and D. Khemani. A clustering algorithm using an evolutionary programming-based approach. *Pattern Recognition Letters*, 18(10): 975–986, 1997.
- S. Z. Selim and K. Alsultan. A simulated annealing algorithm for the clustering problem. *Pattern recognition*, 24(10):1003–1008, 1991.
- J. Shawe-Taylor, N. Cristianini, et al. *Kernel methods for pattern analysis*. Cambridge university press, 2004.
- A. S. Shirkhorshidi, S. Aghabozorgi, and T. Y. Wah. A comparison study on similarity

- and dissimilarity measures in clustering continuous data. *PLOS ONE*, 10(12):1–20, 12 2015. doi: 10.1371/journal.pone.0144059.
- K. Sörensen and F. W. Glover. *Metaheuristics*, pages 960–970. Springer US, Boston, MA, 2013.
- C. S. Sung and H. W. Jin. A tabu-search-based heuristic for clustering. *Pattern Recognition*, 33(5):849–858, 2000.
- P.-N. Tan, M. Steinbach, and V. Kumar. *Introduction to data mining*. Pearson Addison Wesley, MA, USA, 2006.
- M. Turkensteen and K. A. Andersen. A tabu search approach to clustering. In B. Fleischmann, K.-H. Borgwardt, R. Klein, and A. Tuma, editors, *Operations Research Proceedings 2008*, pages 475–480, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.
- J. Tvrdik and I. Křivý. Hybrid differential evolution algorithm for optimal clustering. *Applied Soft Computing*, 35:502–512, 2015.
- S. W. J. van der Merwe. The concept of insurance and the insurance contract. *The Comparative and International Law Journal of Southern Africa*, 3(2):149–167, 1970. ISSN 00104051. URL <http://www.jstor.org/stable/23240782>.
- N. X. Vinh, J. Epps, and J. Bailey. Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. *Journal of Machine Learning Research*, 11(Oct):2837–2854, 2010.

- C. S. Wallace and D. L. Dowe. Intrinsic classification by mml-the snob program. In *Proceedings of the 7th Australian Joint Conference on Artificial Intelligence*, volume 37, page 44. World Scientific, 1994.
- F. Wang, D. Zhang, and N. Bao. Fuzzy document clustering based on ant colony algorithm. In W. Yu, H. He, and N. Zhang, editors, *Advances in Neural Networks – ISNN 2009*, pages 709–716, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.
- X. Wang and E. Keogh. *A Clustering Analysis for Target Group Identification by Locality in Motor Insurance Industry*, pages 113–127. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008. ISBN 978-3-540-79005-1. doi: 10.1007/978-3-540-79005-1_7. URL https://doi.org/10.1007/978-3-540-79005-1_7.
- Z. Weili. An improved entropy-based ant clustering algorithm. In *2009 WASE International Conference on Information Engineering*, volume 2, pages 41–44, July 2009.
- W. J. Welch. Algorithmic complexity: three np- hard problems in computational statistics. *Journal of Statistical Computation and Simulation*, 15(1):17–25, 1982.
- G. Werner, C. Modlin, and W. T. Watson. *Basic Ratemaking*. 2016. URL https://www.casact.org/library/studynotes/Werner_Modlin_Ratemaking.pdf.
- D. Xu and Y. Tian. A comprehensive survey of clustering algorithms. *Annals of Data Science*, 2(2):165–193, 2015.
- X.-S. Yang. Optimization and metaheuristic algorithms in engineering. In X.-S. Yang, A. H. Gandomi, S. Talatahari, and A. H. Alavi, editors, *Metaheuristics in Water*,

- Geotechnical and Transport Engineering*, pages 1 – 23. Elsevier, Oxford, 2013. ISBN 978-0-12-398296-4.
- Y. Yang and M. S. Kamel. An aggregated clustering approach using multi-ant colonies algorithms. *Pattern Recognition*, 39(7):1278 – 1289, 2006. ISSN 0031-3203.
- J. Yao. Clustering in ratemaking: Applications in territories clustering. Discussion Paper Program - Applying Multivariate Statistical Models, 2008. URL <https://www.casact.org/pubs/dpp/dpp08/08dpp170.pdf>.
- A. Yeo, K. Smith-Miles, R. Willis, and M. Brooks. Clustering technique for risk classification and prediction of claim costs in the automobile insurance industry. *Intelligent Systems in Accounting, Finance and Management*, 10:39–50, 03 2001.
- J. Zhang, M. Marszałek, S. Lazebnik, and C. Schmid. Local features and kernels for classification of texture and object categories: A comprehensive study. *International journal of computer vision*, 73(2):213–238, 2007.
- T. Zhang, R. Ramakrishnan, and M. Livny. Birch: an efficient data clustering method for very large databases. In *ACM SIGMOD Record*, volume 25, pages 103–114, 1996.
- Q. Zou and R. Diehl. Modifying insurance rating territories via clustering. In *Proceedings of the Northeast SAS Users Group: Health Care and Life Sciences*, Baltimore, Maryland, USA, Nov 14–17 2010.