Optimizer augmented clustering: a study over iterative k-means

A thesis submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy

Subhajit Ghosh

Registration No. 007 of 2011

School of Engineering

Department of Computer Science and Engineering

Tezpur University

February, 2011

Dedicated to Dadu, Dida, Mejoma & Tuwa

The void created by your absence can never be filled

To my in-laws

Fate willed that we never meet, yet your blessings are a constant source of Encouragement in our lives...

Abstract

Clustering often forms the first-stage analysis before applying other data mining techniques. By performing a cluster analysis, the user can ideally gain an overview on the major characteristics of a data set without any previous knowledge. However, in practice, performing a cluster analysis is often not easy, since most clustering algorithms require numerous input parameters. The selection of seeds for K-Medoids or the numbers of clusters for K-means are a few examples of input parameters. Without background knowledge on the data, it is often difficult to find a suitable parameterization. Often, parameters need to be adjusted in a time consuming trial and error procedure. It cannot be guaranteed that a useful parameterization can be detected by doing so. Outliers and noise points in real world data additionally complicate the search for a suitable parameterization.

The performance of several popular clustering algorithms such as K-means also depends on the choice of several user-defined parameters such as how the initial points would be selected, or which distance measure to use to compute the similarity of the data points. It is in this context that we define a *free parameter* in clustering. Many clustering algorithms have one or more parameters that must be set according to the problem at hand. These are *free parameters*, and influence the process of clustering. As an example, the clustering algorithms like CURE, shifting grid, training neural networks, global FCM, and evidential C-Means (ECMs) are sensitive to the initial parameter settings, namely, the initialization. One can say that the initial parameter choices for these algorithms are all *free parameters*.

In our work we propose a optimizer augmented clustering (OAC), i.e., a method employing an inner optimization loop, in addition to the clustering algorithm, to achieve higher clustering performance through a better choice of these *free parameters*. As a case study, we have tried to achieve better centroids for the iterative K-means algorithm. The improvement achieved replacing mean computations for new centroids in successive iterations of K-means (and thereafter of Maximin) with more optimized values obtained from different optimizing techniques have been investigated. An experimental study is presented in the domain of K-means and Maximin. We explore four different optimization techniques to optimize the centroids of the two algorithms.

Initial experiments were conducted in the domain of textual data BIGCHECK. For the document source, we used 1060 records created for journal articles from the Information Science and

Abstracts (ISA) database and computer science technical reports collected from various sites on the Internet. This is a comprehensive subset of the entire database. Each document record includes a complete abstract, title, author, and subject keywords. BIGCHECK is a text dataset of 1060 records of ACM Citations. Four optimizing techniques viz., *Genetic Algorithm (GA) Steady State, Simulated Annealing, Differential Evolution (DE) & Particle Swarm Optimization (PSO)* were used in an inner loop to arrive at better centroids compared with an independent K-Means run. The performance of these methods and an independent K-means run are observed for a fixed number of generations. The results clearly demonstrate that a superior clustering performance (as measured by an appropriate, suitably defined metric) is achieved with such an optimization-based clustering approach, as compared to a normal single-level clustering using a fixed choice of the parameters. Results indicated GA & DE gives the best values, with PSO performing rather poorly.

To assess the improvement in cluster quality of the methods, we use a quantitative measure *Silhouette coefficient (SC)*. Three datasets were used for the experimentation viz., synthetic dataset BIGCHECK (ACM citation dataset used in earlier experiment), and two standard datasets SOYBEAN & WATER PLANT TREATMENT. In addition to the earlier four methods, two more GA methods were used: *Roulette Method & Grouping GA (GGA)*. GGA emerges as the most effective technique for arriving at better centroids of K-Means for high dimensional datasets. Using GGA improves cluster quality by a substantial margin, for instance, from SC value of 0.32 to .78 for the WATERPLANT TREATMENT dataset. In general, GA & DE followed by SA gives much improved values. As in the first experiment, here too PSO lagged behind the other methods used in improving the centroids.

The experiments reported in the thesis have been done using Euclidean distance for computing the similarity of points. For high dimensional data, Pearson Correlation Coefficient is an effective similarity measure. The same experiments for computing silhouette coefficients were also performed using Pearson distance as the distance measure (instead of Euclidean) for K-Means & other hybrid optimized schemes. The results obtained are similar.

Consequent to the experiments with silhouette coefficient, another well-known validity index, Davies Bouldin index , have been used to assess the results. The results obtained by this index also confirm that GGA is the best optimizer, while simulated annealing is the second best performer for the analyzed problem.

To estimate the performance of these optimizers on other clustering algorithms, a second clustering algorithm, Maximin clustering algorithm, was chosen and the set of experiments with the same set of optimizers were performed for two datasets viz., BIGCHECK & WATER TREATMENT PLANT. More or less similar results were obtained when Maximin replaced K-Means as the algorithm under study.

Keywords - Clustering, Optimization, GA, SA, PSO, DE, fitness function, Silhouette Coefficient, Davies Bouldin Index, Euclidean Distance, Pearson distance, K-Means, Maximin

TEZPUR UNIVERSITY

Certificate of the Supervisor

This is to certify that the thesis entitled **Optimizer augmented clustering: a study over iterative k-means** submitted to Tezpur University in the Department of Computer Science and Engineering under the School of Engineering in partial fulfillment of the award of the degree of Doctor of Philosophy in Computer Science and Engineering is a record of research work carried out by Mr. Subhajit Ghosh under our supervision and guidance.

All helps received by him from various sources have been duly acknowledged.

No part of this thesis has been produced elsewhere for award of any other degree.

Signature of Co-Supervisor
(SNEHASIS MUKHOPADHYAY)

Designation: PROFESSOR

School: SCIENCE

Department: COMPUTER & INFORMATION SCIENCE, IUPUI

Date: OCTOBER 24, 2011

Place: INDIANAPOLIS, INDIANA, USA

Signature of Supervisor

Designation: Professor

School : Engineering

Department: Computer Science and Engineering

Date:

Place:

DECLARATION

Subhajit Ghosh, hereby declare that the thesis entitled "*Optimizer augmented clustering ... study over iterative k-means*" submitted to the Department of Computer Science and ...gineering under the School of Engineering, Tezpur University, in partial fulfillment of ... requirements for the award of the degree of Doctor of Philosophy in Computer Science ...d Engineering, is based on bona fide work carried out by me. The results embodied in ... Thesis have not been submitted in part or in full, to any other university or institute for ...ard of any degree or diploma.

(Subhajit Ghosh)

Acknowledgements

I would like to convey my sincere gratitude to my guide, Dr. Shyamanta M. Hazarika, Professor, Dept of Computer Sc and Engineering, Tezpur University (TU), whose guidance and insightful suggestions helped me in all aspects of my research. I really learnt the art of visualization and approach of a solution of a complex problem from him. The dissertation would also have been incomplete without the assistance of my co-supervisor, Dr. Snehashis Mukhopadhyay, Professor, Dept. of Computer and Information Sciences, Indiana University-Purdue University, Indianapolis. I would also like to thank Dr. Utpal Sharma, HOD of Computer Science, TU, for the assistance offered during the period of investigation. I would also like to thank Prof. Malay Dutta, Prof. D.K.Bhattacharyya, Prof. D. K. Saikia, Prof. Nrityanand Sharma, Prof. Sarat Saharia, Sr. Lecturer Bhabesh Nath, researcher Juwesh Binong, Achintya Sharma, Nayan Kakoty and Sajid Nagi, and my previous employers and colleagues for the help and support extended at various stages of my research. I appreciate the suggestion provided by my colleague Mr. Manoj Jain to crack certain programming intricacies which arose at the initial stages. I am grateful to my M.Tech students who took my courses on Data Mining and Warehousing, and Soft Computing which helped me to enhance my knowledge in the area. I would also like to acknowledge all the authors of the articles and the books and other reference material listed at the end.

I appreciate the work of the Machine Intelligence Unit of Indian Statistical Institute, Kolkata for the various schools they had organized and in which I had participated. I am thankful for the fruitful discussions I have had with many of its researchers and others at various stages of my work, and also acknowledge the Center for Soft Computing Research: A National Facility at Indian Statistical Institute, Kolkata for allowing me to visit their Center and utilize the resources therein. Technical Assistant Ajay of Computer Sc. dept, TU has been ever helpful in certain technical matters as has been the staff of the TU CSE dept office. Saurabh Sharma and Rosy Das, Assistant Professor at CSE dept, TU, demonstrated their methodology of handling clustering schemes when I was struggling during the coding phase which proved helpful....

I would also like to say 'thank you' to my family, colleagues and friends for their patience and support. Last, but certainly not the least, my six and a half year old son, Amartya Vishnu Ghosh

(Bonbon), for whom *I have promises to keep* and whether I walk a few yards before I sleep is something only the future will unfold and posterity will judge...

I would be happy if the present work is of some significance to anyone exploring the fascinating world of clustering and optimization and it's far reaching impact in our everyday life.

Contents

List of Tables

List of Figures

List of Algorithms

Chapter 1

Introduction

In this chapter, we provide an introduction to our work. The chapter is organized as follows. Section 1.1 outlines the motivation of the work. In Section 1.2, we provide the problem statement. In the next section, the relevance of the work is highlighted. In Section 1.4, we deal with the contribution of the thesis. In Section 1.5, we outline the content of the dissertation.

One of the most important tasks in the data mining process is that of clustering. **Clustering** is an **unsupervised learning** technique and aids in assorting the data into **similar** and **dissimilar** groups. The groups are called clusters. In the context of data mining, where one is handling gigantic sizes of databases, clustering can identify dense and sparse regions and therefore discover overall distribution patterns and interesting correlations among data attributes.

Clustering consists of four basic steps[127]:

1. *Feature Selection or Extraction.* Feature Selection chooses distinguishing features from a set of candidates, while feature extraction utilizes some transformations to generate useful and novel features.

2. *Clustering algorithm design or selection.* The step is generally combined with the proximity measure selection and the criterion function construction. The proximity measure directly affects the formation of the resulting clusters. Once it is chosen, the clustering criterion construction makes the partition of clusters an optimization problem, which is well defined mathematically.

3. *Cluster validation.* Effective evaluation standards and criteria are important to provide the users with a degree of confidence for the clustering results derived from the used algorithms.

4. *Results interpretation.* Experts in the relevant fields interpret the data partition. Further analysis, even experiments, may be required to guarantee the reliability of extracted knowledge.

There are two major styles of clustering: *partitioning* (often called *k-clustering*), in which every object is assigned to exactly one group, and *hierarchical clustering*, in which each group of size greater than one is in turn composed of smaller groups. Both hierarchical clustering and k-clustering had been studied extensively by the mid-1970s, and comparatively little clustering research was carried out in the 1980s. In recent years, however, the advent of the World Wide Web search engines (and specifically the problem of organizing the large amount of data they produce) and the concept of "data mining" massive databases has lead to a renewal of interest in clustering algorithms [3]

1.1 Motivation

The utility of optimization in data mining algorithms is far-reaching. Take the case of Regression analysis which in its simplest form involves building a predictive model to relate a predictor variable, X, to a response variable Y through a relationship of the form Y= aX+b. The optimization algorithm can be quite simple but nonetheless essential in the case of Linear Regression: a and b can be expressed as explicit functions of the observed values of 'spending' and 'income' [56].

Unsupervised learning means that only the intrinsic structure of the data defines the groups of data. The difficulty here is that usually there is **no a priori information about the structure of the data set or potential parameters.** To solve this problem, many methods make assumptions or select a model to fit the data. Usually, the assumptions are made according to the distribution of data set or the shape of data set. For example, typically K-means algorithms assume that the shapes of clusters are convex. For the same reason, another method uses a set of parameters to control the performances of the algorithms. But these kinds of **parameters are usually set manually and hard to find the ideal values.** To get the good results, the user needs to run the algorithms several times, moreover, in some cases, it is difficult to decide on which is the best result because of the lack of information on the data set. All the above reasons can make the clustering algorithms inefficient and unstable.

Clustering often forms the first-stage analysis before applying other data mining techniques. It is important in many (perhaps most) domains that the clustering algorithm is fast and must be able to handle very large datasets. In fact, clustering is often used as a pruning mechanism so that un-interesting and obvious clusters as well as outliers can be discarded before proceeding with further analysis [82]. By performing a cluster analysis, the user can ideally gain an overview on the major characteristics of a data set without any previous knowledge. However, in practice, **performing a cluster analysis is often not easy, since most clustering algorithms require numerous input parameters whose values affect the quality of clustering results** [91]. The selection of seeds for K-Medoids and the numbers of clusters for K-means are a few examples of input parameters. Without background knowledge on the data, it is often **difficult to find a suitable parameterization.** Often, parameters need to be adjusted in a time consuming trial and error procedure. It cannot be guaranteed that a useful parameterization can be detected by doing so. Outliers and noise points in real-world data additionally complicate the search for a suitable parameterization.

The performance of several popular clustering algorithms such as K-means [4] also depends on the choice of several user-defined parameters such as how the initial points would be selected, or which distance measure to use to compute the similarity of the data points. It is in this context that we define a *free parameter* in clustering. Many clustering algorithms have one or more parameters that must be set according to the problem at hand. These are *free parameters*, and influence the process of clustering. As an example, the clustering algorithms like CURE [101], shifting grid, training neural networks, global FCM [104], and evidential C-Means (ECMs) [103]are sensitive to the initial parameter settings, namely, the initialization. One can say that the initial parameter choices for these algorithms are all *free parameters*. One can only guarantee that the best clustering solution for a fixed value of k would be found if all possible initial configurations of prototypes were evaluated. Of course, this approach is not computationally feasible in practice, especially for large data sets and large k. Running the algorithm only for a limited set of initial prototypes, in turn, may be either inefficient or not computationally attractive, depending on the number of prototype initializations to be performed.

In our work we propose a optimizer augmented clustering (OAC), i.e., a method

employing an optimization loop, in addition to the clustering algorithm, to achieve higher clustering performance through a better choice of these *free parameters*. As a case study, we have tried to achieve better centroids for the iterative K-means algorithm by means of a combinatorial search. The improvement achieved replacing mean computations for new centroids in successive iterations of K-means with more optimized values obtained from different optimizing techniques have been investigated. **The experimental study is presented firstly in the domain of K-means, and thereafter tested with another clustering algorithm, viz., Maximin clustering algorithm.** We explore four different optimization techniques to optimize the centroids of the iterative K-means algorithm, and that of the Maximin algorithm.

There exists literature on optimization-based clustering methods. In them, one finds the use of Genetic Algorithm (GA), Particle Swarm Optimization (PSO), and other optimization techniques in optimizing criterion like Sum Squared Error (SSE) and others in standard clustering algorithms. The well-known clustering algorithm **K-means** is one of the most popular partition clustering algorithms. There is still a considerable interest in this 50 year old algorithm and in its improvement. A couple of criteria influence the performance of this algorithm greatly. One of them includes the manner in which the initial points are selected, which is largely a random selection technique. This mostly leads to the problem of what is known as **local convergence** in the literature. To overcome this problem and to arrive at a solution that converges globally, there has been several attempts using GA or simulated annealing. The reason one often uses a conventional clustering algorithm like K-means is because they are powerful, time-tested, easy to implement and linear in complexity, and also implementations are freely available. How does one **measure the benefits obtained by use of some of the recent optimization schemes on aspects of clustering algorithms like K-Means that requires selection of parameters by a trial method, or an algorithm like Maximin that requires the use of a thresholding parameter?**

Another interesting study would be to **make a comparative performance analysis of a few well-known optimizing schemes on sparse/ high-dimensional/binary / standard datasets.** We have evaluated four optimization techniques viz., GA, Simulated Annealing (SA), PSO & Differential Evolution (DE) in an attempt to arrive at better centroids for K-means and Maximin clustering.

The performance of these methods and an independent K-means and Maximin run are observed for terminations condition/fixed number of generations.

1.2 Problem Statement

The general objective of clustering is to find a set of equivalence classes over a set of data points to optimize a suitably defined criterion. More precisely, given a set of data points $X = \{X_1, X_2, ..., X_n\}$, the objective is to find a partition $D = \{D_1, D_2, ..., D_k\}$, with $D_i \bigcap D_j = \Phi, \bigcup_i D_i = X$, so as to optimize a suitably defined criterion $f(D)$. Since the number of possible partitions (the number of potential equivalence relations) defined on the set X of n points is given by the Bell number B_n which grows fast with n, an exhaustive search for the optimal D is impractical. Numerous heuristic clustering algorithms (such as k-means and Maximin) have been proposed that find a good, but sub-optimal partition. Frequently, a clustering algorithm has several free parameters α and the resulting partition $D(\alpha)$ and the quality of clustering $f(D(\alpha))$ varies significantly with the choice of α. An example of such a free parameter is the initial choice of centroids in the k-means algorithms. The problem addressed in our investigation is to find the optimal choice of the parameters α so as to optimize $f(D(\alpha))$.

Even **the problem of finding optimal α may be a combinatorially hard one.** For example, the number of possible selections of k centroids from a set of n points is obviously given by $C(n,k)$ which can grow exponentially with n. We propose to use a novel multi-loop optimal clustering approach where the inner loop employs a combinatorial heuristic optimization algorithm to improve upon the choice of the free parameters to result in higher quality clustering. We experiment with the k-means and Maximin clustering algorithm and several different optimization algorithms such as genetic algorithms, simulated annealing, etc in the inner loop, and demonstrate that such an optimizer augmented clustering approach can improve the clustering performance (as measured by $f(D(\alpha))$) significantly over

that with regular non-optimized clustering.

1.3 Relevance of the work

Most technical fields, including all those in engineering, involve some form of optimization that is required in the process of design. Since design is an open-ended problem with many solutions, the quest is to find the best solution according to some criterion. In fact, almost any optimization process involves trade-offs between costs and benefits because finding optimal solutions is analogous to creating designs – there can be many solutions, but only a few might be optimum or useful, particularly when there is a non linear relationship between performance and cost. Optimization, in its most general form, involves finding the most optimum solution from a family of reasonable solutions according to an optimization criterion. For all but a few trivial problems, finding the global optimum (the best optimum solution) can never be guaranteed [65].

In large databases where we have hundreds of dimensions and tens of thousands to millions of records, better optimized values exhibit the greatest value. The reason is simple: a clustering session on a large database is a time-consuming affair. Hence a refined point can insure that the time investment pays off. The refinement algorithm operates over small sub-samples of the database and hence run-times needed to determine a "good" point (which speeds the convergence on the full data set) are orders of magnitude less than the total time needed for clustering in a large-scale situation. Optimizing techniques have been used for arriving at such 'good' point(s).

The optimization of iterative K-Means has been studied in the literature, but such a comprehensive study for OAC has not been reported to the best of our knowledge. Further, the use of three different approaches of Genetic Algorithm for the optimization of the objective function of the clustering process is a detailed look and investigation on the attempted problem. We further experiment with a second clustering algorithm, Maximin, to support the encouraging results of the first set of experiments with the K-Means algorithm.

The use of Grouping Genetic Algorithm (GGA) [92], an effective optimizer in many optimization problem like Bin Packing [105], have not been reported (we couldn't find any

mention) for clustering problems, and our work highlights the applicability of this method as an effective optimizer in clustering algorithms like K-Means & Maximin.

We list a good number of similar problems in clustering requiring optimization of free parameters

(i) *Seed initialization of K-means*

(ii) No of clusters k in K-means

(iii) The 'distance' measure used in the K-means algorithm

(iv) Several research work were focused on improving cluster algorithms, such as CURE [101], shifting grid, training neural networks, global FCM [104], and evidential C-Means [103] (ECMs). However, these clustering algorithms are sensitive to the initial parameter settings, namely, the initialization [32].

(v) Partitioning around Medoids (PAM) algorithm which arbitrarily chooses objects as the initial medoids or seed points.

(vi) The leader clustering algorithm chooses a leader as a representative (center) of a cluster. The algorithm depend on a threshold value to determine whether an object is similar (close) enough to the leader to in order to lie in the same partition.

(vii) The algorithm CLARA draws a sample from the data set and uses the PAM (Partitioning around Medoids) algorithm to select an optimal set of medoids from the sample. To alleviate sampling bias, CLARA repeats the sampling and clustering process several times and subsequently selects ***(optimizes)*** *the best set of medoids as the final clustering.*

(viii) The initialization of weights during the training phase of Kohonen Self Organizing Map (SOM) is done with random values. Kohenen's SOM finds applicability in document clustering.

(ix) Methods for initialization of Reference Vectors are done randomly in LVQ (Learning Vector Quantization). Vector Quantization is a standard statistical clustering technique.

(x) Random initialization of seed of the algorithm fuzzy c-means

(xi) Initial selection of medoids in K-medoids clustering algorithm

(xii) Gaussian Mixture clustering (Expectation Maximization Algorithm) - The clustering optimization problem is that of finding parameters associated with the mixture model which maximize the likelihood of the data given the model.

(xiii) ISODATA method is a popular clustering algorithm developed at the Stanford Research Institute by G.H.Ball & D.J.Hall [5]. This method requires that the number of

clusters 'm' be specified and threshold values t1, t2 & t3 be given or determined for use in splitting, merging or discarding clusters respectively. During the clustering process, the thresholds are used to determine if a cluster should be split into two clusters , merged with other clusters or discarded (when too small) [57]

(xiv) In Approximation Clustering, clustering within (1+ε) of the optimum cost, ε is user defined tolerance.

The utility of optimized schemes can be gauged from reported results on the **seed initialization problem of K-means**. [17] have run 20 procedures of *k*-means algorithm on KDD99 data set with parameter $k = 4$. The results show that 80% of random initial centers lead *k*-means algorithm to some local optimum with cost larger than 110,000, while 10% of procedures end with cost smaller than 100,000. Most of the computation time is wasted on the useless iterations if the initial centers are not well chosen. To discover such bad initial centers as early as possible, [17] focus on deriving a lower bound on the local optimums achievable in the future iterations.

1.4 Contributions of the Thesis

The first contribution of the thesis is the demonstration of a **superior clustering performance (using metrics for compactness of clusters) achieved with an optimization-based clustering approach, as compared to standard iterative K-Means on a Textual dataset**. The dataset ISA (BIGCHECK) used for experimentation contains approximately 5,000 documents culled from abstracts of ACM. We have experimented using a subset of 1060 documents which is a fairly representative subset of the dataset.

The second contribution **is a systematic, in-depth investigation using four recent optimizing techniques to improve the well-known iterative K-means algorithm and Maximin clustering algorithm** and analyze the comparative performance of the techniques. The relative comparison of these cutting-edge optimization techniques in the optimization loop is very interesting and reveals useful results.

The third contribution is **identifying Grouping Genetic Algorithm (GGA) as the most effective technique for arriving at better centroids of K-Means for high dimensional datasets. GGA (with Pearson distance) emerges as the most effective optimizer that**

can effectively be used for augmentation of iterative K-Means.

The fourth contribution is the revelation of **Maximin arriving at better centroids in comparison to the popular K-Means,** though in terms of efficiency, K-Means proved to be much better.

The fifth contribution is the discovery that the **Silhouette index gives slightly more accurate results than the Davies-Bouldin index.** However owing to computational complexity, Silhouette index may be unsuitable for real-time operation.

1.5 Outline of the Thesis

Chapter 1 provides the Introduction, and defines the problem under investigation. The thesis includes five more chapters.

In **Chapter 2,** a survey of the current state of the art in the field of clustering is presented [1] [3 8 9 30]. We investigate optimization based clustering methods [25 27 28 29 43 44], and also outline the Vector Space Model for text clustering [31]. It also provides a basic outline of the four optimizing techniques (Genetic Algorithm, Simulated Annealing, Particle Swarm Optimization, Differential Evolution) used in the experimentation, it underlines how our schemes differs from some of the schemes used by other researchers who have used these techniques (the differences of our GA with work of Laszlo [28] et al., for example). It provides the background of document clustering with example, Term frequency, Document Frequency & Inverse Document Frequency. We also look into the K-Means algorithm, its areas of improvement, and highlight studies that has been made to refine centroids of the K-Means algorithm [10 11 12 14 15 16 17 18 22 23], or related studies on the K-Means algorithm [19 20 21], or initial seed selection of clustering algorithms in general [24 26]. The chapter also provides a description of Cluster Validity with particular reference to Silhouette Coefficient (SC) & Davies Bouldin Index (DBI), and the datasets that we have worked with during the investigation.

Chapter 3 contains an introduction to our proposal. We ***propose a multi-level method employing an optimization loop, in addition to the clustering algorithm, to achieve***

higher clustering performance through an automatic, better choice of the free parameters. We provide the Architecture of Optimizer Augmented Clustering (OAC), and its potential utility. We opine on the choice of our problem selection, and also highlight the differences in our solving techniques with some standard techniques reported in the literature.

Chapter 4 provides the details of the experimental study with K-Means. Initial experiments were conducted in the domain of textual data sets. Four optimizing techniques viz., *Genetic Algorithm (GA) Steady State, Simulated Annealing, Differential Evolution (DE) & Particle Swarm Optimization (PSO)* were used in an inner loop to arrive at better centroids compared with an independent K- Means run. The performances of these methods and an independent K-means run were observed for a fixed number of generations. The results clearly demonstrate that a superior clustering performance (as measured by an appropriate, suitably defined metric) is achieved with such an optimization-based clustering approach, as compared to a normal single-level clustering using a fixed choice of the parameters.

In **Chapter 5**, in '***Results and Discussion***' we discuss the results of the experiments with K-Means. The findings clearly demonstrate that a superior clustering performance (as measured by an appropriate, suitably defined metric) is achieved with such a multi-level, optimization-based clustering approach, as compared to a normal single-level clustering using a fixed choice of the parameters.

Chapter 6 provides the details of the experimental study with Maximin algorithm. The performance of Maximin and the same set (used earlier with K-Means) of optimizers augmented Maximin were compared to validate the claims made earlier during the experiments with K-Means. In general, the results obtained using Maximin support our K-Means results discussed in the earlier chapter.

Chapter 7 discusses the experimental results with Maximin algorithm.

Chapter 8 presents the final conclusion of the dissertation.

Chapter 2

Literature Survey

In this chapter, we provide the background to our work. The chapter is organized as follows. Section 2.1 gives a comprehensive survey of clustering. In section 2.2, we discuss the four optimization based schemes used for investigation. Section 2.3 deals with Text Clustering and various related concepts. Section 2.4 outlines the K-Means algorithm, its local convergence problem and approaches used to overcome it. Section 2.5 contains the description of the datasets used in the experiments. Section 2.6 outlines two quantitative measures of cluster quality – silhouette coefficient & Davies Bouldin Index used in the experiments. Section 2.7 discusses the two similarity measures used in the experiments- Euclidean and Pearson distance.

2.1 A Survey of Clustering

The most important role of clustering is in achieving ***data abstraction***. Clustering is one of the dominant techniques of exploratory data analysis. The literature on clustering is enormously rich; one may refer to some classical references such as [2 5 6 13] as well as the ones that concentrate on knowledge based approaches [58]. Clustering is usually performed when no information is available concerning the membership of data items to pre-defined classes. Clustering discover patterns within data of large sizes. In its early days the main emphasis has been to cluster with precision. This made the I/O cost high. Therefore, **classical clustering algorithms developed in statistics were not relevant in the context of data mining** where devising efficient clustering algorithms which minimize I/O cost became essential.

Clustering techniques consider data tuples as objects. They partition the objects into groups or clusters, so that objects within a cluster are 'similar' to one another and 'dissimilar' to objects in other clusters. Similarity is defined in terms of how close the objects are in space, based on a distance function. ***The quality of a cluster may be represented by its diameter, the maximum distance between any two objects in the***

cluster. Centroid distance is an alternative measure of cluster quality and is defined as the average distance of each cluster object from the cluster centroid. Clusters can be seen either as compact sets or as dense sets separated by low-density regions. Unlike density, Compactness usually has strong implications on the shape of the clusters, so methods that focus on compactness should be distinguished from methods that focus on the density.

2.1.1 Categories of Clustering Algorithms

The various categories of clustering algorithms are shown in Table 2.1. ***Hierarchical Clustering*** builds a cluster hierarchy, a tree of clusters, also known as a *dendrogram*. Every Cluster node contains Child cluster; Sibling clusters partition the points covered by their common parents. Such an approach allows exploring data at different levels of granularity. They can be agglomerative (bottom up) or divisive (top-down) [2 7]. The sequence of partitioning operation can be done bottom-up, performing repeated amalgamation of groups of data until some pre-defined threshold is reached, or top-down where recursively the data is divided until some pre-defined threshold is reached. *Hierarchical Clustering* is frequently used in document and text analysis. Advantages of *Hierarchical clustering* include embedded flexibility regarding the level of granularity, ease of handling of any forms or distance and applicability to any attribute types. The disadvantage of *hierarchical clustering* is vagueness of termination criteria, and no re-tracing of once constructed clusters (intermediate) for improvement.

Given a set of N input patterns $\mathbf{X} = \{x_1, \ldots\ldots, x_j, \ldots\ldots, x_N\}$, where $x_j = (x_{j1}, x_{j2} \ldots\ldots\ldots\ldots\ldots, x_{jd})^T \varepsilon\ R^d$ and each x_{ji} measure is said to be a feature (attribute, dimension or variable), Hierarchical clustering [127]attempts to construct a tree-like nested structure partition of $\mathbf{X}$, $\{H = H_1, \ldots\ldots, H_Q\}$ $(Q \leq N)$, such that $C_i \in H_m$, $C_j \in H_l$, and $m > l$ imply $C_i \in C_j$ or $C_i \cap C_j = \emptyset$ for all $i, j \neq i, m, l = 1, \ldots\ldots, Q$.

Clusters are either isotropic or non-isotropic. A cluster is isotropic when it has equal tendency to growth in all directions or when the variance of samples in the cluster is nearly the same in different directions. The **hierarchical algorithms are more versatile than the partitioning algorithms**. For example the single-link clustering algorithm[128] works well on data sets containing non-isotropic clusters including well-

separated, chain-like & concentric clusters, whereas a typical partitioning algorithm like the K-Means algorithm works well only on data sets having isotropic clusters [30]. On the other hand, **the time and space complexities of the partitioning algorithms are typically lower than those of the hierarchical algorithms.** It is possible to have a hybrid algorithm that exploits the good features of both categories. One of the most remarkable developments of *hierarchical* algorithm is the algorithm *BIRCH* [33]. Scalability is the major achievement of this strategy. In BIRCH, a data structure is used to store relevant information like centroid, variance of patterns in a cluster. Hierarchical clustering of large data sets can be sub-optimal, even if data fits into memory. Compressing data may improve performance of hierarchical algorithms.

***Partitioning* algorithms** learn clusters directly. They try to discover clusters by iteratively relocating points between subsets, or try to identify clusters as areas highly populated with data. Algorithm of the first kind, Partitioning Relocation Method, is further classified into probabilistic clustering (*EM* framework, algorithm *SNOB, AUTOCLASS* [112], *MCLUST*), *K-medoids* methods (algorithms *PAM, CLARA, CLARANS* [119] and its extension) and *K-Means* methods. In *K-Means*, given a pre-specified k, the algorithm partitions the data set into k clusters, which optimizes an objective function. The objective function tries to minimize the sum of the squared distances of objects from their cluster centers. Given a set of N input patterns $\mathbf{X} = \{x_1, \ldots\ldots, x_j, \ldots\ldots, x_N\}$, where $x_j = (x_{j1}, x_{j2} \ldots\ldots\ldots\ldots\ldots, x_{jd})^T \varepsilon\ R^d$ and each x_{ji} measure is said to be a feature (attribute, dimension or variable), Hard partitional clustering [127]attempts to seek a K-partition of **X**, $C = \{C_1, \ldots\ldots, C_K\}$ $(K \leq N)$, such that $C_i \neq \emptyset, i = 1, \ldots\ldots\ldots, K$; $\cup_{i=1}^{K} C_i = \mathbf{X}$; $C_i \cap C_j = \emptyset$, $i,j = 1, \ldots\ldots, K$ and $i \neq j$. The various deterministic and stochastic search approaches to solve partitioning clustering mostly use the **squared-error criterion function**. The clusters generated by these approaches are not as versatile as those generated by *hierarchical* algorithms. *Hierarchical* algorithms are computationally expensive [1]. Another approach for partitional clustering is to allow splitting and merging of clusters. Here merging is performed based on the distance between the centroids of two clusters. A cluster is split if its variance is above a certain threshold [64]. One proposed algorithm performing this is called ISODATA.

Hierarchical method • Agglomerative algorithm • Divisive algorithm Optimized for large (BIRCH [111]) & small data sets (ROCK [109]), able to handle Outliers, arbitrary shaped clusters	***Partitioning method*** Relocation algorithms • Probabilistic clustering • K-medoids method • K-Means method Optimized for small(PAM) & large data sets(CLARA), spherical clusters	***Density based method*** • Density based connectivity clustering • Density functions clustering Optimized for large data sets, able to Complexity is O (n log n)
Grid-based method Optimized for high dimensional large data sets, Complexity O (n)	***Method based on Co-occurrence of Categorical data*** CACTUS (Clustering Categorical Data Using Summaries) [108] Algorithm STIRR (Sieving Through Iterated Reinforcement) [107]	***Constraint based clustering*** The COD (Clustering with Obstructed Distance) algorithm [110].
Clustering Algorithms used in machine learning • Gradient Descent & Artificial Neural Networks •Evolutionary methods (GA & others) Globalised sol" convergence	***Scalable clustering algorithms*** BIRCH	***Algorithms for high dimensional data*** • Subspace Clustering [113] • Projection techniques • Co-clustering techniques

Table 2.1: Types of clustering algorithms

Density Based partitioning (Fig. 2.1) try to discover dense connected components of data, which are flexible in terms of their shape. Density based connectivity is used in the algorithms *DBSCAN* [90],*OPTICS* [114], *DBCLASD* [115], whereas the algorithm *DENCLUE* [117] exploits space density functions. These algorithms are less sensitive to outliers and can discover clusters of irregular shapes. They usually work with low-dimensional data of numerical attributes, known as spatial data. They quantize the space of the data items into a finite number of cells and only retain for further processing the cells having a high density of items; isolated data items are not considered. Quantization steps and density

thresholds are common parameter for these methods, and these are also free parameters for these clustering algorithms. Some interesting recent work on density based clustering is using 1-class support vector machines [59]. Many of the graph clustering methods is related to density based clustering. The data items are represented as nodes in a graph and the dissimilarity between two items is the 'length' of the edge between the corresponding nodes. However, some other graph theoretic methods are more related to squared error methods. Graph based clustering has paved the way for significant interest recently in Spectral Clustering [59]. ***Grid based methods*** frequently use *hierarchical* agglomeration as one phase of processing. Algorithms *BANG, STING*[118] *and WAVECLUSTER* [116]are of this type. Grid based methods are fast and handle outliers well. Grid based methodology is also used as an intermediate step in many other algorithms (for example like *CLIQUE* [113], *MAFIA*).

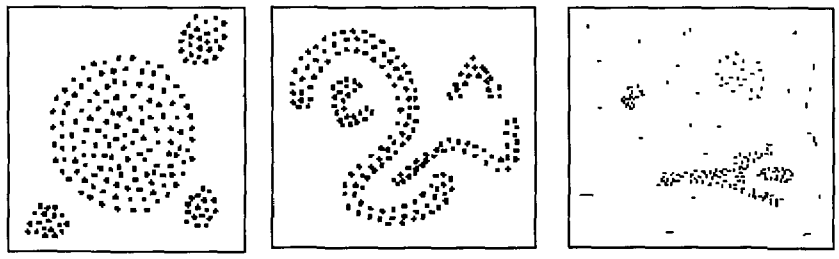


Fig 2.1 Density based clustering depicting clusters of arbitrary shapes

The methodology of clustering and subsequent techniques calls for a number of enhancements to become fully operational in the setting of data mining. The first enhancement comes in the form of *context-oriented clustering* in which contexts are regarded as useful hints delivered by the user and helping to develop a certain point of view of the data; in this sense we make the notion of interestingness more operational. The second modification comes in the form of partial supervision in which some hints needs to be incorporated as an important part of the clustering algorithm.

The most general approach to clustering is to view it as a density estimation problem. We assume that in addition to the observed variables for each data item, there is a *hidden*, unobserved variable indicating the "cluster membership" of the given data item. Hence the data is assumed to arrive from a *mixture model* and the mixing labels (cluster identifiers)

are hidden. Many methods assume that the number of clusters K is known or given as input. **The clustering optimization problem is that of finding parameters associated with the mixture model M (W_i and parameters of components C_i) which maximize the likelihood of the data given the model.** The probability distribution specified by each cluster can take any form. The *EM* algorithm [142] is a well-known technique for estimating the parameters in the general case. *K-Means* clustering is a popular method (historically also known as Forgy's method [85], or MacQueen's [4] algorithm. It is really a special case of *EM* that assumes:

1. Each cluster is modeled by a spherical Gaussian distribution;
2. Each data item is assigned to a single cluster;
3. The mixture weights (W_i) are equal.

2.1.2 Clustering Using Soft Computing Approaches:

2.1.2.1 Neural Networks for Clustering

Neural networks have been successfully applied in a wide range of supervised and unsupervised learning applications. Neural network models are not commonly used for data mining tasks because they produce incomprehensible models and require long training times. However, of late neural-network learning algorithms are able to produce comprehensible models, which do not require excessive training times. They're inherently parallel and distributed and learn the weights adaptively. ANN processes numerical vectors. This restricts the representing of patterns by quantitative features only [1]. The well-known examples of ANN used for clustering are Kohonen's learning vector quantization (LVQ) and self-organizing map [60]. These are single layered nets. Patterns are presented at the input and are associated with the output nodes. The weights between the input nodes and the output nodes are iteratively changed until a termination criterion is satisfied.

In competitive neural networks, active neurons reinforce their neighborhood within certain regions, while suppressing the activities of other neurons. These includes Learning Vector Quantization (LVQ) and Self-Organizing Feature Maps (SOFM) [131] [132]. Intrinsically, LVQ performs supervised learning, and is not categorized as a clustering algorithm [132] [133]. The learning properties of LVQ provide an insight to describe the potential data structure using the prototype vectors in the competitive layer. By pointing out the limitations of LVQ, including

sensitivity to initiation and lack of a definite clustering object, Pal, Bezdek and Tsao proposed a general LVQ algorithm for clustering, known as GLVQ [133]. They constructed the clustering problem as an optimization process based on minimizing a loss function, which is dependent on the locally weighted error between the input pattern and the winning prototype.

Adaptive resonance theory (ART) was developed by Carpenter and Grossberg [134 135] ART can learn arbitrary input patterns in a stable, fast and self-organizing way, thus overcoming the effect of learning instability that plagues many other competitive networks. ART is not a neural network architecture. It is a learning theory, that resonance in neural circuits can trigger fast learning. Extension includes ART1 (binary input patterns) [134], although it can be extended to arbitrary input patterns by a variety of coding mechanisms. ART2 extends the applications to analog input patterns [136] and ART3 introduces a new mechanism originating from elaborate biological processes to achieve more efficient parallel search in hierarchical structures [137]. By incorporating two ART modules, which receive input patterns (ART *a*) and corresponding labels (ART *b*) respectively, with an inter-ART module, the resulting ARTMAP system can be used for supervised classifications [138]. The match tracking strategy ensures the consistency of category prediction between two ART modules by dynamically adjusting the vigilance parameter of ART *a*. One of the big challenges of clustering is organization and retrieval of documents from archives. [60] have demonstrated the utility of a self-organizing map (SOM) with more than one million nodes to partition a little less than seven million patent abstracts where the documents are represented by 500-dimensional feature vectors.

2.1.2.2 Evolutionary Approaches for Clustering

From an optimization perspective, clustering can be formally considered as a particular kind of NP-hard grouping problem [92]. This has stimulated the search for efficient approximation algorithms, including not only the use of *ad hoc* heuristics for particular classes or instances of problems, but also the use of general-purpose metaheuristics [139]. Particularly, evolutionary algorithms are metaheuristics widely believed to be effective on NP-hard problems, being able to provide near-optimal solutions to such problems in reasonable time. Under this assumption, a large number of evolutionary algorithms for solving clustering problems have been proposed in the literature. These algorithms are based on the optimization of some objective function (i.e., the so-called fitness function) that guides the evolutionary search.

Evolutionary approaches make use of evolutionary operators and a population of solutions to

obtain the globally optimal partition of the data. The most commonly used evolutionary operators are selection, recombination and mutation [34]. The most well known evolutionary algorithms are genetic algorithms (GA), evolution strategies (ES) and evolutionary programming (EP). Of the three approaches, GA has been most frequently used for clustering [1]. In GA, the solutions are as binary strings. Selection operator reproduces solutions (chromosomes) to the next population depending on their fitness value. Crossover takes as input a pair of chromosomes (called parents) and outputs a new pair of chromosomes (called children or offspring). After a certain crossover point, it exchanges the parts of the parents. Mutation complements the bit value at an arbitrary selected location of the input chromosome resulting in a new chromosome. GA depends on the crossover operator to explore the search space and mutation is used in GA to ensure no part of the search space is left unexplored.

Evolutionary approaches are globalized search techniques, whereas the other approaches like the statistical algorithms like the K-Means algorithm, fuzzy clustering algorithms and tabulate search are localized search techniques. ANN and GA are inherently parallel and so they can be implemented using parallel hardware to improve the speed of processing. Evolutionary approaches are population based and they search using multiple solutions at a time and the others are based on using a single solution at a time. ANN and GA are sensitive to the selection of various learning/control parameters. [130] [141]presents a survey of evolutionary algorithms designed for clustering tasks. [140] provides an extensive review of evolutionary algorithms for data mining applications, but the work focuses on specific evolutionary approaches (GAs and Genetic Programming) and is mainly intended for classification tasks, clustering being just slightly touched in a peripheral section.

2.1.2.3 Fuzzy Clustering

The degree of membership of a data item to a cluster is either in [0, 1] if the clusters are fuzzy or in {0, 1} if the clusters are crisp. For fuzzy clusters, data items can belong to some degree to several clusters that don't have hierarchical relations with each other. The clustering result changes depending on its fuzziness/ crispness. The output of such algorithm is a clustering, but not a partition. Crisp Clusters can always be obtained from fuzzy clusters.

When a fuzzy clustering algorithm is applied to a data set with N objects, the final result is a

partition of the data into a certain number *k* of *fuzzy clusters*, such that

$$\begin{cases} P = [\mu_{ij}]_{k \times N} \\ \mu_{ij} \in [0,1] \end{cases} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (2.1)$$

where P is a $k \times N$ *fuzzy partition matrix* whose element μ_{ij} represents the fuzzy membership of the jth object to the ith fuzzy cluster. When μ_{ij} is limited to the extreme values of its feasibility interval, i.e., $\mu_{ij} \in \{0,1\}$, then P degenerates to a soft partition. Besides, if the additional constraint $\sum_i \mu_{ij} = 1$ is imposed to every column j of the matrix, then P degenerates to a standard hard partition. A fuzzy partition matrix provides additional information about the data that is not available in its soft or hard counterparts. In fact, the fuzzy membership values μ_{ij} can help discover more sophisticated relations between the corresponding data objects and disclosed clusters [127].

Fuzzy version of methods based on the squared error was defined, beginning with the Fuzzy C-Means. When compared to their crisp counterparts, fuzzy methods are more successful in avoiding local minima of the cost function and can model situations where clusters actually overlap. To make the results of clustering less sensitive to outliers several fuzzy solutions were put forward, based on robust statistics or the use of a 'noise cluster'.

In fuzzy clustering, each cluster is a fuzzy set of all the patterns. Fuzzy K-Means algorithm though betters the K-Means algorithm in avoiding local minima; it still converges to local minima of the squared-error criterion [61]. The design of membership function is the most important problem in fuzzy clustering.

Regardless of the fixed or variable nature of the number of clusters, the evolutionary algorithms for fuzzy clustering are mostly based on extensions – to the fuzzy domain – of the fundamental ideas for hard partitional clustering. This is in conformity with the fact that most fuzzy clustering algorithms are based on generalizations of traditional algorithms for hard clustering, as it is the case of the well-known Fuzzy C-Means (FCM) algorithm and its variants [145 146 147]. These are essentially generalizations of the classic *k*-means algorithm.

Recent research into subspace [84] and online clustering algorithms has been performed.

'Online' implies that the algorithm can be performed dynamically as the data are generated, and thus it works well for dynamic databases. Additionally, some adaptive algorithm allows the user to change the number of clusters dynamically. These adaptive algorithms avoid having to completely re-cluster the database if the users' needs change. One recent online approach represents the cluster by profiles such as cluster mean and size. These profiles are shown to the user, and the user has the ability to change the parameters (number of clusters) at any time during processing. One recent clustering approach is both online and adaptive, (OAK Online Adaptive Clustering). OAK can handle outliers effectively by adjusting a viewing parameter, which gives the user a broader view of clustering and help in obtaining the desired clusters. For clustering algorithms embedded within World Wide Web search engines, time efficiency is imperative because search engine users want results very fast. There have been others efforts of clustering using competitive agglomeration [83] and at parallelization and distributed clustering [62] [63] with promising results. This comprises a brief survey of the various clustering approaches [81] (see Table 2.2) and the trends within the clustering community. Choosing a clustering algorithm for a particular problem can be a daunting task. **There are no single algorithms that can effectively perform tackling all the major challenges of clustering like high dimensionality, scalability & outlier detection that is so crucial while choosing a clustering algorithm.** Some scores over the others in some respect while lacking in other aspects of clustering.

Perhaps the most important criterion for matching an instance of a clustering problem to a clustering algorithm is the nature of the data and the anticipated clusters. Analysis and results have shown that if the data is numerical and the clusters are believed to be spherical or ellipsoidal, then approaches based on mixture models may be appropriate. On the other hand, if data is non-numerical, and little is known a priori about the geometry of the clusters, one of the graph-based approaches may be most appropriate.

The context of clustering plays an important role in discovering knowledge nuggets – rare yet essential pieces of information. Without any direction imposed by the user, one could be easily washed away in a mass of useless but frequent data (which is statistically meaningful). The filtering of data accomplished by the context prevents this from happening.

Algorithm	Type	Space	Time	Remarks
Single link	Hierarchical	$O(n^2)$	$O(k n^2)$	Not incremental
Average link	Hierarchical	$O(n^2)$	$O(k n^2)$	Not incremental
Complete link	Hierarchical	$O(n^2)$	$O(k n^2)$	Not incremental
MST	Hierarchical/ Partitional	$O(n^2)$	$O(n^2)$	Not incremental
Squared error	Partitional	O(n)	O(tk n)	Iterative
K-Means	Partitional	O(n)	O(tk n)	Iterative, no categorical
Nearest Neighbour	Partitional	$O(n^2)$	$O(n^2)$	Iterative
PAM	Partitional	$O(n^2)$	$O(tk (n-k)^2)$	Iterative, adaptive agglomerative, Outliers
BIRCH	Partitional	O(n)	O(n) No rebuild	CF- tree, Incremental, Outliers
CURE	Mixed	O(n)	O(n)	Heap, k-D tree, Incremental, Outliers, Sampling
ROCK	Agglomerative	$O(n^2)$	$O(n^2 \lg n)$	Sampling, Categorical, Links
DBSCAN	Mixed	$O(n^2)$	$O(n^2)$	Sampling, Outliers

Table 2.2 Comparison of various clustering algorithms

2.2 Optimization based clustering methods

2.2.1 Genetic Algorithm

2.2.1.1 Genetic Algorithms

Genetic algorithms are general purpose search algorithms inspired by Darwin's principle of the 'survival of the fittest' to solve complex optimisation problems [34 76]. An initial population of possible solutions evolves over time to converge to an optimal solution. **Although it is not guaranteed to find the optimum, the use of a population helps to avoid local maxima.**

A solution is represented by a chromosome, consisting of several genes. A genetic algorithm begins with an initial population of randomly generated chromosomes. **During successive iterations, called generations, the initial chromosomes advance towards stronger chromosomes by reproduction among members of the previous generation.** New generations are created by three genetic operators: **selection, crossover** and **mutation.** Selection of the best chromosomes makes sure that only the best chromosomes can crossover or mutates by rating the individual chromosomes by their adaptation or associated fitness.

There have been clustering techniques based on the use of genetic algorithms [97].**A genetic algorithm performs a parallel, non comprehensive search for the global optimum values.** To determine how to perform clustering with genetic algorithms, we must first determine how to represent each cluster. One simple approach would be to use a bit map representation for each possible cluster. Given a data base with four items {W, X, Y, Z} we could represent one solution to creating two clusters as 1001 and 0110. This represents the two clusters {W, Z} and {X, Y}.

Algorithm 2.2 shows an iterative refinement technique for clustering that uses a genetic Algorithm [27 93]. An initial random solution is given and successive changes to this converge on a local optimum. A new solution is generated from the previous solution using crossover and mutation operations. The new solution must be created in such a way that it represents a valid clustering. Crossover is the most popular recombination operator.

2.2.1.2 Basic Steps in GA The basic steps are outlined in Algorithm 2.1

In GA, Mutation is used to make sure that no part of the search space is left unexplored. Crossover and Mutation are applied with some pre-specified probabilities, which depend on the fitness values. A fitness solution must be used and may be defined based

Algorithm 2.1 Genetic Algorithm

Input: a random initial population of possible solution

Output: optimized solution for the problem

Begin

1. *t= 0*
2. *initialize population P(t)*
3. *compute fitness P(t)*
4. *t= t+1*
5. *if termination criterion achieved go to step 10*
6. *select P(t) from P(t-1)*
7. *crossover P(t)*
8. *mutate P(t)*
9. *go to step 3*
10. *Output best and stop*

End

on an inverse of the squared error. Because of the manner in which crossover works, genetic clustering algorithms perform a global search rather than a local search of potential solutions. The basic steps of GAs, which are also followed in the GA-clustering algorithm [102], are shown in Algorithm. 2.2.

Algorithm 2.2 **GA Clustering Algorithm**

Input: $D = \{n_1, n_2, \ldots..n_n\}$ *// set of elements*

k *number of desired clusters*

Output: *K // set of clusters*

Randomly create an initial solution;

Repeat

Use crossover to create a new solution;

Until termination criteria is met

2.2.1.3 Population initialization

Each individual (chromosome) represent a possible solution to a given problem. Creating the initial population provides the starting point for the algorithm. Typically, this process is done by creating chromosomes randomly, but it can also be done by seeding the population with known fit chromosomes. One important constraint to this algorithm is that the initial population must be diverse. Diversity can be checked by running an additional step to confirm that each chromosome is different to some degree. Without reasonable diversity, the algorithm may not produce good solutions.

2.2.1.4 Fitness computation

The evaluation step provides a way to rate how each chromosome (candidate solution) solves the problem at hand. The step involves decoding the chromosome into the various space of the problem and then checking the result of the problem using these parameters. The fitness is computed from this result.

2.2.1.5 Selection

The selection process selects chromosomes from the mating pool directed by the survival of the fittest concept of natural genetic systems. In the proportional selection strategy, a chromosome is assigned a number of copies, which is proportional to its fitness in the population, and which goes into the mating pool for further genetic operations. Roulette wheel selection is one common technique that implements the proportional selection strategy.

2.2.1.6 Crossover

Crossover is a probabilistic process that exchanges information between two parent chromosomes for generating two child chromosomes. The crossover operator takes two chromosomes, separates them at a random site in both chromosomes, and then swaps the tails of the two, resulting in two new chromosomes. Cutting the chromosome at one location, called single-point crossover, is often used. Multipoint crossover can also be used. The crossover operator does not create new material within the population but inter-mixes the existing population to create new chromosomes. This allows the genetic algorithm to search the solution space for new candidate solutions to solve the problem at hand. **The crossover operator is generally accepted as the most important operator.**

2.2.1.7 Mutation

The mutation operator introduces a random change into a gene in the chromosome (sometimes more than once). The mutation operator provides the ability to introduce new material into the population. Because chromosomes intermix with existing chromosomes, mutation provides the opportunity to shake up the population to expand the solution space.

2.2.1.8 Termination criterion

The termination criterion could be a fixed number of iterations, or a condition where the fitness value remains the same over a few successive iterations.

2.2.2 Simulated Annealing

Simulated annealing (SA) is a heuristic based probabilistic method for the global optimization problem. It provides a good approximation to the global minimum of a given function in a large search space [98] [99]. It is often used when the search space is discrete (e.g., centroid selection from a set of points of the dataset). **For certain problems, simulated annealing**

may be more effective than exhaustive enumeration — provided that the goal is merely to find an acceptably good solution in a fixed amount of time, rather than the best possible solution.

Annealing, a technique in metallurgy, involves heating and controlled cooling of a material to increase the size of its crystals and reduce their defects. The heat causes the atoms to wander randomly from their initial positions (a local minimum of the internal energy) through states of higher energy; the slow cooling gives those more chances of finding configurations with lower internal energy than the initial one. Each step of the SA algorithm replaces the current solution by a random 'nearby' solution, chosen with a probability that depends on the difference between the corresponding function values and on a global parameter T (called the temperature), that is gradually decreased during the process. The dependency is such that the current solution changes almost randomly when T is large, but increasingly 'downhill' as T goes to zero. The allowance for 'uphill' moves saves the method from converging at local minima.

The method has been independently described by [70] in 1983, and by V. Černý [71] in 1985 . The method is an adaptation of the Metropolis-Hastings algorithm, a Monte Carlo method to generate sample states of a thermodynamic system, invented by N. Metropolis et al. [72] in 1953.

2.2.2.1 Overview

In the simulated annealing (SA) method, each point s of the search space is analogous to a state of some physical system, and the function E(s) to be minimized is analogous to the internal energy of the system in that state. The goal is to bring the system, from an arbitrary initial state, to a state with the minimum possible energy.

2.2.2.2 The basic iteration

At each step, the SA heuristic considers some neighbour s' of the current state s, and probabilistically decides between moving the system to state s' or staying in state s. The probabilities are chosen so that the system ultimately tends to move to states of lower energy. Typically this step is repeated until the system reaches a state that is good enough for the application.

2.2.2.3 The neighbors of a state

The neighbors of each state (the candidate moves) are specified by the user, usually in an application-specific way. For example, in clustering problem each state may be represented by a fitness value and neighbors represent potential solution state in the neighborhood with its associated fitness value.

2.2.2.4 Acceptance probabilities

The probability of making the transition from the current state s to a candidate new state s' is specified by an acceptance probability function P (e, e', T), that depends on the energies e = E(s) and e' = E(s') of the two states, and on a global time-varying parameter T called the temperature.

One essential requirement for the probability function P is that it must be nonzero when e' > e, meaning that the system may move to the new state even when it is worse (has a higher energy) than the current one. It is this feature that helps the method from converging to a local minimum—a state that is worse than the global minimum, yet better than any of its neighbors.

On the other hand, when T goes to zero, the probability P (e, e', T) must tend to zero if e' > e, and to a positive value if e' < e. For sufficiently small values of T, the system will tend to move 'downhill' (to lower energy values), and bypass moves that go 'uphill'. In particular, when T becomes 0, the procedure will reduce to the greedy algorithm—which makes the move only if it goes downhill. The P function is usually chosen so that the probability of accepting a move decreases when the difference e' − e increases (small uphill moves are more likely than large ones).

The evolution of the state s depends on the temperature T, and is sensitive to coarser energy variations when T is large, and to finer variations when T is small.

2.2.2.5 The annealing schedule

The SA method reduces the temperature gradually. Initially, T is set to a high value (or infinity), and it is decreased at each step according to some annealing schedule—which may be specified by the user, but must end with T = 0 towards the end of the allotted time. The system is expected to wander initially towards a broad region of the search space containing good solutions, ignoring small features of the energy function; then drift towards low-energy regions

Algorithm 2.3 Simulated Annealing

Input: a random initial population of possible solution
Output: optimized solution for the problem

```
s ← s0; e ← E(s)                    // Initial state, energy.
sb ← s; eb ← e                      // Initial "best" solution
k ← 0                               // Energy evaluation count.
while k < kmax and e > emax         // While time remains & not good enough:
    sn ← neighbor(s)                //  Pick some neighbor.
    en ← E(sn)                      //  Compute its energy.
    if en < eb then                 //  Is this a new best?
        sb ← sn; eb ← en            //  Yes, save it.
    if P(e, en, temp(k/kmax)) > random() then  //  Should we move to it?
        s ← sn; e ← en              //  Yes, change state.
    k ← k + 1                       //  One more evaluation done
return sb                           // Return the best solution found.
```

that become narrower and narrower; and finally move downhill according to the steepest descent heuristic. For any given finite problem, the probability that the simulated annealing algorithm terminates with the global optimal solution approaches 1 as the annealing schedule is prolonged.

2.2.2.6 Pseudocode

The pseudo code (see Algorithm[75] 2.3) implements simulated annealing, starting from state s0 and continuing to a maximum of kmax steps or until a state with energy emax or less is found. The call neighbor(s) should generate a randomly chosen neighbor of a given state s; the call random () should return a random value in the range [0,1). The annealing schedule is defined by the call temp(r), which should yield the temperature to use, given the fraction r of the time spent thus far. Saving the best state is not necessarily an improvement, since one may have to specify a smaller kmax in order to compensate for the higher cost per iteration. However, the step sb← sn happens only on a small fraction of the moves. Therefore, the optimization is usually worthwhile, even when state-copying is an expensive operation.

2.2.2.7 Selecting the parameters

In order to apply the SA method to a specific problem, one must specify the following parameters: *the state space, the energy (goal) function E (), the candidate generator procedure neighbor (), the acceptance probability function P (), and the annealing schedule temp ()*. These choices can have a significant impact on the method's effectiveness. Unfortunately, there are no choices of these parameters that will be good for all problems, and there is no general way to find the best choices for a given problem.

2.2.2.8 Diameter of the search graph

Simulated annealing may be modeled as a random walk on a search graph, whose vertices are all possible states, and whose edges are the candidate moves. An essential requirement for the neighbor () function is that it must provide a sufficiently short path on this graph from the initial state to any state which may be the global optimum. (In other words, the diameter of the search graph must be small).

2.2.2.9 Transition probabilities

For each edge (s, s') of the search graph, one defines a transition probability, which is the probability that the SA algorithm will move to state s' when its current state is s. This probability depends on the current temperature as specified by temp (), by the order in which the candidate moves are generated by the neighbor () function, and by the acceptance probability function P().

2.2.2.10 Acceptance probabilities

The specification of neighbor (), P (), and temp () is partially redundant. In practice, it's common to use the same acceptance function P () for many problems, and adjust the other two functions according to the specific problem. In the formulation of the method by Kirkpatrick et al., the acceptance probability function P (e, e', T) was defined as 1 if e' < e, and exp ((e − e') / T) otherwise. This formula corresponds to the Metropolis-Hastings algorithm, in the case where the proposal distribution of Metropolis-Hastings is symmetric. However, this acceptance probability is often used for simulated annealing even when the neighbor () function, which is analogous to the proposal distribution in Metropolis-Hastings, is not symmetric, or not probabilistic at all. As a result, the transition probabilities of the simulated annealing algorithm and the long-term distribution of states at a constant temperature T need not bear any resemblance to the thermodynamic equilibrium distribution over states of that physical system, at any temperature. Nevertheless, most descriptions of SA assume the original acceptance function, which is probably hard-coded in many implementations of SA.

2.2.2.11 Efficient candidate generation

When choosing the candidate generator neighbor (), one must consider that after a few iterations of the SA algorithm, the current state is expected to have much lower energy than a random state. Therefore, as a general rule, **one should skew the generator towards candidate moves where the energy of the destination state s' is likely to be similar to that of the current state. This heuristic (which is the main principle of the Metropolis-Hastings algorithm) tends to exclude "very good" candidate moves as well as "very bad" ones; however, the latter are usually much more common than the former, so the heuristic is generally quite effective.**

In the traveling salesman problem, for example, swapping two consecutive cities in a low-energy tour is expected to have a modest effect on its energy (length); whereas swapping two arbitrary cities is far more likely to increase its length than to decrease it. Thus, the consecutive-swap neighbor generator is expected to perform better than the arbitrary-swap one, even though the latter could provide a somewhat shorter path to the optimum (with n − 1 swaps, instead of n (n −1) / 2). A more precise statement of the heuristic is that one should try first candidate states s' for which P (E(s), E(s'), T) is large. For the standard acceptance function P, it means that E(s') − E(s) is on the order of T or less. Thus, in the traveling salesman example, one could use a neighbor () function that swaps two random cities, where the probability of choosing a city pair vanishes as their distance increases beyond T.

2.2.2.12 Barrier avoidance

When choosing the candidate generator neighbor () one must also try to reduce the number of "deep" local minima — states (or sets of connected states) that have much lower energy than all its neighboring states. Such **closed catchment basins of the energy function** may trap the SA algorithm with high probability (roughly proportional to the number of states in the basin) and for a very long time (roughly exponential on the energy difference between the surrounding states and the bottom of the basin). It is generally impossible to design a candidate generator that will satisfy this goal and also prioritize candidates with similar energy. On the other hand, one can often vastly improve the efficiency of SA by relatively simple changes to the generator.

2.2.2.13 Cooling schedule

The physical analogy that is used to justify SA assumes that the cooling rate is low enough for the probability distribution of the current state to be near thermodynamic equilibrium at all times. Unfortunately, the relaxation time—the time one must wait for the equilibrium to be restored after a change in temperature—strongly depends on the "topography" of the energy function and on the current temperature. In the SA algorithm, the relaxation time also depends on the candidate generator, in a very complicated way. All these parameters are usually provided as black box functions to the SA algorithm. Therefore, in practice the ideal cooling rate cannot be determined beforehand, and should be empirically adjusted for each problem. The variant of SA known as thermodynamic simulated annealing tries to avoid this problem by dispensing with the cooling schedule, and instead automatically adjusting the temperature at each step based on the energy difference between the two states, according to the laws of thermodynamics.

2.2.2.14 Restarts

Sometimes it is better to move back to a solution that was significantly better rather than always moving from the current state. This is called restarting. To do this we set s and e to sb and eb, and perhaps restart the annealing schedule. The decision to restart could be based on a fixed number of steps, or based on the current energy being too high from the best energy so far.

2.2.3 Particle Swarm Optimization

The particle swarm optimization (PSO) is a type of derivative free, direct search method used to find an optimal solution to an objective function (fitness function) in a search space. They depend only on the evaluation of the objective function. PSO is a stochastic, population-based computer algorithm inspired by swarm intelligence, which is based on social-psychological principles providing insights into social behavior proving useful for engineering problems[95]. The method was first proposed in 1995 [73].

Social influence and social learning enable a person to maintain cognitive consistency. People solve problems by interacting with other people and consequently their beliefs, attitudes, and behaviors change. The individuals move toward one another. **PSO simulates** this kind of **social optimization**. Given a problem, some mechanism to evaluate a proposed solution to it exists in terms of a fitness function. A communication or social network is established by assigning neighbors for interaction of each individual. Initialization of candidate solutions is done as a population of individuals through random guesses of the solution of the problem. A particle is a candidate solution. An iterative process to improve these candidate solutions is set in motion. The particles iteratively evaluate the fitness of the candidate solutions. They memorize the location where they had their best success. **The individual's best solution is called the particle best or the local best.** Each particle conveys this information to their neighbors. They are also able to see where their neighbors have had success. The solution is guided by these successes, with the population usually converging at a better solution in comparison to a non-swarm approach.

The swarm is modelled by particles in multidimensional space. The particles have a position and a velocity. These particles fly through hyperspace (i.e., R^n) and have two essential reasoning capabilities: their **memory of their own best position** and **knowledge of the global or their neighborhood's best**. They communicate good positions to each other, and adjust their own position and velocity based on these good positions. A particle has the following information to adjust its position and velocity:

*A **global best** immediately updated when a new best position is found by any particle in the swarm.*

*A **Neighborhood best** the particle obtains by communicating with a subset of the swarm.*

*The **local best**, the best solution witnessed by the particle.*

The particle position and velocity update equations in the simplest form are given by

$$v_{ij} \leftarrow c_0 v_i + c_1 r_1 (globalbest_j - x_{ij}) + c_2 r_2 (localbest_{ij} - x_{ij}) + c_3 r_3 (neighborhoodbest_j - x_{ij}) \quad \ldots\ldots \quad (2.2)$$

$$x_{ij} \leftarrow x_{ij} + v_{ij} \quad \ldots\ldots \quad (2.3)$$

The iteration of the swarm improves the fitness of the global best solution. The fitness may hit a plateau despite repeated runs if all particles being influenced by the global best eventually approach the global best. The particles may also exhibit 'convergence' by moving about in the search space in close proximity to the global best leaving the rest of search space unexplored. If the inertial coefficient of the velocity is small, all particles could slow down until they approach zero velocity at the global best. **The selection of coefficients in the velocity update equations affects the convergence and the ability of the swarm to find the optimum.** Reinitializing the particles positions at intervals or when convergence is detected can overcome this situation.

Some research approaches investigated the application of constriction coefficients and inertia weights. There are numerous techniques for preventing premature convergence. Many variations on the social network topology, parameter-free, fully adaptive swarms, and some highly simplified models have been created. PSO in its basic form is best suited for continuous variable, i.e., the objective function can be evaluated for even the tiniest increment. The method has been adapted as a binary PSO to also optimize binary variables which take only one of two values. Several methods exist to handle discrete variables which may be in one of multiple states.

The algorithm below uses the global best and local bests but ignores neighborhood bests. Neighborhood bests allow parallel exploration of the search space and help to avoid local convergence, but it slows down the speed of arriving at good solutions. A new best position discovered by a particle's neighborhood would be communicated to another particle's neighborhood at the next iteration of the PSO algorithm. Smaller neighborhoods lead to slower convergence, while larger neighborhoods to faster convergence, with a global best representing a neighborhood consisting of the entire swarm.

A single particle by itself is unable to accomplish anything. The power is in interactive collaboration.

Let $\boldsymbol{f}: \rightarrow \boldsymbol{R^m}$ be the fitness function

Let there be n particles, each with associated positions $x_i \varepsilon R^m$ and velocities $v_i \varepsilon R^m$, i= 1,.....,n,

Let $\hat{y}$ be the current best position of each particle and let $\hat{g}$ be the global best.

Initialize x_i and v_i for all i. One common choice is to take $x_{i,j} \varepsilon U[a_j,b_j]$ and $v_i = 0$ for all i and j=1,....,m, where a_j, b_j are the limits of the search domain in each dimension, and U represents the Uniform distribution (continuous).

$$\hat{y} \leftarrow x_i and\ \hat{g} \leftarrow arg^{min}_{x_i} f(x_i), i = 1, \ldots., n \qquad \ldots\ .. \quad (2.4)$$

While not converged:

For each particle $1 \leq i \leq n$

Create random vectors r_1, r_2 : r_{1j} and r_{2j} and for all j, by taking r_{1j}, $r_{2j} \in U[0,1]$ for j= 1,...,m

r_{2j}

Update the particle velocities:

$$v_i \leftarrow \omega v_i + c_1 r_1(\hat{y} - x_i) + c_2 r_2(\hat{g} - x_i) \qquad \ldots\ .. \quad (2.5)$$

Update the particle positions:

$$x_i \leftarrow x_i + v_i \qquad \ldots\ .. \quad (2.6)$$

Update the local bests If $f(x_i) < f(\hat{y})$, $\hat{y} \leftarrow x_i$ ... (2.7)

Update the global best If $f(x_i) < f(\hat{g})$, $\hat{g} \leftarrow x_i$ ... (2.8)

$\hat{g}$ is the optimal solution with fitness $f(\hat{g})$

ω is an inertial constant. Good values are usually slightly less than 1.

c_1 and c_2 are constants indicating how much the particle is directed towards good positions. They represent a "cognitive" and a "social" component respectively. They affect how much the particle's personal best and the global best respectively influence its movement. Usually we take

$c_1, c_2 \approx 2$

r_1, r_2 are two random vectors with each component generally a uniform random number between 0 and 1.

The pseudo code of the basic PSO (see Algorithm 2.4) with random vectors r_1, r_2 implemented as scalars inside the dimension loop is given below.

By studying this algorithm, we see that we are essentially carrying out something like a discrete time simulation where an iteration of it represents a tick of time. **The particles communicate information they find about each other by updating their velocities in terms of local and global bests. When a new best is found, the particles will change their positions accordingly so that the new information is broadcast to the swarm.** The particles are always drawn back both to their own personal best positions and also to the best position of the entire swarm. They also have stochastic exploration capability via the use of the random multipliers r_1, r_2. The vector, floating-point nature of the algorithm suggests that high-performance implementations could be created that take advantage of modern hardware extensions pertaining to vectorization.

Typical convergence conditions include reaching a certain number of iterations, reaching a certain fitness value, and so on.

There are a number of considerations in using PSO in practice; say one might wish to limit the velocities to a certain maximum amount. **The considerable adaptability of PSO to variations and hybrids is seen as strength over other robust evolutionary optimization mechanisms, such as genetic algorithms.** For example, one common, reasonable modification is to add a probabilistic bit-flipping local search heuristic to the loop. Normally, a stochastic hill-climber risks getting stuck at local maxima, but the stochastic exploration and communication of the swarm overcomes this. Thus, **PSO can be seen as a basic search that can be adapted as needed for the problem at hand.**

The research literature has uncovered many heuristics and variants determined to be better with respect to convergence speed and robustness, such as clever choices of ω, c_i, and r_i. There are also other variants of the algorithm, such as discretized versions for searching over subsets of Z^n rather than R^n. There has also been experimentation with co-evolutionary versions of the PSO algorithm with good results reported. Very frequently the value of ω is taken to decrease over time; e.g., one might have the PSO run for a certain number of iterations and decrease linearly from a starting value (0.9, say) to a final value (0.4, say) in order to facilitate exploitation over exploration in later states of the search. The literature is full of such

Algorithm 2.4 Particle Swarm Optimization

Input: a random initial population of possible solution
Output: optimized solution for the problem

```
// Initialize the particle positions and their velocities
for I = 1 to number of particles n do
    for J = 1 to number of dimensions m do
        X[I][J] = lower limit + (upper limit - lower limit) * uniform random number
        V[I][J] = 0
    enddo
enddo
fitness_gbest = inf; // Initialize the global and local fitness to the worst possible
for I = 1 to number of particles n do
    fitness_lbest[I] = inf
enddo
// Loop until convergence, in this example a finite number of iterations chosen
for k = 1 to number of iterations to do
// evaluate the fitness of each particle
        fitness_X = evaluate_fitness(X)
// Update the local bests and their fitness
 for I = 1 to number of particles n do
        if (fitness_X[I] < fitness_lbest[I])
                fitness_lbest[I] = fitness_X[I]
                for J = 1 to number of dimensions m do
                        X_lbest[I][J] = X[I][J]

                enddo
        endif
  enddo
   // Update the global best and its fitness
  [min_fitness, min_fitness_index] = min(fitness_X)
       if (min_fitness < fitness_gbest)
            fitness_gbest = min_fitness
              for J = 1 to number of dimensions m do
                  X_gbest[J] = X(min_fitness_index,J)
              enddo
       endif
// Update the particle velocity and position
for I = 1 to number of particles n do
      for J = 1 to number of dimensions m do
            R1 = uniform random number
            R2 = uniform random number
          V[I][J] = w*V[I][J] + C1*R1*(X_lbest[I][J] - X[I][J])
                  + C2*R2*(X_gbest[J] - X[I][J])
          X[I][J] = X[I][J] + V[I][J]
      enddo
  enddo

enddo
```

heuristics. In other words, the canonical PSO algorithm is not as strong as various improvements which have been developed on several common function optimization benchmarks and consulting the literature for ideas on parameter choices and variants for particular problems is likely to be helpful.

Significant, non-trivial modifications have been developed for multi-objective optimization, versions designed to find solutions satisfying linear or non-linear constraints, as well as niching versions designed to find multiple solutions to problems where it is believed or known that there are multiple global minima which ought to be located. There is also a modified version of the algorithm called **repulsive particle swarm optimization, in which a new factor, called repulsion, is added to the basic algorithm step. Although a relatively new paradigm, PSO has been applied to a variety of tasks**, such as the training of artificial neural networks and for finite element updating. Very recently, PSO has been applied in combination with grammatical evolution to create a hybrid optimization paradigm called "grammatical swarms".

2.2.3.1 Comparisons between Genetic Algorithm and PSO

Most of evolutionary techniques have the following procedure:

1. Random generation of an initial population

2. Reckoning of a fitness value for each subject. It will directly depend on the distance to the optimum.

3. Reproduction of the population based on fitness values.

4. If requirements are met, then stop. Otherwise go back to 2.

From the procedure, we can see that PSO shares many common points with GA. Both algorithms start with a group of a randomly generated population, both have fitness values to evaluate the population. Both update the population and search for the optimium with random techniques. Both systems do not guarantee success. However, **PSO does not have genetic operators like crossover and mutation.** Particles update themselves with the internal velocity. They also have memory, which is important to the algorithm. Compared with genetic algorithms (GAs), the information sharing mechanism in PSO is significantly different. In GAs, chromosomes share information with each other. So the whole population moves like a one group towards an optimal

area. **In PSO, only gBest (or lBest) gives out the information to others. It is a one way information sharing mechanism.** The evolution only looks for the best solution. **Compared with GA, all the particles tend to converge to the best solution quickly** even in the local version in most cases.

2.2.4 Differential Evolution The global optimization problem minimize $f(x)$ subject to $x \varepsilon D$, where x is a continuous variable with the domain $D \subset \mathbb{R}^d$, and $f(x)$: $D \longrightarrow R$ is a continuous function. The domain D is defined by specifying lower (a_j) and upper (b_j) limits of each component x_j, $D = \prod^d_{j=1} [a_j, b_j]$,

$a_j < b_j$, $j = 1,2, ..., d$. This specification of D is also called the *boundary (box) constraints*. The global minimum point $x^* = \arg\min_{x \varepsilon D} f(x)$ is the solution of the problem. The algorithm of differential evolution in pseudo-code is given as Algorithm 2.5.

2. 2.4.1 Differential evolution and its control parameters

The differential evolution (DE) introduced by Storn and Price has become one of the most frequently used evolutionary algorithms for solving the continuous global optimization problems in recent years.

A new trial point y (line 4 in Algorithm) is generated by using mutation and crossover. There are various strategies, how to create the mutant point **u**. The most popular strategy called **DE/rand/1/** generates the point **u** by adding the weighted difference of two points $\mathbf{u} = r_1 + F(r_2 - r_3)$, where r_1, r_2 and r_3 are three mutually distinct points taken randomly from population P, not coinciding with the current x_i, and $F>0$ is an input parameter.

Another strategy called **DE/best/2/** generates the point **u** according to formula $\mathbf{u} = x_{min} + F(r_1 + r_2 - r_3 - r_4)$, where x_{min} is the point of P with minimal function value, r_1, r_2, r_3, r_4 are four mutually distinct points taken randomly from P not coinciding with the current x_i or x_{min}, and $F>0$ is an input parameter. The efficiency of differential evolution is very sensitive to the setting of values F and CR. The values recommended in literature are $F=0.8$ and $CR=0.5$, but even Storn and Price in their numerical tests set up values of F and CR depending on the results of preliminary tuning, $0.5 \leq F \leq 1$, and $0 \leq CR \leq 1$. Several papers deal with the setting of control parameters. Zaharie derived the critical intervals for the control parameters. Some other attempts to the adaptation of DE control parameters are summarized in Liu and Lampinen.

Algorithm 2.5 Differential Evolution

Input: a random initial population of possible solution

Output: optimized solution for the problem

1 generate an initial population $P = (x_1, x_2, ..., x_N)$, $x_1 \varepsilon D$

2 ***repeat***

3 ***for i= 1 to N do***

4 generate a new trial vector y

5 ***if*** $f(y) < f(x_1)$ ***then*** *insert y into new generation Q*

6 ***else*** *insert* x_1 *into new generation Q*

7 ***endif***

8 ***endfor***

9 P:=Q

10 ***until*** *stopping condition*

2.3 Text Clustering

Text mining applies data mining techniques, such as clustering, classification or association rule search to textual information. Typically high-dimensional (or irreducible to a simple vector representation) data, Text mining is highly dependent on document representation, or index. Background knowledge on information retrieval is required. Data preprocessing (=feature selection) is essential (stopword filtering, term stemming).

Among the text mining tasks are (a) **Data preparation and pre-processing (feature selection)** which includes *tokenization, stopword filtering, term stemming, vectorization, and dictionary compilation* (b) **Prediction** by methods such as naive Bayes and advanced linear models (c) **Information retrieval** by k- nearest neighbors and document matching (d) **Document clustering** and (e) **Information extraction of named entities**.

In the case of document clustering, high dimensionality is not a problem for computing the distance of two documents. Their vectors are sparse, so that only a small fraction of the theoretically possible M component wise differences need to be computed. Centroids, however, are dense since they pool all terms that occur in any of the documents of their clusters. As a result, distance computations are time consuming in a naive implementation of K-Means.

2.3.1 **Vector Space Model**

A starting point for applying clustering algorithms to unstructured text data is to create a *vector space model*, alternatively known as a *bag-of-words* model. The core idea is (i) extract distinct content bearing words from the collection of documents and consider these words as features (ii) Thereafter represent each document as a vector of certain weighted word frequencies in the feature space.

The vector space model computes a measure of similarity by defining a vector that represents each document, and a vector that represents the query [77]. The meaning of a document is conveyed by the words used. If one can represent the words in a document by a vector, it is possible to compare documents with queries to determine how similar their content is. If a query is considered to be like a document, a similarity coefficient (SC) that measures the similarity between a document and a query can be computed. Documents whose content, as measured by the terms in the document, correspond most closely to the content of the query are judged to be the most relevant.

The model involves constructing a vector that represents the terms in the document and another vector that represents the terms in the query. Next, a method must be chosen to measure the closeness of any document vector to the query vector. One could look at the magnitude of the difference vector between two vectors, but this would tend to make any large document appear

to be not relevant to most queries, which typically are short. The traditional method of determining closeness of two vectors is to use the size of the angle between them. This angle is computed by using the inner product (or dot product); however, it is not necessary to use the actual angle. Any monotonic function of the angle suffices. Often the expression "similarity coefficient" is used instead of an angle. Computing this number is done in a variety of ways, but the inner product generally plays a prominent role. Underlying this is the idea that a document and a query are similar to the extent that their associated vectors point in the same general direction.

There is one component in these vectors for every distinct term or concept that occurs in the document collection. Consider a document collection with only two distinct terms, α and β. All vectors contain only two components, the first component represents occurrences of α, and the second represents occurrences of β. The simplest means of constructing a vector is to place a one in the corresponding vector component if the term appears and a zero if the term does not appear. Consider a document D_1, that contains two occurrences of term α and zero occurrences of term β. The vector <1, 0> represents this document using a binary representation. This binary representation can be used to produce a similarity coefficient, but it does not take into account the frequency of a term without a document. By extending the representation to include a count of the number of occurrences of the terms in each component, the frequency of the terms can be considered. In this example, the vector would now appear as <2, 0>.

Instead of simply specifying a list of terms in the query, a user is often given the opportunity to indicate that one term is more important than another. This was done initially with manually assigned term weights selected by users. Another approach uses automatically assigned weights—typically based on the frequency of a term as it occurs across the entire document collection. **The idea was that a term that occurs infrequently should be given a higher weight than a term that occurs frequently**. Similarity coefficients that employed automatically assigned weights were compared to manually assigned weights [77]. It was shown that automatically assigned weights perform at least as well as manually assigned weights. Unfortunately, these results did not include the relative weight of the term across the entire collection.

This example illustrates the use of weights based on the collection frequency. Weight is computed using **Inverse Document Frequency (IDF)** corresponding to a given term. To construct a vector that corresponds to each document, we define

t= number of distinct terms in the document collection

tf_{if} = *number of occurrences of term* t_j *in Document* D_i. This is the referred to as the **term frequency.**

df_j= *number of documents which contain* t_j. This is the **document frequency.**

The vector for each document has n components and contains an entry for each distinct term in the entire document collection. The components in the vector are filled with weights computed for each term in the document collection. The terms in each document are automatically assigned weights based on how frequently they occur in the entire document collection and how often a term appears in a particular document. **The weight of a term in a document increases the more often the term appears in one document and decreases the more often it appears in all other documents.**

A weight computed for a term in a document vector is non-zero only if the term appears in the document. For a large document collection consisting of numerous small documents, the document vectors are likely to contain mostly zeros. For example, a document collection with 10,000 distinct terms results in a 10,000-dimensional vector for each document. A document with only 100 distinct terms will have a document vector that contains 9,900 zero-valued components.

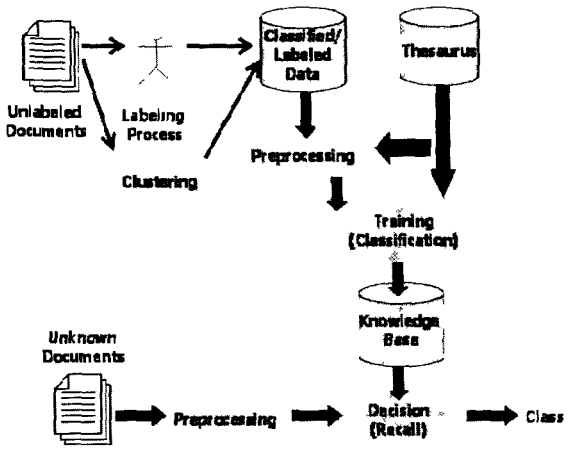


Fig 2.2: Document Clustering

The weighing factor for a term in a document is defined as a combination of term frequency, and inverse document frequency. That is, to compute the value of the jth entry in the vector corresponding to document i, the following equation is used:

$d_{ij}=tf_{ij} \times idf_j$ ..... (2.9)

Document clustering is done by utilizing mathematical algorithms against document content. The resulting values are then compared to one another such that documents with like content can then be grouped together. Document clustering algorithms can be hierarchical or partitioned. Hierarchical algorithms find successive clusters using previously established clusters, whereas partitioned algorithms determine all clusters at once.

Once data is loaded after the conversion to numeric values, the documents can now be processed and clustered together based upon like content via the algorithms like K-Means.
The clustering process actually involves the following steps:

(a) converting the text documents into its numeric equivalent dataset (pre-process to form the vector space model)

(b) read the numeric dataset from disk

(c) Use a particular clustering algorithm

2.4 K-Means

K-Means is the most commonly used partition clustering algorithm. The challenge of designing improved versions of this 50 year old algorithm is still being felt as the size of datasets and databases continues to grow. Numerous approaches have been employed on K-means to better its performance. This demonstrates the importance of K-means as a clustering technique. Some of these approaches like the standard K-means, FCM or KSOM converge to a local minimum. All the reported findings of the optimization based methods overcome the limitations of local convergence, and provide a global minimum. This proves the superiority of the optimization based approaches. An excellent article highlighting the relevance of K-Means in comparison

to other clustering algorithms can be found in the paper [9], whereas Table 2.3 below lists some approaches used on K-Means to improve its performance.

Algorithm	Input Parameter	Structure	Convergence	Optimised for	Search technique	Time complexity
K-means	No of clusters	Hyper-spherical clusters	Convergences to locally optimal solutions Dependent on initial seed selection	Separated clusters, Large data sets	Localised	O (nkl)
FCM	No of clusters	Non convex clusters	Converges to local minima of the squared error criterion Dependent on initial seed selection	Large data sets	Localised	O(n)
GA based FCM	Automatic	Non Convex clusters	Parameters dependent (population size, crossover and mutation probabilities)	One dimensional Small data sets	Globalised	Execution time is very high for large data sets
KSOM	Automatic	Hyper spherical clusters	Parameters dependent (learning rate and neighbourhood of the winning node)	Large data sets	Localised	Execution time is very high for large data sets
Simulated Annealing	Acceptance probabilities, global parameter T (temperature), annealing schedule, state space, Energy (goal) function E	Compact Clusters	Finds global minima, and overcomes the problem of local convergence of k means even for simple datasets	Significant improvement of performance over K Means in the case of overlapping and complex data sets	Globalised	Execution time is dependent on input parameters

Table 2.3: K-Mean vis-à-vis improvement techniques (Assorted K-means improvement techniques)

The *K-Means* algorithm is simple and fast [100]. Sensitivity to initial points and convergence to local optima are usually among the problems affecting the interactive techniques such as K-Means [4 11]. K-Means is one of the simplest unsupervised learning algorithms. The procedure follows a simple and easy way to classify a given data set through a certain number of clusters (assume k clusters) fixed a priori. The main idea is to define k centroids, one for each cluster. These centroids should be placed in a cunning way because different location causes different result. So, the better choice is to place them as much as possible far away from each other. The next step is to take each point belonging to a given data set and associate it to the nearest centroid.

When no point is pending, the first step is completed and an early grouping is done. At this point we need to re-calculate k new centroids as centers of the clusters resulting from the previous step. After we have these k new centroids, a new binding has to be done between the same data set points and the nearest new centroid. A loop has been generated. As a result of this loop we may notice that the k centroids change their location step by step until no more changes are done. In other words centroids do not move any more. Finally, this algorithm aims at minimizing an *objective function*, in this case a squared error function.

Although it can be proved that the procedure will always terminate, the K-Means algorithm does not necessarily find the most optimal configuration, corresponding to the global objective function minimum. Minimizing this objective function is known to be an NP-hard problem (even for K = 2) [74]. Thus ***K-Means, which is a greedy algorithm, can only converge to a local minimum,*** even though recent study has shown with a large probability K-Means could converge to the global optimum when clusters are well separated [9 16].

The K-Means algorithm is also known to be too slow for practical databases [20]. The K-Means algorithm can be run multiple times to reduce this effect. Unfortunately there is no general theoretical solution to find the optimal number of clusters for any given data set. A simple approach is to compare the results of multiple runs with different k classes and choose the best one according to a given criterion. The K-Means algorithm [69] is as shown as Algorithm 2.6. The objective function which is a chosen distance measure between a data point and the cluster centre is an indicator of the distance of the *n* data points from their respective cluster centers. The objective of K-Means is to minimize the average squared Euclidean distance of objects or documents from their cluster centers where a cluster center is defined as the mean or *centroid.*

The first step of K-Means is to select as initial cluster centers k randomly selected points, the *seeds.* The algorithm then moves the cluster centers around in space in order to minimize squared Euclidean distance. This is done iteratively by repeating two steps until a stopping criterion is met: reassigning points to the cluster with the closest centroid; and re-computing each centroid based on the current members of its cluster. One can apply one of the following termination conditions :

(i) *A fixed number of iterations has been completed.* This condition limits the runtime of the

Algorithm 2.6 K-Means Algorithm

1. Select K points as initial centroids
2. **repeat**
3. Form K clusters by assigning each point to its closest centroid
4. Re-compute the centroid of each cluster
5. **until** Centroids do not change

clustering algorithm, but in some cases the quality of the clustering will be poor because of an insufficient number of iterations.

(ii) *Assignment of points to clusters (the partitioning function) does not change between iterations.* Except for cases with a bad local minimum, this produces a good clustering, but runtime may be unacceptably long.

(iii) *Centroids do not change between iterations.* This is equivalent to partitioning function not changing.

(iv) *Terminate when error value falls below a threshold.* This criterion ensures that the clustering is of a desired quality after termination. In practice, we need to combine it with a bound on the number of iterations to guarantee termination. For small threshold, this indicates that we are close to convergence. Again, we need to combine it with a bound on the number of iterations to prevent very long runtimes.

There are two main problems for K-Means algorithm. *First, in each iteration step, much computation time is spent on assigning every point in the data set to its new nearest center. Second, the algorithm is easy to be trapped in some local optimum, which can be much worse than the global optimum.*

For the first problem, there have been several works on accelerating the nearest center search Procedure based on triangle inequality or indexing structures, which can work

well in low dimensional space. Compared with the first problem, the second problem has not been well addressed yet. Although there are some studies on the choices of initial centers to avoid those local optimums, these methods do not show too much advantage over the simple random selection in the data set. Hence, a study employing different optimizing scheme on improving the iterative K-Means can bring out the relative strength and weaknesses of the optimizing techniques and interestingly reveal the best method to use for this very well-known problem.

The complexity of K-Means algorithm can be estimated as below

Assume computing distance between two instances is O(m) where m is the dimensionality of the vectors.

Reassigning clusters: O(kn) distance computations, or O(knm).

Computing centroids: Each instance vector gets added once to some centroid: O(nm).

Assume these two steps are each done once for I iterations: O(Iknm).

Linear in all relevant factors, assuming a fixed number of iterations,

- *more efficient than $O(n^2)$ than HAC*

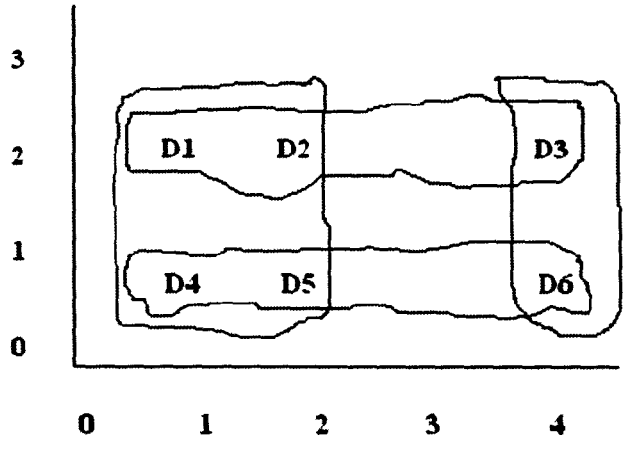


Fig 2.3 The outcome of K-Means depends on its initial seed. For seeds D2 and D5, K-Means converges to {D1, D2, D3} & {D4,D5,D6}, a sub optimal clustering. For seeds D2 and D3, it converges to {D1,D2,D4,D5} & {D3,D6}, a global optimum for K=2.

The K-Means algorithm is an iterative algorithm for minimizing the sum of distance between each data point and its cluster center (centroid). There are various ways of measuring distance such as **squared Euclidean distance, city-block, cosine dissimilarity, Hamming distance** and a few others. **Squared Euclidean distance is the most widely used distance measure for K-Means**, and the corresponding criterion function that a Euclidean K-Means algorithm optimizes is the **sum-squared-error (SSE)** criterion. K-Means clustering and Gaussian mixture clustering may not converge to the global optimum. The performance of K-Means and Gaussian mixture clustering strongly depends on the initial starting points [10].

Several random initialization methods for K-Means have been developed. Two classical methods are random seed and random partition. Random seed randomly selects k instances (seed points), and sends each of the other instances to the cluster with the nearest seed point. Random partition assigns each data instance into one of the k clusters randomly. To prevent from getting stuck at a local minimum, one can apply r random starts. One can perform either of the above methods to initialize K-Means, repeat the process r times, and select the final clustering with the minimum SSE from the r runs. The problem with random methods is that they're not repeatable (unless one stores all the starting points applied or the seeds of the random number generator), and they may still lead to a solution with bad quality unless we allow r to be very large making the clustering time-consuming for large data sets.

[12] have outlined a procedure for computing a refined starting condition from a given initial one that is based on an efficient technique for estimating the modes of a distribution. The refined initial starting condition leads to convergence to better local minima. Refinement run time is considerably lower than the time required to cluster the full database. The method is scalable and can be coupled with a scalable clustering algorithm to address the large-scale clustering in data mining. By initializing a general clustering algorithm near the modes, not only are the true clusters found more often, but it follows that the clustering algorithm will iterate fewer times prior to convergence.

Effective heuristics for seed selection include

(i) *excluding outliers from the seedset*;

(ii) *trying out multiple starting points and choosing the clustering with lowest cost*; and

(iii) *obtaining seeds from another method such as hierarchical clustering.*

Since deterministic hierarchical clustering methods are more predictable than K-Means, a hierarchical clustering of a small random sample of size ik (e.g., for i=5 or i=10) often provides good seeds (Buckshot algorithm). Other initialization methods compute seeds that are not selected from the vectors to be clustered. A robust method that works well for a large variety of document distributions is to select i (e.g.,i=10) random vectors for each cluster and use their centroid as the seed for this cluster.

Because of the problems with using randomly selected initial centroids, which even repeated runs may not overcome, other techniques are often employed for initialization. One effective approach is to take a sample of points and cluster them using a hierarchical clustering technique. K clusters are extracted from the hierarchical clustering, and the centroids of these clusters are used as the initial centroids. This approach often works well, but is practical only if (1) the sample is relatively small, a few hundred to a few thousand as hierarchical clustering is expensive & (2) k is relatively small compared to the sample size.

In K-Means, the process of centroid selection is done randomly. This leads to sub optimal solutions, especially for complex datasets [143]. **In large databases** where we have hundreds of dimensions and tens of thousands to millions of records, **better centroids exhibits the greatest value.** The reason is simple: a clustering session on a large database is a time-consuming affair. Hence a refined centroid can insure that the time investment pays off. The refinement algorithm can operate over small sub-samples of the database and hence run-times needed to determine "good" centroid points (which speeds the convergence on the full data set) are orders of magnitude less than the total time needed for clustering in a large-scale situation.

Our initial experiments were conducted in the domain of textual clustering. Fast and high-quality document clustering algorithms play an important role in effectively navigating, summarizing, and organizing information. In recent years, it has been recognized that the partitional clustering technique (like K-Means) is well suited for clustering a large document dataset due to their relatively low computational requirements [35 36]. One algorithm for enhancing the text clustering process by reducing the dimensionality of feature space and thus reducing time for processing and enhancing clustering procedure is Vector–Space Model. The algorithm has used the document as a document to term matrix for calculation procedure.

What is the time complexity of K-Means? Most of the time is spent on computing vector distances. One such operation costs θ (M). The reassignment step computes KN distances, so its overall complexity is θ (KNM). In the re-computation step, each vector gets added to a centroid once, so the complexity of this step is θ (NM). For a fixed number of iterations I, the overall complexity is therefore θ (IKNM). Thus, K-Means is linear in all relevant factors: *iterations, number of clusters, number of vectors* and *dimensionality of the space.* **This means that K-Means is more efficient than the hierarchical algorithms.** In most cases, K means quickly reaches either complete convergence or a clustering that is close to convergence. In the latter case, a few points would switch membership if further iterations were computed, but this has a small effect on the overall quality of the clustering. Even a linear algorithm can be quite slow if one of the arguments of θ (...) is large, and M usually is large.

2.4.1 Local Convergence of K-Means and its Avoidance

The K-Means algorithm requires **three user-specified parameters**: number of clusters *K*, cluster initialization [126], and distance metric [9]. The most critical choice is *K*. While no perfect mathematical criterion exists, a number of heuristics [66] are available for choosing *K*. Typically, K-Means is run independently for different values of *K* and the partition that appears the most meaningful to the domain expert is selected.

Different initializations can lead to different final clustering because K-Means only converges to local minima. One way to overcome the local minima is to run the K-Means algorithm, for a given K, with multiple different initial partitions and choose the partition with the smallest squared error. Here we look into some efforts using different approaches towards overcoming the local convergence of K-Means

2.4.2 Lloyd, K-Means and the Continuous K-Means

A fundamental problem in cluster analysis is although the algorithm will produce the desired number of clusters, the centroids of these clusters may not be particularly representative of the data. **What determines a good or 'representative' clustering? In a cluster, if the data points are tightly clustered around the centroid, the centroid will be representative of all the points in that cluster.** The standard measure of the spread of a group of points about its mean is its variance, or sum of the squares of the distance between each point and the mean. If the data points are close to the mean, the variance will be small. A generalization of the

variance, in which the centroid is replaced by a reference point that may or may not be a centroid, is used in cluster analysis to indicate the overall quality of a partitioning; specifically the error measure E is the sum of all the variances

$$E = \sum_{i=1}^{k} \sum_{j=1}^{n_i} ||x_{ij} - z_i||^2 \quad .... \quad (2.10)$$

where x_{ij} is the j^{th} point in the i^{th} cluster, z_i is the reference point of the ith cluster, and n_i is the number of points in that cluster. The notation $|| x_{ij} - z_i ||$ stands for the distance between x_{ij} and z_i. Hence, the error measure E indicates the overall spread of data points about their reference points. **To achieve a representative clustering, E should be as small as possible.** The error measure provides an objective method for comparing partitioning as well as a test for eliminating unsuitable partitioning. Finding the best partitioning (clustering most representative of a data set) requires generating all possible combinations of clusters and comparing their error measures. This can be done for small datasets with a few dozen points, but not for large sets – the number of different ways to combine 1 million data points into 256 clusters, for example, is 256 X 1,000,000 / 256!,where $256! = 256\ X\ 255\ X\ .....X\ 2\ X\ 1$ This number is greater than 1 followed by 2 million zeros [14].

When clustering is done for the purpose of data reduction, the goal is not to find the best partitioning. It is desirable to want a consolidation of N data points into k clusters, and some efficient way to improve the quality of the initial partitioning.

Iterative algorithms begin with a set of k reference points whose initial values are usually chosen by the user. Initially, the data points are partitioned into k clusters: a data point x becomes a member of cluster i if z_i is the reference point closest to x. The reference points and the assignment of the data points to clusters are then adjusted during successive iterations. **Iterative algorithms are thus similar to fitting routines, which begin with an initial guess for each fitted parameter and then optimize their values.** Algorithms within this family differ in the details of generating and adjusting the partitions. Lloyd's algorithm, the standard K-Means algorithm and a continuous K-Means algorithm first described in 1967 by J. MacQueen belong to this category [14]. In Lloyd's algorithm [67] all the data points are partitioned into k clusters by assigning each point to the cluster of the closest reference points. Adjustments are made by calculating the centroids of each of those clusters and then using those centroids as reference points for the next partitioning of all the data points. A

local minimum of the error measure E corresponds to a 'centroidal Voronoi' configuration where each data point is closer to the reference point of its cluster than to any other reference point, and each reference point is the centroids of its cluster. The purpose of iteration is to move the partition closer to this configuration and thus to approach a local minimum for E.

For Lloyd's and other iterative algorithms, improvement of the partitioning and convergence of the error measure E to a local minimum is generally very fast, even when the seed points are badly chosen. Different initial partitioning generally does not produce the same set of final clusters. No doubt the final partitioning would better the initial cluster, but it will not necessarily be the best possible partitioning.

The standard K-Means algorithms differ from Lloyd's in its better use of information at every step. Reference points are chosen and all the data points are assigned to clusters.
As with Lloyd's, the K-Means algorithm then uses the cluster centroids as reference points in subsequent partitioning- but the centroids are adjusted both during and after each partitioning. There are a number of variants of the K-Means algorithm. In some versions, the error measure E is evaluated at each step, and a data point is reassigned to a different cluster only if that reassignment decreases E. The K-Means algorithm constantly updates the clusters (assign data points to cluster, re-compute the centroids, shift the reference point to the centroids) is unlikely to require as many iterations as the less efficient Lloyd's algorithm and is therefore considerably faster.

The continuous K-Means algorithm is faster than the standard version and has proved scalable for larger datasets. It differs from the standard version in how the initial reference points are chosen and how data points are selected for the updating process. In the standard algorithm the initial reference points are chosen more or less arbitrarily. In the continuous algorithm, reference points are chosen as a random sample from the whole population of data points. If the sample is sufficiently large, the distribution of these initial reference points should reflect the distribution of points in the entire set. The standard algorithm examines all the data points in sequence. The continuous algorithm, on the other hand, examines only a random sample of data points. If the dataset is very large and the sample is representative of the dataset, the algorithm should converge much more quickly than an algorithm that examines every point in sequence.

The modification to the standard algorithm greatly accelerates the clustering process. Since both the reference point and the data points for the updates are chosen by random sampling, more reference points will be found in the densest regions of the dataset and the reference points will be updated by data points in the most critical regions. The initial reference points are already members of the dataset and require fewer updates. Even when applied to a large data set the algorithm normally converges to a solution only after a small fraction (10 to 15 percent) of the total points have been examined.

This rapid convergence distinguishes the continuous K-Means from less efficient algorithms. **Clustering with the continuous K-Means algorithm is about ten times faster than clustering with Lloyd's algorithm.** Two features of the continuous K-Means algorithm – convergence to a feasible group of reference points after very few updates and greatly reduced computer time per update – are highly desirable for any clustering algorithm.

2.4.3 Kaufman Approach (KA) has been proposed by [7]. Here the initial clustering is obtained by the successive selection of representative instances until K instances have been found. The first representative instance is the most centrally located instance in the database. The rest of the representative instances are selected according to the heuristic rule of choosing the instances that promise to have around them a higher number of the rest of instances.

In a comparative study of four K-Means initialization methods [15], KA demonstrates its capability to induce to the K-Means algorithm a more desirable behavior than the other three methods.

2.4.4 Buckshot Algorithm

Buckshot algorithm tries to improve the performance of K-Means algorithm by choosing better initial cluster centroids. It uses Hierarchical Agglomerative Clustering (HAC) algorithm, and considers each point as a separate cluster and combines the cluster with the maximum similarity. Similarity between clusters is measured as a group average. When the no. of clusters left equals the no. of required clusters, the algorithm is stopped. And the centroids of these clusters are taken as initial centroids for the K-Means algorithm.

Fig 2.4 clarifies the distinction between a local minimum and a global minimum. In the fig. one finds the graph of a function and points A and B. The overall or global minimum as one can see is at point B, which is smaller than point A, so A corresponds only to a local minimum. It is desirable to get to B and not get stopped at A itself, in the pursuit of a minimum for the function. If point C is reached, one would like the further movement to be toward B and not A. Similarly, if a point A is reached, the subsequent movement should avoid reaching or settling at A but carry on to B. Perturbation techniques are useful for these considerations.

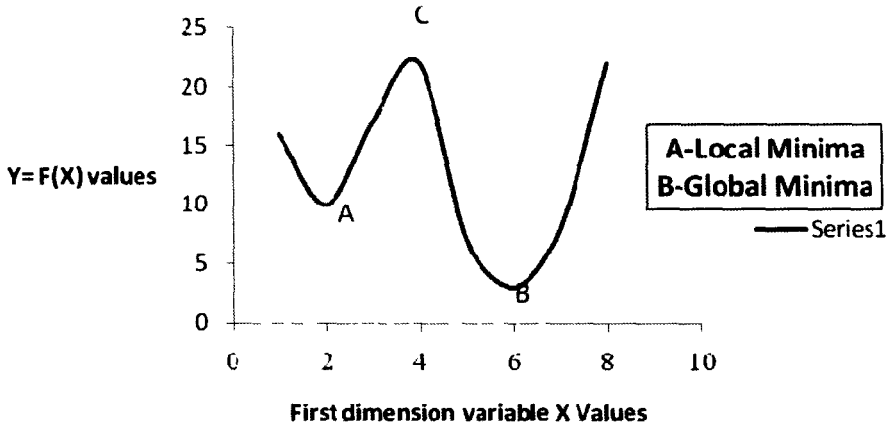


Fig 2.4: Local and Global Minima Src: [67]

2.4.5 Comparative Table of Approaches in improving K-Means

#	Technique of initial seed selection in K-Means	Reported Research work	Comments on methods and Performance
1	**Iterative**	Refer [67]	Refines clusters iteratively • Clusters points using Voronoi partitioning of the centers • Centroids of the clusters determine the new centers Local convergence is achieved very fast
		Refer [85]	Forgy initialization: choose *k* points at random as starting center locations. Not a very good method.
		Standard K-Means, Continuous K-Means	See description against technique 2 below as randomized based approach.
2	**Randomized based**	Continuous K-Means	Uses a sample of data points for initial seed selection and even for subsequent updates. Two features of continuous K-Means algorithm – convergence to a feasible group of reference points after very few updates and greatly reduced computer time per update – are highly desirable for any clustering algorithm. **Continuous K-Means is 10 times faster than Lloyd's algorithm.**
		Standard K-Means	Similar with Lloyd's, in standard K-Means algorithm the centroids are adjusted both during and after each partitioning. **Considerably faster than Lloyd's algo**
		Random partitions	Divides the data points randomly into *k* subsets.
3	**Sampling based**	(i)Continuous K-Means, (ii) Refer [11] (iii) Subset Furthest First (SFF) Algorithm (iv) Refer [12]	1. As above 2. [11] discuss ways to refine the selection of starting centers through repeated sub-sampling and smoothing. Use K-Means M times for M random subsets of the original data. 3.A random sample includes many representative points, but few outliers found by algorithm Furthest First (FF). 4.[12] indicates that the solutions obtained by clustering over a small subsample may provide good refined

			initial estimates of the true means or centroids in the data.
4	**Clustering based**	Buckshot	It uses HAC algo for arriving at better initial centroids.
5	**Heuristic based**	(i) Refer [24] (ii) An approach to finding the c-cluster starting point from the solutions to the (c-1) cluster problem Refer [86]	(i) The "furthest first" algorithm (FF): 1. Pick first center randomly. 2. Next is the point furthest from the first center. 3. Third is the point furthest from both previous centers. 4. In general: next center is $\text{argmax}_x \min_c d(x,c)$ **This is a smart initialization procedure** (ii) The solution for the one-cluster problem is the tota sample mean; the starting point for the c-cluster problem can be the final means for the (c-1) cluster problem plus the sample that is farthest from the nearest cluster center.
6	**Statistical based**	(i) Refer [23] (ii) Refer [19] (iii) Refer [22] (iv) Refer [12]	In [23], a kd-tree used to calculate an estimate of the density of data and to select the number of clusters. The paper [12] presents a procedure for computing a refined starting condition from a given initial one that is based on an efficient technique for estimating the modes of a distribution. The refined initial starting condition leads to convergence to "better" local minima.
7	**Optimization based**	Refer [25]	Genetic K-Means algorithm. Hybrid scheme based on Genetic Algorithm - Simulated annealing with new operators to perform global search and rapid convergence.

8	**Density based**	Kaufman Approach. Refer [7]. Here the initial clustering is obtained by the successive selection of representative instances until K instances have been found. The first representative instance is the most centrally located instance in the database. The rest of the representative instances are selected according to the heuristic rule of choosing the instances that promise to have around them a higher number of the rest of instances.	In a comparative study of four K-Means initialization methods [15], KA demonstrates its capability to induce to the K-Means algorithm a more desirable behavior than the other three methods.
9	**Hybrid based**	Refer [87]	The work [87] does not depend on the initial centers. Algorithm PSO-SA-K combines the algorithms "Particle Swarm Optimization ,"Simulated Annealing " and K-Means.
10	**Recursive**	Refer [88]	A recursive method for initializing the means by running *K* clustering problems is mentioned in [86] for K-Means. A variant consists of taking the mean of the entire data and then randomly perturbing it K times [88]
11	**Outlier based**	Refer [24]	(i) Furthest First (FF) finds outliers, by definition not good cluster centers To make the results of clustering less sensitive to outliers several fuzzy solutions were put forward, based on robust statistics or the use of a 'noise cluster'

Table 2.4 Improvement Approaches of K-Means

2.5 *Datasets used in experimentation*

2.5.1 ISA DATASET (BIGCHECK) - Synthetic dataset over ACM Citations. For the document source of the first experiment, we used 1060 records created for journal articles from the Information Science and Abstracts (ISA) database and computer science technical reports collected from various sites on the Internet. This is a comprehensive subset of the entire database. Each document record (sample shown below) includes a complete abstract, title, author, and subject keywords.

BEHAVIOR-/INFORMATION-SYSTEMS/PERCEPTION-/RESEARCH-/USAGE-STUDIES/VALIDATION-
Subject: Biosignal pattern recognition and interpretation systems.
Part 2 of 4:
Methods for feature extraction and selection.

The wide variety of techniques existent for feature extraction presents two problems: 1) which techniques should be used and 2) how to select from among the features that each extraction technique generates. Selected features are "best" only by some standard (i.e., criterion); therefore techniques for generation of features tend not to be very portable from one pattern processing problem to another. Production of salient features is the connecting link between prototypical and symbolic representations of a class. Often, thresholds govern the selection of features. Many techniques do not generate independent features; therefore there is redundancy in the data, which potentially affects both efficiency and accuracy in pattern recognition.
Ciaccio, E.J.//Dunn, S.M.//Akay, M.
IEEE-Engineering-in-Medicine-and-Biology Vol. 12, Issue 4, p. 106-113, Dec 1993, 22

2.5.2 SOYBEAN [84]- Standard dataset of 307 records over 35categorical attributes

There are 19 classes, only the first 15 of which have been used. The last four classes are unjustified by the data since they have so few examples. Of the categorical attributes, some are nominal and some ordered. The **soybean dataset** is one of the largest among crop species.

2.5.3 WATER TREATMENT PLANT [121]- Standard dataset of 527 records over 38 attributes.

This dataset comes from the daily measures of sensors in a urban waste water treatment plant. All atrributes are numeric and continuous. The attributes are integer and real, and the dataset has been used generally to estimate the task of clustering.

2. 6 Measures of Cluster Quality:

2.6.1 Silhouette Coefficient [149]

Silhouette Coefficient (SC) is based on the idea that a good clustering should consist of well-separated, cohesive clusters. For an individual point *i*

Calculate ***a*** = average distance of *i* to the points in its cluster

Calculate ***b*** = min (average distance of *i* to points in another cluster)

The silhouette coefficient for a point is then given by

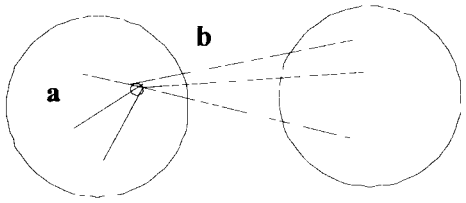


Fig 2.5 Silhouette Coefficient

$s = 1 - a/b$ if $a < b$, (Or $s = b/a - 1$ if $a \geq b$, not the usual case)

Typically SC lie between 0 and 1.

Higher the Silhouette Coefficient, more compact the cluster. The closer the values are to 1 the better.

2.6.2 Davies Bouldin Index [144]

The Davies-Bouldin index is a measure of the uniqueness of clusters in a given clustering. A measure for dispersion of a single cluster and dissimilarity between a pair of clusters is required. A simple definition for each is that the dispersion of a cluster S is the average Euclidean distance of all points to the cluster center. The dissimilarity D is defined as the distance between the two cluster centers.

It is a function of the ratio of the sum of within-cluster (i.e. intra-cluster) scatter to between cluster (i.e. inter-cluster) separations. Let $C = \{C_1, \ldots, C_k\}$ be a clustering of a set of N objects :

with $DB = \frac{1}{k} \cdot \sum_{i=1}^{k} R_i$ and $R_i = \max_{j=1,..k, i \neq j} R_{ij}$ $R_{ij} = \frac{\text{var}(C_i) + \text{var}(C_j)}{\| c_i - c_j \|}$

where C_i is the i^{th} cluster and c_i is the centroid for cluster i

Numerator of R_{ij} is a measure of intra-cluster similarity while the denominator is a measure of inter-cluster separation. Note, $R_{ij}=R_{ji}$

A low scatter and a high distance between clusters lead to low values of R_{ij} ; therefore **for a compact cluster a minimization of DB index is desired**

2.6.3 Inter Cluster Distance

Distance between the centroids measures the inter cluster distance.

2.7 Distance measure

2.7.1 Euclidean distance

Definition: A distance measure between two points. In a plane with p_1 at (x_1, y_1) and p_2 at (x_2, y_2), it is $\sqrt{((x_1 - x_2)^2 + (y_1 - y_2)^2)}$.

Note: In N dimensions, the Euclidean distance between two points p and q is $\sqrt{\sum_{i=1}^{N} (p_i-q_i)^2}$
where p_i (or q_i) is the coordinate of p (or q) in dimension i.

2.7.2 Pearson Correlation distance [122]

Pearson Correlation measures the similarity in shape between two profiles. The formula for the Pearson Correlation distance is:

$$d = 1 - r$$

where $r = Z(x)\cdot Z(y)/n$ is the dot product of the z-scores of the vectors x and y. The z-score of x is constructed by subtracting from x its mean and dividing by its standard deviation.

Chapter 3

Optimizer augmented clustering (OAC)

3.1 Why Optimization in Clustering?

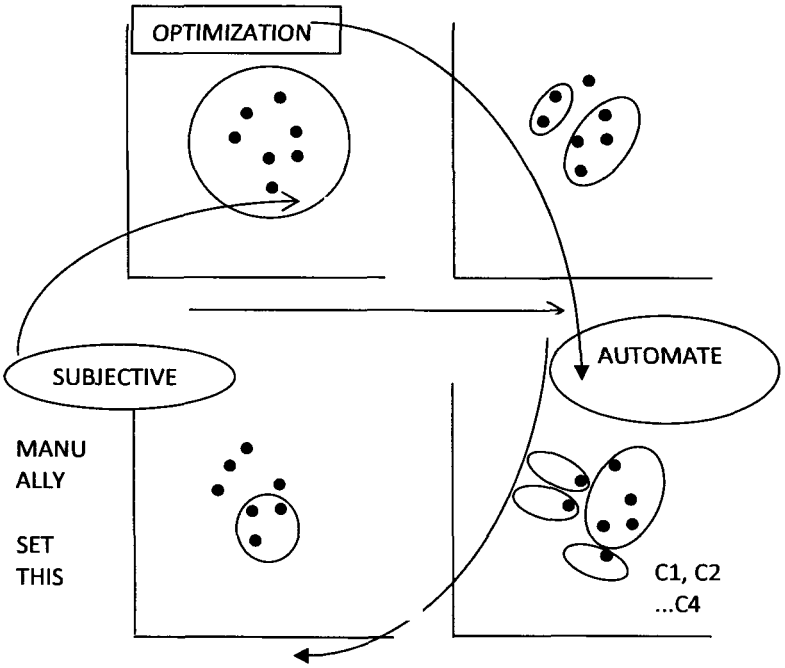


Fig 3.1 : Benefit of optimization in determining the right number of clusters

The diagram highlights the utility of optimization in the context of clustering. If one need to cluster the set of given points depicted in the figure, what should be the right number of clusters? This is a subjective issue, and as witnessed in the diagram above, should have ideally stopped when the number of clusters is three (3), as increasing the number of clusters (4 or 5) tends to make the clusters look rather sparse. In algorithms like K-Means, the user has to specify the number of clusters k.

3.2 Focus of Investigation *The central thrust of the dissertation is to investigate the benefits of employing heuristic and other evolutionary based optimization techniques such as genetic algorithm,*

particle swarm optimization, differential evolution, simulated annealing among many others on free parameters of clustering. We ***propose a multi-level method employing an optimization loop, in addition to the clustering algorithm, to achieve higher clustering performance through an automatic, better choice of these free parameters.***

As a case study, we have studied it on the benefits obtained for the iterative centroid improvement of K-Means, and thereafter on another clustering algorithm Maximin to estimate the recurrence of the improved results obtained using K-Means. Some initial experiments have been performed in the area of text clustering. For the document source, we used 1060 records created for journal articles from the Information Science and Abstracts (ISA) database and computer science technical reports collected from various sites on the Internet. This is a comprehensive subset of the entire database. The dataset ISA used for experimentation contains approximately 5,000 documents culled from abstracts of ACM. We have experimented using a subset of 1060 documents (referred as Bigcheck) which is a fairly representative subset of the entire dataset. Each document record includes a complete abstract, title, author, and subject keywords.

After the initial experiments conducted in the domain of textual data sets, we have later extended our experiments to standard dataset SOYBEAN and WATER TREATMENT PLANT. ***Four optimizing techniques viz., GA, SA, DE and PSO were used*** *to arrive at better centroids compared with an independent K-Means and Maximin. The* ***performances of these methods and an independent K-Means (& later with Maximin) run were compared*** *and studied* ***using metrics for compactness of clusters****. To* ***assess the improvement in cluster quality*** *of the methods, we* ***use two quantitative measure Silhouette coefficient and Davies Bouldin Index****.*

3.3 Optimizer Augmented Clustering (OAC)

Section 3.2 contains an introduction to our investigation. In this section, we represent pictorially the basic OAC scheme and the proposed work. We provide the architecture of Optimizer Augmented Clustering (OAC). The basic schematic of the OAC technique is shown in Fig. 3.2. Algorithm 3.1 depicts K-Mean algorithm vis-à-vis OAC.

As can be observed from Table 2.3, **a comprehensive systematic study of recent optimization based methods (evolutionary and heuristic based) on free parameters of some important clustering algorithms (like K-means & Maximin) can reveal useful information about performance related issues of such optimizer augmented algorithms. This is the prime objective of this investigation.**

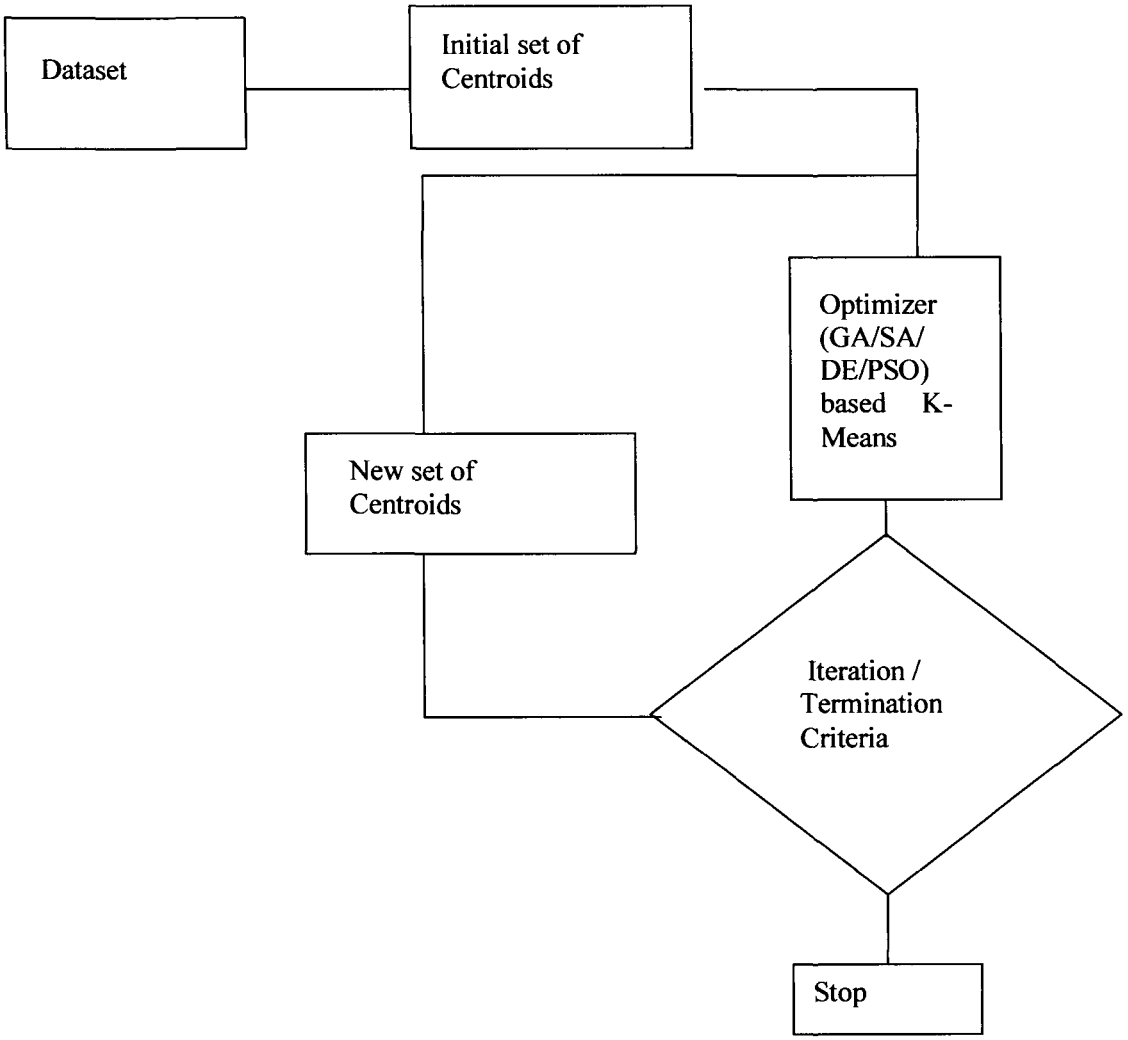


Fig **3.2: Basic schematic of OAC**

3.4 Optimization Approaches Used For Experimentation

3.4.1 Genetic Algorithms:

By using exhaustive search techniques, we can solve small problems to optimality, although the time complexity may be enormous. For certain applications, it may well pay to spend extra time to be certain of the optimal solution [96]. The use of **Genetic Algorithms (GA)** in improving the K-means clustering process has already been reported [27 28]. [27] exploits the searching capability of genetic algorithms in order to search for appropriate cluster centers in the feature space such that a similarity metric of the resulting clusters (within cluster spread) is optimized. [28] present a genetic algorithm for selecting centers to seed the

Algorithm 3.1 **K-Mean algorithm vis-à-vis OAC**

The K-Mean algorithm's major steps can be summarized as follows:

Step 1 - begin with an arbitrary assignment of samples to the clusters;

Step 2 - compute the sample mean of each cluster;

Step 3 - reassign each sample according to the nearest mean;

Step 4 - if the classification of all samples has not changed, stop; else go to step 2.

The OAC algorithm can be summarized as follows:

Step1 – begin with an arbitrary assignment of initial population

Step 2- compute fitness values of individuals

Step 3- Use fitness (maximize) to generate candidate solution/ improve solution, forming a new set of population

Step 4- If termination criteria is met, stop; else go to step 2

popular K-means method for clustering. Using a novel crossover operator that exchanges neighboring centers, the GA identifies superior partitions using both benchmark and large simulated data sets.

Though essentially our work can be considered as similar with Laszlo et al. in that the objective of investigation is the same i.e., finding good centers for K-means, yet the approach adopted is vastly different. With reference to [28] '*A genetic algorithm that exchanges neighboring centers for k-means clustering*', the following difference was observed

1. Laszlo et al. uses a novel region based crossover operator that exchanges neighboring centers. However, in our crossover approach, we're using a random 5-point crossover approach.
2. Laszlo et al. have used a crossover operator that exchanges subsets of centers that occupy the same region of space. Whereas our crossover operator exchanges random subsets of centers between parents.

3. As their GA proceeds, Laszlo et al. evolves individuals with different no. of genes. In our case, the no of genes in individual evolution remains the same

4. For selection, Laszlo et al. uses a roulette-wheel sampling with replacement, whereas we have used steady state replacement based on mating pool and higher fit values for selection

5. Laszlo et al. have used mutation with a low probability, whereas we have not used mutation.

6. The work of Laszlo et al. is inspired by a data structure oriented approach, being based on their earlier work employing hyper-quadtrees which they claim gave good results for low dimensional data.

We are using GA to arrive at better selection of centroids in the inner loop. Thereafter, the K-Means algorithm is applied and clustering is performed. In our method, we have used a random 5-point crossover approach, exchanging random subsets of centers between parents for crossover, and keeping the number of genes in individual evolution the same. We have used steady state replacement based on 20 % mating pool and higher fit values for selection. Specific to our work on using Steady GA, it may be mentioned that our work is vastly different from the commonly cited works on using GA to clustering using K-Means. Some of these, employed GA for clustering, and instead of crossover used K-Means as a genetic operator for clustering [25]A few other approaches used a novel region based crossover operator that exchanges neighbouring centers, evolving individuals with different number of genes using a roulette-wheel sampling with replacement, and also use mutation with a low probability [28]..

3.4.2 Particle Swarm Optimization (PSO):

PSO is another computational intelligence method that we have used. It has already been applied to image clustering and other low dimensional datasets [V. D. Merwe et al., 2003][M. Omran et al., 2002]. However, to the best of the author's knowledge, PSO has not been used to cluster text documents [Xiaohui Cui et al., 2005]. In this investigation, in the initial experiments, the **corpus of documents have been reduced to an equivalent numeric dataset** on which PSO augmented K-Means clustering algorithm have been implemented and evaluated for performance. Compared with genetic algorithms (GAs), the information sharing mechanism in PSO is significantly different. **In GAs, chromosomes share information with each other. So the whole population moves like a one group towards an optimal area. In PSO, only gBest (or lBest) gives out the information to others. It is a one -way information sharing mechanism.** The evolution only looks for the best solution. Compared with GA, all the particles tend to converge to the best solution quickly even in the local version in most cases.

In our PSO optimizer, we have considered the **set of centroids (fifteen for k=15) as a particle.** A collection of fifty such particles, the same set that was used during the experimentation with GA, forms the collective population or the **swarm.** During initialization of the parameters, we set the max iteration to 50 to find the particle best position. Each iteration updates the particle position and its velocity. Thereafter, the particle best and global best values are computed. The fitness value of the swarm is computed for successive iterations and compared. We terminate the procedure when the fitness value doesn't improve over five generations / iterations.

Contrary to the localized searching of the K-means algorithm, the PSO clustering algorithm performs a globalized search in the entire solution space. In the experiments we conducted, we applied the PSO in the inner loop before application of the K-Means clustering algorithm to obtain better centroids for iterative K-Means. The results highlight that the hybrid PSO algorithm can generate more compact clustering results than the K-Means algorithm.

3.4.3 Simulated Annealing:

The third optimizer we used, **Simulated Annealing (SA) optimizer,** uses an iterative procedure. Each step of the SA algorithm replaces the current solution by a random "nearby" solution, chosen with a probability that depends on the difference between the corresponding function values and on a global parameter T (called the *temperature*), that is gradually decreased during the process. The dependency is such that the current solution changes almost randomly when T is large, but increasingly "downhill" as T goes to zero. The allowance for "uphill" moves saves the method from becoming stuck at local minima—which are the bane of greedier methods. The perturbation operator used in general annealing has a simple meaning in clustering: it amounts to a relocation of a point from its current to a new randomly chosen cluster (very similar to *k*-means scheme).

In our experiments, the Maximum number of iterations before the cooling schedule is applied has been set to 10. The global parameter T have been set to 100, and thereafter a cooling schedule of ti = ti * 0.95 was used. The 'uphill' (bad) solutions were accepted with a probability of 0.9 to overcome local convergence. The convergence criteria taken was T=0. Since a particular iteration required a number of cooling schedules, taking quite a good amount of time with each cooling schedule, fitness function was computed after clustering with values obtained after each cooling schedule.

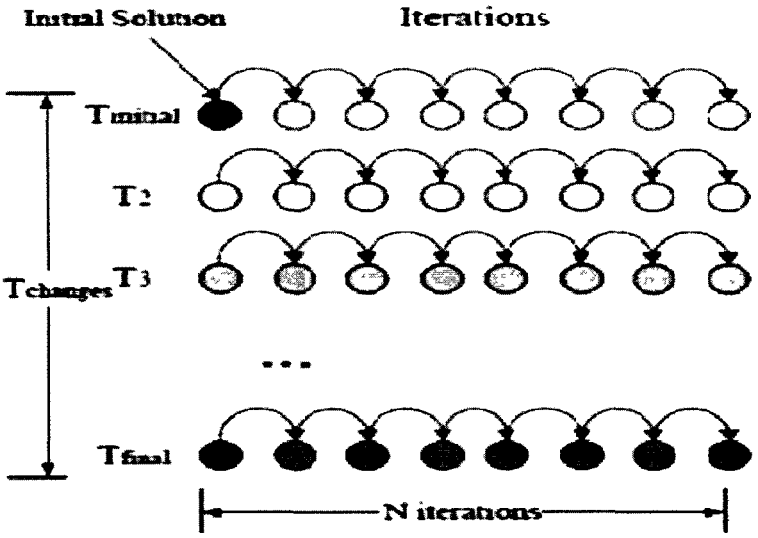


Fig 3.3 SA Structure [89]

3.4.4 Differential Evolution:

The fourth optimizer, **Differential Evolution**, has become very popular recently as an effective method for global optimization problems. In this technique, a new trial point y is generated by using mutation and crossover. There are various strategies on how to create the mutant point u. One of the most popular strategy called **DE/rand/1/** generates the point u by adding the weighted difference of two points

$u = r_1 + F(r_2 - r_3)$ where r_1, r_2 and r_3 are three mutually distinct points taken randomly from population P, not coinciding with the current x_i and $F>0$ is an input parameter. Another strategy called **DE/best/2/** generates the point u according to formula $u = x_{min} + F(r_1 + r_2 - r_3 - r_4)$ where x_{min} is the point of P with minimal function value, r_1, r_2, r_3, r_4 are four mutually distinct points taken randomly from P not coinciding with the current x_i or x_{min} and $F>0$ is an input parameter. **In our experiments, we have used /DE/best/2/ method.** The efficiency of differential evolution is very sensitive to the setting of values F. The values recommended in literature are $F=0.8$ and we have used this value in our experiments.

3.4.5 Cluster Quality:

To assess the improvement in **Cluster Quality** from the use of these optimizer methods, we use a quantitative measure *Silhouette coefficient (SC)*. Three datasets were used for the experimentation viz., synthetic text dataset (ISA dataset, named as BIGCHECK & used in the earlier experiment), and two standard datasets SOYBEAN & WATER TREATMENT PLANT. In addition to the earlier

four methods, two more flavors of GA methods were used: *Roulette Method & Grouping GA (GGA)*. The number of instances in SOYBEAN dataset is 307, and the number of attributes is 35. The number of instances in WATER TREATMENT PLANT dataset is 527, and the number of attributes is 38. The Silhouette Coefficient values of the resulting clusters were computed for each of these three datasets (ISA, SOYBEAN & WATER TREATMENT PLANT) for each of the following seven methods: *K-Means, GA Steady State K-Means, GA Roulette Wheel K-Means, Grouping GA K-Means, PSO K-Means, SA K-Means & DE K-Means.*

The results obtained were further verified using another quantitative measure, the popular *Davies Bouldin* index. The same three datasets (BIGCHECK, SOYBEAN & WATER TREATMENT PLANT) and the seven methods (*K-Means, GA Steady State K-Means, GA Roulette Wheel K-Means, Grouping GA K-Means, PSO K-Means, SA K-Means & DE K-Means)* were used for the experiments. The experimental results are discussed in the next chapter.

Chapter 4

Experiments with K-Means

4.1 **Introduction**

In this chapter, we provide the details of the experimental study. Initial experiments were conducted in the domain of textual data sets. The programs developed were mostly under Matlab 7.1 [120], and also C++ and Data Mining software Weka were used for some results. **Why work with documents?** This is because large collections of data are becoming increasingly common. The public internet currently has more than 1.5 billion web pages, while private intranets also contain an abundance of text data. There exists an information goldmine in the gigantic volume of important scientific data appearing as technical abstracts and paper. It is important to organize such large document collections into structured ontology. The necessity is felt of automatic methods such as clustering to organize unlabelled document collections into clusters of related documents. It is a contemporary challenge to efficiently preprocess and cluster very large document collections [38].

Firstly, we converted the document dataset to its numeric equivalent dataset using the Vector Space Model; thereafter the numeric dataset was reduced to a binary dataset for computational simplicity. The investigation unfolded the following: (1) It clearly establishes the marked benefits of using optimizing schemes in the clustering process (2) It provides an insight and a comparative assessment into the performance benefits of using four well-known optimizing techniques on a textual/binary data set. For the document source, we used 1060 records created for journal articles from the Information Science and Abstracts (ISA) database and computer science technical reports collected from various sites on the Internet. This is a comprehensive subset of the entire database. Each document record includes a complete abstract, title, author, and subject keywords. We used a text dataset of 1060 such records. Four optimizing techniques viz., *Genetic Algorithm (GA) Steady State, Simulated Annealing, Differential Evolution (DE) & Particle Swarm Optimization (PSO)* were used in an inner loop to arrive at better centroids compared with an independent K-Means run. The performances of these methods and an independent K-means run were observed for a fixed number of generations. The results clearly demonstrate that a superior clustering performance (as measured by an appropriate, suitably defined metric) is achieved with such an optimization-based clustering approach, as compared to a normal single- level clustering using a fixed

choice of the parameters. Results indicated GA and DE give the best values, with PSO performing rather poorly.

The thesaurus used for the experiments is listed below

PROGRAM 0	OPERATING 5	MEMORY 9	DATABASE 15
PROGRAMS 0	SYSTEM 6	MEMORIES 9	DATABASES 15
PROGRAMMING 0	SYSTEMS 6	DISTRIBUTED 10	MANAGEMENT 16
LANGUAGE 1	SCHEDULE 7	AUTOMATA 11	INDEX 17
LANGUAGES 1	SCHEDULES 7	AUTOMATON 11	INDEXES 17
GRAMMAR 2	SCHEDULER 7	COMPLEXITY 12	INDEXING 17
GRAMMARS 2	SCHEDULING 7	COMPLEXITIES 12	QUERY 18
SYNTAX 3	CONCURRENT 8	DETERMINISTIC 13	QUERIES 18
SEMANTIC 4	CONCURRENCE 8	DETERMINISM 13	QUERYING 18
SEMANTICS 4	CONCURRENCY 8	COMPLETENESS 14	FILE 19
FILES 19	INTELLIGENCE 22	LEARNING 25	NETWORKING 27
RETRIEVE 20	INTELLIGENT 22	PARALLEL 26	HUMAN 28
RETRIEVAL 20	KNOWLEDGE 23	PARALLELISM 26	HUMANS 28
RETRIEVING 20	EXPERT 24	NETWORK 27	COMPUTER 29
ARTIFICIAL 21	LEARN 25	NETWORKS 27	COMPUTING 29
	LEARNS 25		

Thesaurus Adaptation: We used the ACM *Computing Reviews* Classifications Scheme and the ASIS Thesaurus to choose what we believe to be an adequate thesaurus for the domain of computer and information science. When such standardized sources are not available or are not believed to be adequate, there is a need to incorporate new terms automatically. Standard statistical approaches exist for automated thesaurus generation [77]. Several other approaches based on machine learning and NLP techniques have also been reported in the literature for automatic term discovery [78 79] and refinement [80]. Of course, users should also have the option to introduce new terms to suit their individual needs. Whenever there is a change in the thesaurus, there is also a need to update the centroids of the document clusters and/or reclustering.

4.1.1 Term Frequency Obtained for the Textual Dataset ISA (BIGCHECK)

{program, programs, programming} = 194+129+256 = 579

{language, languages} = 270+163 = 433

{grammar, grammars} = 36+43 = 79

{syntax} = 15

{semantic, semantics} = 58+95= 153

{operating} = 112

{system, systems} = 770+972= 1742

{schedule, schedules, scheduler, scheduling} = 16+8+1+56= 81

{concurrent, concurrence, concurrency} = 75+0+35= 110

{memory, memories} = 411+2 = 413

{distributed} = 212

{automata, automaton} = 50+12 = 62

{complexity, complexities} = 93+3 = 96

{deterministic, determinism} = 33+6= 39

{completeness} = 10

{database, databases} = 271+144 = 415

{management} = 204

{index, indexes, indexing} = 57+38+69 = 164

{query, queries, querying} = 161+53+2 = 216

{file, files} = 125+56= 181

{retrieve, retrieval, retrieving} = 8+192+18= 218

{artificial} = 173

{intelligence, intelligent} = 168+62 = 230

{knowledge} = 234

{expert} = 115

{learn, learns, learning} = 12+2+179 = 193

{parallel, parallelism} = 244+18= 262

{network, networks, networking} = 219+196+12= 427

{human, humans} = 101+1= 102

{computer, computing} = 807+149 = 956

In the same manner, document frequency and term weights were computed for generating the numeric dataset equivalent to BIGCHECK.

4.2 Using GA: In the first experiment, we have compared the results obtained using K-Means independently, and using GA to arrive at better selection of centroids in the inner loop followed by clustering using the K-Means algorithm. We have taken 50 individuals (each individual consists of 15 centroids, one each for the assumed fifteen clusters) as population. Thereafter K-Means was run for 50 iterations, and in each run its **fitness value (inverse of Euclidean distance)** is computed. Thereafter, while implementing GA augmented K-Means, the algorithm started with the same population of individuals as was used by K-Means at the outset. Mating individuals has been picked using the best fitness values (mating pool is taken to be about a fifth of the size of population), new population is created retaining old population plus 10 off-springs resulting from the mating pool using 5 point random crossover, exchanging random subsets of centers between parents for crossover, thus forming new population of sixty which was eventually trimmed to fifty (original size of population) based on its fitness value (high fit values retained). We kept the number of genes in individual evolution the same.

Iteration#	Fitness values of K-means	Fitness values of GA K-means	Iteration#	Fitness values of K-means	Fitness values of GA K-means
1	780	773	29	783	881
2	782	789	30	773	881
3	757	810	31	780	882
4	770	815	32	764	882
5	773	819	33	775	883
6	774	825	34	767	883
7	760	834	35	779	883
8	778	839	36	771	883
9	767	839	37	766	885
10	784	843	38	768	885
11	773	848	39	757	885
12	768	851	40	782	885
13	777	853	41	763	885
14	777	859	42	777	885
15	774	862	43	770	886
16	766	865	44	771	888
17	762	865	45	769	888
18	771	867	46	752	888
19	783	868	47	772	889
20	782	870	48	769	889
21	773	873	49	774	892
22	782	876	50	781	893
23	767	878			
24	794	878			
25	777	879			
26	779	879			
27	780	879			
28	776	880			

Table 4.1: Fitness Values of K-Means & GA K-Means for 50 iterations (BIGCHECK i.e., ISA)

In brief, we have used steady state replacement based on 20 % mating pool and higher fit values for selection. After implementation of GA using steady state replacement, we have obtained fitness results which is given in Table 4.1

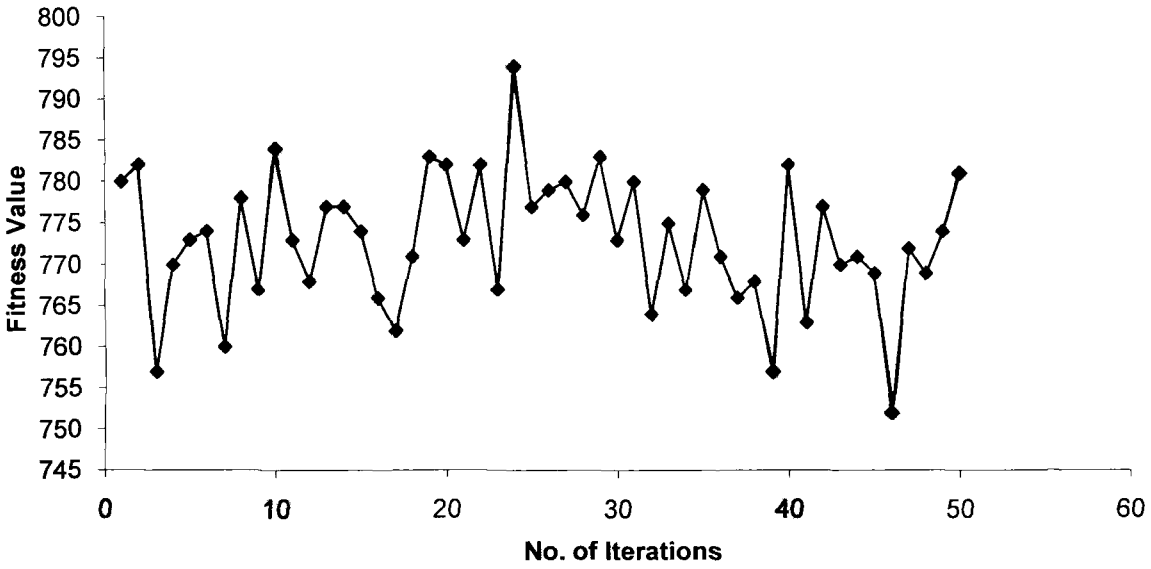


Fig 4.1 Fitness values vs. Iteration for K-Means K=15; Dataset - BIGCHECK

Fig 4.2 Fitness values vs. Iterations for GA K-means K=15; Dataset - BIGCHECK

Fig. 4.1 & Fig. 4.2 show plots of fitness values against the number of successive iteration for the two algorithms, K-Means & GA K-Means.

The results thus far obtained clearly demonstrate (Fig. 4.1, 4.2 & 4.3) that a superior clustering performance (as measured by an appropriate, suitably defined metric such as Euclidean distance)

is achieved with such a multi-level, optimization-based clustering approach, as compared to a normal single-level clustering using a fixed choice of the parameters.

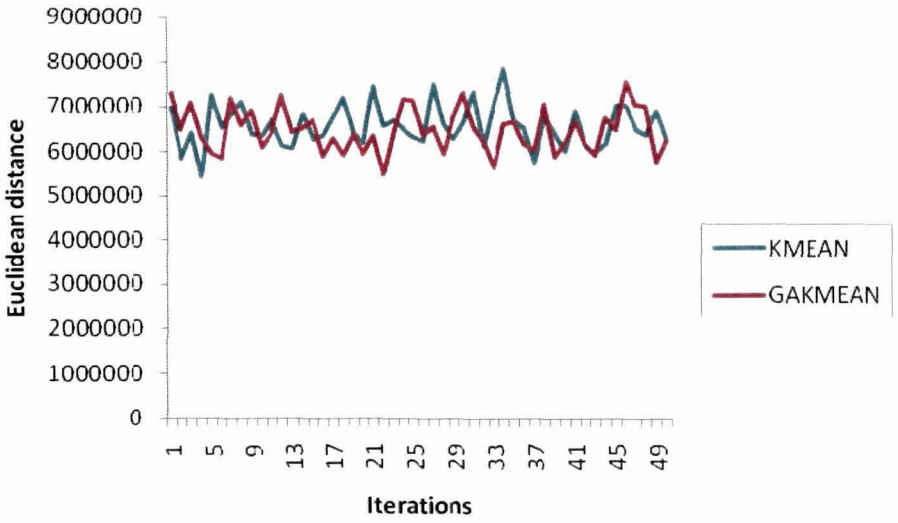


Fig 4.3 GA K-Means Vs. K-Means (Measure – Euclidean Distance) Dataset – BIGCHECK

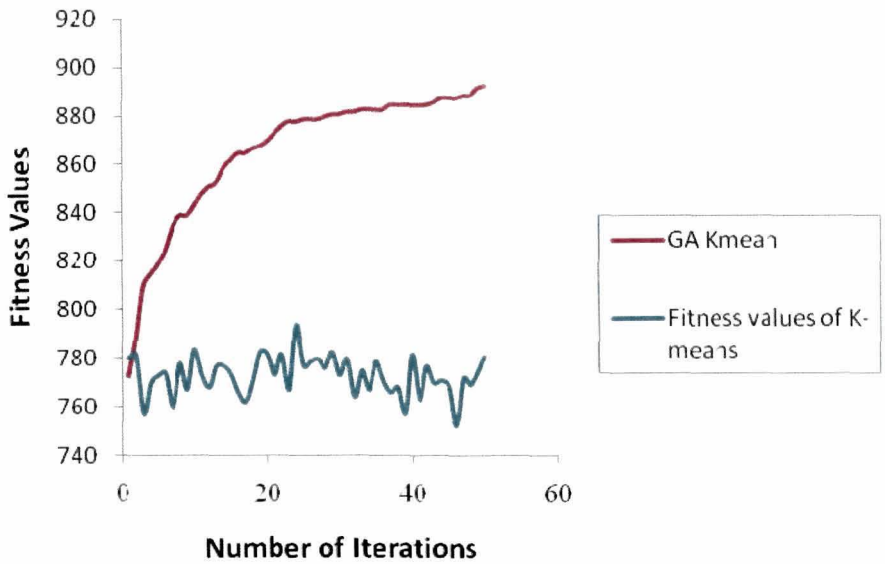


Fig 4.4 GA K-Means vs. K-Means (Measure – Fitness Values) Dataset - BIGCHECK

Fig 4.3 reveals Euclidean distance is less in case of GA K-means. Fig 4.4 reveal fitness value is more in case of GA K-Means i.e., GA K-Means has obtained better centroids for DATASET BIGCHECK.

In Table 4.1, the fitness values of K-Means and GA K-Means for 50 iterations for the textual dataset we have experimented with is shown. The improvement achieved using GA K-Means over K-Means is depicted in Fig. 4.4.

4.3 Using Particle Swarm

4.3.1 Experimental Results

Since our objective is to find better centroids for K-means, we have taken 50 individuals (each individual consists of k centroids, one each for the assumed k clusters) as population. Thereafter K-means was run for 50 iterations, and in each run its fitness value (inverse of Euclidean distance) is computed.

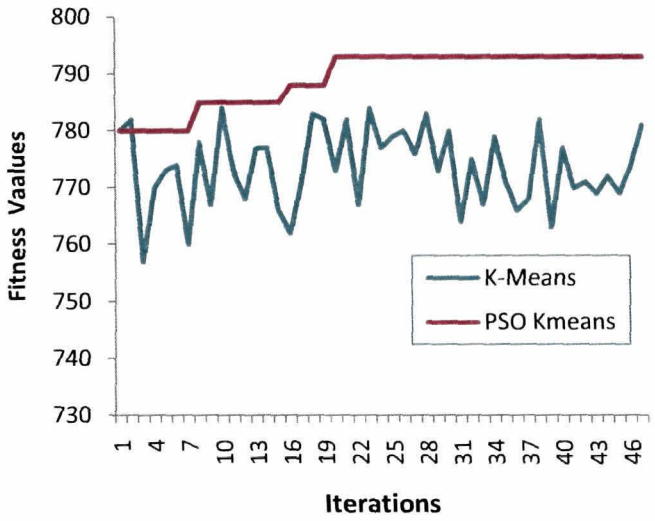


Fig 4.5 Fitness values obtained using K-means & PSO K-means (K=15) Dataset - BIGCHECK

Thereafter, while implementing PSO, the algorithm started with the same population of individuals as was used by K-means at the outset. The fitness values obtained for K-means and PSO based K-means are provided in Table 4.2. Fig. 4.5 & Fig. 4.6 demonstrate the benefit of using the PSO method.

	K=5		K=10		K=15		K=20	
S#	K-means	PSO K-means	K-means	PSO	K-means	PSO	K-means	PSO
1	1652	1656	1070	1041	780	780	590	587
2	1626	1656	1083	1051	782	780	595	615
3	1619	1656	1062	1088	757	780	587	615
4	1667	1656	1040	1088	770	780	603	615
5	1638	1656	1063	1088	773	780	604	615
6	1646	1656	1052	1088	774	780	596	615
7	1672	1656	1044	1088	760	780	586	615
8	1638	1656	1051	1088	778	785	591	615
9	1647	1656	1067	1088	767	785	586	615
10	1663	1656	1053	1088	784	785	609	615
11	1667	1657	1082	1088	773	785	600	615
12	1635	1657	1052	1088	768	785	597	615
13	1639	1663	1055	1088	777	785	597	615
14	1658	1663	1060	1088	777	785	603	615
15	1654	1663	1041	1088	774	785	597	615
16	1664	1663	1037	1088	766	785	602	615
17	1621	1664	1054	1088	762	788	605	615
18	1645	1664	1040	1088		788	585	615
19	1662	1664	1044	1088	783	788	606	615
20	1666	1664	1056	1088	782	788	597	615
21	1653	1664	1064	1088	773	793	592	615
22	1630	1664	1049	1088	782	793	600	615
23	1622	1666	1036	1088	767	793	595	615
24	1650	1666	1034	1088	794	793	590	615
25	1644	1666	1058	1088	777	793	601	615
26	1671	1666	1047	1088	779	793	593	615
27	1639	1666	1029	1088	780	793	602	615
28	1644	1666	1033	1088	776	793	604	615
29	1647	1666	1051	1088	783	793	610	615
30	1630	1671	1059	1088	773	793	596	615
31	1666	1671	1057	1088	780	793	590	615
32	1626	1671	1050	1088	764	793	597	615
33	1650	1671	1057	1088	775	793	601	615
34	1655	1671	1063	1088	767	793	602	615
35	1645	1671	1066	1088	779	7933	592	615
36	1652	1671	1064	1088	771	793	598	615
37	1638	1671	1048	1088	766	793	596	615
38	1685	1671	1060	1088	768	793	609	615
39	1633	1671	1056	1088	757	793	588	615
40	1653	1671	1059	1088	782	793	598	615
41	1623	1671	1054	1088	763	793	613	615
42	1642	1671	1048	1088	777	793	604	615
43	1646	1671	1047	1088	770	793	590	615
44	1636	1671	1051	1088	771	793	589	615
45	1663	1671	1051	1088	769	793	594	615
46	1656	1671	1063	1088	752	793	593	615
47	1651	1671	1061	1088	772	793	608	615
48	1636	1671	1053	1088	769	793	601	615
49	1651	1671	1060	1088	774	793	604	615
50	1632	1671	1071	1088	781	793	591	615

Table 4.2: PSO results Fitness values for different K

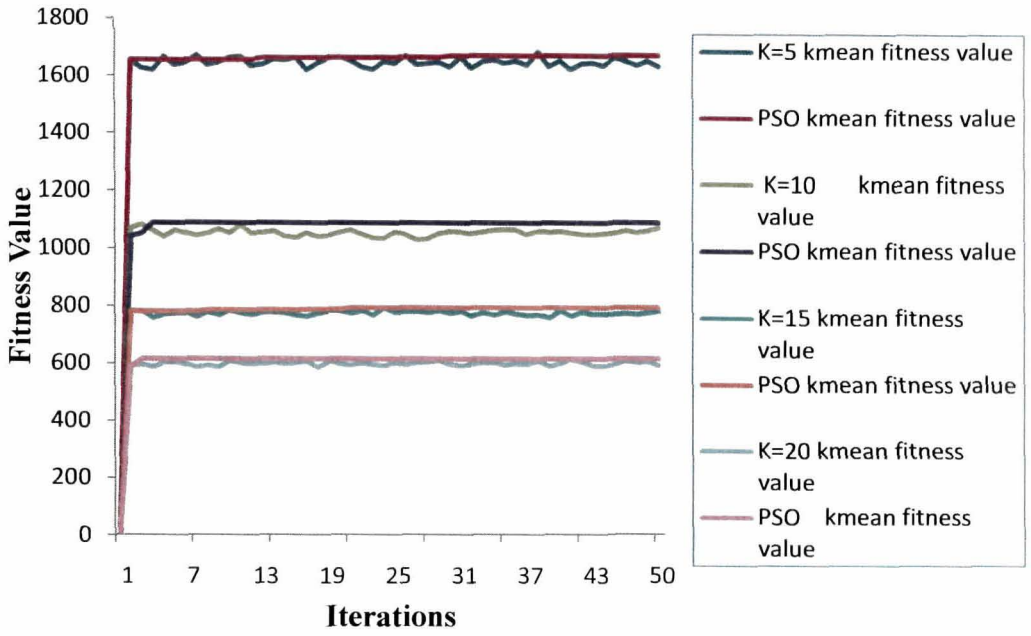


Fig 4.6 Fitness values obtained using K-means & PSO K-means for k=5, 10, 15 & 20. DATASET - BIGCHECK

Fig 4.6 shows clearly that PSO K-means can obtain better centroids.

4.4 Using Differential Evolution Results of our experiments using /DE/best/2/ are given below in Table 4.3 and shown in Fig 4.7.

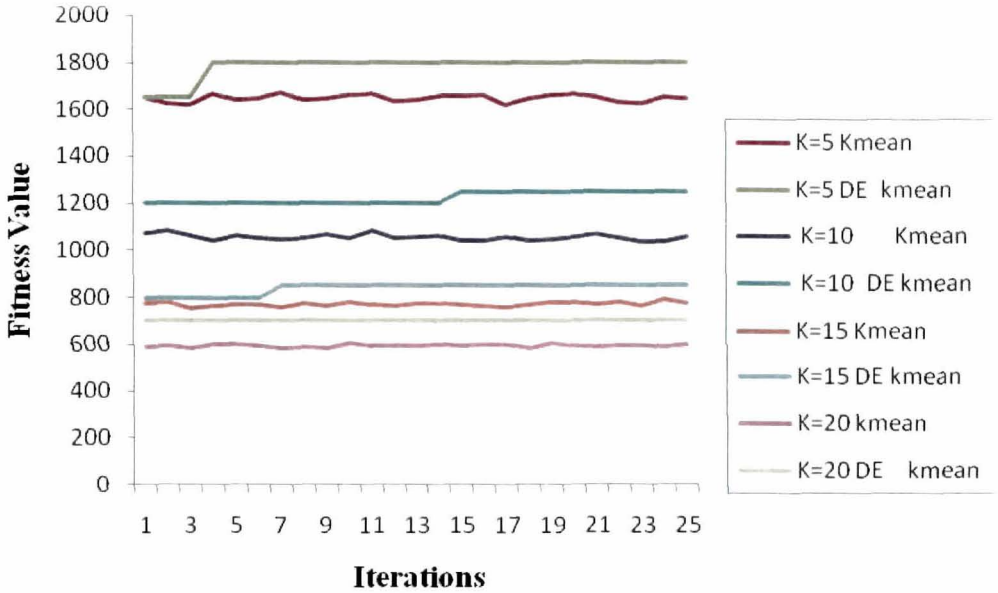


Fig 4.7 Fitness values obtained using K-means & DE K-means for k=5, 10, 15 & 20. DATASET - BIGCHECK

	K=5		K=10		K=15		K=20	
S#	K-means	DE K-means	K-means	DE K-mean	K-means	DE kmean	K-means	DE
1	1652	1650	1070	1200	780	800	590	700
2	1626	1650	1083	1200	782	800	595	700
3	1619	1650	1062	1200	757	800	587	700
4	1667	1800	1040	1200	770	800	603	700
5	1638	1800	1063	1200	773	800	604	700
6	1646	1800	1052	1200	774	800	596	700
7	1672	1800	1044	1200	760	850	586	700
8	1638	1800	1051	1200	778	850	591	700
9	1647	1800	1067	1200	767	850	586	700
10	1663	1800	1053	1200	784	850	609	700
11	1667	1800	1082	1200	773	850	600	700
12	1635	1800	1052	1200	768	850	597	700
13	1639	1800	1055	1200	777	850	597	700
14	1658	1800	1060	1200	777	850	603	700
15	1654	1800	1041	1250	774	850	597	700
16	1664	1800	1037	1250	766	850	602	700
17	1621	1800	1054	1250	762	850	605	700
18	1645	1800	1040	1250		850	585	700
19	1662	1800	1044	1250	783	850	606	700
20	1666	1800	1056	1250	782	850	597	700
21	1653	1800	1064	1250	773	850	592	700
22	1630	1800	1049	1250	782	850	600	700
23	1622	1800	1036	1250	767	850	595	700
24	1650	1800	1034	1250	794	850	590	700
25	1644	1800	1058	1250	777	850	601	700

Table 4.3: DE results Fitness values for different K for 25 iterations DATASET- BIGCHECK.

4.5 Using Simulated Annealing

The results obtained using the method is incorporated in Table 4.4

4. 6 EXPERIMENTAL RESULTS: A COMPARATIVE STUDY

Table 4.4 below depicts the movement of fitness values of the various methods for 25 iterations. Fig. 4.8 shows the plot of these values and exhibits the relative performance.

Iter#	K-means	GA K-means	PSO K-means	DE K-means	SA K-means
1	780	782	780	800	787
2	782	795	780	800	787
3	757	806	780	800	791
4	770	809	780	800	791
5	773	820	780	800	791
6	774	830	780	800	791
7	760	834	780	850	791
8	778	836	785	850	791
9	767	842	785	850	791
10	784	845	785	850	791
11	773	848	785	850	791
12	768	848	785	850	791
13	777	850	785	850	791
14	777	851	785	850	791
15	774	853	785	850	791
16	766	856	785	850	791
17	762	859	788	850	791
18	771	863	788	850	791
19	783	863	788	850	791
20	782	866	788	850	791
21	773	866	793	850	791
22	782	868	793	850	791
23	767	871	793	850	791
24	794	873	793	850	791
25	777	873	793	850	791

Table 4.4: Comparative results of K-Means Vs. 4 optimizers DATASET-BIGCHECK.

In fig. 4.8 random centroid selection method K-Means is compared with four optimizer based techniques (GA/SA/PSO/DE) for evolving better centroids iteratively. The four optimizer based methods were found to have brought about a significant improvement as compared to K-Means (as can be seen from the graph of fig. 4.8).

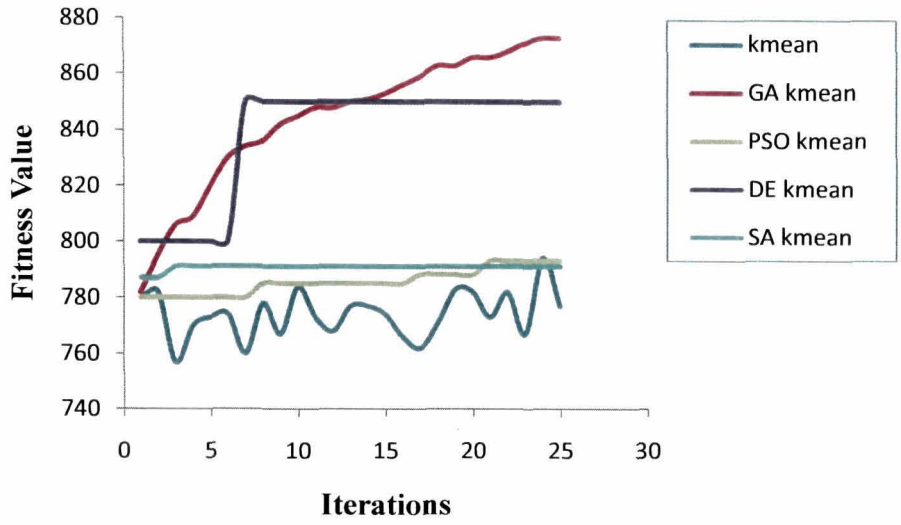


Fig 4.8 Comparison of performance of the various methods for k=15 dataset - BIGCHECK

It was known that free parameters could have major impacts on the standard K-means performance. With a more sophisticated procedure, using optimization techniques like GA or PSO, the K-means results ought likely to be better. **Experiments with our textual datasets show that GA and DE arrived at better values whereas the least improvement was witnessed using the swarm technique PSO.**

4.6.1 Initial Observations: The Experimental results justify the use of optimizing methods in improving the centroids of the iterative K-Means algorithm. All the optimizers improve the solution in comparison to random selection methods like K-Means. In the first experiments on text dataset, **GA Steady State emerges as the best optimizer** of the four methods.

4.7 Cluster Quality:

To assess the improvement in **Cluster Quality** from the use of these optimizer methods, we use a quantitative measure *Silhouette coefficient (SC)*. Three datasets were used for the experimentation viz., synthetic text dataset (ISA dataset, named as BIGCHECK & used in the earlier experiment), and two standard datasets SOYBEAN & WATER TREATMENT PLANT. In addition to the earlier four methods, two more flavors of GA methods were used:

Roulette Method & Grouping GA (GGA). The number of instances in SOYBEAN dataset is 307, and the number of attributes is 35. The number of instances in WATER TREATMENT PLANT dataset is 527, and the number of attributes is 38. The Silhouette Coefficient values of the resulting clusters were computed for each of these three datasets (ISA, SOYBEAN & WATER TREATMENT PLANT) for each of the following methods: *K-Means, GA Steady State K-Means, GA Roulette Wheel K-Means, Grouping GA K-Means, PSO K-Means, SA K-Means & DE K-Means.*

Below is shown some data containing the values of experiments related to determining silhouette co-efficient (SC). The following abbreviations were used

KM	**K-Means**
GASS	**GA Steady State**
GAR	**GA Roulette**
GGA	**Grouping GA**
PSOKM	**PSO K-Means**
DEKM	**DE K-Means**
SAKM	**SA K-Means**

DATASET	**KM**	**GASS**	**GAR**	**GGA**	**PSOKM**	**DEKM**	**SAKM**
Bigcheck(ISA)	0.15	0.22	0.17	0.22	0.21	0.15	0.18
SOYBEAN	0.34	0.56	0.65	0.67	0.5	0.55	0.61
WATER TREATMENT PLANT	0.32	0.7	0.67	0.78	0.67	0.7	0.68

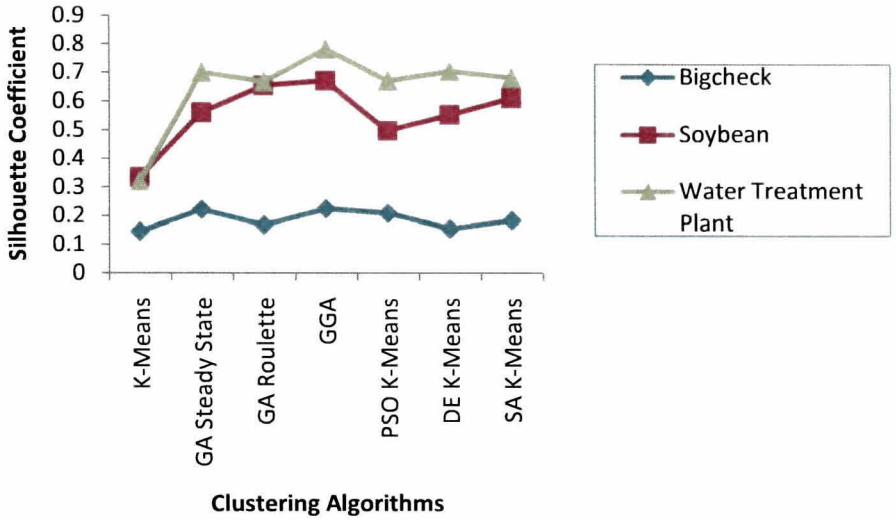


Fig 4.9 Silhouette Coefficient vs. Clustering algorithms Dataset – Bigcheck, Soybean & Water Treatment Plant

The experiment with Silhouette Coefficient for K =15 and using EUCLIDEAN Similarity measure is given above in Fig 4.9

The experiment with Silhouette Coefficient for K =15 and using PEARSON Similarity measure is given below in Table 4.5 and depicted in Fig 4.10.

	K-Means	GA Steady State	GA Roulette	GGA K-Means	PSO K-Means	DE K-Means	SA K-Means	K=15 PEARSON
BIGCHECK	0.145451	0.229298021	0.204755042	0.286028558	0.181910273	0.212482658	0.209103676	
WATER PLANT	0.432237	0.674580835	0.66494231	0.791291461	0.666407021	0.692104934	0.681505693	
SOYBEAN	0.3227173	0.628285016	0.60317329	0.721212052	0.507436482	0.58429609	0.6300013	

Table 4.5: SC Values (Pearson)

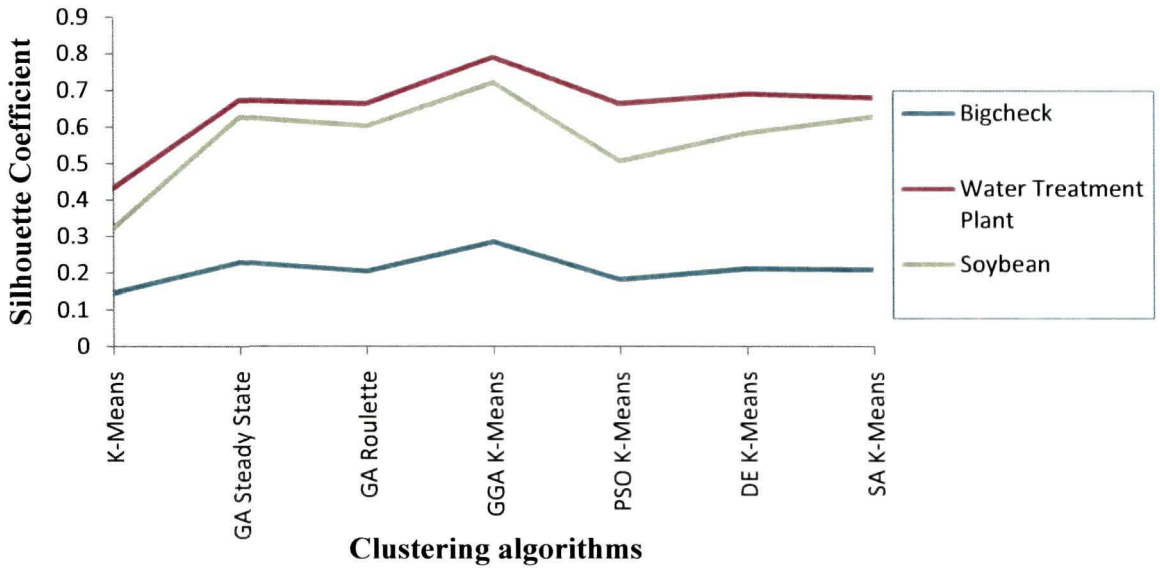


Fig 4.10 SC (Pearson) Vs. Clustering algorithms

For the above experiments, on observation it becomes clear that GGA outputs the best optimized values. The other optimizers improve appreciably upon the results obtained using K-Means independently.

Thereafter, we also experimented with another quantitative cluster validity index, viz., **Davies Bouldin Index (DBI)**. Table 4.6 and Fig. 4.11 is a result corresponding to the SOYBEAN dataset using the Davies Bouldin Index. Low values of DBI indicate a good performance, and Grouping GA also appears as the best optimizer from this experiment.

K=15	DAVIES BOULDIN INDEX			SOYBEAN		
	K-Means	GA Steady State	GA Roulette	Grouping GA	PSO K-Means	DE K-Means	SA K-Means
Euclidean	0.98434	0.834427	0.68593	0.623	0.80431	0.79348	0.75452
Pearson	.63335	.4831	.4569	0.4208	0.47028	0.4528	0.46232

Table 4.6: DB Index for SOYBEAN dataset

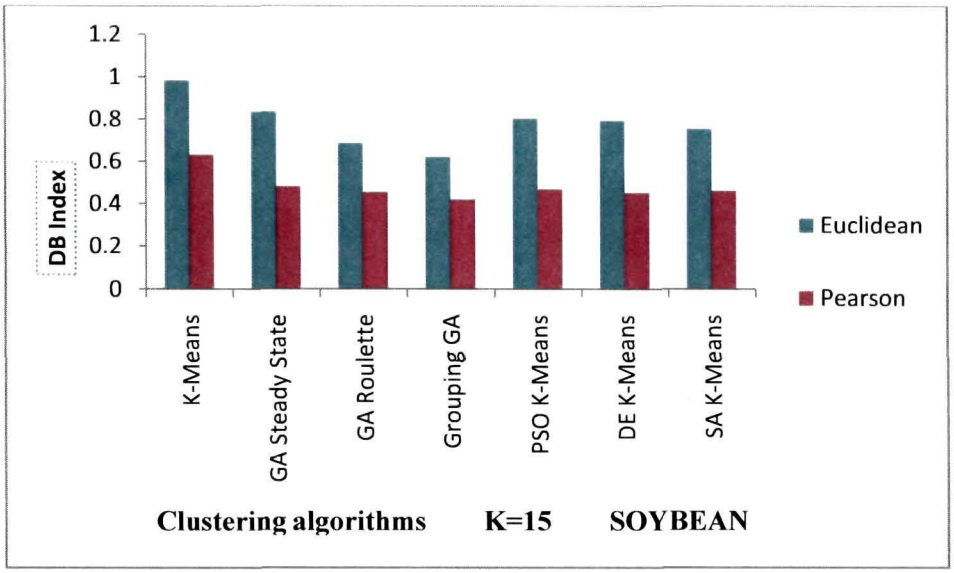


Fig 4.11. DB Index vs. Clustering algorithm. Dataset - Soybean

The comprehensive results corresponding to DB Index for three datasets is shown in Table 4.7 and Fig 4.12.

DATASET	KM	GASS	GAR	GGA	PSOKM	DEKM	SAKM
BIGCHECK (ISA)	0.99564	0.89574	0.74723	0.743	0.84561	0.8247	0.88581
SOYBEAN	0.98434	0.834427	0.68593	0.623	0.80431	0.79348	0.75452
WATER TREATMENT PLANT	0.97214	0.64224	0.58373	0.5208	0.70211	0.68128	0.66232

Table 4.7 DBI for 3 datasets

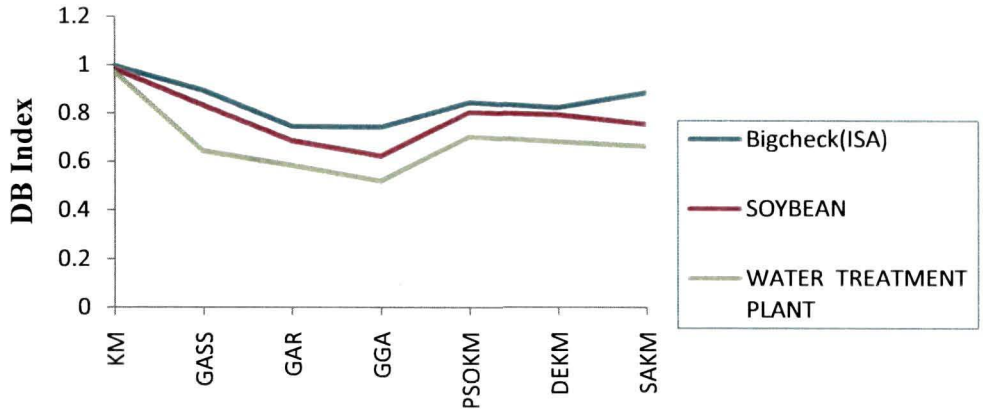


Fig 4.12 DB Index vs. Clustering Algorithms (Measure – Euclidean distance)

The comprehensive result, corresponding to the **WATER TREATMENT PLANT dataset, which has the best cluster structure of the three datasets** that we have considered for our experiments, is shown below in Table 4.8 and Fig. 4.13.

K=15	DAVIES BOULDIN INDEX			WATER TREATMENT PLANT			
	K-Means	GA Steady State	GA Roulette	Grouping GA	PSO K-Means	DE K-Means	SA K-Means
Euclidean	0.97214	0.64224	0.58373	0.5208	0.70211	0.68128	0.66232
Pearson	0.77223	0.60273	0.51218	0.4128	0.55218	0.49332	0.48411

Table 4.8 DB Index values for WATER TREATMENT

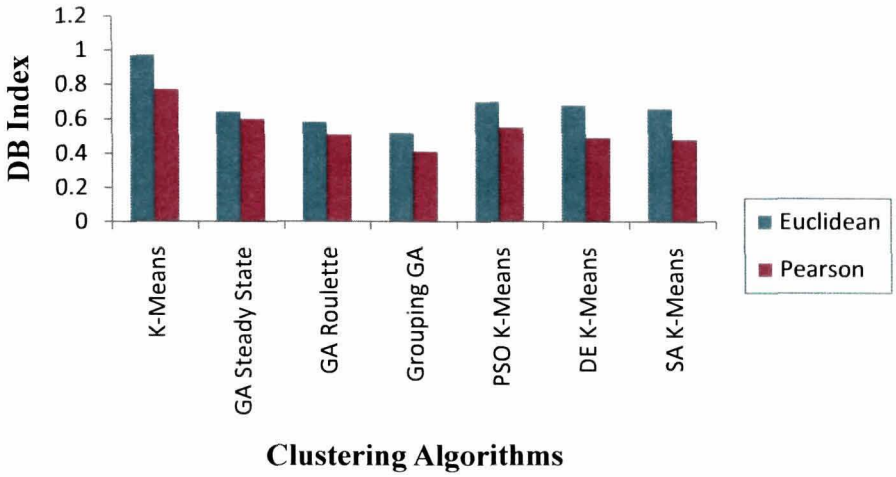


Fig 4.13 DB Index (Euclidean vs. Pearson) for dataset - Water Treatment Plant

GGA emerges as the most effective technique for arriving at better centroids of K-Means for high dimensional datasets. Using GGA improves quality of cluster obtained by a substantial margin, for instance, from SC value of 0.32 to 0.78 for the WATERPLANT TREATMENT dataset. In general, **GA and DE followed by SA gives much improved values. PSO lagged behind the other methods** used in improving the centroids. **GGA (with Pearson distance) emerges as the most effective optimizer that can be effectively used for augmentation of iterative K-Means.**

The significance of all the above experiments can't be overlooked. Overall, all the optimizers have successfully brought about an appreciable improvement in reaching better centroids after a few iterations. As the sizes of data sets continues to grow rapidly these days with easy availability of relevant data for analysis via better data acquisition tools and the Internet, as well as the urgency for mining them, the use of these recent cutting edge optimizers would befit the optimization of the objective function of complex modern applications. **Also of note is that most of these optimizers quickly reach an improved state, approximately close to the best solutions, after just the first ten iterations, it would make sense to run such optimizers for about 10-12 iterations for discovering the good centroid points for large scale, data clustering problems.**

Chapter 5

K-Means: Results and Discussion

The objective of our investigation is to estimate the benefit of using optimization on free parameters of clustering. As a case study, we decided to improve upon the centroids of iterative K-Means. Firstly, we experimented with a textual dataset of 1060 records which was converted to a numeric dataset using the Vector Space model. Estimating term frequencies of about 30 terms, each term comprising an alternative of 2 or 3 near similar terms (e.g., computer and computers) from the perspective of categorization was difficult from programming point of view. Such large array sizes were not supported in C++. A divide and conquer approach was used. Thereafter, one needed to multiply term frequencies by inverse document frequencies. That is, count n (k), the number of documents that contain term k. Then multiply the frequency of the kth term in each document by log (N/n (k)). The resulting weight vectors are ready to generate the corresponding equivalent numeric dataset. After the numeric dataset was generated, we have converted it to a binary dataset for the sake of simplicity.

In the initial experiment, we made an estimate of the benefits resulting from using GA in clustering. Fig 5.1 & Fig 5.2 depict an instance of clustering using K-Means independently and using GA augmented K-Means clustering respectively. From the figures, it does appear that the points are more uniformly spread among the clusters in Fig 5.2.

The experimentation with GA augmented K-Means showed improvement in the fitness values obtained in comparison to K-Means. The results are depicted in Table 4.1, and Fig 4.1, 4.2, 4.3 & 4.4 as shown in the previous chapter. Thereafter, we also experimented with three more cutting edge optimizers, viz., Particle Swarm optimizer, Simulated Annealing & Differential Evolution on the textual dataset BIGCHECK, and these results are shown in Fig 4.5, 4.6, 4.7 & 4.8, and Tables 4.2, 4.3 & 4.4.

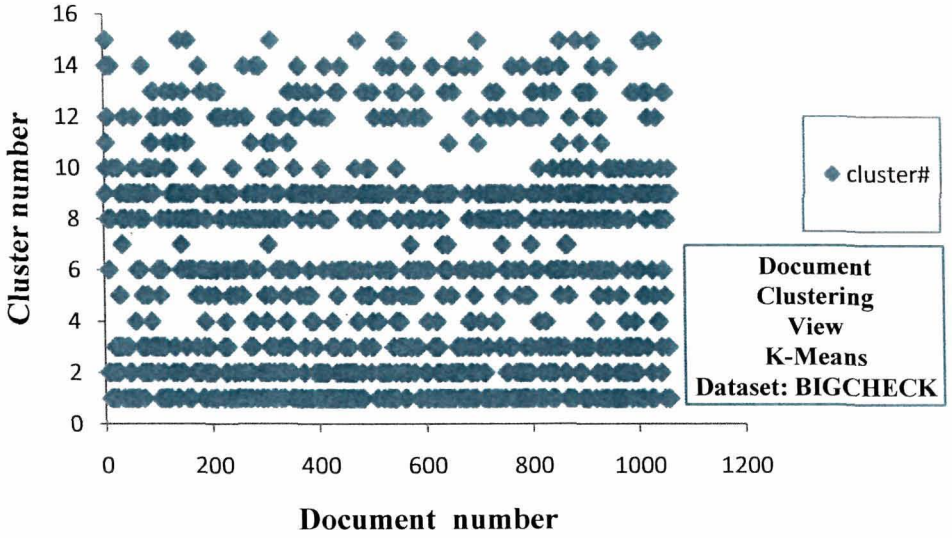


Fig 5.1 Instance of clustering using K-Means

Y-Axis: Cluster # X-Axis: Document #

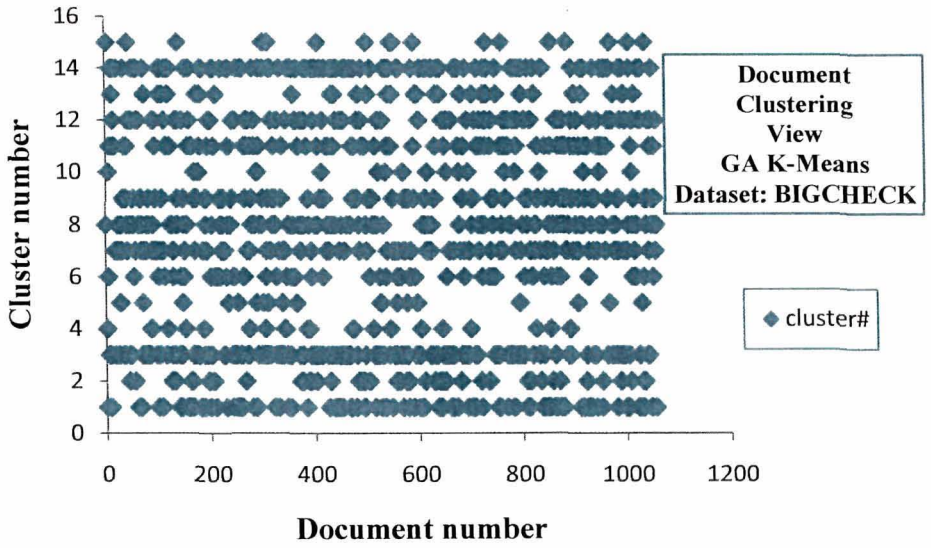


Fig 5.2 Instance of Clustering obtained thru better GA seeds

Y-Axis: Cluster # X-Axis: Document #

We also conducted an experiment using GA augmented K-Means varying the number of clusters to measure inter-cluster distance (cluster quality). A few problems was encountered during GA implementation

1. The code was becoming I/O intensive as there are no concepts of pointers in MATLAB.

2. Inclusion of highest fit individuals into the next generation was not yielding expected results initially. Ideally, fitness should increase, or remain the same if crossover is not bringing about improvements in the solution. The problem was resolved when we added the best fit solutions into the population, and then truncated the population to keep the size of the population constant at fifty (size of population initially decided upon). These fifty individuals were the highest fit among the aggregated group.

For GA augmented k-means, results varying the number of clusters and finding inter-cluster distance for measuring clustering quality, the result of the experiment is depicted in Table 5.1 **The results of Table 5.1 confirm the fact that better centroids (estimated using higher fitness value) in case of lesser number of clusters such as k=3 actually results in better clustering quality, measured by inter-cluster distance in the subsequent step** which is least when k=3. (Distance between the centroids measures the inter cluster distance. The centroids were determined by running the inner GA loop for the mentioned number of iterations). **The better centroids obtained through GA in the K-Means can actually be used for determination of optimal clusters as well.** However, this requires further investigation; which we conduct in terms of compactness of clusters.

Estimation of inter-cluster variation of fitness value and inter-cluster distance with variation in K in GA K-Means is depicted in Fig 5.3 & 5.4.

Iter#	K=3 Fitness Value	K=5 Fitness Value	K=10 Fitness Value	K=15 Fitness Value	K=20 Fitness Value
1	1961	1633	1066	782	610
2	1979	1652	1072	795	620
3	2000	1669	1082	806	628
4	2014	1687	1089	809	632
5	2016	1703	1103	820	634
6	2017	1706	1111	830	637
7	2020	1716	1122	834	639
8	2023	1732	1124	836	640
9	2024	1741	1141	842	641
10	2024	1746	1145	845	645
11	2025	1754	1146	848	650
12	2026	1755	1149	848	651
13	2027	1760	1149	850	653
14	2027	1762	1154	851	654
15	2031	1763	1158	853	657
16	2032	1768	1160	856	665
17	2035	1773	1169	859	665
18	2037	1777	1172	863	665
19	2037	1777	1175	863	666
20	2037	1777	1175	866	667
21	2038	1778	1176	866	668
22	2040	1778	1176	868	669
23	2041	1778	1180	871	669
24	2041	1778	1181	873	670
25	2043	1779	1182	873	671
26	2044	1787	1182	874	672
27	2045	1789	1182	874	672
28	2047	1795	1184	875	672
29	2047	1798	1184	875	673
30	2048	1800	1185	875	674
31	2048	1800	1185	875	675
32	2049	1801	1187	877	675
33	2050	1803	1188	877	676
34	2050	1804	1188	877	678
35	2050	1804	1188	877	678
	Inter-cluster distance For k=3	**Inter-cluster distance For k=5**	**Inter-cluster distance For k=10**	**Inter-cluster distance For k=15**	**Inter-cluster distance For k=20**
	8.3208	**32.6797**	**163.2386**	**377.9324**	**791.9232**

Table 5.1: Inter-Cluster distance using GA

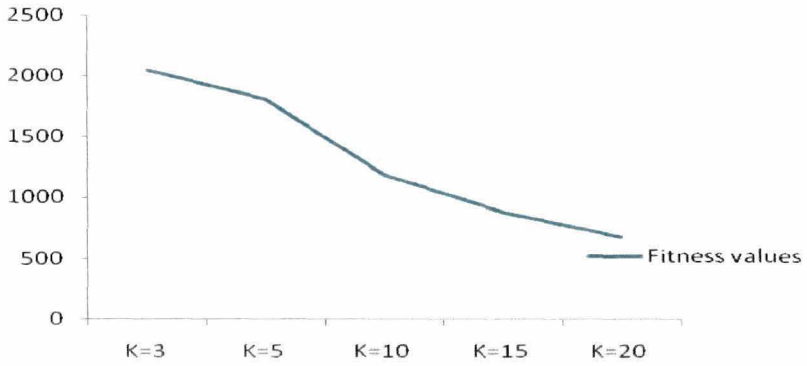


Fig 5.3 GA K-Means: Inter-Cluster Variation with variation in k in GA K-Means

Y-axis: Fitness Value **X-axis: no of clusters**

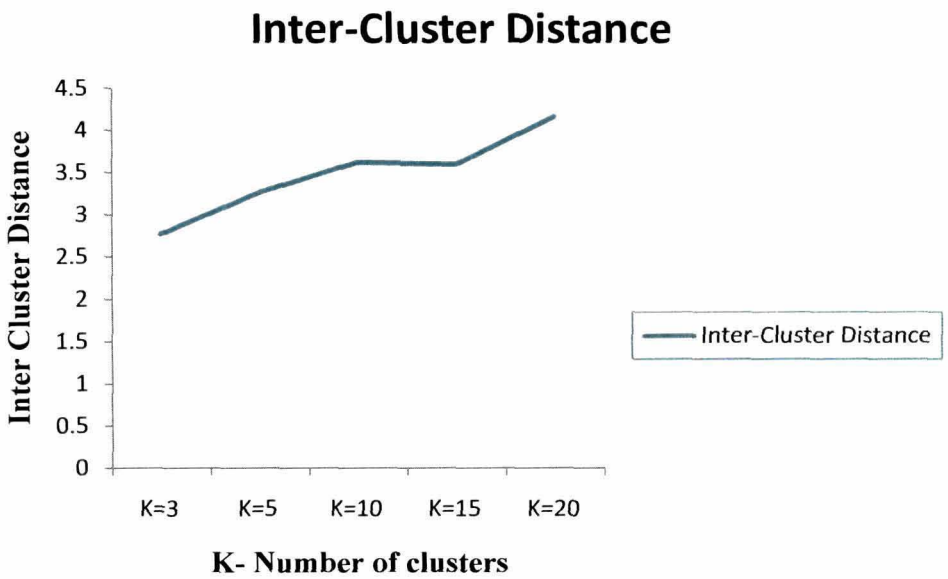


Fig 5.4 GA K-Means: Inter-Cluster Variation with variation in k in GA K-Means

The experimentation with the four optimizers on the textual dataset yielded positive results. **Thereafter, we decided to investigate the improvement in cluster quality through a quantitative measure, Silhouette Coefficient. The results were not all that encouraging. The improvement was marginal. We next decided to experiment again with Silhouette Coefficient, and added two more standard datasets for our experiments.**

To assess the improvement in cluster quality of using these techniques, three datasets were used for the experimentation viz., synthetic dataset BIGCHECK (ACM citation dataset used in earlier experiment), and two standard datasets SOYBEAN & WATER TREATMENT PLANT. **In addition to the earlier four methods, two more GA methods were used: Roulette Method & Grouping GA (GGA).** The Silhouette Coefficient values of the resulting clusters were computed for each of these three datasets (BIGCHECK, SOYBEAN & WATER TREATMENT PLANT) for each of the following methods: *K-Means, GA Steady State K-Means, GA Roulette Wheel K-Means, Grouping GA K-Means, PSO K-Means, SA K-Means & DE K-Means.* **Results indicate that GGA comes up with the best values, while in general GA & DE followed by SA gives much improved values.** Using GGA improves cluster quality by a substantial margin, for instance in the WATERPLANT TREATMENT dataset, the SC value improves from 0.33 to 0.78.

The working of GGA is as follows. An initial population is generated. The population size, n^{pop}, is a parameter of the GGA. According to this approach, each newly generated pair of offspring is inserted immediately into the current population where it replaces the two worst individuals. This incremental approach ensures the survival of the best solution over the whole search and prevents the occurrence of duplicate individuals. Thus, it often reveals higher performance than generational replacement (Davis, 1991). In order to detect duplicates, a simple comparison of objective values proved satisfactory for GGA, so it was preferred over any other time consuming procedure seeking genotypical or phenotypical differences between individuals. The crossover and mutation are employed in a sequential fashion. The crossover operator is applied to each selected pair of parent chromosomes with probability $pcross$, whereas the mutation operator is applied to each offspring with probability p^{mut}. Both $pcross$ and p^{mut} are parameters of the GGA. If the crossover operator is not applied according to its execution probability $pcross$, the children are simply clones of their parents. Similarly, if the mutation operator is not applied, the offspring leave the mutation operator unchanged. The search terminates after a given total number, n^{max}, of individuals has been generated without improvement but no later than

Algorithm 5.1 Grouping Genetic Algorithm (GGA)

Input: a random initial population of possible solution

Output: optimized solution for the problem

initialize population P and evaluate all individuals in P ;

while not *(termination criterion is met)*

select a pair of individuals x, y *from P as parents;*

generate two children x', y' *applying the crossover operator to* x *and* y *with probability* p_{cross};

generate two modified children x'', y'' *applying the mutation operator to* x' *and* y'*, respectively, with probability* p_{mut};

evaluate x'' *and* y''*, insert them into P and in turn remove the two worst individuals from P ;*

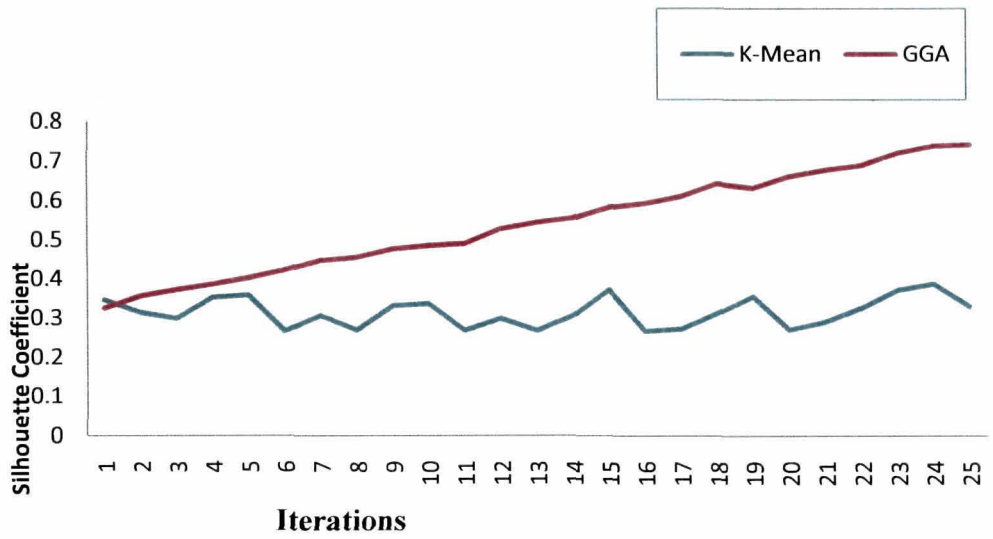
return *best individual from P as solution.*

Algorithm 5.1 describes the general search scheme of GGA [92].

after a given maximum number, n_{max}, of generated individuals has been reached.

Falkenauer [92] describes a high-level paradigm (metaheuristic) that can be adapted to deal with grouping problems broadly defined, showing that it is useful for several applications – e.g., bin packing, economies of scale, conceptual clustering, and equal piles. Data partitioning problems are not the primary focus of Falkenauer's book [92]. Nevertheless, it is worth mentioning that, in order to pave the way for the proposed paradigm, the author investigates, among other issues, key aspects of some genetic algorithms designed for data partitioning problems until 1998. Most importantly, the concepts underlying such a paradigm allow delving into important features of evolutionary algorithms for data partitioning problems.

GGA emerges as an effective technique for arriving at better centroids of K-Means for high dimensional datasets. Fig 5.5 plots the variation of Silhouette Coefficient for 25 iterations for the two algorithms, K-Means & GGA K-Means for the SOYBEAN dataset using Euclidean distance.



Variation of Silhouette Coefficient (DATASET: SOYBEAN; K=15; Measure – EUCLIDEAN)

Fig 5.5: K-Means vs. GGA K-Means using Silhouette Coefficient for 25 iterations (k=15)

All the experiments reported till now have been done using Euclidean distance for computing the similarity of points. It is generally accepted that Euclidean doesn't perform well in high dimension. **For high dimensional data, Pearson Coefficient is an effective similarity measure.** The same experiments for computing silhouette coefficients were also performed using Pearson distance as the similarity measure (in stead of Euclidean) for K-Means & the six other hybrid optimized schemes. The results attest to the above claims made about GGA and the other methods, and the results improve from the Euclidean measure. Fig 5.6 plots the variation of

Silhouette Coefficient for 25 iterations for K-Means & GGA K-Means for the SOYBEAN dataset using Pearson distance.

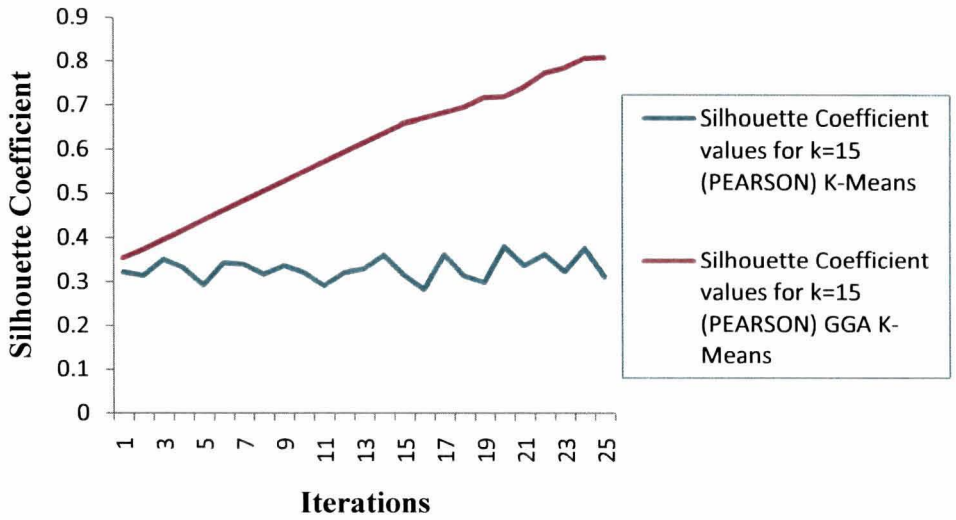


Fig 5.6 Variation of Silhouette Coefficient (DATASET: SOYBEAN PEARSON)

Y-Axis: Silhouette Coefficient X-Axis: no of iterations

Consequent to the experiments with *Silhouette Coefficient*, another well-known validity index, the *Davies Bouldin index (DBI)*, have been used to assess the results. The results

obtained using index also confirms that GGA is the best optimizer. For example, for SOYBEAN dataset (k=15), the results obtained using DBI is shown in Fig 4.11.

Shown in the following pages the graph in Fig 5.7 & 5.8 compares the variation of K-Means & GGA K-Means using Davies Bouldin Index, varying the number of specified clusters k in the first experiment, and for successive number of iterations for k=15 in the next.

The following are the observations vis-à-vis DB index

- The results obtained using this index also conforms generally to the results obtained earlier using silhouette coefficient.

- The experimental values indicate WATER TREATMENT PLANT has a **better cluster structure** as compared to the other two datasets used, since it gives better values as compared to SOYBEAN.
- Earlier experiment with silhouette coefficient showed SOYBEAN has a better structure than Textual dataset BIGCHECK.

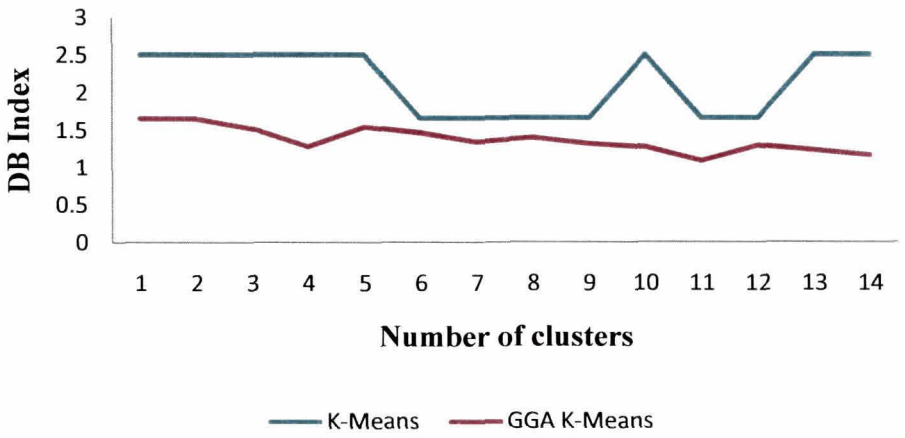


Fig 5.7 K-Means vs. GGA K-Means using DBI for varying no. of clusters; dataset - Soybean

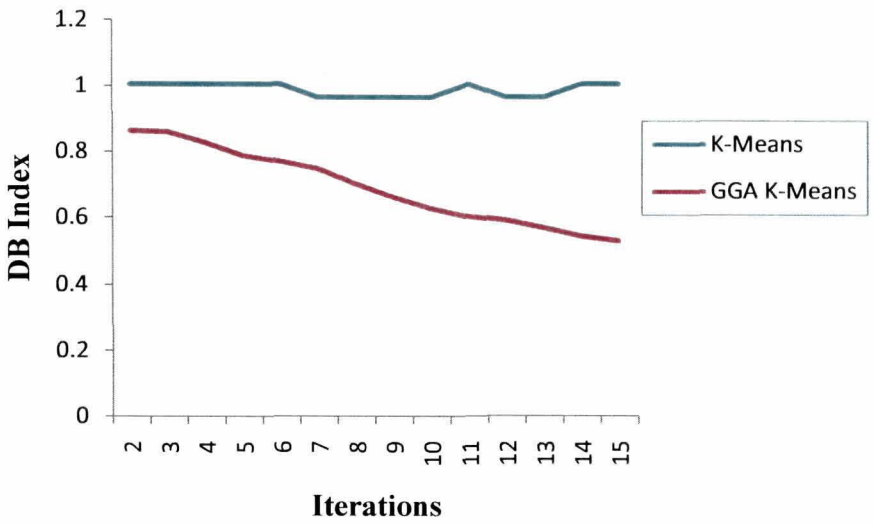


Fig 5.8 K-Means vs. GGA K-Means using DB Index for specified no. of iterations; dataset – Water Treatment Plant

To summarize briefly, the core objective of this research sought to **measure the benefits obtained by use of recent optimization schemes on aspects of clustering that requires selection of free parameters by a trial method.** At the initial stage of the research, an attempt to **orient the investigation process by experimenting with Textual Dataset** and create a base for future work in the Text Mining / Web Mining areas was made. Text Mining requires knowledge of Information Retrieval techniques like Vector Space Model. A voluminous collection of documents requiring investigation is firstly pre-processed involving concepts like Term Frequency, Document Frequency and Vocabulary (Thesaurus) generation and converted into an equivalent numeric dataset. The converted dataset is now ready for data mining tasks such as clustering & classification. This forms the basis to explore document clustering. However, since **our central thrust was to estimate the utility of optimizer augmented clustering on free parameters, we decided to experiment further with standard datasets thereafter, since the synthetic textual dataset that we experimented with didn't exhibit a good clustering quality.**

Another objective of this research was to **make a comparative performance analysis of the recent well-known optimizing schemes on sparse, moderately high-dimensional/binary dataset.** The thesis presents the results of a comprehensive study using four optimization techniques viz., Genetic Algorithm, Simulated Annealing, Particle Swarm Optimization & Differential Evolution in an attempt to arrive at better values for centroids of K-means clustering.

The performance of these methods and an independent K-means run are observed for a fixed number of generations/termination criteria.

There were several papers in the past which dealt with the use of PSO, GAs, in solving the K-Means clustering problem; specifically, there were papers on selecting initial seeds for the K-Means algorithm based on a variety of stochastic search algorithms including GAs, Simulated annealing and Tabu Search. Also, there were recently proposed algorithms like the KMEANS++ which have guaranteed behaviour [46]. Ours is a systematic study of the approach of combining efficient optimization algorithms with time-tested clustering algorithms, in a multi-loop iterative technique. As a case study, we **explored the benefit of optimization in the iterative K-Means algorithm and thereafter for Maximin,** for obtaining compact clusters (over a fixed number of clusters) through minimization of Euclidean & Pearson distance of cluster points from centroid of a given cluster. The relative comparison of several cutting-edge optimization techniques in the inner optimization loop of K-Means was also very interesting. Finally, the experimental study with real-world text clustering problem and two standard data sets unraveling important pointers on performance issues of some of the very popular optimizers is also an important practical contribution we feel.

The first experiment on BIGCHECK revealed encouraging results for GA Steady as an optimizer. It improved upon K-Means on metrics such as fitness function and sum of squared Euclidean distance. Proceeding with the next optimizer, Particle Swarm, it brought about a marginal improvement in the values of fitness function, but soon hit a plateau. Experimenting with DE optimizer did produced impressive results in comparison with PSO, but GA Steady remained a notch better. Using SA optimizer, a particular iteration required a number of cooling schedules, taking quite a good amount of time with each cooling schedule, and so the fitness function was computed after clustering with values obtained after each cooling schedule. The result with SA couldn't match that of DE, and was way below what GA Steady achieved with the BIGCHECK dataset.

We decided to assess the improvement in cluster quality on BIGCHECK achieved by these optimizers incorporating two more flavours of GA viz., Roulette wheel based GA & Grouping GA through the use of a popular quantitative measure, Silhouette Coefficient. The improvement witnessed was marginal, from 0.15 for K-Means to 0.22 which was obtained when the GGA optimizer was used. It was felt that BIGCHECK didn't exhibit a good cluster structure, and so we

decided to experiment further with two standard datasets, viz., SOYBEAN & WATER TREATMENT PLANT. With SOYBEAN, the improvement achieved using the cluster validity index Silhouette Coefficient with the best optimizer was nearly two-fold, while in WATER TREATMENT PLANT, it surpassed even that mark and was nearly a two-and-a-half fold betterment for the experiment with Euclidean distance similarity. The SC values for the three datasets improved marginally with the use of Pearson similarity. In this set of experiments with Pearson, the most significant improvement was seen in the case of SOYBEAN. In the case of experiments with the second quantitative measure, Davies Bouldin index (DBI) almost all the optimizers brings about a significant betterment, and the best optimizer brings about nearly a two-fold improvement. **WATER TREATMENT PLANT proved a better dataset for optimization and achieved lower values for DBI (signifying enhanced performance) among the three datasets.** The results obtained using Pearson vis-à-vis Euclidean was indicative of a marginal improvement. A comparative performance between the best optimizer and K-Means for 25 iterations showed a near two-fold improvement for the SOYBEAN dataset (Euclidean), while with Pearson the improvement was even more than the double mark.

As a conclusion, **Grouping GA emerges as the best optimizer** of the methods that have been used in the experiments with three datasets (one synthetic and two standard datasets). Further, though optimization of iterative K-Means have been studied in the literature, **such a comprehensive study exploring four recent optimizing techniques for OAC have not been reported.**

Chapter 6

Experiments with Maximin

The encouraging results obtained with K-Means intrigued our curiosity to experiment with another clustering algorithm. We choose the Maximin Clustering Algorithm.

6.1 Maximin Clustering Algorithm

This is an iterative algorithm that eliminates some of the problems of the threshold based clustering algorithm. Secondly, the threshold value is adaptive.

Let $\{x_1, x_2, \ldots\ldots\ldots x_N\}$ be the given set of patterns of unknown classes.

1) Choose an appropriate similarity (distance)

2) Choose $z_1 = x_1$ (cluster center 1). $N_c = 1$ (number of clusters)

3) Compute all inter-pattern distances

4) Find the farthest sample from x_1 and assign that pattern as cluster center z_2. Set $N_c = 2$.

5) Compute distances between all other samples to $z_1, \ldots, z_{N_c}$.

Compute $d_i = Min \quad \| x_i - z_j \| \; for \; i = 1, 2, \ldots\ldots N$

$i \leq j \leq N_c$

Compute $Max \quad \{d_i\} = d_k$

$1 \leq i \leq N$

6) If $d_K > K \left(\| z_i - z_j \| \right)$, i, j = 1, 2, ..., N_c, $i \neq j$. Then $N_c = N_c + 1$, $z_{N_c} = .x_K$

Go to 5.

Else assign remaining patterns to closest cluster center.

Stop.

$(\| z_i - z_j \|)$ is a "typical" intercluster distance and may be chosen to be the smallest intercluster distance or average intercluster distance. K is a fraction that is determined by the user according to the problem.

Firstly, we computed the fitness value (using inverse of Euclidean distance) for the Maximin algorithm for the dataset BIGCHECK using the same set of initial centroids that was used for experiments with K-Means reported in an chapter 4. The comparative values for the two algorithms K-Means and Maximin are shown in Table 6.1

Iteration#	Fitness values of K-means	Fitness values of Maximin	Iteration#	Fitness values of K-means	Fitness values of Maximin
1	780	924	29	783	933
2	782	926	30	773	933
3	757	926	31	780	933
4	770	927	32	764	934
5	773	929	33	775	934
6	774	929	34	767	934
7	760	929	35	779	934
8	778	929	36	771	934
9	767	930	37	766	935
10	784	930	38	768	935
11	773	930	39	757	935
12	768	930	40	782	935
13	777	931	41	763	935
14	777	931	42	777	935
15	774	931	43	770	936
16	766	932	44	771	936
17	762	932	45	769	937
18	771	932	46	752	937
19	783	932	47	772	937
20	782	932	48	769	939
21	773	932	49	774	939
22	782	932	50	781	941
23	767	932			
24	794	932			
25	777	933			
26	779	933			
27	780	933			
28	776	933			

Table 6.1: Fitness values of K-Means & Maximin; dataset - BIGCHECK

Fig 6.1 shows the comparative performance of the two algorithms.

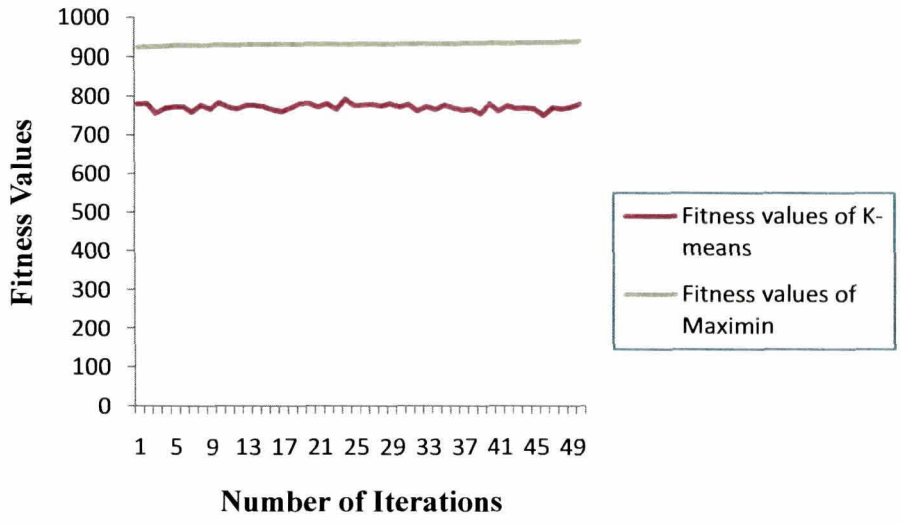


Fig 6.1: K-Means Vs. Maximin; dataset - BIGCHECK

The results corresponding to the experiments with the optimizers being used for investigation of improvement are shown in Table 6.2 and Fig. 6.2

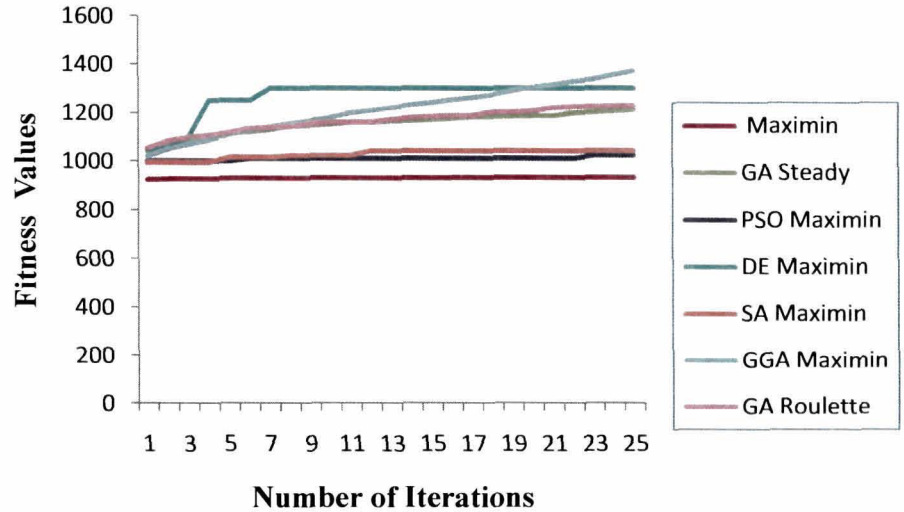


Fig 6.2: Maximin Vs. Optimizer based Maximin; dataset - BIGCHECK

Iteration#	Maximin	GA Steady	PSO Maximin	DE Maximin	SA Maximin	GGA Maximin	GA Roulette
1	924	1041	1000	1050	995	1020	1055
2	926	1073	1000	1050	995	1049	1083
3	926	1091	1000	1100	995	1069	1099
4	927	1103	1000	1250	995	1085	1107
5	929	1117	1000	1250	1016	1111	1119
6	929	1121	1010	1250	1016	1126	1136
7	929	1130	1010	1300	1016	1141	1140
8	929	1144	1010	1300	1024	1154	1144
9	930	1148	1010	1300	1024	1167	1154
10	930	1153	1010	1300	1024	1181	1161
11	930	1160	1010	1300	1025	1200	1162
12	930	1161	1010	1300	1043	1210	1162
13	931	1165	1010	1300	1043	1220	1173
14	931	1167	1010	1300	1043	1232	1180
15	931	1171	1010	1300	1043	1241	1185
16	932	1176	1010	1300	1043	1253	1188
17	932	1182	1010	1300	1043	1262	1189
18	932	1182	1010	1300	1043	1273	1203
19	932	1185	1010	1300	1043	1288	1203
20	932	1186	1010	1300	1043	1305	1208
21	932	1186	1010	1300	1043	1315	1220
22	932	1199	1010	1300	1043	1328	1226
23	932	1203	1023	1300	1043	1338	1226
24	932	1208	1023	1300	1043	1357	1228
25	933	1213	1023	1300	1043	1372	1228

Table 6.2: Fitness values of Maximin & Optimizer augmented Maximin; dataset -BIGCHECK

The experiments with the Silhouette Coefficient yielded the values as shown in Table 6.3 and in Fig. 6.3

Algorithm	SC Values
Maximin	0.094675
GA Steady	0.18381
GA Roulette	0.220203
GGA Maximin	0.266851
SA Maximin	0.127282
DE Maximin	0.233653
PSO Maximin	0.128508

Table 6.3: Silhouette Coefficient values of Maximin & Maximin augmented optimizer

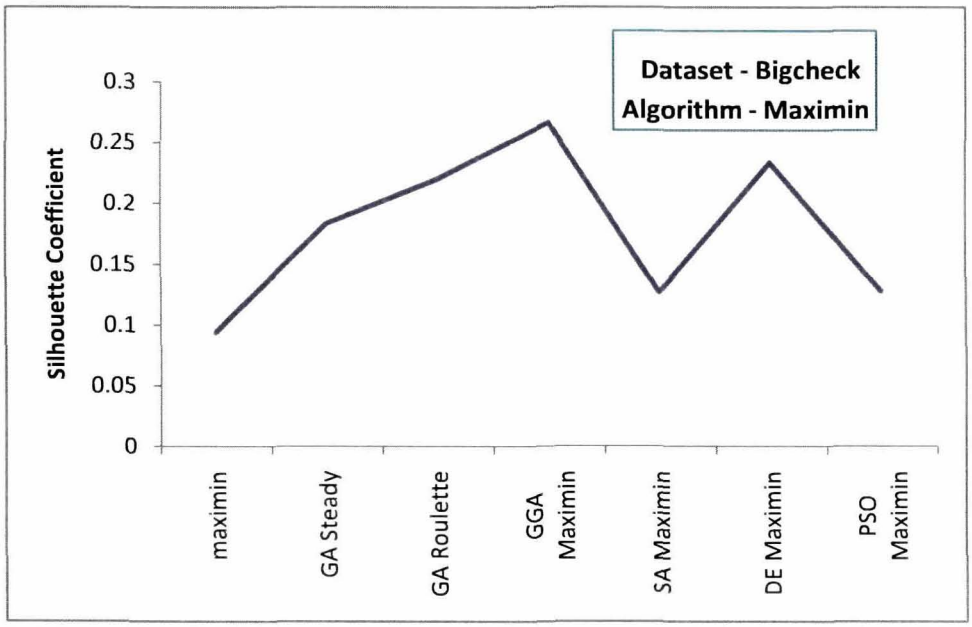


Fig 6.3: Silhouette Coefficient Vs. clustering algorithms; dataset - BIGCHECK

The experiments with the Davies Bouldin index yielded the values as shown in Table 6.4.and Fig 6.4

Algorithm	DBI Values
Maximin	2.58
GA Steady Maximin	1.65
GA Roulette Maximin	1.77
GGA Maximin	1.28
SA Maximin	1.87
DE Maximin	1.58
PSO Maximin	1.92

Table 6.4: Davies Bouldin index values of Maximin & Maximin augmented optimizer

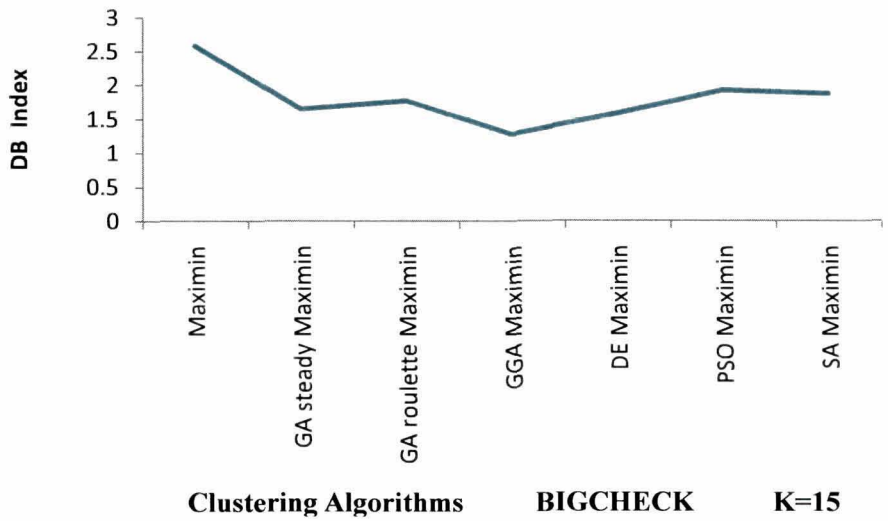


Fig 6.4: DB Index vs. clustering algorithms; dataset - BIGCHECK

The experiments with the Silhouette Coefficient for Water Treatment Plant dataset yielded the values as shown in Table 6.5 and Fig 6.5

Algorithm	SC Values
Maximin	**0.4319**
GA Steady Maximin	**0.5974**
GA Roulette Maximin	**0.5764**
GGA Maximin	**0.6081**
SA Maximin	**0.5475**
DE Maximin	**0.5946**
PSO Maximin	**0.5319**

Table 6.5: Silhouette Coefficient of Maximin & Maximin augmented optimizer

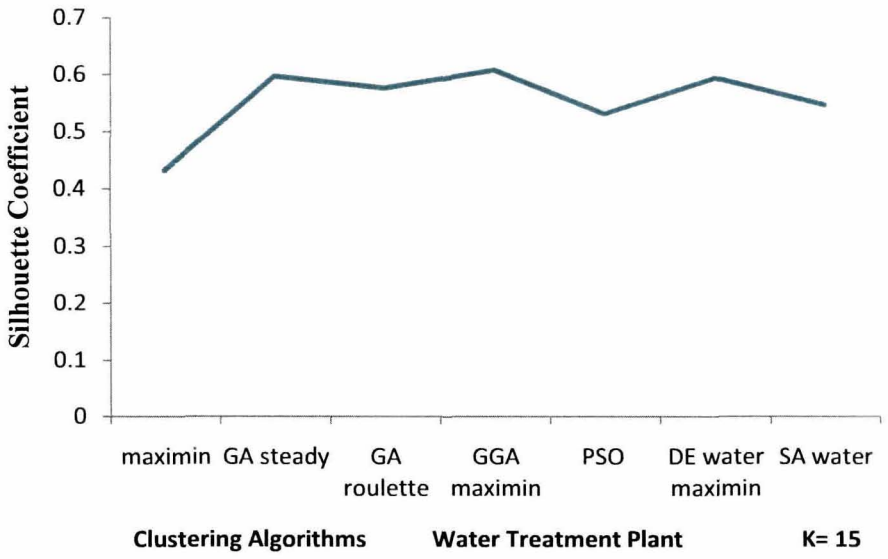


Fig 6.5: SC vs. clustering algorithms; dataset – WATER TREATMENT PLANT

Chapter 7

Discussion on Maximin results

The set of experiments with Maximin revealed that Maximin **arrived at better centroids in comparison to the popular K-Means**, though in terms of efficiency, K-Means proved to be much better. This is quite reasonable, because the complexity of K-Means is linear whereas that of Maximin is quadratic. For experiments with Maximin, the threshold was varied from 100 to 10,000 in steps of 100 while computing Max distance d_k and evolving subsequent clusters.

In Table 6.1, in the previous chapter, it is shown that Maximin continuously improves upon the previous solution, whereas K-Means doesn't. Where efficiency is not a major issue, Maximin would be preferable over K-Means. In the following Table 6.2 we report the results with the optimizer(s) augmented clustering using Maximin as the base algorithm instead of K-Means. The results clearly bring out the superiority of the optimizer augmented clustering approach. Table 6.2 unfolds the sudden improvement brought about by an optimizer like DE (say after 10 iterations), whereas GGA have demonstrated its superiority as the number of iterations increases.

The experiments with the Silhouette Coefficient yielded some improvement as shown by the values in Table 6.3 and in Fig. 6.3. GGA, DE & GA Roulette performed significantly well. The experiments with DBI only indicate a strong performance by GGA Maximizer.

For the second dataset WATER TREATMENT PLANT, the experiments with Silhouette Coefficient shows a comparable performance by all the optimizers.

The experimental results show that the strategy that uses the Silhouette index gives slightly more accurate results than the strategy that employs the Davies-Bouldin index. **Fig 4.9 and Fig 4.12 clearly shows that Silhouette Coefficient can capture the separation between the curves for the three datasets in a better fashion than DB Index where the separation between the curves is minimal.** However, the computation of the Davies-Bouldin index is much less complex than the computation of the Silhouette index, which is a very important advantage regarding eventual real-time operation.

The time complexity of the Silhouette index computation is quadratic in the number of vectors involved in the clustering, whereas the time complexity of the Davies-Bouldin index computation is linear in the number of clustered vectors. In the case of the algorithm that uses the Silhouette

index, a relatively small improvement in correctness of the results over the algorithm that uses the Davies-Bouldin index is penalized with a significant increase in computational complexity. This may make the algorithm that uses the Silhouette index unsuitable for real-time operation [148].

As seen in the experiments with K-Means, here GGA and DE scores over the other optimizers for the two datasets viz., BIGCHECK & WATER TREATMENT PLANT used for the experiments.

Chapter 8

Conclusion and Future works

In this Chapter, we summarize our work briefly, and highlight future work in this direction.

Evolutionary algorithms essentially evolve clustering solutions through operators that use probabilistic rules to process data partitions sampled from the search space [92]. Roughly speaking, more fitted partitions have higher probabilities of being sampled. Thus, the evolutionary search is biased towards more promising clustering solutions and tends to perform a more computationally efficient exploration of the search space than traditional randomized approaches (e.g., multiple runs of K-Means). Besides, traditional randomized approaches do not make use of the information on the quality of previously assessed partitions to generate potentially better partitions. For this reason, these algorithms tend to be less efficient (in a probabilistic sense) than an evolutionary search. In spite of the theoretical advantages (in terms of computational efficiency) of evolving clustering solutions, much effort has also been undertaken towards showing that evolutionary algorithms can provide partitions of better quality than those found by traditional algorithms. In fact, this may be possible provided that the parallel nature of the evolutionary algorithms allows them to handle multiple solutions, possibly guided by different distance measures and different fitness evaluation functions.

Despite the simplicity involved in these and other related approaches, plenty of computational resources may be wasted to figure out and/or fix invalid solutions. The real impact of such computational burden into the efficiency of the evolutionary search relies on several factors that are hard to theoretically assess, such as the overall design of the evolutionary algorithm and the application in hand. Therefore, it is more conservative not to make any sharp claims concerning the generalization of the efficacy provided by any algorithm and/or application [130].

8.1 Conclusion and Major Achievements:

We have done the following:

1. We investigated the benefits of optimization on free parameters of the clustering algorithm. **Case Study: Iterative K-Means algorithm, Maximin algorithm.**

2. Initial experiments were conducted in the domain of textual data. 1060 records of ACM Citations were converted to numeric values for experimentation. Four optimizing techniques viz., GA Steady State, Simulated Annealing, Differential Evolution & Particle Swarm Optimization were used in an inner loop to arrive at better centroids compared with an independent k-means run. Results indicate GA & DE gives the best values, with PSO performing rather poorly.

3. To assess the improvement in cluster quality of using these techniques, we use a quantitative measure Silhouette coefficient (SC). Three datasets were used for the experimentation viz., synthetic dataset BIGCHECK (ACM citation dataset used in earlier experiment), and two standard datasets SOYBEAN & WATER PLANT TREATMENT. In addition to the earlier four methods, two more GA methods were used: Roulette Method & Grouping GA (GGA). Results indicate that GGA comes up with the best values, while in general GA & DE followed by SA gives much improved values.

4. Using GGA improves cluster quality by a substantial margin, for instance in the K-Means experiment with WATERPLANT TREATMENT dataset, the SC value improves from 0.33 to .78

5. GGA emerges as an effective technique for arriving at better centroids of K-Means for high dimensional datasets.

6. The experiments were firstly done using Euclidean distance for computing the similarity of points. For high dimensional data, Pearson Coefficient is an effective similarity measure. The same experiments for computing silhouette coefficients has been done using Pearson distance as the distance measure (in stead of Euclidean) for K-Means &

other hybrid optimized schemes. The results attest to the above claims made, and the results improve from the Euclidean measure.

7. Consequent to the experiments with silhouette coefficient, another well-known validity index, Davies Bouldin index have been used to assess the results. The results indicate that **GGA (with Pearson distance) emerges as the most effective optimizer that can effectively be used for augmentation of iterative K-Means.**

    - ☐ This is an **interesting observation,** as even though GGA has been used in optimization problems, **no mention/reference is found in the literature regarding its usage in augmentation of iterative K-Means.**

GGA (with Pearson distance) can effectively be used for augmentation of iterative K-Means. This is a significant contribution of this investigation, as because **even though Grouping GA has been used in optimization problems, no mention/reference was found in the literature regarding its usage in augmentation of iterative K-Means.** The benefits employing DE & SA and the other two GA based methods viz., GA Steady State & GA Roulette are comparable in terms of their improvement factor, and can form a second choice, next to GGA.

In brief, the contributions of the investigation are:

(1) **It clearly establishes the marked benefits of using optimizing schemes in the clustering process**

(2) **It provides an insight and a comparative assessment into the performance benefits of using four well-known optimizing techniques on a textual/binary/numeric data set.** The same set of initial centroids was provided to both K-Means and optimizer augmented K-Means, and after iteration of the algorithm the fitness value (indicator of the quality of the centers) was determined. Independent K-Means run came up with poor centroids which would affect large scale clustering problems. If we use the inner optimizer loop to find better centroids by running the optimizer based algorithm for a reasonable amount of time, say for 10-15 iterations or exit beforehand upon some termination criteria such as no improvement reported for five successful iterations, we can eventually come up with far better clusters. The computational cost for larger

datasets would be a cause of concern for any method, but this method can work effectively for mid-size datasets, typically in the range of 1000 – 1,00,000 points.

(3) **GGA (with Pearson distance) can effectively be used for augmentation of iterative K-Means.** This is a significant contribution of this investigation, as because **even though Grouping GA has been used in optimization problems, no mention/reference was found in the literature regarding its usage in augmentation of iterative K-Means.**

(4) The fourth contribution is the revelation of **Maximin arriving at better centroids in comparison to the popular K-Means**, though in terms of efficiency, K-Means proved to be much better.

(5) The fifth contribution is the discovery that the **Silhouette index gives slightly more accurate results than the Davies-Bouldin index.** However owing to computational complexity, Silhouette index may be unsuitable for real-time operation.

6.2 Further Work:

A lot of interesting possibilities still remain to be explored in the future. Bag-of-words model used for experimentation is not a good model for text clustering. Semantic relationships are ignored in VSM. Experimenting with Semantic model can overcome the limitations that we faced in our experiments with Textual dataset.

As an extension to the present study, one may attempt to estimate the benefits of using such optimizing schemes on more free parameters of clustering algorithms sensitive to the initial parameter settings, namely, CURE, shifting grid, training neural networks, global FCM, and evidential C-Means (ECMs).

There are also several other techniques and papers that talk about hyper-parameters screening and optimization and we have not referred to any of them [25] [123] [124] [125]

The paper by Bengio [123] 'Gradient based...' presents a methodology for optimizing several hyper-parameters and is far too mathematical in nature as well. The paper "R. Kohavi and G.H. John [124]. Automatic Parameter Selection by Minimizing Estimated Error' deal mostly with a statistical cross validation technique to arrive at optimal values of the parameters, and the experimentation have been performed employing a mix of best first search and decision-tree and modifying Quinlan C4.5 algorithm. The paper [125] is basically a novel incremental approach of solving the local convergence problem of K-Means by adding centroids, one at a time, in an orderly optimized manner in the clustering process. It may also be mentioned that our method is vastly different from the cited work by [25] on using GA to clustering using K-means. [25] employed GA for clustering, and instead of crossover used K-means as a genetic operator for clustering. We have used GA as a meta-clusterer to arrive at better selections of centroids in the inner loop. Thereafter, the clustering algorithm K-means was applied.

Future work can investigate other potential optimization techniques. There are a plethora of optimization techniques/ methods which may also be probed for their ability to optimize free parameters to improve clustering/iterative K-Means/Maximin/other clustering algorithms.

1. Ant Colony [94] [106]
2. Hybrid techniques like fuzzy genetic, hybrid PSO
3. Bacterial Foraging
4. Tabu search
5. Evolutionary K-means
6. Memetic Algorithms
7. Quantum Annealing
8. Adaptive simulated annealing
9. Biogeography based optimization (BBO)
10. Other GA based methods (Elitist, Tournament Selection)
11. Learning Automata
12. Best First Search
13. Harmony Search
14. Integer Programming
15. Dynamic Programming
16. Branch and Bound
17. Cellular Automata
18. Modified PSO using a new concept 'repulsion', called repulsive particle swarm optimization

19. Graphical Techniques, which reportedly have proved successful in overcoming the local convergence of K-Means.

One may also like to experiment with more standard datasets. We have concentrated only on the improvement of numeric datasets. The improvement achieved using these techniques on other types of datasets may further be investigated. We have experimented using a single objective function i.e., we tried to optimize the centroids of iterative K-means/Maximin. One can also study the effect of Multi-Objective optimizers e.g., Multi Objective GA (MOGA) on the free parameters of the clustering process. Multi-objective could be Optimization of (1) centroid of iterative K-Means (2) Selection of seed for K-Mean, or the number of clusters for K-Means(user specified), or the minimization of within cluster spread of the objects for compactness (Inter & Intra Cluster distance).

K-Means and Maximin were specific instances of a clustering algorithm that we have considered. One can explore results of more such free parameters for the clustering algorithm k-medoid for instance. There have been studies employing some of these optimizers like GA & SA on popular clustering algorithms like K-Means, and we have not done comparison/benchmarking.

Further, a variety of clustering algorithms has been proposed and known in many different domains of applications. How such a framework can be adapted to these **OAC** algorithms and applications needs to be investigated. Finally, in many domains, prior or experiential knowledge may be available in many cases about a good, but not necessarily optimal guess concerning many free parameters. How to combine such prior knowledge with an automatic optimization algorithm to achieve fast convergence with high performance is an open question at the current time.

References:

[1] Murty, M.N. Clustering large data sets, *Soft Computing approach to Pattern Recognition and Image Processing* **53**, 41-63 (World Scientific, Singapore, 2002) ISBN: 981-238-251-8

[2] Jain, A. K. and Dubes, R.C. *Algorithms for clustering data* (Prentice Hall, New Jersey, 1988)

[3] Fasulo, D. An Analysis of Recent Work on Clustering Algorithms (1999), http://citeseer.ist.psu.edu/fasulo99analysi.html

[4] MacQueen, J. Some methods for classification and analysis of multivariate observations, Proc. 5th Berkeley Symp. Math. Stat. and Prob., **1**, 281-97 (Univ. of Calif. Press, 1967)

[5] Anderberg, M.R. *Cluster Analysis for Applications* (Academic Press, New York, 1973)

[6] Everitt, B.S. *Cluster Analysis* (John Wiley & Sons, New York., 1974)

[7] Kaufman, L. and Rousseeuw, P.J. *Finding Groups in Data. An Introduction to Cluster Analysis* (John Wiley & Sons, Canada, 1990)

[8] Berkhin, P. Survey of Clustering Data Mining Techniques (2002), http://citeseer.ist.psu.edu/berkhin02survey.html

[9] Jain, A. K. Data Clustering: 50 Years beyond K-Means, *Pattern Recognition Letters.* **31**, 8, 651-666. (2009)

[10] Su, T. and Dy, J. G. In search of deterministic methods for initializing K-means and Gaussian mixture clustering, *Intelligent Data Analysis* **11** (4), 319-338, (IOS Press, 2007)

[11] Bradley, P. S. and Fayyad, U. M. Refining Initial Points for K-Means Clustering, Proc. 15th Intl Conf on Machine Learning, 91 – 99, (Morgan Kaufmann Publishers, San Francisco, 1998) ISBN: 1-55860-556-8 , http://research.microsoft.com/~fayyad

[12] Fayyad, U., Reina, C. and Bradley, P.S. Initialization of Iterative Refinement Clustering Algorithms, Proc. 4th Int'l Conf. on Knowledge Discovery and Data Mining (KDD98), (AAAI press, 1998), http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.44.4060

[13] Hartigan, J. A. *Clustering Algorithms* (John Wiley & Sons, New York, 1975)

[14] Faber, V. Clustering and the Continuous k-means Algorithm, *Los Alamos Science, **22***, 138-144 (1994)

[15] Pena, J.M., Lozano, J.A. & Larranaga, P., An empirical comparison of four initialization methods for the K Means algorithm, ***Pattern Recognition Letters*** , **20** , 10, 1027 - 1040 (1999) ISSN:0167-8655

[16] Meila, M. and Heckerman, D. An experimental comparison of several clustering and initialization methods (1998), http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.8.1159

[17] Zhang, Z., Dai, B. T. and Tung, A. K.H. On the Lower Bound of Local Optimums in K-Means Algorithm, Proc. 6th Intl Conf on Data Mining ICDM'06, 775 – 786 (IEEE Computer Society, Washington DC, 2006) ISBN: 0-7695-2701-7, DOI: 10. 1109/ ICDM.2006.118

[18] Daoud, M. A., Venkateswarlu, N. B. and Roberts, S. New Methods for the Initialization of Clusters, ***Pattern Recognition Letters*** **17**, 5, 451 - 455 (1996) ISSN: 0167-8655

[19] Pelleg, D. and Moore, A. X-means: Extending K-means with Efficient Estimation of the number of clusters, ***LNCS***, **1983**, Proc. 2nd Intl Conf on Intelligent Data Engineering and Automated Learning, Data Mining, Financial Engineering, and Intelligent Agents, 17-22, (Springer-Verlag, London, 2000) ISBN:3-540-41450-9

[20] Pelleg. D. and Moore, A. Accelerating Exact k-means Algorithms with Geometric Reasoning, Proc. 5th ACM SIGKDD Intl Conf on Knowledge Discovery and Data Mining, 277–281 (ACM New York, 1999) ISBN:1-58113-143-7

[21] Kanungo, T., Mount, D.M., Netanyahu, N. S., Piatko, C. D., Silverman, R.and Wu, A. Y. A Local Search Approximation Algorithm for k-Means Clustering, 18th Annual ACM Symposium on Computational Geometry, 10–18 (SoCG, Barcelona, 2002)

[22] Deelers, S. and Auwatanamongkol, S. Enhancing K-Means Algorithm with Initial Cluster Centers Derived from Data Partitioning along the Data Axis with the Highest Variance, Proc. of world Academy of Science, Engineering and Technology, **26** (2007)

[23] Redmond, S. J. & Heneghan, C. A method for initializing the K-means Clustering algorithm using kd-trees, ***Pattern Recognition Letters***; **28**, 8, 965-973 (2007) ISSN: 0167-8655

[24] Hochbaum, D. and Shmoys, D. A best possible heuristic for the k-center problem, ***Mathematics of Operations Research***, **10**(2):180-184 (1985)

[25] Krishna, K. and Murty, M. Genetic K-means algorithm, ***IEEE Trans. Syst., Man, Cybern. B., Cybern.*** **29**, 3, 433 – 439 (1999)

[26] Epter, S., Krishnamoorthy, M. and Zaki, M. Clusterability detection and initial seed selection in large data sets, Technical Report 99-6, Department of Computer Science, Rensselaer Polytechnic Institute (1999), http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.30.1108

[27] Maulik, U. and Bandopadhyay, S. Genetic Algorithm based Clustering technique, ***Pattern Recognition*** **33**,1455-1465 (2000)

[28] Laszlo, M. and Mukherjee, S. A genetic algorithm that exchanges neighboring centers for k-means clustering, ***Pattern Recognition Letters*** **28** , 2359–2366 (2007)

[29] Murthy, C.A. and Chowdhury, N. In search of optimal clusters using genetic algorithms, ***Pattern Recognition Letters***, **17**, 14, 825–832 (1996)

[30] Jain, A.K., Murty M.N., Flynn P.J. Data Clustering: A review, ***ACM Computing Surveys***, **31**(3), 264-323 (1999)

[31] Grossman, D. A. and Frieder, O. *Information Retrieval: Algorithms and Heuristics* (Springer, 2004) ISBN 978-81-8128-917-9

[32] Huang, H., Mok, P.Y., Kwok, Y. L & Au, S. C. A parameter free approach for clustering analysis, X. Jiang & N. Petkov (Eds.): CAIP 2009, ***LNCS* 5702**, 816-823 (Springer-Verlag, Berlin Heidelberg, 2009) DOI 10.1007/978-3-642-03767-2

[33] Zhang, T. Data Clustering for Very Large Datasets Plus Applications, Ph. D thesis (University of Wisconsin at Madison, 1997) Order Number: UMI Order No. GAX97-11817

[34] Goldberg, D. E. *Genetic Algorithms in Search, Optimization and Machine Learning*, (Addison Wesley, Reading, 1989)

[35] Steinbach, M., Karypis, G., Kumar, V. A Comparison of Document Clustering Techniques, Text Mining Workshop, KDD (2000), http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.34.1505

[36] Zhao, Y. and Karypis, G. Empirical and Theoretical Comparisons of Selected Criterion Functions for Document Clustering, ***Machine Learning*, 55** (3): 311-331. (2004)

[37] Mirkin, B. Computational Intelligence and Visualization, www.dcs.bbk.ac.uk/~mirkin/advanced

[38] Dhillon, I.S., Fan, J. and Guan, Y. Efficient Clustering of very large document collections in Data Mining for scientific and engineering applications, C., P.Kegelmeyer, and R. L. Grossman (Eds.), 357-381 (Kluwer Academic, 2001)

[39] Eberhart, R. C. and Shi, Y. Comparison between genetic algorithms and particle swarm optimization, Proc. 7th ann. conf. on evolutionary conf., ***LNCS*** Evolutionary programming **1447** (Springer Berlin, Heidelberg, 1998) ISBN 978-3-540-64891-8, 611-616

[40] Shi, Y. and Eberhart, R. C. Parameter selection in particle swarm optimization, Evolutionary Programming VII: Proc. EP 98, 591-600 (Springer-Verlag, New York, 1998)

[41] Paterlini, S. and Krink, T. Differential evolution and particle swarm optimization in partitional clustering, ***Comput. Stat. Data Anal.*** **50** (2006)

[42] Mitra, S., Pal, S. K. and Mitra, P. Data Mining in Soft Computing Framework: A Survey, ***IEEE Transaction on Neural Networks***, **13** (I), (2002)

[43] Selim, S. Z. and Sultan, K. A. A Simulated Annealing Algorithm for the Clustering Problem, ***Pattern Recognition***, **24**, 1003-1008 (Pergamon Elsevier Sc)

[44] Babu, G. P. and Murty, M. N. Simulated annealing for selecting optimal initial seeds in the K-Means Algorithm, ***Indian J. pure appl. Math.*** , **25**(1 & 2); 85-94 (1994)

[45] Kanungo, T., Mount, D. M., Netanyahu, N. S., Piatko, C. D., Silverman, R. & Wu, A. Y. An Efficient k-Means Clustering Algorithm: Analysis & Implementation, ***IEEE Transactions On Pattern Analysis And Machine Intelligence***, **24** (7), 881-892 (2002)

[46] Arthur, D., Vassilvitskii, S. kmeans++: The Advantages of Careful Seeding, Proc. 18th ACM-SIAM symposium on Discrete algorithms, 1027 – 1035 (New Orleans Louisiana, 2007) ISBN: 978-0-898716-24-5

[47] Bandyopadhyay S., and Maulik U. An evolutionary technique based on K-means algorithm for optimal clustering in RN, ***Information Sciences***, **146**, 221-237 (2002)

[48] Mostafa, J., Mukhopadhyay, S., Lam, W. and Pallakal, M. A Multilevel Approach to Intelligent Information Filtering: Model, System, and Evaluation, ***ACM Transactions on Information Systems***, **15**(4), 368–399 (1997)

[49] Abraham, A., Das, S. and Konar, A. Document Clustering Using Differential Evolution, Evolutionary Computation, CEC 2006. Congress on, 1784-1791 (2006) ISBN: 0-7803-9487-9 DOI: 10.1109/CEC.2006.1688523

[50] Paterlini, S. & Krink, T. High performance clustering with differential evolution, Evolutionary Computation, CEC 2004. Congress on, **2,** 2004-2011 (2004) ISBN: 0-7803-8515-2

[51] Cheo, C. Y & Ye, F. Particle Swarm Optimization Algorithm and Its Application to Clustering Analysis, Proc. IEEE Intl Conf on Networking, Sensing & Control (Taipei, 2004), **2,** 789-794, ISSN: 1810-7869 Print ISBN: 0-7803-8193-9 DOI: 10.1109/ICNSC.2004.1297047

[52] Arthur, D. and Vassilvitskii, S. How slow is the K-Means method, Proc. 22nd annual symposium on Computational Goemetry *SCG'06,* Sedona, Arizona, 144-153 (ACM, New York, 2006) ACM 1-59593-340-9/06/0006 ISBN: 1-59593-340-9

[53] Pearson, R. K., Zylkin, T., Schwaber, J. S. and Gonye, G. E. Quantitative Evaluation of Clustering Results Using Computational Negative Controls, © SIAM http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.118.436

[54] Shin, M., Kang, E. M. and Park, S. H. Automatically Finding Good Clusters with Seed K-Means, ***Genome Informatics Series*** **14**:326-327 (2003)

[55] Li, M. J. , Ng, M. K., Cheung, Y. M. and Huang, J. Z. Agglomerative Fuzzy K-means Clustering Algorithm with Selection of Number of Clusters, ***IEEE Transactions On Knowledge And Data Engineering***, **20** (11), 1519 - 1534 , ISSN: 1041-4347 DOI: 10.1109/TKDE.2008.88 (IEEE, Los Angeles, 2008)

[56] Hand, D. J. , Mannila, H. & Smyth, P., *Principles of Data Mining* (MIT Press, Cambridge MA , 2001)

[57] Patterson, D. W. *Introduction to Artificial Intelligence and Expert Systems* (Prentice-Hall Inc., USA, 1990) ISBN-978-81-203-0777-3

[58] Backer, E. Computer Assisted Reasoning in Cluster Analysis (Prentice Hall, 1995)

[59] Ben-Hur, A., Horn, D., Siegelmann, H. T. and Vapnik, V. Support Vector Clustering, ***Journal of Machine Learning Research***, **2**:125-137 (2002)

[60] Kohonen, T., Kaski, S., Lagus, K. Salojarvi, J., Honkela, J., Paatero, V. & Saarela, A. *Self-Organization of a massive document collection,* ***IEEE Trans. Neural Networks***, **11**, 574-585 (2000)

[61] Bezdek, J.C. A convergence theorem for the fuzzy isodata clustering algorithms, ***IEEE Trans. Pattern Analysis and Machine Intelligence, 2, 1.*** (1980)

[62] Januzaj, E., Kriegel, H. P. & Pfeifle, M. Towards effective and efficient distributed Clustering, Workshop on Clustering Large Data Sets (ICDM2003), Melbourne, FL (2003)

[63] Al-Mubaid, H. & Umair, S. A. A New Text Categorization Technique using Distributional Clustering and Learning Logic, ***IEEE Transactions on Knowledge and Data Engineering*** **18**(9), Sept 2006 (2006)

[64] Dunham, M. H. *Data Mining: Introductory and Advanced Topics*, Pearson Education ISBN-10: 0130888923 ISBN-13: 9780130888921 (Prentice Hall, 2003)

[65] Ross, T. J. *Fuzzy Logic with engineering applications.* ISBN: 0470860758, ISBN13: 9780470860755 (John Wiley & Sons, 2004)

[66] Tibshiraniet al. Stanford Technical Report, http://www- stat.stanford.edu/~tibs/lab/ publications.html (2001)

[67] Lloyd, S. Least square quantization in PCM, ***IEEE Trans. Inform. Theory, 28,*** 129–137 (1982)

[68] Rao, V. & Rao, H. *C++ Neural Networks and Fuzzy Logic* (BPB, 1996)

[69] Tan, P. N., Steinbach, M. & Kumar, V. *Introduction to Data Mining,* ISBN 81-317-1472-1 (Pearson Education, 2006)

[70] Kirkpatrick, S., Gelatt, C. D., Jr. & Vecchi , M. P. Optimization by Simulated Annealing, ***Science,*** **220** (4598) 671-680, DOI: 10.1126/science.220.4598.671 (1983)

[71] Cerny, V. Thermodynamical approach to the traveling salesman problem: an efficient simulation algorithm. ***Journal of Optimization Theory and Applications***. 45:41--51 (1985).

[72] Metropolis, N., Rosenbluth, A.W., Rosenbluth, M.N., Teller, A.H. and Teller, E. Equations of State Calculations by Fast Computing Machines. ***Journal of Chemical Physics,*** **21**(6):1087-1092, 1953.

[73] Kennedy, J. and Eberhart, R. C. Particle Swarm Optimization. Proceedings of the 1995 IEEE International Conference on Neural Networks, 1942-1948 (IEEE , Piscataway New Jersey, 1995)

[74] Drineas P., Frieze, A., Kannan, R., Vempala, S. & Vinay, V. Clustering in large graphs and matrices, Proc. Of SODA, SIAM (1999)

[75] http://en.wikipedia.org/wiki/Simulated_annealing

[76] Holland, J. H. *Adaptation in Natural and Artificial Systems* (Ann Arbor: The University of Michigan Press, 1975)

[77] Salton, G. and McGill, M.J. 1983. *Introduction to Modern Retrieval. McGraw Hill*

[78] Futrelle, R. P., Xiaolan, Z., & Sekiya, Y. Corpus linguistics for establishing the natural language content of digital library documents. In *Digital Libraries.* ***LNCS*****, 916**, 165–180 (Springer-Verlag, New York, 1994)

[79] Guntzer, U., Juttner, G., Seegmuller, G., & Sarre, F. Automatic thesaurus construction by machine learning from retrieval sessions. In *Proceedings of RIAO: User-Oriented Content-Based Text and Image Handling,* 587–596 (MIT, Cambridge, Mass., 1988)

[80] Liddy, E. D., Paik, W., & Yu, E. S. Text categorization for multiple users based on semantic features from a machine-readable dictionary, ***ACM Trans. Inf. Sys. 12*****, 3**, 278 –295 (1994)

[81] Ghosh, S. a survey of techniques and trends in clustering for data mining. Proc. Natl Conf. for Advanced Computing Technologies 2007, Tezpur University. In D. K. Bhattacharyya and S.M.Hazarika, editors, *Networks, Security and Soft Computing: Trends and future directions*, 125-132 (Narosa Publication, 2007)

[82] Banerjee, A. Scalable Clustering Algorithm, PhD Thesis, University of Texas at Austin (2005)

[83] Frigui, H. and Krishnapuram, R. Clustering by competitive agglomeration, ***Pattern Recognition*****, 30**(7):1109-1119 (1997)

[84] SOYBEAN dataset, http://www.cs.sfu.ca/~wangk/ucidata/dataset/soybean/

[85] Forgy, E. Cluster analysis of multivariate data: Efficiency vs. interpretability of classifications, ***Biometrics, 21***(3):768 (1965)

[86] Duda, R.O. and Hart, P.E. *Pattern Classification and Scene Analysis* (John Wiley and Sons, New York, 1973)

[87] Firouzi, B. B., Niknam, T. and Nayeripour, M. A New Evolutionary Algorithm for Cluster Analysis, Proceedings of world Academy of Science, Engineering and Technology **36** (2008),

[88] Thiesson, B., Meek, C., Chickering, D. and Heckerman, D. Learning Mixtures of Bayesian Networks (Microsoft Research Technical Report TR-97-30, Redmond, WA, 1997)

[89] Maria, A. & Garces, G. Development of a Methodology to Solve the Line Balancing Problem with Parallel Workstations, http://gradworks.umi.com/14/40/1440672.html (2007)

[90] Ester, M., Kriegal, H.P., Sander, J. and Xu, X. A density based algorithm for discovering clusters in large spatial databases with noise, Proc. 2nd Intl Conf on Knowledge Discovery and Data Mining (1996)

[91] Liu, D., Sourina, O. Free-Parameters Clustering of Spatial Data with non-uniform Densitv, Proc. 2004 IEEE Conference on Cybernetics and Intelligent Systems Singapore (2004)

[92] Falkenauer, E. *Genetic Algorithms and Grouping Problems*. ISBN 978-0-471-97150-4 (John Wiley & Sons, Chichester, England, 1997)

[93] Painho, M. and Bação, F. Using Genetic Algorithms in Clustering Problems, *http://www.geocomputation.org/2000/GC015/Gc015.htm* (2000)

[94] Kangavari, M. R. & Fakhar, B. A New Clustering Method Using Ant Colony Optimization Algorithm, IDMC' 2007 (2007)

[95] http://en.wikipedia.org/wiki/Particle_swarm_optimization

[96] Skiena, S. S. The Algorithm Design Manual, Dept of Computer Sc, State University of New York, Stony Brook, NY 11794-4400 (Springer, 2008)

[97] Bandyopadhyay, S. and Maulik, U. Non-parametric Genetic Clustering : Comparison of Validity Indices, ***IEEE Transactions on Systems, Man and Cybernetics Part-C***, **31** (1), 120-125 (2001)

[98] Brown, D. E. and Huntley, C. L. A practical application of simulated annealing to clustering, ***Pattern Recognition,*** **25** (4), 401-412 (1992)

[99] Klein, R. **W.** and Dubes, R. C. Experiments in projection and clustering by annealing, ***Pattern Recognition,*** **22**(2), 213-220 (1989)

[100] K-Means Clustering. http://people.revoledu.com/kardi/tutorial/kMean/

[101] Guha, S., Rastogi, R. & Shim, K. CURE: An efficient clustering algorithm for large databases, Proc. 1998 ACM-SIGMOD Int'l Conf. on Management of Data (SIGMOD '98) Seattle, WA, 73-84 (1998)

[102] Wu, F. X. Genetic weighted k-means algorithm for clustering large-scale gene expression data, ***BMC Bioinformatics***, **9**(6):S12doi:10.1186/1471-2105-9-S6-S12 (2008)

[103] Masson, M. H. & Denoeux, T. ECM: An evidential version of the fuzzy c-means

algorithm, ***Pattern Recognition***, **41**(4), 1384-1397 doi: 10.1016/j.patcog.2007.08.014 (2008)

[104] Wang, W., Zhang, Y. , Li, Y. & Zhang, X. The Global Fuzzy C-Means Clustering Algorithm, Intelligent Control and Automation. WCICA 2006. The Sixth World Congress on, 3604-3607, ISBN 1-4244-0332-4 doi: 10.1109/WCICA.2006.1713041 (2006)

[105] Falkenauer, E., a hybrid grouping genetic algorithm for bin packing, http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.109.3436 (1996)

[106] Kuo, R.J., Wang, H.S., Hu T., Chou, S.H., Application of Ant K-Means on Clustering Analysis, Computers and Mathematics with Applications **50**, 1709-1724 (2005)

[107] Gibson, D., Kleinberg, J., & Raghavan, P.. Clustering categorical data: An approach based on dynamical systems, Proceedings of the 24th International Conference on Very Large Databases (1998)

[108] Ganti, V., Gehrke, J., & Ramakrishnan, R. CACTUS clustering categorical data using summaries, Proceedings of the fifth ACM SIGKDDInternational Conference on Knowledge Discovery and data mining, 73-83 (1999)

[109] Guha, S., Rastogi, R., & Shim, K. ROCK: A robust clustering algorithm for categorical attributes, Proceedings of the 1999 International Conference on Data Engineering (ICDE '99), 512-521 (1999).

[110] Tung A. K. H., Ng R., Lakshmanan L. V. S. and Han J. Constraint based clustering in large databases. In Proc. ICDT, 405-419 (2001)

[111] Zhang T., Ramakrishnan R. and Livny M. BIRCH: an efficient data clustering method for very large databases. In Proc. ACM-SIGKDD Int. Conf. Managament of Data, 103-114 (1996)

[112] Castro, E. V. and Lee I. Autoclust: Automatic clustering via boundary extraction for mining massive point-data sets. In Proc. 5th International Conference on Geocomputation (2000)

[113] Agrawal R., Gehrke J., Gunopulos D. and Raghavan P. Automatic subspace clustering of high dimensional data for data mining applications. In Proc. ACM-SIGMOD Int. Conf. Management of Data (SIGMOD'98), 94-105 (1998)

[114] Ankerst M., Breunig M, Kriegel H.-P. and Sander J. Optics: Ordering points to identify the clustering structure. In Proc. ACM-SIGMOD Conf. on Management of Data, 49-60 (1999)

[115] Xu , X., Ester , M., Kriegel , H. P. , Jörg., S A distribution-based clustering algorithm for mining in large spatial databases,

http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.72.414 (1998)

[116] Sheikholeslami, G., Chatterjee, S. and Zhang, A. WaveCluster: A Multi-Resolution Clustering Approach for Very Large Spatial Databases. Proc. 24th Int. Conf. on Very Large Data Bases (1998)

[117] Hinneburg, A., Keim, D.A. An Efficient Approach to Clustering in Large Multimedia Databases with Noise. Proc. 4th Int. Conf. on Knowledge Discovery and Data Mining (1998)

[118] Wang, W., Yang, J., Muntz , R. STING: A Statistical Information Grid Approach to Spatial Data Mining. Proc. 23rd Int. Conf. on Very Large Data Bases 1997 (1997)

[119] Ng, R.T. , Han, J. Efficient and Effective Clustering Methods for Spatial Data Mining. Proc. 20th Int. Conf. on Very Large Data Bases, 144-155 (1994)

[120] MathWorks, http://www.mathworks.com/, MatLab Version 7.1.

[121] WATER TREATMENT PLANT dataset, http://archive.ics.uci.edu/ml/datasets/Water+Treatment+Plant

[122]http://www.improvedoutcomes.com/docs/WebSiteDocs/Clustering/Clustering_Parameters/Pearson_Correlation_and_Pearson_Squared_Distance_Metric.htm

[123] Bengio, Y.. Gradient based optimization of hyperparameters, Neural Computation,12, (2000)

[124] Kohavi, R. and John, G.H. Automatic Parameter Selection by Minimizing Estimated Error. In A, Prieditis and S. Russell, editors, Proceedings of the Twelfth International Conference on Machine Learning , San Francisco C, A, (MorganKaufmann, 1995)

[125] Likas, A., Vlassis, N. and Verbeek, J.J. The global k-means algorithm. ***Pattern Recognition.*** **36** (2003) 451-461

[126] Khan, S. S., Ahmad, A. Cluster center initialization algorithm for K-Means Clustering. ***Pattern Recognition letters.*** **25**, 1293-1302 (2004)

[127] Xu, R. and Wunsch, D.I.I. Survey of clustering algorithms. ***IEEE Transactions on Neural Networks***, 16(3): 645-678 (2005)

[128] Sneath, P. H. A. and Sokal, R. R. *Numerical Taxonomy*. Freeman (1973)

[129] Sheikh, R..H, Raghuwanshi, M.M. and Jaiswal, A. N. Genetic Algorithm Based Clustering: A Survey. ICETET. First International Conference on Emerging Trends in Engineering and Technology, 314-319. ISBN: 978-0-7695-3267-7, DOI Bookmark: http://doi.ieeecomputersociety.org/10.1109/ICETET.2008.48 (2008)

[130] Hruschka, E.R., Campello, R. J. G. B., Freitas, A.A. and Carvalho, A. C. P. L. F. A Survey of Evolutionary Algorithms for Clustering. ***To appear in IEEE Transactions on Systems, Man, and Cybernetics - Part C: Applications and Reviews***

[131] Kohonen, T. The self-organizing map. ***Proceedings of the IEEE***, 1464-1480 (1990)

[132] Kohonen, T. *Self-organizing maps*. 3rd edition, Springer (2001)

[133] Pal, N., Bezdek, J. and Tsao, E. Generalized clustering networks and Kohonen's self-organizing scheme, ***IEEE Transactions on Neural Networks***, 549-557 (1993)

[134] Carpenter, G. and Grossberg, S. A massively parallel architecture for a self-organizing neural pattern recognition machine. ***Computer Vision, Graphics, and Image Processing*** 54-115 (1987)

[135] Carpenter, G. and Grossberg, S. The ART of adaptive pattern recognition by a self-organizing neural network. ***IEEE Computer.*** 77-88 (1988)

[136] Carpenter, G. and Grossberg, S. ART2: Self-organization of stable category recognition codes for analog input patterns, ***Applied Optics***, 4919-4930 (1987)

[137] Carpenter, G. and Grossberg, S. ART3: Hierarchical search using chemical transmitters in self-organizing pattern recognition Architectures. ***Neural Networks*** 129-152 (1990)

[138] Carpenter, G., Grossberg, S. and Reynolds, J. ARTMAP: Supervised real-time learning and classification of non stationary data by a self-organizing neural network, ***Neural Networks***, 169-181 (1991)

[139] Smith, V.J.R. Metaheuristics for Clustering in KDD. ***Proc. IEEE Congress on Evolutionary Computation***, pp. 2380-2387 (2005)

[140] Freitas, A. A. A Review of Evolutionary Algorithms for Data Mining, ***Soft Computing for Knowledge Discovery and Data Mining***, pp. 61-93, O. Maimon; L. Rokach (Editors), Springer (2007)

[141] Sarafis, I. Data Mining Clustering of High Dimensional DataBases with Evolutionary Algorithms.

PhD Thesis, Heriot-Watt University, UK (2005)

[142] Dempster, A. P., Laird, N., Rubin, D. B. Maximum Likelihood from Incomplete Data via the EM Algorithm, ***Journal of the Royal Statistical Society***, Vol. B39, pp. 1-38 (1977)

[143] Alves, V. S., Campello, R. J. G. B., Hruschka, E. R. Towards a Fast Evolutionary Algorithm for Clustering. ***Proc. IEEE Congress on Evolutionary Computation***, pp. 6240-6247 (2006)

[144] Davies, D. L., Bouldin, D. W. A Cluster Separation Measure. ***IEEE Trans. on Pattern Analysis and Machine Intelligence***, Vol.1, pp. 224-227 (1979)

[145] Babuška, R. Fuzzy Modeling for Control. ***Kluwer*** (1998)

[146] Bezdek, J. C. Pattern Recognition with Fuzzy Objective Function Algorithm. ***Plenum Press*** (1981)

[147] Höppner, F., Klawonn, F., Kruse, R., Runkler, T. Fuzzy Cluster Analysis: Methods for Classification, Data Analysis and Image Recognition. ***Wiley*** (1999)

[148] Petrovi´c, S. A Comparison Between the Silhouette Index and the Davies-Bouldin Index in Labelling IDS Clusters

[149] Rousseeuw P., Silhouettes: a Graphical Aid to the Interpretation and Validation of Cluster Analysis, ***J. Comput. Appl. Math.***, 20, 53-65 (1987)