Parallel Clustering Algorithms: Survey

Wooyoung Kim
CSC 8530 Parallel Algorithms
Spring 2009

Abstract

Clustering is grouping input data sets into subsets, called 'clusters' within which the elements are somewhat similar. In general, clustering is an unsupervised learning task as very little or no prior knowledge is given except the input data sets. The tasks have been used in many fields and therefore various clustering algorithms have been developed.

Clustering task is, however, computationally expensive as many of the algorithms require iterative or recursive procedures and most of real-life data is high dimensional. Therefore, the parallelization of clustering algorithms is inevitable, and various parallel clustering algorithms have been implemented and applied to many applications.

In this paper, we review a variety of clustering algorithms and their parallel versions as well. Although the parallel clustering algorithms have been used for many applications, the clustering tasks are applied as preprocessing steps for parallelization of other algorithms too. Therefore, the applications of parallel clustering algorithms and the clustering algorithms for parallel computations are described in this paper.

Contents

1 Introduction

Clustering is defined as dividing input data sets called ‘clusters’. As a unsupervised learning tasks, clustering tasks have been exploited in many fields including image/video processing, machine learning, data mining, biochemistry and bioinformatics. Depending on the data properties or the purpose of clustering, different types of clustering algorithms have been developed, such as, partitional, hierarchical, graph-based clustering etc.

Most of the clustering task requires iterative procedures to find locally or globally optimal solutions from a high-dimensional data sets. In addition, very rarely real-life data present a unique clustering solution and it is also hard to interpret the ‘cluster’ representations. Therefore, it requires many experimentations with different algorithms or with different features of the same data set. Hence, they are

computationally expensive and saving computational complexity is a significant issue for the clustering algorithms. Therefore, the parallelization of clustering algorithms is very practical approach, and the parallel methods will be various for different algorithms.

In this paper, we first classify some of existing clustering algorithms and observe the properties. The literatures about clustering algorithms [42; 41; 76; 7] classify many clustering algorithms into different point of views. Mostly following the categorization in the paper [7], clustering algorithms can be categorized into 6 types of algorithms: partitional, hierarchical, evolutionary, dense-based, model-based and graph-based clustering algorithms. The parallelization of each type of algorithms have been developed and applied for some applications as well. We observed that most of the parallel clustering algorithms used message-passing model in a master-slave architecture. We also observed other techniques. For instance, in the hierarchical clustering algorithms using a specific data structure, hypercube or butterfly network of processes were used to save the update time as well. The parallel clustering algorithms have been applied to many fields of study and there is an interesting aspect of application of clustering algorithms for parallelization itself.

The remaining of this paper is as the followings. We first give general information about clustering analysis including the critical steps and desirable features for a good clustering and the difficulties in section 2. A variety of clustering algorithms are discussed in section 3 and the parallel versions clustering algorithms follows them. In section 5, we observe some of the applications of parallel clustering algorithms, followed by the applications of clustering tasks for parallelization in section 6.

2 General Information of Clustering Analysis

Jain et al. [41] defined the cluster analysis as the "organization of a collection of patterns into clusters based on similarity". The difficulty lies in the definition and the scope of ‘cluster’ in the data set. The book [42] lists some of the definitions of a cluster, which are introduced in [27]:1) A cluster is a set of similar objects and the objects in different clusters should not be similar; 2) A cluster is the set of points which are aggregated in the testing environment such that the distance between two points in a cluster is less than the distance between the points of other clusters; 3) Clusters can be densely connected regions in a multi-dimensional space separated by loosely connected points. Based on the purposed

of the application, different concept of clusters and different clustering algorithms can be applied. The first concept focuses on the minimum intra-cluster, while the second considers between-clusters and intra-clusters at the same time. Graph-based and dense-based clustering algorithms are developed mostly based on the last concept.

2.1 Distinguishing features of Clustering Analysis

Clustering analysis is distinguished with other analysis in the following criterion [41; 76]. First, clustering analysis is unsupervised classification. Unlike supervised classification whose goal is to assign an input data into one of the classes based on the classification learned with labeled training data [13], unsupervised classification is given unlabeled data. The goal is to separate or partition a set of unlabeled data sets into several clusters based on the *conceptual* or *hidden* properties of the input data sets. Second, clustering analysis is different from unsupervised predictive learning. Unsupervised predictive learning includes vector quantization [31], probability density function estimation and entropy maximization [13], and these provide an accurate characterization of unobserved samples generated from the same probability distribution. In the other hand, clustering analysis is unsupervised 'nonpredictive' learning which divides the data sets into several subsets on their subjective measurements, which is not based on the 'trained characterization'.

2.2 Clustering Analysis Components

A clustering task needs several essential steps as discussed in [41] and those steps are emphasized in the paper [76] as well. Jain *et al* [41] described the steps as the followings: 1) pattern representation, 2) measurements appropriate to the data domain, 3) clustering or grouping, 4) data abstraction and 5) assessment of output. Figure 1 shows the steps with a feedback path where the clustering output can change the feature selection/extraction process in return. First, the patterns are represented in some way through feature selection or feature extraction process which identifies the most significant features from the original patterns or produces new prominent features respectively. The second step is to define an appropriate metric for data to cluster and design a clustering algorithm. The simplest example of measurements is Euclidean distance. In this step, an objective function and any constraints are also defined. The grouping step, as the next step, is implemented with different clustering algorithms. Different clustering algorithms

and their relationships are described further in section 3. The fourth step, data abstraction process refers to the process of extracting a compact representation of a data set. Typically in the clustering task, the compact representation is a set of centers or prototypes of each cluster. The last step is the evaluation of clustering results. Any clustering algorithm will produce clusters, even though the data sets do not include any clusters in nature. Therefore, the assessment of clusters offers several aspects: cluster tendency, which assesses the data domain; cluster validity, which is the evaluation of a clustering output. Three types of validation, internal, external and relative examination exist. Internal validity is to decide if the structure is proper for the data, external validity is to compare the recovered structure to an *a priori* structure, and the relative validity is to compare two structures.

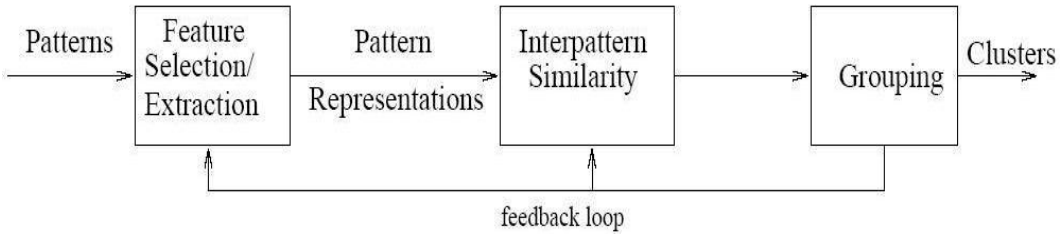

Figure 1: Clustering procedure with a feedback pathway [41].

Xu *et al* illustrated the procedure of clustering task similarly as the followings: 1) Feature selection or extraction, 2)Clustering algorithm design or selection, 3) cluster validation and 4) Results interpretation. As are in the paper [41], the four steps of [76] have a feedback pathway, and they are closely related to each other and affect the calculated clusters as depicted in 2.

Given a data samples, clustering tasks first perform a feature selection or feature extraction as in the first step in [41]. Then the second and the third steps of [41] are combined as the second step of 'clustering algorithm design or section' in this paper [76]. The selected clustering algorithm will perform the clustering task to obtain a set of clusters and the data is abstracted as a compact representation, a set of cluster prototypes. Then, Xu *et al* [76] describes the cluster validation step as the next step. Previously in [41], the step is included in the last step of 'assessment of output'. The study of cluster validation involves an effective evaluation standard, and three testing criteria; internal, external and relative cluster validity. 'Results interpretation' step is the process which gives users meaningful interpretation of data sets, which usually involves the experts of original data sets. This

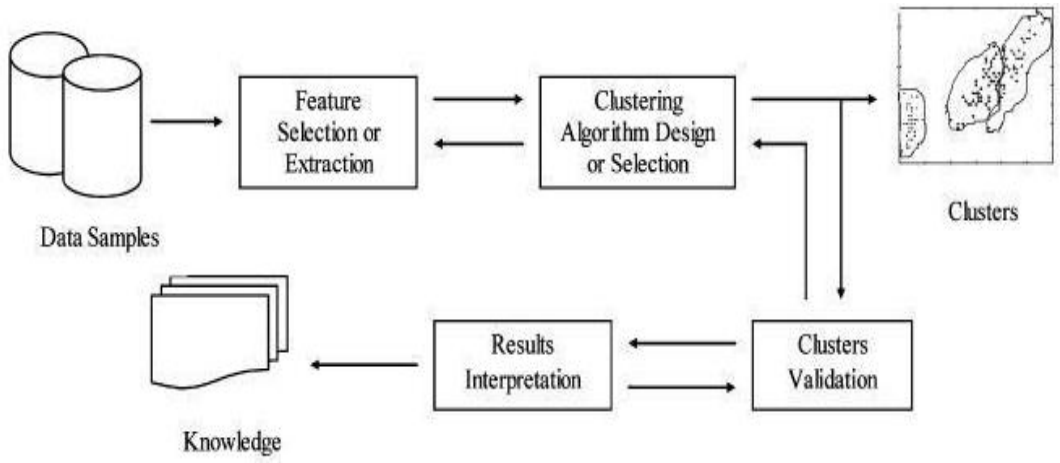

Figure 2: Clustering procedure with a feedback pathway [76].

step is also the first step in the feedback path.

2.3 Desirable Features of Clustering Analysis

In general, there are a set of desirable features for a clustering, described by Andreopoulos *et al* [7] as the followings.

Scalability The temporal and spatial complexity should not explode on large data sets.

Robustness The process should detect outliers in the data set.

Order insensitivity The algorithm should not be sensitive to the ordering of the input data.

Minimum user-specified input The number of user-specified parameters should be minimized.

Mixed data type Data can be represented as numbers or binary attributes or mixed of them.

Arbitrary-shaped clusters The clusters can be shaped arbitrary.

Point proportion admissibility Duplicating data set and re-clustering task should not change the clustering results.

Different clustering algorithms produce different results with different features. Therefore, a clustering algorithm should be chosen based on the applications as the desirable features are application dependent.

2.4 Challenges in Clustering Analysis

clustering analysis is a challenging task. There is no standard solution that can answer the questions [41] such as, How to normalize the data? What is the appropriate similarity measurement to the data? How to incorporate the knowledge in the data domain? How to cluster a large data set efficiently?

Therefore we can list some of challenges of clustering algorithms [76]. First, most of the clustering algorithms need a number of repetitions or trials. And no universal guide of feature selection or extraction. In addition, no universal validation criteria for the quality of the results and no standard solution exists. Therefore various clustering algorithms have been developed for overcoming those drawbacks and we briefly review a number of clustering algorithms in the next section.

3 Various Approaches to Data Clustering and Applications

Before we discuss parallel clustering algorithm, we review various sequential clustering algorithms and extend the concept based on the same categorization.

3.1 Taxonomy of Clustering Algorithms

Various data clustering algorithms are categorized into various ways as we can see in the papers [41; 76; 7]. Jain *et al* provided the taxonomy of the clustering algorithms in a hierarchical structure [41] as depicted in figure 3. Most of the clustering algorithms are divided into one of the two methods; hierarchical and partitioning clustering algorithms.

The classification of clustering algorithms are based on the several cross-cutting aspects of as described in [41].

- Agglomerative vs. Divisive: It refers to algorithmic structure and operation, since an agglomerative approach is a bottom-up construction while divisive

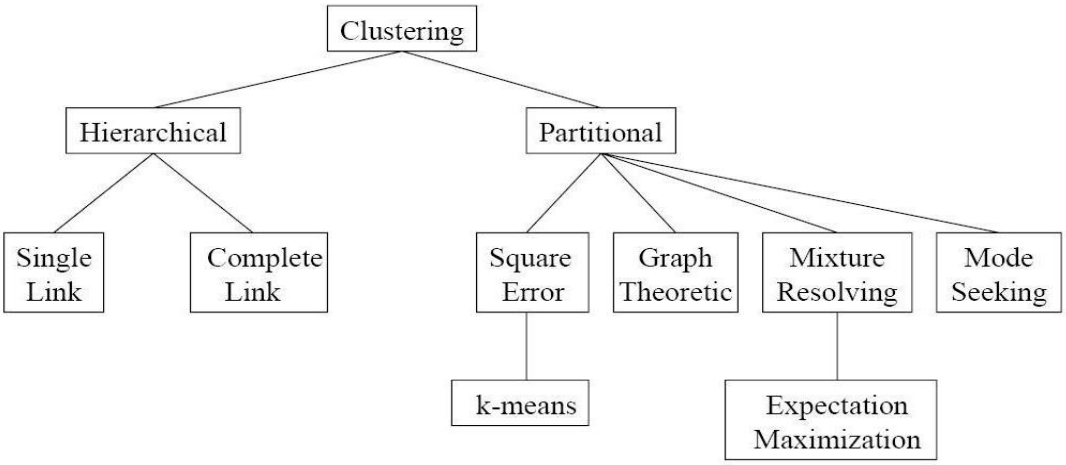

Figure 3: A taxonomy of clustering approaches [41].

is top-down approach. In general this aspect distinguish the hierarchical clustering algorithms.

- Monothetic vs. Polythetic: It refers to the different use of features in the process, either sequential or simultaneous. While most of algorithms are polythetic, Anderberg reported a simple monopthetic algorithm in [5].

- Hard vs. Fuzzy: This aspect refers to the membership of the data. A hard clustering assigns one data point to only one cluster while a fuzzy clustering assigns one to multiple clusters. Some of applications in fact require the fuzzyness.

- Deterministic vs. Stochastic: It is related to the optimal techniques the clustering algorithm is using. Most of partitioning clustering algorithms use either a deterministic objective function or a random search technique for optimization.

- Incremental vs. Non-incremental: The algorithm where the size of data can be increased is considered as incremental otherwise non-incremental.

3.2 Various Clustering Algorithms

Because of the different aspects of the clustering algorithms in 3.1, categorization of existing clustering algorithms are also various as we can see in the survey papers of clustering algorithms [41; 76; 7]. In this paper, we are mostly following the classification of Andreopoulos *et al.* [7] as it is most recent survey paper, and the classification is nicely structured.

Andreopoulos *et al.* [7] divided the whole clustering algorithms into 6 categories: Partitioning, Hierarchical, grid-based, density-based, model-based and graph-based clustering algorithms. We omit the discussion of grid-based algorithm. Instead, we add another category, evolutionary clustering algorithms, described in [41]. Note that this categorization is a soft classification as some of them can belong to several categorizes.

3.2.1 Partitioning Clustering Algorithms

Partitioning clustering methods are useful for the applications where a fixed number of clusters are required and Andreopoulos *et al.* [7] further divided it into numerical methods and discrete methods. Partitioning algorithms assign a number of data into k number of clusters [76]. K-means algorithm and Farthest First Traversal k-center (FFT) algorithm [35], K-medoids [49] or PAM (Partitioning Around Medoids), CLARA (Clustering Large Applications)[50], CLARANS (Clustering Large Applications Based Upon Randomized Search)[60] and Fuzzy K-means [34] belong to numerical methods according to [7].

In k-means algorithm[5], the number of clusters k is the user-specified parameter. With k number of initial mean vectors, k-means algorithm iteratively assign the objects into one of cluster whose center is the closest with it. The process continues until there is no other re-assignment or it depends on the user-specified threshold.

Despite of its simplicity and easy implementation, k-means has several drawbacks as well. For instance, there is no standard for a 'good' set of initial centers. Instead of several runs with random choices, Bradley and Fayyad [14] presents a refining k-means algorithms, where several preliminary k-means results can provide the initial points for the next run of the algorithm, so that it leads to a better local minimum points. Likas *et al.* [57] suggested a global k-means algorithm which includes a set of k-means processes with various number of clusters. They also suggested the extension of the parallel implementation of the algorithm. Instead of given k, Ball and Hall [10] proposed the technique of ISODATA where

the number of cluster k is estimated with merging and splitting processes. But it also involves another user-specified threshold for those processes.

Traditional k-means algorithm takes $O(Nkd)$ at each iteration where N is the number of data and d is dimension. Kanungo *et al.* [48] presented an efficient implementation of Lloyd's k-means algorithm which is the filtering algorithm using kd-tree data structure. In fact, utilization of kd-structure was proposed by Alsabti *et al.* in [4]. The algorithm [48] starts by storing all data points in a kd-tree, and maintains a subset of candidate centers. The candidate centers are filtered as they are passed to its children. As the kd-tree is constructed based on the data points, it does not need to be updated at each iteration, which saves the time overall.

Ng and Han [60] developed a new clustering method called CLARANS which is dynamic version of CLARA [50] applicable for a large data set by sampling inputs, for the application to spatial databases. CLARANS is based on randomized search to choose samples. The parallelization of CLARANS is available in [80].

Discrete methods include K-modes [39], Fuzzy K-modes [38] etc. Huang [39] provided k-modes algorithm which extends the k-means methods to categorical domains, as k-means only deal with numerical data sets. Many of data mining applications consists of categorical data, not numerical data. Therefore, converting these data into numerical to apply conventional K-means algorithm often reveal meaningless results. Therefore the author of this paper suggests K-modes algorithm using new similarity measurements for the categorical objects.

Previously, *k-prototypes* are proposed by Huang [40] in 1997, to cluster large data sets with mixture of numeric and categorical measurements. One of the problems using k-prototypes algorithm is to choose a proper weight for the categorical measurements. The k-modes algorithm, therefore, is improved version of k-prototypes, as it only takes categorical attributes for the measurements. If there is any numeric attribute, then it can be converted into categorical attribute. The centers are called modes in k-modes algorithm, and the modes are updated with frequencies. The advantages of k-modes algorithm over k-means are efficiency and scalability. It is faster than k-means as it does not need the distance of each data, but need the frequencies. Also it is scalable as it only need to update the frequencies of a new data.

K-prototypes [40] is a mixed of discrete and numerical clustering methods. The algorithms except K-means are in fact modification of K-means algorithms with various purposes.

Recently, a non-negative matrix factorization (NMF) method is applied for clustering micro array data as we can see in [52]. With the sparse coefficient matrix factored from the original data set, the membership of each data can be

assigned one of the clusters.

3.2.2 Hierarchical Clustering Algorithms

Hierarchical clustering algorithms divide the data into a tree of nodes, where each node represents a cluster [7]. Hierarchical clustering algorithms are often divided into two categories based on their methods or the purposes: Agglomerative vs. Divisive; Single vs. Complete vs. Average linkage. In some applications including bioinformatics, hierarchical clustering methods are more popular as natures can have various levels of subsets. But hierarchical methods are slow, errors are not tolerable and information losses are common when moving the levels. Like partitioning methods, hierarchical methods consists of numerical methods and discrete methods. BIRCH [79], CURE [32] and Spectral clustering [74] are numerical methods while ROCK [65] and LIMBO [8] are discrete methods.

The paper [79] proposes a Balanced Iterative Reducing and Clustering using Hierarchies (BIRCH) as a new data clustering method using clustering feature (CF)-tree data structure, and suggests that this method can be easily parallelized. Because of growing number of data and and concerns for the robustness to outliers, BIRCH is developed. The CF-tree is a height-balanced tree designed to store the summaries of the input. The tree structure stores the clustering information while requires less storage. After the CF-tree is constructed, an agglomerative hierarchical clustering is applied to perform global clustering, and the time complexity is linear.

In [79], the authors provided CURE clustering which abbreviate Clustering Using Representatives. This is one of agglomerative hierarchical clustering algorithm and it deals with large scale data sets. While centroid-based hierarchical clustering algorithm such as BIRCH [79] have the restriction in the shape of cluster, CURE is not limited to the shapes nor sizes of the clusters. CURE uses a set of well-scattered points to form each cluster, then the clusters are further shrunk toward the cluster center with an adjustable parameter to limit the effects of outliers. To reduce computational complexity, CURE use random sampling and partition techniques.

3.2.3 Evolutionary Clustering Algorithms

Evolutionary approaches use evolutionary operators (such as selection, recombination and mutation) and a population to obtain the optimal partition of the input data [41]. The first step of these algorithms is to choose a random population of

solutions, which is usually a valid partition of data with a fitness value. As a next step, they use the evolutionary operators to generate the next population. A fitness function, which determines a population's likelihood of surviving into the next generation, is applied to the solutions. The two steps are repeated until it finds the required solution meeting some conditions. Generic algorithms (GA)[37] and evolution strategies (ES)[33] belong to this category.

3.2.4 Density-based Clustering Algorithms

Density-based clustering algorithms use a local density standard. Clusters are dense subspaces separated by low density spaces. One of the examples is DBSCAN introduced in [26]. DBSCAN was developed to cluster large-scale data sets in the context of data mining. It requires that the density in a neighborhood for a data should be high enough if it belongs to a cluster. A new cluster from one data point is created by including all points in its neighborhood. The threshold of neighborhood of a data point is user-specific. DBSCAN uses $R*$-tree structure for more efficient queries. The authors showed the effectiveness and efficiency of DBSCAN using synthetic data and SEQUOIA 2000 benchmark date as well.

Other density-based clustering algorithm include CLIQUE [2] and HIERDENC (Hierarchical Density-based Clustering)[6], etc.

3.2.5 Model-based Clustering Algorithms

Model-based clustering uses a model which is often derived by a statistical distribution. AutoClass [17] is the most popular example of this category. This paper [17] is about AutoClass system which is based on Bayesian method for determining optimal classes in large data sets. In the category of [76], it belongs to the mixture densities-based clustering as well. In the probabilistic view, data points are assumed to be generated according to probability distributions. Combining it with clustering point of view, each cluster is represented with different probability distributions, (different type or different parameters). The algorithms belonging to this category mostly use expectation-maximization(EM)approach. It first initialize the parameters of each cluster. It computes the complete data log-likelihood in e-step and select new parameters maximizing the likelihood function. AutoClass considers a number of families of probability distributions including Gaussian, Poisson and Bermoulli, for different data types. A Bayesian approach is used in AutoClass to find out the optimal partition of the given data based on the prior probabilities.

3.2.6 Graph-based Clustering Algorithms

According to the paper [7], graph-based clustering algorithms were applied to interactomes for complex prediction and to sequence networks. Junker and Schreiber [45] reviewed some of graph-based clustering algorithms for bioinformatics applications. Clustering algorithms are very useful for biological networks such as Protein-Protein Interaction (PPI), Transcriptional Regulatory Network and Metabolic Networks. As they can be represented as a graph, the chapter of *Network Clustering* in this book uses graph structure for the network. Then it reviews Clique-based and Center-based clustering techniques for small data sets.

For the large data sets, it refers some techniques including distance k-neighborhood, k-cores and quasi-cliques as well.

3.3 Clustering Algorithms and Applications

Clustering analysis has been applied in a variety of applications. For example, as in the engineering field including machine learning, artificial intelligence, pattern recognition, mechanical engineering and electrical engineering; computer science researches of web mining, spatial database analysis, textual document collection and image segmentation; life and medical science consisting of genetics, biology, microbiology, paleontology, psychiatry, clinic and pathology; earth science; social science; and economics etc. For bioinformatics applications, [43; 45] listed a number of algorithms with specific applications.

4 Parallel Implementation of Clustering Algorithms

Most of the clustering algorithms are computationally expensive as it involves non-specified number of iterations and validation steps. Therefore, parallel implementation of the clustering algorithms are somewhat inevitable. Recently parallel clustering algorithms have been developed and implemented for practical applications. [16; 3; 71; 75; 30; 55; 56; 1; 16; 59; 9; 66; 68; 12; 58; 28; 61; 62; 77; 80; 63; 23; 44; 29; 72; 47; 15; 19; 25]

Talia [72] identified three main strategies in the parallelism used in data mining algorithms as the followings. 1) Independent parallelism where each processor accesses to the whole data to operate but do not communicate each other. 2) Task parallelism where each processor operate different algorithms on the partitioned or on the whole data set. 3) SPMD (Single Program Multiple Data) parallelism

where multiple processors execute the same algorithm on different subsets and exchange the partial results to cooperate each other.

Most of parallel clustering algorithms follow the combinations of task and SPMD parallelism with master-slave architecture. In this section, we observe some of parallel implementations of existing clustering algorithms with those strategies, based on the categorization discussed in the section 3, partitioning, hierarchical, evolutionary, dense-based, model-based and graph-based parallel clustering algorithms.

4.1 Parallel Partitioning Clustering Algorithms

K-means clustering algorithm is the most popular partitioning clustering algorithms. In 1991, the paper [66] describes partitioning clustering algorithms using squared error metric for SIMD (Single Input Multiple Data) hypercubes. The square error for each cluster is defined as the sum of square distance between each member in the class and the center, called intra-distance. Therefore the objective function in the square error clustering algorithm is to minimize the sum of all intra-distances. Traditional sequential algorithm starts with initial set of k clusters, and move each pattern (or data) to a cluster if it minimizes the square error. With single processor, one iteration takes $O(nmk)$. Li and Fang [55] introduced the algorithm which takes $O(klognm)$ for this one pass, using SIMD hypercube network. For further improvement, this paper [66] uses subhypercube concept for broadcasting data or computing data. Therefore, the time is reduced into $O(k+lognm)$ with nm processors, and in $O(lognmk)$ with nmk processors.

On the other hand, without specific topology of connection, some parallel partitioning clustering methods are implemented using master/slave message passing architecture as shown in [44; 47; 23; 28; 80]. Some of the parallelization comes from the distance computation in parallel [47] or the cluster assignment in parallel[44].

In the paper by *Judd et al.* [44], algorithmic enhancements are described which reduces large computational efforts in mean square-error data clustering. These improvements are incorporated into a parallel data-clustering tool. The basic CLUSTER and Parallel-CLUSTER (P-CLUSTER) algorithm was first reviewed in this paper, then provided the results of tasks on NOWs (network of workstations) platforms. The basic CLUSTER program consists of two components of a *K-means pass* and a *forcing pass*. In a *K-means pass*, after the first clustering is created, the subsequent clusterings are created based on the previous clustering. Then in a *forcing pass*, another set of clusters by merging existing

clusters is created, two at a time, to determine whether a better clustering is being achieved. The basic P-CLUSTER algorithm uses a client-server(or master-slave) process configuration. The data set is partitioned into blocks by the server process, sending the initial centroid list and job assignments to each of the clients. Each client process takes each block of data set to assign it to the appropriate cluster. The server then collect the information of each clients and calculates new centers to send them back to clients. By observing that after a small number of passes, relatively few changes occur, the authors of this paper proposes three techniques to reduce the number of distance calculations to reduce the total computational time. One is to pre-compute a set of radii for each centroid, called sphere of guaranteed assignment. The second technique is using maximum movement effect. That is, if there is no changed membership for a pattern then all computation for the pattern is avoided. Lastly, partial sum technique is used to reduce additional computations for floating point addition: they maintain a set of partial sums associated with each block of data. Patterns that change assignment are subtracted from the old set of sums and added to the new set. So a computation is carried out only when a memberships change takes place.

The simple parallel implementation of k-means algorithms was applied in [47]. Using the intra-cluster, (inner summation) and inter-cluster (outer summation) criterions, the parallel K-means algorithm is developed on the message-passing model of a network of workstations (NOWs).

The master process first randomly form k equal subsets and send each to each k slaves. Each slave process compute the mean of each subset then broadcast it to other slaves. Each slave compute the distance of each data from each mean, then choose the closest vector members for each k subset, which is also broadcast. Then for the new subset by collecting those closest members, and send it back to the master process.

The total time estimated with this algorithm is $O(2R_pk|P|)$ where R_p is the number of iterations and $|P|$ is the size of subset and the total space complexity is $O(n)$, where n is the number of data. In a similar way, the authors of the paper [23] proposed a parallel implementation of the k-means clustering algorithm using the message passing model, based on the single program multiple data (SPMD) model. Message-passing model assumes that p processors are connected with its own local memory. After initial k centers are selected, n data points are divided into p blocks. Observing that each processor can carry out the distance calculation in parallel if the centers are available to them, each processor deals with only n/p number of data points to assign to clusters. Each iteration consists of an asynchronous computation phase and a synchronous communication phase. In

conclusion, the paper shows a nearly linear speedups with multiple processors, and it is also scalable with increasing data points. The method proposed by Forman and Zhang [28] is also similar.

Not only the parallel k-means, but also other partitioning clustering methods are implemented. For example, Zhang *et al.*[80] introduced a parallel CLARANS (Clustering Large Applications Based Upon Randomized Search) algorithms using PVM (Parallel Virtual Machine). PVM supports a complete message-passing model and make a collection of distributed computers function as a single machine. In the PVM, all the computers are working in master/slave mode. Master program can assign tasks to other slaves and the communication between computers is based on the message-passing. CLARANS algorithm randomly draw a sample at each iteration, and the k-Medoids are chosen from this sample. In PVM, the master program assigns each tasks to k slaves, and each slave randomly select a neighbor of each center and compute cost differential of the two nodes. With the smaller cost, the center can be replaced. This parallel version of CLARANS is called ICLARANS (Improved CLARANS) and it outperforms CLARANS in terms of time complexity, and scalability.

4.2 Parallel Hierarchical Clustering Algorithms

The parallelization of hierarchical clustering algorithms use both of the network topology of processors and the splitting data accessing. The hierarchical clustering algorithms with single-linkage metric utilizing the MST(minimum spanning tree) structure for clustering can be paralleled with hypercube network of processors [62]. The time of calculating data distance and the updating step can be reduced with parallel access by multiple processors. The PBIRCH [30] utilize the message-passing method to distribute the data to each processor to build its own data structure. Then they exchange the information to update the clusters.

Naive hierarchical clustering algorithms in sequential take $O(n^3)$ where n is the number of data. The paper by Ranka and Sahni[56] presents parallel hierarchical clustering algorithms on SIMD machine in 1990. The parallel version of single-link and complete-link hierarchical clustering algorithms proposed in this paper use an alignment network, especially shuffle-exchange network. The data matrix of $n \times m$ with m number of features (pattern matrix or proximity matrix) is first stored in the central memory then it is distributed to p number of processor modules. With two storage rules proposed in the paper, the data matrix is stored in rows, columns, folds partial-row-pairs and partial-column pairs which is accessed in parallel. As a result, if $n < Q$, then the time becomes $O(nlogp)$ and $O(n^2logp)$

for the simple-link and complete-link respectively. If $n \geq p$ then they become $O(n^{\ulcorner} n/p^{\urcorner} logp)$ and $O(n^{2^{\ulcorner}} n/p^{\urcorner} logp)$. In other word, how to distribute the data of matrix into the processors decides the computational efficiency. With different metrics, (single, complete or average linkage), sequential hierarchical algorithms can reduce the time to $O(n^2)$. The algorithms with single-linkage metric first construct MST tree to form a number of clusters. The parallel implementation of such algorithms focuses on reducing time for MST construction.

Olson [62] described this kind of parallel algorithms for agglomerative hierarchical clustering tasks in 1995. The author first reviewed the sequential algorithms with two distance metrics. The first metric is graph metrics. They determine inter-cluster distances according to the cost functions of the edges of members in the two clusters, including single, average and complete link. The other metric is geometric metrics which determines intra-cluster distances between the cluster center and the members including centroid, median and minimum variance, etc.

Sequentially, clustering with the single link metric is mostly done by finding the Euclidean minimal spanning tree and the time is $O(n^2)$ as proved in [78]. For the median and centroid metrics, Day and Edelsbrunner [22] showed that it takes overall $O(n^2)$ time as well. In general case, if the modified inter-cluster distance can be updated in a constant time, the clustering takes $O(n^2 logn)$ time as shown in [22]. Previous parallel clustering algorithm with single linkage metric on SIMD array processor was discussed by Rasmussen and Willett [67]. They parallelized SLINK algorithm [70], Prim's minimal spanning tree algorithm [64] and Ward's minimum variance method [73]. Li and Fang [55] parallelized hierarchical clustering using the single link metric on hypercube network and butterfly, by parallelizing the Kruskal's minimal spanning tree algorithm [54]. The n-nodes hypercube network takes $O(nlogn)$ time while the n-nodes butterfly takes $O(nlog^2 n)$ time. However, these times are incorrect as they did not count the step of updating distances of clusters, which will increase the time to $O(n^2)$. The relaxed heap introduced by Driscoll et al. [25] is used for parallel implementation of Dijkstra's minimal spanning tree [24] and reduces the time into $O(nlogn)$ with $m/(nlogn)$ processors on a PRAM. Bruynooghe [15] describes a parallel nearest neighbors clustering algorithm for a parallel supercomputer, using message-passing master/slave model.

Then the author[62] proposed the following algorithm for each of the metrics on PRAMs and butterflies. With single-linkage, centroid and median metrics, finding MST techniques are used. According to the algorithm introduced in this paper, the inter-cluster distance and nearest neighbor arrays are obtained in $O(n/p)$ at each step, where n is the number of data and p is the number of

processors. As the number of steps is $O(n)$, total time would be $O(n^2/p)$. With minimum variance, average and complete link metrics, each processor store the location of each cluster center and find the nearest neighbor in $O(logn)$ with $O(n/logn)$ processors, leading total $O(nlogn)$ time. For the general metrics, create a priority queue in $O(n)$ time on a processor and in $O(logn)$ time on a n processor PRAM. After each iteration the priority queue is updated, total n steps lead $O(nlogn)$ time for updating queue.

In 1999, Kantabutra and Couch [21] provided parallel algorithms for hierarchical clustering methods and apply it to split decomposition in graphs. The paper observed that both of a single linkage method and the algorithm determining overlap components can be the sub-procedures in an efficient parallel algorithm for split decomposition. Therefore they provided a parallel algorithm for the single linkage method, computing a minimum spanning tree (MST) with multiple processors. The overlap components can be determined in linear time sequentially and in $O(logn)$ with $O(n)$ processors using parallel prefix computation. Finally, an efficient parallel split decomposition algorithm is presented using the above two algorithms.

Garg *et al*. [30] described a parallel BIRCH [79] clustering algorithms. PBIRCH algorithm is applied to the SPMD model with message-passing. First, the data is divided equally into the processors. Each processor constructs each CF-tree. Initially one processor chooses k initial centers for k-means algorithm, broadcast it to all processors. Then each processor use the initial centers to cluster its own CF-tree locally. The centers of the local clusters are exchanged, then the global centers are computed. Now having k global centers, each processor recomputes the clusters and the processes are repeated until it converges. If new data are entered they are cyclically distributed as well. In other words, PBIRCH applied k-means algorithm to distribute data to each sub-processors.

4.3 Parallel Evolutionary Clustering Algorithms

The paper [9] described parallel ES algorithm. Evolutionary strategy (ES) is one of the best-known evolutionary techniques developed in 1981 by Schwefel [33]. It is population-based stochastic optimization technique tool, and is modeled using biological evolution concepts with selection, recombination and mutation. According to [36], if we set μ be the number of parents, λ be the number of offspring, the ES model is represented as (μ, λ)-ES. The early (μ, λ)-ES model has only one parent and one offspring , that is, (1+1)-ES. Later multi-membered ES with $(\mu + 1)$-ES where $\mu > 1$ is proposed by introducing recombination operator.

The $(\mu+\lambda)$-ES and (μ, λ)-ES where $\mu > 1$ and $\lambda > 1$ were introduced for parallel computation of ES. For selecting parents, $(\mu + \lambda)$-ES uses all individuals of the next population while (μ, λ)-ES uses only the given offsprings.

After brief description of how ES can be used to optimize a function, Babu and Murty [9] discussed clustering with ES. In clustering, two types of objective functions are described: centroid type and non-centroid type. The centroid type of objective function, where objective function is based on the centroid of clusters, is used to find optimal cluster centers. ES is used to optimize this object function, in case of hard and fuzzy clustering as well. The non-centroid type of objective function is applied to discrete optimization problems. The entire mutation is controlled by global mutation and a solution string is an ordered sequence of T discrete parameters, each of which is the cluster label of a data. Using ES, the objective function takes a solution string as input and outputs the clusters based on the assignment.

Parallel ES algorithm is described in terms of the master/slave model in [9]. Assume that the number of processors are $O(\mu)$. Then there is a master processor that maintains a number of parent solutions and processes selection operation and passes parent solutions to child(slave) processors. The child processors perform mutation and recombination processes and evaluate with the objective function. New solutions are sent back to the master process and it repeats the procedures until a near-optimal or optimal solution is found.

4.4 Parallel Density-based Clustering Algorithms

DBSCAN is one of density-based clustering algorithms [26], which can have arbitrary shape of clusters and efficient enough to be used for large spatial data sets as it uses spatial access methods. The authors [77] introduced PDBSCAN, a parallel DBSCAN algorithm using shared-nothing architecture with multiple computers interconnected through a network. As a data structure, the dR*-tree is introduced for a distributed spatial index.

PDBSCAN algorithm with master-slave configuration has three steps. First, divide the input data into several partitions, and distribute them to available machines, which is a data replacement step. Each partition is concurrently clustered using DBSCAN. The last step is merging the local clusters into global clusters for the whole data. The parallel model is master/slaves model. A master program starts a clustering slave on each available machine and distributes the data set to the slaves. Each slave clusters calculate the data assigned to it locally. The $R* -$ data structure for index gives an efficient access of data and the commu-

nication between processors is message-passing model. The results suggests that PDBSCAN offers nearly linear speedup.

4.5 Parallel Model-based Clustering Algorithms

Foti *et al.* [29] illustrates the design and implementation of parallel version of AutoClass system [17] with SPMD and call it a P-AutoClass. The SPMD parallel can be used in AutoClass by dividing the input into the processors and updating parameters for the classifications. It avoids replicating the whole input on each processor and also avoids balancing problems as each processor perform the same function on equal size of data. In addition, not much exchanges are necessary as they are locally executed. The process is similar to the parallel k-means algorithms in [44] as both of algorithms distribute a block of data set into each slave process to update the membership.

4.6 Parallel Graph-based Clustering Algorithms

In the paper [1], parallel processors are used for ordering the graph vertices for efficient implementation. The paper [1] described one of the parallel graph theory-based clustering algorithm used for non-hierarchical clusters. It studied graph clustering on bipartite and chordal graphs and illustrated two types of clustering problems: one is a partition of the vertices of a graph with minimum diameters of subgraphs; the other one is a subset of fixed number of vertices which induces maximum possible number of edges. Putting main interests onto the first problem, the author proves that the partitioning problem is NP-complete on bipartite and chordal graphs. But, this paper also described linear time sequential algorithms on bipartite permutation and interval graphs. By proving and giving efficient parallel algorithm for constructing a certain ordering of the vertices on the biconvex graphs, called a biconvex straight ordering, the author presented efficient parallel algorithms for the general problem on biconvex and interval graphs. At last, the paper also provided an efficient parallel algorithm for solving the second problem on a class of unit interval graphs.

In some of the graph-based algorithm, each processor is assigned to each node. As illustrated in [12] traditionally, clustering in the network is performed in two steps: clustering setup by picking cluster-heads and clustering maintenance by collecting the neighbors of cluster-heads. However existing clustering algorithms are based on the assumption that the nodes are not moving in the process. Mobility-adaptive algorithm is introduced for mobile networks, whose decision is

based on the local topology of head's neighbors with the largest degree. In this paper [12], a distributed clustering algorithm (DCA) and a distributed mobility-adaptive (DMAC) algorithm are proposed. The authors partition the nodes in a fully mobile network or ad hoc network, and organize a hierarchical structure. The DCA generalizes the common clustering in network, and it is executed at each node, each of which communicate with others with message-based approach. It selects the cluster-heads based on the new weight-based criterion, and each node decides its role with its current one hop neighbors as opposed to one and two hop neighbors by previous algorithms.

As DCA algorithm is suitable for the quasi-static networks, the author propose DMAC for highly mobile networks. Without the assumption that during the clustering process the nodes are not moving, DMAC lets each node to properly react to the failure of a link with another node on top of the messages from others.

5 Applications of Parallel Clustering Algorithms

We observe that the parallel clustering algorithms can be applied to any of applications using clustering algorithms for efficient computing. In this section, we review several papers that implemented parallel clustering algorithms for specific applications.

One of the common application of clustering algorithms is image segmentation. Khotanzad and Bouarfa presented a non-parametric parallel clustering algorithm in [51] for the task of unsupervised image segmentation. It builds a multi-dimensional histogram of the feature vectors to see the distribution of feature vectors and approximate the density of the data. Each peak is considered as a cluster. The process of identifying clusters are automatic as it proposes systematic methods for choosing an appropriate histogram cell size. After the clustering, the segmentation task is followed and they are implemented in a SEQUENT parallel computer. The parallel computation is done using message-passing protocol as well.

A parallel bottom-up clustering algorithms were applied to partition circuits in VLSI in the paper [20]. Unlike many top-down partitioning algorithms for layouts, the bottom-up layout clustering algorithms are useful as circuit sizes increase. The parallel clustering algorithm was implemented by dividing the graph into multiple processors. Each processor forms clusters of the graph, occasionally swaps parts so that cliques to be found. The bottom-up clustering tasks also are used as preprocessing for partitioning as each cluster is then assigned to each

node.

Bandyopadhyay and Coyle presented a distributed and randomized clustering algorithm to layout the sensors in a wireless sensor network [11]. On top of the clustering network sensors, the authors organized the cluster-heads hierarchically, so that it saves the energy more. It assumes the sensors in a network are simple and all the sensors communicate at a fixed power level. Using the experimental fact that a network with one level of clustering has an optimal number of cluster-heads minimizing the energy, this paper proposed an analysis to obtain the optimal number of cluster-heads at each level of clustering.

Anantharaman *et al.* [46] developed a software called PaCE which is parallel and fast clustering method of large-scale EST (Expressed Sequence tags) in gene identification. ESTs are sequences at most a few hundred base pairs determined by single-pass sequencing of the 5' or 3' ends of cDNA. EST representations provide a dynamic view of genome content and expression. With the huge EST data library, more than 5 million human ESTs, the primary goal in ESTs is to derive sets of unique genes from the data which represent the transcriptome of each species. Therefore, the first step would be clustering ESTs to remove redundancy in EST sets. The authors of this paper developed a parallel software program, named PaCE, which clusters ESTs into the same gene or paralogous genes. Then each cluster is further processed with an assembly tool for generating one consensus sequence per putative transcript. The memory with PaCE in $O(n)$ and the input size of the assembly tools is now reduced to the size of each cluster.

The PaCE implements one of the bottom-up hierarchical clustering in parallel. That is, each pair of clusters are grouped recursively. The PaCE software has two phases: GST(generalized suffix tree) construction as a preprocessing phase, and pair generation, pairwise alignment and EST cluster management as the clustering phase. That is, a distributed representation of the GST is first built in parallel. A master processor maintains and updates the EST clusters and the rest of processors generate batches of promising pairs and perform pairwise alignment on selected promising pairs. Master process is also responsible for selecting promising pairs at each step.

Kraj *et al.* illustrated the parallel k-means algorithm, called paraKMeans, for general laboratory use in [53]. The parallel computation is performed to the cluster assignment by dividing the data set into sub-processors to assign the memberships. The ParaKMeans is a software program that can be used for practical applications.

6 Applying Clustering Algorithms for Parallelization

Clustering algorithms and parallelization is closed related because they are useful applying the 'divide and conquer' strategy in algorithms so that they reduce the time complexities. Therefore, while some parallel techniques used for efficient clustering algorithms, some clustering algorithms used for parallel tasks as well. In this section, we provide a number of literatures which used clustering algorithms for efficiency.

Fukunaga and Narendra (1975) used ISODATA partitioning clustering algorithm [42] to decompose the patterns for efficient computation of nearest neighbors.

Sadayappan and Ercal addressed the task-to-processor mapping problem, which is essentially NP complete [69]. In other words, clustering task is used for distributing tasks for parallel computations. Assuming that the multiprocessors are connected with a hypercube interconnection network, two cluster-based mapping strategies are introduced: a nearest neighbor approach and a recursive clustering.

There are two distinct settings for the task-mapping problem: Task Precedence Graph (TPG) model and Task Interaction Graph (TIG) model. Each model can be represented as a graph. The vertices in a TPG graph represent tasks, and directed edges represent execution dependencies. Vertex weight is the execution time and edge weight is the amount of data to be communicated between vertices. Since the parallel program modelled with TIG graph does not capture the execution dependencies between tasks, the TIG model is more suitable for more complex parallel programs. The parallel computer in the paper [69] is also represented as a graph where the vertices represent the processors and the edges represent the communication link, focusing on static mapping with TIG model. The nearest-neighbor mapping strategy maps any pair of nodes sharing an edge in TIG to nearest-neighbor processors. It starts from an initial mapping of clustering nodes of TIG onto the clusters of processors holing nearest-neighbor properties. Then it modifies the mapping by reassigning with a boundary-refinement process to improve the mapping cost defined in some way. The recursive-clustering strategy has two phases: cluster formation and processor allocation. In the cluster formation phase, the TIG is first partitioned into as many clusters as the number of processors. It is done with top-down fashion, recursively. Then the clusters are allocated to some processors in order to minimize the total inter-processor communication cost.

Chen *et al.* [18] used two clustering algorithms for efficient sequence motif discovering problems . Fuzzy-C means algorithm is exploit to divide the large data set into smaller subsets so that parallel computation is possible. Each processor then run k-means clustering algorithms to find out protein sequence motifs. Protein motif sequence is a repeated pattern in a set of protein sequences, which are potentially important biologically and chemically. However, with large data set, discovering motifs becomes hard as the task of grouping the proteins with similar patterns should proceed. Hence, in this paper, a number of protein sequence which represent its protein family information is segmented by shifting 9 window size, and they are clustered first to find consensus at each cluster. The consensus sequence at each cluster is now considered a potential sequence motif.

Clustering task is performed to divide the data sets into parallel for efficient computation of clustering algorithm itself. For example, PBIRCH algorithm [30], which is a parallel clustering algorithm discussed in section 4 uses the clustering algorithm as a preprocessing for an efficient assignment of sets as well.

7 Conclusion

In this study, we reviewed the papers about clustering algorithms and the corresponding parallel clustering algorithms. Clustering algorithms are categorized to five classifications: partitioning, hierarchical, evolutionary approach, dense-based, model-based and graph-based algorithms. Partitioning and hierarchical algorithms are the most popular clustering algorithms and many of parallel version of the algorithms also have been developed.

Parallelism in the clustering algorithms have been used both for efficient clustering strategy and for the efficient distance-computation.

K-means algorithm is the representative partitioning clustering algorithm and the parallelization was implemented mostly in message-passing model. For efficient data communication, some of the algorithms used the properties of specific interconnection network topology including hypercube or butterfly while others use master-slave configuration to parallelize the distance computation or the cluster assignment.

Hierarchical clustering algorithms is more expensive than others in computation as it involves computing the level of clustering as well. Parallel version of those algorithms with single-linkage metric use hypercube network to reduce the computation of MST, or use message-passing architecture to efficiently compute the similarities and to update the memberships of each data.

Other clustering algorithms also use similar implementation schemes; master-slave configuration using message-passing model. In summary, parallel clustering algorithms exploit the combinations of task implementation and SPMD parallelism introduced in [72].

We also reviewed some of applications used the parallel clustering algorithms and observed the clustering algorithms have been used for other parallel algorithms as well. In other words, parallelism and clustering task is closely related since it can divide the large data set into smaller subsets.

Parallel clustering algorithms are practical approaches for many of applications of large data sets. However, there are several architectural issues which the parallel clustering algorithms should consider when implemented as Talia [72] pointed out. The issues including the different implementation for distributed memory versus shared memory machine; the configuration of interconnection network topology of processors; how to optimize communications; load balancing; spatial memory optimization and the I/O impact.

References

[1] Nesrine Abbas. Graph Clustering: Complexity, Sequential and Parallel Algorithms, 1995.

[2] R. Agrawal, J. Gehrke, and D. Gunopulos. Automatic subspace clustering of high dimensional data for data mining applications. In *Proceedings of the ACM-SIGMOD'98 International Conference on Management of Data*, pages 94–105, Seattle, WA, 1998.

[3] S.G. Akl. *Parallel Computation: Models and Methods*. Prentice Hall, 1997.

[4] Khaled Alsabti, Sanjay Ranka, and Vineet Singh. An Efficient k-Means Clustering Algorithm. In *Proceedings of IPPS/SPDP Workshop on High Performance Data Mining*, 1998.

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

[6] B. Andreopoulos, A. An, and X. Wang. Hierarchical density-based clustering of categorical data and a simplification. In *Proceedings of the 11th Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 11–22, Nanjing, China, 2007. Springer LNCS.

[7] Bill Andreopoulos, Aijun An, Xiaogang Wang, and Michael Schroeder. A roadmap of clustering algorithms: finding a match for a biomedical application. *Brief Bioinform*, pages bbn058+, February 2009.

[8] P. Andritsos, P. Tsaparas, and R. Miller. LIMBO: Scalable clustering of categorical data. In *Proceedings of the 9th international conference on Extending Database Technology*, 2004.

[9] G. Phanendra Babu and M. Narasimha Murty. Clustering with evolution strategies. *Pattern Recognition*, 27(2):321 – 329, 1994.

[10] G. Ball and D. Hall. A clustering technique for summarizing multivariate data. *Behav. Sci.*, 12:153–155, 1967.

[11] Seema Bandyopadhyay and Edward J. Coyle. An energy efficient hierarchical clustering algorithm for wireless sensor networks. volume 3, pages 1713–1723 vol.3, March-3 April 2003.

[12] S. Basagni. Distributed clustering for ad hoc networks. pages 310–315, 1999.

[13] C.M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[14] P.S. Bradley and U.M. Fayyad. Refining initial points for k-means clustering. *Proceedings of the Fifteenth International Conference on Machine Learning*, pages 91–99, 1998.

[15] M. Bruynooghe. Parallelism implementation of fast clustering algorithms. In *Proc. int. Symp. On High Performance Computing*, pages 65–78, 1989.

[16] Rajkumar Buyya. *High performance cluster computing / edited by Rajkumar Buyya* . Prentice Hall PTR, Upper Saddle River, N.J. :, 1999.

[17] Peter Cheeseman and John Stutz. Bayesian classification (AutoClass): theory and results. pages 153–180, 1996.

[18] B. Chen, P. C. Tai, R. Harrison, and Y. Pan. FGK model: A Efficient Granular Computing Model for Protein Sequence Motifs Information Discovery. In *The IASTED International Conference on Computational and Systems Biology*, pages 56–61, 2006.

[19] P. D. Coddington. Parallel cluster algorithms. In *Nuclear Physics B, Proceedings Supplements, 20:76–79*, 1991.

[20] Jason Cong and M'Lissa Smith. A parallel bottom-up clustering algorithm with applications to circuit partitioning in VLSI design. In *DAC '93: Proceedings of the 30th international conference on Design automation*, pages 755–760, New York, NY, USA, 1993. ACM.

[21] Elias Dahlhaus. Parallel Algorithms for Hierarchical Clustering and Applications to Split Decomposition and Parity Graph Recognition. *Journal of Algorithms*, 36(2):205 – 240, 2000.

[22] William H. Day and Herbert Edelsbrunner. Efficient algorithms for agglomerative hierarchical clustering methods. *Journal of Classification*, 1(1):7–24, December 1984.

[23] Inderjit S. Dhillon and Dharmendra S. Modha. A Data-Clustering Algorithm on Distributed Memory Multiprocessors. In *Revised Papers from Large-Scale Parallel Data Mining, Workshop on Large-Scale Parallel KDD Systems, SIGKDD*, pages 245–260, London, UK, 2000.

[24] E.W. Dijkstra. A note on two problems in connection with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[25] James R. Driscoll, Harold N. Gabow, Ruth Shrairman, and Robert E. Tarjan. Relaxed heaps: an alternative to Fibonacci heaps with applications to parallel computation. *Commun. ACM*, 31(11):1343–1354, 1988.

[26] Martin Ester, Hans peter Kriegel, Jorg S, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise . pages 226–231. AAAI Press, 1996.

[27] B. S. Everitt. *Cluster analysis*. Heinemann Educational [for] the Social Science Research Council, London :, 1974.

[28] George Forman and Bin Zhang. Distributed data clustering can be efficient and exact. *SIGKDD Explor. Newsl.*, 2(2):34–38, 2000.

[29] D. Foti, D. Lipari, Clara Pizzuti, and Domenico Talia. Scalable Parallel Clustering for Data Mining on Multicomputers. In *IPDPS '00: Proceedings of the 15 IPDPS 2000 Workshops on Parallel and Distributed Processing*, pages 390–398, London, UK, 2000. Springer-Verlag.

[30] Ashwani Garg, Ashish Mangla, Neelima Gupta, and Vasudha Bhatnagar. PBIRCH: A Scalable Parallel Clustering algorithm for Incremental Data. *Database Engineering and Applications Symposium, International*, 0:315–316, 2006.

[31] A. Gersh and R.M. Gray. *Vector Quantization and Signal Compression*. Kluwer Acad. Press, 1992.

[32] Sudipto Guha, Rajeev Rastogi, and Kyuseok Shim. CURE: an efficient clustering algorithm for large databases. *Information Systems*, 26(1):35 – 58, 2001.

[33] Eldon R. Hansen. Numerical Optimization of Computer Models (Hans-Paul Schwefel). *SIAM Review*, 25(3):431–433, 1983.

[34] F. H*ö*pner, F. Klawonn, and R. Kruse. *Fuzzy Cluster Analysis: Methods for Classification, Data Analysis, and Image Recognittion*. Wiley, New York, NY., 1999.

[35] D. Hochbaum and D. Shmoys. A best possible heuristic for the k-center problem.

[36] Frank Hoffmeister and Thomas Bäck. Genetic Algorithms and Evolution Strategies - Similarities and Differences. In *PPSN I: Proceedings of the 1st Workshop on Parallel Problem Solving from Nature*, pages 455–469, London, UK, 1991. Springer-Verlag.

[37] J.H. Holland. *Adaption in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, 1975.

[38] Z. Huang and M. Ng. A fuzzy k-modes algorithm for clustering categorical data. In *IEEE Trans Fuzzy Syst*, pages 446–452, 1999.

[39] Zhexue Huang. A Fast Clustering Algorithm to Cluster Very Large Categorical Data Sets in Data Mining. In *In Research Issues on Data Mining and Knowledge Discovery*, pages 1–8, 1997.

[40] Zhexue Huang. Clustering large data sets with mixed numeric and categorical values. In *In The First Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 21–34, 1997.

[41] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM Comput. Surv.*, 31(3):264–323, 1999.

[42] Anil K. Jain and Richard C. Dubes. *Algorithms for clustering data*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1988.

[43] Neil C. Jones and Pavel A. Pevzner. *An Introduction to Bioinformatics Algorithms*. MIT Press, 2004.

[44] Dan Judd, Philip K. McKinley, and Anil K. Jain. Large-Scale Parallel Data Clustering. *IEEE Trans. Pattern Anal. Mach. Intell.*, 20(8):871–876, 1998.

[45] Bjvrn H. Junker and Falk Schreiber. *Analysis of Biological Networks*. Wiley, 2008.

[46] Anantharaman Kalyanaraman, Srinivas Aluru, Suresh Kothari, and Volker Brendel. Efficient clustering of large EST data sets on parallel computers. *Nucl. Acids Res.*, 31(11):2963–2974, 2003.

[47] S. Kantabutra and A.L. Couch. Parallel K-Means Clustering Algorithm on NOWs. *NECTEC Technical Journal*, 1(1), 1999.

[48] Tapas Kanungo, David M. Mount, Nathan S. Netanyahu, Christine D. Piatko, Ruth Silverman, and Angela Y. Wu. An Efficient k-Means Clustering Algorithm: Analysis and Implementation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(7):881–892, 2002.

[49] L. Kaufmann and P. Rousseeuw. Clustering by means of medoids. *Elsevier Science*, pages 405–416, 1987.

[50] L. Kaufmann and P. Rousseeuw. *Finding Groups in Data: An introduction to Cluster Analysis*. John Wiley, 1990.

[51] Alireza Khotanzad and Abdelmajid Bouarfa. Image segmentation by a parallel, non-parametric histogram based clustering algorithm. *Pattern Recognition*, 23(9):961 – 973, 1990.

[52] H. Kim and H. Park. Sparse Non-negative Matrix Factorizations via Alternating Non-negativity-constrained Least Squares for Microarray Data Analysis. *Bioinformatics*, 23(12):1495–1502, 2007.

[53] Piotr Kraj, Ashok Sharma, Nikhil Garge, Robert Podolsky, and Richard A. Mcindoe. ParaKMeans: Implementation of a parallelized K-means algorithm suitable for general laboratory use. *BMC Bioinformatics*, 9:200+, April 2008.

[54] Joseph B. Kruskal. On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem. *Proceedings of the American Mathematical Society*, 7(1):48–50, February 1956.

[55] X. Li and Z. Fang. Parallel clustering algorithms. *Parallel Comput.*, 11(3):275–290, 1989.

[56] Xiaobo. Li. Parallel algorithms for hierarchical clustering and cluster validity. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 12(11):1088–1092, Nov 1990.

[57] Aristidis Likas, Nikos Vlassis, and Jakob J. Verbeek. The global k-means clustering algorithm. *Pattern Recognition*, 36(2):451 – 461, 2003.

[58] Fionn Murtagh. Comments on 'Parallel Algorithms for Hierarchical Clustering and Cluster Validity'. *IEEE Trans. Pattern Anal. Mach. Intell.*, 14(10):1056–1057, 1992.

[59] H.S. Nagesh, S. Goil, and A. Choudhary. A scalable parallel subspace clustering algorithm for massive data sets. pages 477–484, 2000.

[60] R. Ng and J. Hahn. Efficient and Effective Clustering Methods for Spatial Data Mining. 1994.

[61] Victor Olman, Fenglou Mao, Hongwei Wu, and Ying Xu. Parallel Clustering Algorithm for Large Data Sets with applications in Bioinformatics. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 99(1), 5555.

[62] Clark F. Olson. Parallel Algorithms for Hierarchical Clustering. *Parallel Computing*, (8):1313–1325, 1995.

[63] Fazilah Othman, Rosni Abdullah, Nur'Aini Abdul Rashid, and Rosalina Abdul Salam. Parallel K-Means Clustering Algorithm on DNA Dataset. In *PDCAT*, pages 248–251, 2004.

[64] R. C. Prim. Shortest connection networks and some generalizations. *Bell System Technology Journal*, 36:1389–1401, 1957.

[65] S. Guha Rajeev, R. Rastogi, and K. Shim. ROCK: A robust clustering algorithm for categorical attributes. *Information system*, 25:345–366, 2000.

[66] S. Ranka and S. Sahni. Clustering on a Hypercube Multicomputer. *IEEE Transactions on Parallel and Distributed Systems*, 2(2):129–137, 1991.

[67] E. M. Rasmussen and P. Willett. Efficiency of hierarchic agglomerative clustering using the ICL distributed array processor. *J. Doc.*, 45(1):1–24, 1989.

[68] F. F. Rivera, M. Ismail, and E. L. Zapata. Parallel squared error clustering on hypercube arrays. *J. Parallel Distrib. Comput.*, 8(3):292–299, 1990.

[69] P. Sadayappan and F. Ercal. Cluster-partitioning approaches to mapping parallel programs onto a hypercube. In *Proceedings of the 1st International Conference on Supercomputing*, pages 475–497, New York, NY, USA, 1988. Springer-Verlag New York, Inc.

[70] R. Sibson. SLINK: An optimally efficient algorithm for the single-link cluster method. *The Computer Journal*, 16(1):30–34, 1973.

[71] Kilian Stoffel and Abdelkader Belkoniene. Parallel k/h-Means Clustering for Large Data Sets. In *Euro-Par '99: Proceedings of the 5th International Euro-Par Conference on Parallel Processing*, pages 1451–1454, London, UK, 1999. Springer-Verlag.

[72] Domenico Talia. Parallelism in Knowledge Discovery Techniques. In *PARA '02: Proceedings of the 6th International Conference on Applied Parallel Computing Advanced Scientific Computing*, pages 127–138, London, UK, 2002. Springer-Verlag.

[73] Joe H. Ward. Hierarchical Grouping to Optimize an Objective Function. *Journal of the American Statistical Association*, 58(301):236–244, 1963.

[74] S. White and P. Smyth. A spectral clustering approach to finding communities in graphs. In *Proceedings of the SDM '05*, 2005.

[75] Chin-Hsiung Wu, Shi-Jinn Horng, and Horng-Ren Tsai. Efficient parallel algorithms for hierarchical clustering on arrays with reconfigurable optical buses. *J. Parallel Distrib. Comput.*, 60(9):1137–1153, 2000.

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

[77] Xiaowei Xu, Jochen Jager, and Hans-Peter Kriegel. A Fast Parallel Clustering Algorithm for Large Spatial Databases. *Data Mining and Knowledge Discovery*, 3:263–290(28), 1999.

[78] Andrew C Yao. On constructing minimum spanning trees in k-dimensional spaces and related problems. Technical report, Stanford, CA, USA, 1977.

[79] Miron Livny Zhang, Miron@cs. Wisc. Edu, Tian Zhang, Tian Zhang, Raghu Ramakrishnan, Raghu Ramakrishnan, and Miron Livny. BIRCH: A new data clustering algorithm and its applications. *Data Mining and Knowledge Discovery*, 1:141–182, 1997.

[80] Ya-Ping Zhang, Ji-Zhao Sun, Yi Zhang, and Xu Zhang. Parallel implementation of CLARANS using PVM. *Machine Learning and Cybernetics, 2004. Proceedings of 2004 International Conference on*, 3:1646–1649 vol.3, Aug. 2004.