A publication of ISCA*:
International Society for Computers
and Their Applications

INTERNATIONAL JOURNAL OF COMPUTERS AND THEIR APPLICATIONS

TABLE OF CONTENTS

Page

Guest Editoral: Special Issue from the ISCA CATA 2021 75
Gordon Lee, Ajay Bandi, and Mohammad Hossain

Improvement of Apriori Algorithm for Missing Itemset Identification and Faster Execution 76
Anjan Dutta, Runba Ganguli, Punyasha Chatterjee, Narayan C. Debnath, and Soumya Sen

An Adaptable Memory System Using Reconfigurable Row DRAM to Improve Performance of Multi core for Big Data 84
Nagi Mekhiel

A Survey of Data Clustering for Cancer Subtyping 92
Yan Yan and Frederick C. Harris, Jr.

Clustered Particle Swarm Optimization Using Self-Organizing Maps 115
Yongwon Park, Harika Kilari, and Sanjeev Baskiyar

*** "International Journal of Computers and Their Applications is Peer Reviewed".**

Volume 28, No. 2, June 2021 **ISSN 1076-5204**

International Journal of Computers and Their Applications

A publication of the International Society for Computers and Their Applications

ISCA Headquarters···•···278 Mankato Ave, #220, Winona, MN 55987···•···Phone: (507) 458-4517
E-mail: isca@ipass.net • URL: http://www.isca-hq.org

Guest Editorial:
Special Issue from the ISCA CATA 2021

This Special Issue of the IJCA is a collection of four papers submitted for the 36th International Conference on Computer and Their Applications (CATA 2021). However, the conference was canceled due to the COVID-19 pandemic. The program committee of CATA 2021 carefully reviewed all the submitted papers and invited the authors of eight papers to submit extended conference papers for this special issue. Finally, four papers were selected for publication in ISCA special issue. Each paper was evaluated by at least two reviewers, judging the originality, technical contribution, significance, and quality of the manuscript. The papers in this special issue cover a wide range of research interests in computers and applications. The topics and main contributions of the papers are briefly summarized below.

ANJAN DUTTA of Techno International New Town, Kolkata, India, RUNA GANGULI The Bhawanipur Education Society College, Kolkata, India, PUNYASHA CHATTERJEE School of Mobile Computing and Communication, Jadavpur University, Kolkata, India, NARAYAN C. DEBNATH School of Computing and Information Technology, Eastern International University, Thu DauMot, Vietnam, SOUMYA SEN A.K. Choudhury School of Information Technology, University of Calcutta, India, wrote their paper "Improvement of Apriori Algorithm for Missing Itemset Identification and Faster Execution." In this work, the authors proposed a strategy to identify business-critical infrequent item sets that would otherwise be pruned using the traditional pruning strategy. Furthermore, a novel approach is proposed in the paper to evaluate whether an item is interesting or not at a considerably reduced computational time.

NAGI MEKHIEL of Ryerson University, Toronto, ON, Canada, authored the paper titled "An Adaptable Memory System Using Reconfigurable Row DRAM To Improve Performance Of Multi Core For Big Data." In this paper, the author presented an adaptable memory using reconfigurable row DRAM (RRDRAM) to allow many accesses that map to different physical rows to form one adaptable logical row accessed by multi-core as one physical row. In their experimental results, the author showed that the adaptable memory system improves the scalability of multi-core by up to 300% and could gain more from improving processor speed and global cache miss rate, and memory-processor bus bandwidth. Thus, RRDRAM can reduce the impact of DRAM on the performance and scalability of parallel computing when accessing big data with irregular access patterns.

YAN YAN, FREDERICK C HARRIS, JR. of University of Nevada, Reno, USA, authored the paper "A Survey of Data Clustering for Cancer Subtyping." The authors provided a comprehensive review of the major cluster analysis algorithms from the clinical and computational domains to facilitate the development in cancer subtyping based on microarray data. To complete this survey, the authors have referenced more than 200 articles and used various widely used taxonomy such as hierarchical clustering, partitioning clustering, graph-based clustering, distribution-based clustering, density-based clustering, grid-based clustering, clustering big data, clustering high dimensional data, and other clustering techniques.

YONGWON PARK and SANJEEV BASKIYAR of Auburn University, Auburn, AL, USA, wrote their paper on "Clustered particle swarm optimization using self-organizing maps." In this paper, the authors talked about premature convergence, which causes the PSO to stick in local optima. The authors used an approach called clustered PSO or CPSO, which clusters particles periodically around sample vectors (particles) using self-organizing maps. Their results of simulations showed that CPSO is highly effective in avoiding local minima for multimodal function optimization problems as well as reducing the particle population with every iteration.

We would like to express our sincere appreciation to all the authors and the reviewers for their contributions to this special issue. We hope you will enjoy the special issue and look forward to seeing you at future ISCA conferences. More information about ISCA society can be found at http://www.isca-hq.org.

Guest Editors:
Gordon Lee, San Diego State University, CATA 2021 Program Co-Chair
Ajay Bandi, Northwest Missouri State University, CATA 2021 Program Co-Chair
Mohammad Hossain, University of Minnesota at Crookston, CATA 2021 Program Co-Chair

Improvement of Apriori Algorithm for Missing Itemset Identification and Faster Execution

Anjan Dutta*
Techno International NewTown, Kolkata, INDIA

Runa Ganguli*
The Bhawanipur Education Society College, Kolkata, INDIA

Punyasha Chatterjee†
Jadavpur University, Kolkata, INDIA

Narayan C. Debnath‡
Eastern InternationalUniversity,
Thu DauMot, VIETNAM

Soumya Sen§
University of Calcutta, Calcutta, INDIA

Abstract

Association rule mining (ARM) is an important data mining strategy to analyze the relationship among the items. Apriori algorithm is the most used approach to implement association rule mining. We identified two major issues of Apriori. Apriori follows an iterative approach consisting of multiple database scans for searching frequent itemsets that satisfy certain threshold criteria. The same predefined threshold value is maintained throughout the repetitive stages of the Apriori method and hence there is a huge possibility of discarding higher-order itemsets, though all of its sub-itemsets are frequent. Some of these ignored itemsets if used intelligently have a huge potential for business value addition. Furthermore, in the Apriori procedure, an exponential number of computations is required to check whether an item is important or not and that makes the entire pattern mining system costly. In this study first, we identify the hidden business-critical item sets that are otherwise ignored in the traditional Apriori process. Furthermore, a novel approach is proposed here to evaluate whether an item is interesting or not at a considerably reduced computational time.

*Email: anjan.dutta.edu@gmail.com, runa.ganguli@gmail.com,
†School of Mobile Computing and Communication, Email: punyasha.chatterjee@gmail.com.
‡School of Computing and Information Technology. Email: narayan.debnath@eiu.edu.vn.
§A.K. Choudhury School of Information Technology. Email: iamsoumyasen@gmail.com.

Key Words: Association rule mining, apriori algorithm, data mining, missing itemset, execution time.

1 Introduction

Data mining is an integral part of any business in modern day to improve the performance in terms of profit, sales, forecasting etc. It is comprised of extracting the data, analyzing the data and then generating a report or pattern to ease out the business process. Data mining, also known as Knowledge Discovery in Databases (KDD) [5], is the process of discover-ing hidden and interesting patterns from a huge amount of data for making essential business-oriented decisions. Association rule mining [2, 6, 8] is a popular technique in data mining to analyze the relationship among the different items in a set of transactions. It is conceptualized as, for every occurrence of A there exists certain numbers of occurrence of B in any trans-action database. Knowledge of the frequent sets is generally used to design association rules stating how a set of items (itemset) influences the presence of another itemset in the transaction database. The mining rules are more applicable and useful in the market basket analysis. Association rules are frequently used in business intelligence [12] to help their marketing, advertisement, inventory control, fault prediction, product recommendation etc. Due to its huge business scope, association rule mining is a well-studied research problem among the researchers. Among the several association rule mining techniques Apriori algorithm [2] is the most studied and widely used algorithm for Frequent Pattern Mining (FPM) [6, 9, 13].

Apriori algorithm is used to mine all frequent itemsets in a transaction database. This algorithm begins with defining the support of an item that is the frequency of the occurrence of the items or itemsets in the transactional dataset. An itemset of size k whose support is greater than some user-specified minimum support threshold is said to be a frequent itemset and denoted by L_k, otherwise the items are infrequent. Any candidate itemset of size k, denoted by C_k is a potentially frequent itemset. The algorithm begins by scanning the whole database to find the set of frequent 1-itemsets by counting each item in database. The resulting set is called L_1 which is used to determine the set of frequent 2-itemsets which in turn is used to find the set of frequent 3-itemsets and so on until no more frequent k-itemsets can be found. In this way, it uses an iterative level-wise searching approach where k-itemsets are used to generate (k+1) itemsets. If there are n items, we can generate 2^n numbers of possible combination as given in Figure 1. To reduce the search space and improve the efficiency of this level-wise frequent itemset generation, the concept of pruning is used. This Apriori property states that if an itemset is not frequent, any large subset from it is also non-frequent [2]. This condition leads to pruning of some candidate itemsets from the search space in the database. It is shown in Figure 2.

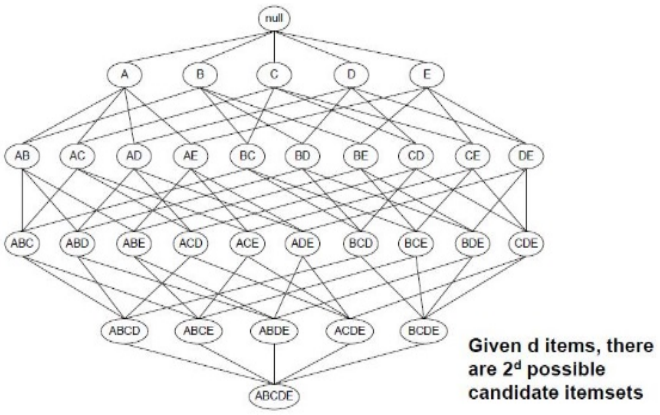

Figure 1: Frequent itemset generation

Although Apriori is a widely used technique for association rule mining several limitations could also be identified. In Apriori, itemsets are discarded based on the given threshold value and that value is also fixed for all the higher order itemsets. This static assumption leads to the discard of some of the interesting patterns by pruning. Another problem of Apriori is high time complexity as every new level or high order itemset (k+1) is generated from the k itemset.

In this paper, we address the above two limitations and propose new methodologies to find out the interesting missing patterns that are pruned by Apriori and also speedup the computation for quick decision making.

The rest of the paper is organized as follows: Section 2 presents the related studies on the different improvised version of Apriori Algorithm and its applications. Section 3 discusses the motivation of the work. In Section 4, the methodology of our work is described and the results are discussed in Section 5. The paper is concluded in Section 6.

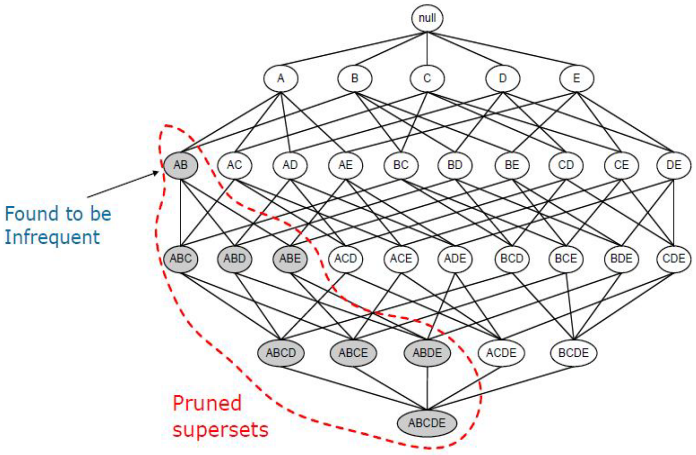

Figure 2: Apriori Principle of reducing number of candidates

2 Related Work

Apriori algorithm is one of the most widely used technique in data mining for frequent itemset calculation. However, it has many drawbacks and these are critical to large dataset mining. As a result, many researchers over the years have pointed out these problems and proposed different versions of improved Apriori algorithm. In [15], a Frequent Pattern (FP) Growth ARM algorithm has been presented that removed the disadvantages of traditional Apriori and proved to be efficient in terms of number of database scan and time. It [15] compresses a large database into a compact data structure Frequent Pattern (FP) tree [10] which is based on FP-Growth algorithm [10] and by recursively searching the tree, all frequent patterns are found. The authors have shown that FP-Growth algorithm outperforms the classical Apriori algorithm in terms of running time, number of database scan, but memory consumption is relatively high. Reducing thenumber of dataset scans for enhancing efficiency of the algorithm have been shown by many researchers [11, 16, 18]. Another approach is seen in [3], where instead of scanning the whole database for frequent itemsets, the authors have focussed on selective scanning of only some specific transactions based on minimum support count. It is seen by experiment that the total number of scanned transaction for candidate itemset generation is less when the same database uses standard Apriori and the improved Apriori [3] reduces the time consumed by 67.38% in comparison with the original Apriori. Singh et. al [17] have also worked towards reducing the database scan time by cutting down unnecessary transaction records and redundant sub-items generation during the pruning stage. The method eliminates the generation of candidates those having infrequent subset i.e., the itemsets having support count less than the specified threshold. The authors have proposed an optimized method for Apriori algorithm which reduces the size of the database by introducing an extra attribute Size_Of_Transaction

(SOT), containing the number of items in individual transaction in database. The improved algorithm [17] not only optimizes the algorithm for reducing the size of the candidate set of k-itemsets, Ck, but also reduces the I/O spending by cutting down transaction records in the database. However, it has the overhead to manage the new database after every generation of frequent set of k-itemsetsLk.

Minimizing the candidate generation is important for increasing efficiency of Apriori algorithm. Factors such as set size and set size frequency are being used [1] to eliminate non-significant candidate itemsets. The author implemented both the original and modified algorithms and average results favoured the modified algorithm by 38% and 33% in terms of execution time and database pass respectively. Apriori-Improve algorithm [4] was mainly proposed to optimize 2-itemset generation and transactions compression. The authors have used hash structure for frequent 2-itemset (L_2) generation directly from one database scan without generation of C_1, L_1 and C_2. The searching cost is also reduced by replacing hash tree by hash table. The algorithm used an efficient horizontal data representation and optimized strategy for saving time and storage space. In another research work [19] two major bottlenecks of FIM were addressed. These are: multitude of candidate 2-itemsets(C_2) and the poor efficiency of counting their support. The proposed algorithm, Reduced Apriori Algorithm with Tag (RAAT), reduces one redundant pruning operation of C_2. It is shown here how the use of transaction tag helps to speed up support calculation. The paper concludes that for relatively small support, RAAT algorithm runs faster than the traditional one. One more enhanced version of Apriori namely DCP (Direct Count of candidates & Prune transactions) [14] was proposed that focused on optimizing the initial iterations of Apriori when datasets are characterized by mainly short or medium length frequent patterns. The main enhancements include database pruning techniques, use of an effective method for storing candidate itemsets and their support counting. Application of improved Apriori algorithm was presented in [7] over a mobile e-commerce recommendation system. The approach converts the transaction database into a corresponding binary matrix to accelerate the algorithm efficiency, initially filtering out unrelated data in the candidate sets and hence improving the mining efficiency too.

3 Motivation and Objective

In the traditional Apriori method a significant number of insignificant items or itemsets having ignorable support count, are generated in the interim stages. Therefore, a pruning strategy is taken to eliminate those item sets. The item sets are eliminated based on two conditions-

If one of the sub-itemsets is infrequent then the itemset is infrequent.

If the support count of the itemset is less than the threshold then the itemset should be pruned.

Example 1: Let us consider a 3-itemset {p, q, r} database whose support count is less than the threshold support count s whereas the support count of all its 2-itemset subset {p, q}, {q, r} and {p, r} is greater than or equal of s. As per the approach of traditional Apriori although all the subset of a set is considered as frequent itemset, the superset {p, q, r} may become infrequent.

In the above approach, there may be itemsets having frequent sub-itemsets though its support count is less than the threshold value. If the sub-itemsets are valuable then keeping these itemsets together may add value to the business e.g. number of purchases of this larger set might increase resulting in the overall business growth.

It is because Apriori algorithm considers static threshold value for all k-itemset. In this example the itemset {p, q, r} can be an interesting pattern that has the potential to add value to the business. In this study, these kinds of valuable patterns that are otherwise ignored in the Apriori method are identified. In real life business applications this can be used as a case where the company wants to sell multiple products together as a package and they can choose the itemset like {p, q, r} which will be become frequent over the time based on their business strategy. Selling multiple products at a time is always a subject of interest for any business organization and this concept will help them in their business planning. We are going to address the issues of missing itemset of Apriori Algorithm that are potentially the subject of interest and have huge business interest apart from making it faster.

Furthermore in traditional Apriori to check whether an k-itemset is valuable or not all the iterative steps starting from computing a single frequent item up to the k-itemset should be completed and that results in the exponential number of computations [2]. Henceforth decreasing the computations to identify business-critical frequent patterns is another challenge. We also address this issue here.

In this work we have two unique objectives – (i) to identify the infrequent itemset, that has certain hidden business critical patterns but is generally pruned by the traditional Apriori method (ii) to reduce the computational complexity to identify the frequent patterns.

4 Methodology

In this work two unique algorithms are proposed to – (i) identify the hidden business critical patterns that are otherwise pruned by the traditional Apriori method (ii) reduce the computational complexity to identify the important patterns. The detailed approach is explained in the next subsections.

4.1 Identifying the Hidden Business Critical Patterns

In this work, a novel strategy is proposed to identify these business critical infrequent itemsets that would be otherwise pruned if the traditional pruning strategy is followed. A modified Apriori algorithm (Algorithm 1) to generate the interesting patterns is proposed and explained here.

Algorithm 1: Algorithm Generate Interesting Patterns
Input: Transactional database, minimum threshold €
Output: All interesting items
Begin
1. Store all the frequent single items in L_1
2. $k \leftarrow 2$
3. While $L_{k-1} \neq \emptyset$
   a. Compute all possible k pair combinations of items in L_{k-1} and store in set C_k

      //Unlike traditional Apriori method, instead of scanning the entire database, the minimum support count of //immediate subsets is compared with the threshold to determine the interesting patterns
   b. For each item p in C_k
      i. Compute minimum support of all subsets of p and store in s
      ii. If $s \geq €$
          $L_k \leftarrow s$
   c. $k = k + 1$
4. return $\bigcup_k L_k$

End

The algorithm (Algorithm 1) begins with identifying all frequent single items by comparing them with the given threshold support count €. After eliminating all the infrequent items the rest are stored in L_1. After that k-item set is found by combining all the k-1 items or itemsets with each other. Thereafter minimum support count of all possible subsets of each of the k-pair itemset is computed. This support count is compared with the threshold support count € and if it is greater than or equal to € then the corresponding itemset is included otherwise it is pruned. Some future potential business-critical items that are pruned in the traditional Apriori process are preserved here as demonstrated by the Example 2.

In Table 1 and Table 2 the support count of 4-pair itemset {A, B, C, D} and support counts of all its possible subsets are shown. The threshold support count is assumed to be 2.

Example 2:

Table 1: Support count of 4-pair item set

Itemset	Support Count
{A, B, C, D}	1

In Table 1 it is observed that {A, B, C, D} has a support count 1 which is less than the threshold support count 2 whereas support counts of all the 3-pair item sub sets are greater than the threshold value. In the traditional Apriori pruning strategy {A, B, C, D} is pruned but in the proposed conditional pruning method since all the subsets are frequent, the superset is considered as an important itemset. The novelty of this study lies here in discovering hidden patterns that will add to the business insight and market forecast.

Table 2: Support counts of 3-pair item sets

Item set	Support Count
ABC	2
ACD	3
ABD	2
BCD	4

4.2 Reduction of the Computational Complexity for Frequent Pattern Identification

In the traditional Apriori method to identify whether an itemset of size k is frequent or not involves a large number of iterative stages. All possible combination of items (except the pruned items) and their support counts are computed until the given itemset is found and it involves nearly n*(k-1) database scans where n is the total number of computed items or itemsets (n≅2^(k-1)) [2]. Therefore these repetitive database scans and large number of computations make the overall Apriori method significantly costly. A unique top-down approach is proposed here to significantly reduce the computational complexity of interesting pattern identification. First frequent single items are found and thereafter all possible two pair itemsets along with their support counts are computed. Here a no pruning strategy of 2-pair itemsets is taken for faster operation. The results are internally stored for later use. These two pair itemsets are periodically refreshed to have the updated items. After computing and storing the interesting one item and two item pairs the next task is to check whether a larger itemset is frequent or important to the business. The decision that whether a given pattern is important for the business or not is taken by observing the support counts of all possible two pair subsets of the larger set. If the support counts of all of the two pair items are greater than the threshold then the itemset is considered valuable. The overall algorithm(Algorithm 2) is described below.

Algorithm 2: **Algorithm Compute Frequent Single And Two Pair Items**
Input: Source transactional database D and minimum support €.
Output: True if the given item set is frequent otherwise false.
Begin
1. Declare is Important = true
2. Search for all single items whose support count is more than the threshold count €
3. Compute all two pair items and store them in a list L.
4. For item set I in L
   a. Compute the frequency of I in the source database D
   b. Store I and corresponding support count

End for

5. Store the items in the given pattern in a set S
6. Compute all possible two pair subsets of set S and store them in Q
7. For each subset s in Q
   a. Get the support count of s from the correspond-ing stored database value and store in s
   b. s ≥ €
      continue
   else
      is Important = false
      exit loop and go to step 5.

End for

8. print is Important

End

A sample source transactional dataset is shown in Table 3. The frequent single itemsets generated by scanning the transaction dataset considering a threshold of two is depicted in Table 4. Again Table 5 depicts the set of all 2-itemsets.

Table 3: Transactional database

Transactions	Items
T1	A, B
T2	B, C, D
T3	A, B
T4	A, C
T5	A, C, D
T6	B, C

Table 4: Frequent single itemsets

Item	**Support count**
A	**4**
B	**4**
C	**3**
D	**2**

Table 5: Two itemsets

Item Sets	Support Count
AB	2
AC	2
BC	2
AD	1
BD	1
CD	2

Now to check a 3-item set {A, B, D} is frequent or not all possible two pair subsets and their support counts are obtained from Table 5. Here the sub sets are {A,B}, {B,D} and {A,D}. Among all these two pair itemsets BD is infrequent since its support count is less than the threshold support count 2. Hence ABD is considered as unimportant itemset. As an another example if itemset {A, B, C} is considered, then all of its two pair subsets are frequent and hence it is considered as an interesting pattern.

5 Results and Discussion

In the traditional Apriori algorithm to check whether an n itemset is important or not, the iterative steps of the Apriori methods have to repeat to obtain all n pair frequent itemsets. Thereafter the given item is searched among those itemsets to know whether it is frequent or not. In this study, a novel methodology is proposed where the traditional Apriori is followed to obtain two itemsets. Thereafter the decision that whether a given itemset is interesting or not is taken by comparing the support counts of all the subsets with the threshold support count. Since only two initial iterations of the traditional Apriori method is performed here, hence this method is computationally faster than the classic Apriori method. The number of computations of Apriori algorithm is exponential and nearly equal to 2^n where n is the number of items. In the proposed methodology since only two pair items are computed the number of computations is reduced to ${}^nC_2$ i.e. Table 6 gives a comparative study with respect to number of computations in both the methods while Figure 3 graphically represents the study.

Table 6: Showing number of computations in both approaches

Number of items	Number of computations in traditional Apriori	Number of computations in the proposed methodology
3	8	3
4	16	6
5	32	10
6	64	15
7	128	21
8	256	28
9	512	36

Apart from reduction in time complexity, mining the interesting patterns that are otherwise unexplored in traditional Apriori is another feature of this study. As an example in Table 7 a sample retail dataset is shown.

Now the single and two pair frequent items are shown in Table 8 and Table 9 considering threshold support count as 2.

From Table 9 frequent 3-itemsets are computed according to traditional Apriori algorithm as shown in Table 10.

According to traditional Apriori only the set { Bread, Butter, Jam } is considered as frequent or as important itemset as its support count is greater than the minimum support count. All other three itemsets (showed in the dashed rectangular area – Table 10) are discarded.

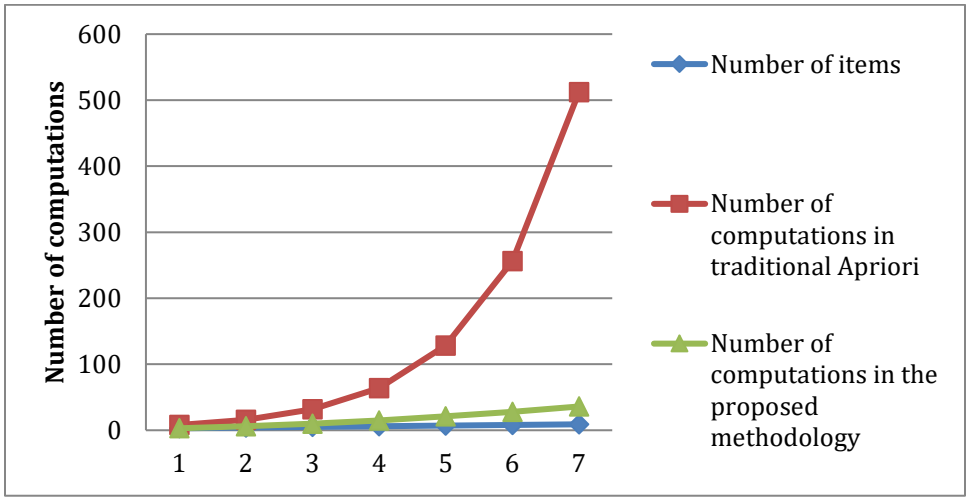

Figure 3: Comparative study of number of computations between traditional Apriori and the proposed method

Table 7: Retail dataset

Transactions	Item pattern
T1	Bread, Butter, Jam
T2	Bread, Jam
T3	Milk, Bread
T4	Milk, Bread, Butter
T5	Bread
T6	Butter, Milk
T7	Bread, Butter, Jam

Table 8: Frequent single itemsets

Items	Support count
Bread	6
Butter	4
Milk	3
Jam	3

Table 9: Frequent 2-itemsets

Item pair	Support count
Bread, Butter	3
Bread, Jam	3
Milk, Bread	2
Milk, Butter	2
Butter, Jam	2
Milk, Jam	1

Table 10: Frequent 3-itemsets

Item set	Support count
Bread, Butter, Jam	2
Bread, Milk, Jam	0
Butter, Milk, Jam	0
Milk, Bread, Butter	1

In this study after computing the frequent two itemsets the repetitive steps of Apriori are not continued, instead a top down approach is taken to mine the interesting patterns. As mentioned in Section 3, a particular itemset is considered as interesting if all the subsets are frequent. Here source database is not scanned again to find the frequency of the pattern. The item superset is considered as valuable if the minimum support count of all the subsets is greater than the threshold support count. The corresponding computations are shown in Table 11.

Table 11: Mining of interesting patterns in the proposed approach

Itemset	Subsets	Min subset support count	Selected / Discarded
{Bread, Butter, Jam}	{Bread, Butter}, {Bread, Jam}, {Butter, Jam}	2	Selected
{Bread, Milk, Jam}	{Bread,Milk}, {Bread,Jam}, {Milk, Jam}	1	Discarded
{Butter, Milk, Jam}	{Butter,Milk}, {Bread,Jam},{ Milk, Jam}	1	Discarded
{Milk,Bread, Butter}	**{Milk, Bread}, {Milk, Butter}, {Bread, Butter}**	**2**	**Selected**

In the above analysis it is observed that in traditional Apriori the combination {Milk, Bread, Butter} is ignored whereas in the proposed methodology this is considered as a potential interesting pattern. Hence for the retail market the combination **{ Milk, Bread, Butter** } can be tried to check

whether it will add value to the business or not. To check if a given itemset of any length is frequent or not, all possible two pair subsets are computed and the minimum support count of the subsets is compared with the threshold support to identify whether it is valuable or not.

6 Conclusion

Apriori algorithm is widely used by different business applications to identify the frequent itemset. But it misses some interesting business patterns. Here we identify these patterns to generate the itemset that can be useful for the organization to sell multiple products as a package. This will help the organization to increase the sales of the products which are below the threshold level. As the users get the chance to use these additional products at a little extra cost (the package of products are sold at discounted price) they may like it and may purchase that product individually also. Without spending a huge amount for the advertisement, the sales of the product will get a boost using our proposed approach. We also address another major drawback of the Apriori algorithm that is the higher number of computations. Our proposed methodology based on 2 itemsets drastically reduces the computational time. In Section 5 we have explained the computational benefit of our proposed method over Apriori. Henceforth the proposed approach will help the organization to identify the missing business pattern and to develop the business strategy based on these additional itemsets at a fraction of time over Apriori. The proposed method can be experimented over different big data tools for faster execution time. Further improvement is possible by incorporating parallelism in computation process may be introduced by using the concept of Mapreduce in Hadoop framework or Sharding in MongoDB.

References

[1] S. A. Abaya, "Association Rule Mining Based on Apriori Algorithm in Minimizing Candidate Generation," *International Journal of Scientific & Engineering Research*, 3(7):1-4, 2012.

[2] R. Agrawal and R. Srikant, "Fast Algorithms for Mining Association Rules," *Proc. 20th Int. Conf. Very Large Data Bases, VLDB*, 1215:487-499, 1994.

[3] M. Al-Maolegi and B. Arkok, "An Improved Apriori Algorithm for Association Rules," *arXiv preprint arXiv:1403.3948,* 2014.

[4] R. Chang and Z. Liu, "An Improved Apriori Algorithm," *Proceedings of 2011 International Conference on Electronics and Optoelectronics*, IEEE, 1:V1-476, 2011.

[5] U. Fayyad, G. Piatetsky-Shapiro and P. Smyth, "From Data Mining to Knowledge Discovery in Databases," *AI Magazine,* 17(3):37-37, 1996.

[6] M. Giridhar, S. Sen, and A. Sarkar, "Share Market Sectoral Indices Movement Forecast with Lagged Correlation and Association Rule Mining," *IFIP International Conference on Computer Information Systems and Industrial Management*, Springer, Cham, pp. 327-340, 2017.

[7] Y. Guo, M. Wang and X. Li, "Application of an Improved Apriori Algorithm in a Mobile E-Commerce Recommendation System," *Industrial Management & Data Systems*, pp. 287-493, 2017.

[8] E. H. Han, G. Karypis and V. Kumar, "Scalable Parallel Data Mining for Association Rules," *AcmSigmod Record,* 26(2):277-288, 1997.

[9] J. Han, H. Cheng and D. Xin, "Frequent Pattern Mining: Current Status and Future Directions," *Data Mining and Knowledge Discovery,* 15(1):55-86, 2007.

[10] J. Han, J. Pei and Y. Yin, "Mining Frequent Patterns without Candidate Generation," *ACM Sigmod,* 29(22)1-12, 2000.

[11] J. L. Lin and M. H. Dunham, "Mining Association Rules: Anti-Skew Algorithms," *Proceedings 14th International Conference on Data Engineering*, IEEE, pp. 486-493, 1998.

[12] G. Maji, S. Sen and A. Sarkar, "Business Intelligence Development by Analysing Customer Sentiment," *2018 7th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO)*, IEEE, pp. 287-290, 2018.

[13] S. Paladhi, S. Chatterjee, T. Goto and S. Sen, "AFARTICA: A Frequent Item-set Mining Method Using Artificial Cell Division Algorithm," *IGI Global Journal of Database Management*, 30(3):71-93, 2019.

[14] R. Perego, S. Orlando and P. Palmerini, "Enhancing the Apriori Algorithm for Frequent Set Counting," *International Conference on Data Warehousing and Knowledge Discovery*, Springer, Berlin, Heidelberg, pp. 71-82, 2001.

[15] S. Rao and P.Gupta, "Implementing Improved Algorithm over Apriori Data Mining Association Rule Algorithm 1," *Citeseer*, pp. 489-493, 2012.

[16] A. Savasere, E. R. Omiecinski, and S. B. Navathe, *An Efficient Algorithm for Mining Association Rules in Large Databases*, Georgia Institute of Technology, 1995.

[17] J. Singh, H. Ram and D. J. S. Sodhi, "Improving Efficiency of Apriori Algorithm using Transaction Reduction," *International Journal of Scientific and Research Publications*, 3(1):1-4, 2013.

[18] H. Toivonen, "Sampling Large Databases for Association Rules," 96:134-145, 1996.

[19] W. Yu; X. Wang, F. Wang, E. Wang and B. Chen, "The Research of Improved Apriori Algorithm for Mining Association Rules," *2007 International Conference on Service Systems and Service Management*, IEEE, pp. 1-4, 2007.

Anjan Dutta has completed his Bachelor of Technology in Information Technology from RCC Institute of Information Technology, Kolkata, India and Master of Technology in Information Technology from Calcutta University, Kolkata, India. Currently, he is an Assistant Professor in the Department of Information Technology at Techno International New Town, Kolkata, India. His research area includes data mining, big data and machine learning.

Runa Ganguli completed her Master of Technology in Computer Engineering and Applications from A.K. Chowdhury School of IT, University of Calcutta, India in 2020, Masters of Science in Computer and Information Science from University of Calcutta, in 2014. She did her Bachelors of Science in Computer Science Honours from Asutosh College, University of Calcutta in 2012. She is currently working as an Assistant Professor in the Department of Computer Science, The Bhawanipur Education Society College, Kolkata, University of Calcutta, India since 2015. She has several papers in reputed peer reviewed journals and international conferences. Her main research interest includes Graph Database, Data mining and Social Network Analysis. She has 6 years of teaching experience.

Punyasha Chatterjee received the B.Tech., M.Tech. and Ph.D. degrees in Information Technology from the University of Calcutta, Kolkata, India, in 2003, 2005 and 2018 respectively. She is presently an Assistant Professor in the School of Mobile Computing and Communication, Jadavpur University, Kolkata, India, since 2012. Her area of interest includes adhoc networks, wireless sensor networks, Internet of Things and pervasive computing. She is a senior member of IEEE and member of ACM.

Narayan C. Debnath earned a Doctor of Science (D.Sc.) degree in Computer Science and also a Doctor of Philosophy (Ph.D.) degree in Physics. Dr. Narayan Debnath is currently the Founding Dean of the School of Computing and Information Technology at Eastern International University, Vietnam. He is also serving as the Head of the Department of Software Engineering at Eastern International University, Vietnam. Dr. Debnath has been the Director of the International Society for Computers and their Applications (ISCA) since 2014. Formerly, Dr. Debnath served as a Full Professor and Chairman of Computer Science at Winona State University, Minnesota, USA. Dr. Debnath has been an active member of the ACM, IEEE Computer Society, Arab Computer Society, and a senior member of the ISCA.

Soumya Sen has received a Ph.D. in Computer Science & Engineering from the University of Calcutta in 2016. He received his M.Tech in Computer Science & Engineering in 2007 and M.Sc. in Computer and Information Science in 2005 also from University of Calcutta. He has joined A. K. Choudhury School of Information Technology under University of Calcutta in 2009. Dr. Sen has around 90 research publications in International Journals and conferences. He has 3 international patents and also published 2 books. Dr. Sen is the TPC member of many conferences across the world and serves as reviewer for many International journals. He is a member of IEEE and ACM. Dr. Sen is a fellow of IETE (Institution of Electronics and Telecommunication Engineers). His current research area interests are Data Warehouse & OLAP Tools, Data Mining, and Distributed Database.

An Adaptable Memory System Using Reconfigurable Row DRAM To Improve Performance Of Multi Core For Big Data

Nagi Mekhiel*
Ryerson University, Toronto, ON, CANADA M5B 2K3

Abstract

Multi-core based systems access DRAM using multiple different addresses that could map to different rows in the same bank at the same time, causing row conflicts forcing them to wait to activate one row at a time. We present an adaptable memory using reconfigurable row DRAM that divides rows into many segments and uses special latches to allow many accesses that map to different physical rows to form one adaptable logical row accessed by multi-core as one physical row. The adaptable row accesses different rows in a pipeline fashion by overlapping the long DRAM access time between the different accesses. The results show that the adaptable memory system improves the scalability of multi-core by up to 300% and could gain more from improving processor speed and global cache miss rate and memory-processor bus bandwidth.

Key Words: Reconfigurable architectures; memory systems; DRAM; access patterns; pipelining; scalability of multi-core; big data.

1 Introduction

The processor speed and computing power have continuously increased due to advancements in technology. This increase in processor power depends on delivering data and instructions to the processor from memory at the processor speed. Unfortunately, current memory systems cannot offer the processor its data at the required rate [1,5,11]. Computer performance depends on the memory system to provide multi core with data at high rate. If the memory system fails to deliver the required data rate, it will become the limiting factor to the system performance [1,2].

The cache system has been used to solve this problem by moving parts of data to a fast memory that can match the processor speed. The cache system cannot fully isolate fast processor from the slow main memory. The cache has misses and the processor have to visit the main memory to get data when information is not in cache (misses). These misses represent the portion of processor time that limits the overall system performance improvements. Furthermore, cache performance cannot be improved continuously by changing any of its parameters (size, associativity, speed, ..) and reaches a point of no return with respect to its parameters [2].

DRAM designs offer high bandwidth that depend on using multibank and fast page access mode. In fast page mode, accesses that exist in same DRAM row can be retrieved at a much faster speed (fast page mode). The large data streams in the applications can be mapped to the same row and benefit from this mode. With multibank, more than one active row can supply a processor with fast accesses from different parts of memory [3,9]. However, a multi-core system has many different accesses to different rows at the same time for the same bank.

The contribution of this paper is to reduce the impact of DRAM on the performance and scalability of parallel computing when accessing big data with irregular access patterns. Irregular accesses map to different DRAM rows, cause row conflicts and increase DRAM access time. Performance gain of parallel computing is limited by the slowest portion that cannot be improved. Our method applies parallelism to the slowest DRAM accesses that is related to activating rows by pipelining and overlapping these accesses when mapped to different rows.

2 Backgound

DRAM has been through numerous changes to its design from the basic DRAM architecture to the asynchronous to the fast page mode (FPM) to the extended data-out (EDO) to the burst-mode EDO to the synchronous(SDRAM) [4].

The changes have been relatively minor in terms of their implementation cost and have increased DRAM throughput significantly. Compared to the asynchronous DRAM, FPM simply allows the row to remain open across multiple CAS commands, requiring very little additional circuitry. To this, EDO changes the output drivers to become output latches so that they hold the data valid on the bus for a longer period. To this, BEDO (EDO with Burst) adds an internal counter that drives the address latch, so that the memory controller need not supply a new address to the DRAM on every CAS command if the desired address is simply one-off from the previous CAS command. Thus, in BEDO, the DRAMs column-select circuitry is driven from an internally generated signal, not an externally

*Department of Electrical, Computer and Biomedical Engineering. Email: nmekhiel@ee.ryerson.ca

generated signal: the source of the control signal is close to the circuitry that it controls in space and therefore time, and this makes the timing of the circuits activation more precise. Lastly, SDRAM takes this perspective one step further and drives all internal circuitry (row select, column select, data read-out) by a clock, as opposed to the RAS and CAS strobes [4].

3 Motivations

In general, DRAM architectures have limitations due to the following problems :

- DRAM architectures allow one active row per bank, which limits the ability of multiple accesses from current multi-core chip to access the same bank if the accesses map into different rows in the same bank.
- Only one row at a time can be activated, thus can have one open row per bank at any time. Multiprocessor shares data that map to the same area of memory, and likely in the same bank with different rows, therefore current architectures are not suitable for multi-core.
- Characteristics of software applications are not suitable for DRAM architectures. Program data could map in two different rows in one bank. Each time the program's accesses one row, the other row is forced to be closed because it is in the same bank, when it finishes, it goes back to the closed row and must open it, necessitating it to close the current row that has just been activated. Each time a row is closed, it precharges the bank, then latches the accessed row which wastes time and limit the ability of DRAM to provide fast accesses to processors.
- As performance of single processor is reaching a diminishing return, processor makers are now moving towards multi-core architecture with 8-core or 16-core in a single chip. The performance of the computer is limited by memory bandwidth [7,8].

What we need is a DRAM that is capable of providing a single processor or multiprocessors with high bandwidth using a new architecture that allows DRAM to have more than one location active in the same bank and is not restricted to one active row per bank in DRAM. Therefore, it must have many partial rows active and map them to multiple of physical DRAM rows in one bank, allowing multiple threads to access these active partial rows as if there is a logical row that changes its mapping to become adaptable to processors access patterns and includes many physical rows in a bank. The active row must be constructed based on the processor order of accesses and not by the order of columns in a physical row (adaptable to processor access patterns) [6].

4 The Concept of Reconfigurable Row DRAM "RRDRAM"

DRAM physical row consisting of a number of columns that are fixed and defined on DRAM array. Through this paper, we assume a simple DRAM array of 1024 rows, each row has 1024 columns that could be accessed when the row becomes active. If the processor requests an access that is not in the same active row in the accessed bank, it must activate a new row and adds all the time delays associated with opening a new row.

The processor access is usually for one location that maps to a single column in the accessed row, then a burst mode is used to transfer one cache block from the same active row. With multi-core many different accesses are requested from the memory, these accesses could map to different rows, forcing memory controller to close some active rows and open new rows which increases the access time.

The frequency of accessing DRAM in a multi-core system is multiple times of the single processor. This causes DRAM to open and close many rows very frequently increasing the accesses time and power. The root of the problem is having one fixed physical row activated per bank. A row consists of a number of columns that are used for accesses, and may be only one location of DRAM array that is 1024x1024, could be used in the fast page mode. This is less than .0001% of the DRAM array size. Having one physical row which either can be active or not at a time is not suitable for accesses that come from the multiprocessor and more likely map into many rows at a time.

We should have a memory that is not restricted to a fixed physical row as the current DRAM architecture dictates. A reconfigurable row is a logical row that is not restricted to the DRAM physical row.

If we divide each physical row into segments or sections, that each represents a smaller row of locations, that has for example, only 64 columns each representing a partial row. The size of row segment or section depends on the cache block size and application requirements and implementation. Each row segment or section could be active, therefore creating a logical row that is adaptable for physical rows when changing [6].

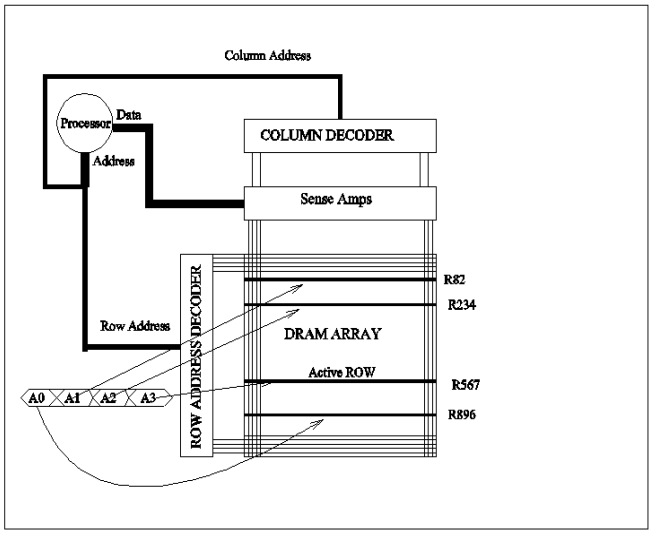

Figure 1: DRAM organization

Figure 1, shows a conventional DRAM organization being

accessed by a processor to deliver data from four locations in the DRAM array: A0, A1, A2, and A3. Assuming that A0 maps to row Row896, A1 maps to Row82, A2 maps to Row234 and A3 maps to Row567. This conventional DRAM system must precharge and activate R896 (Row896) to supply data for A0, then precharge and activate R82 and wait to supply data for A1, followed by the same operations for A2 and A3. This requires that total access time is four of random accesses = 4 *(precharge + ROW access + Column access + Transfer). In the reconfigurable row system, it will only use one active logical row and burst the four accesses from the latched data on sense amps, therefore total time of four accesses is one random access plus three fast page accesses= Precharge + Row access + Column access + 4 *transfer.

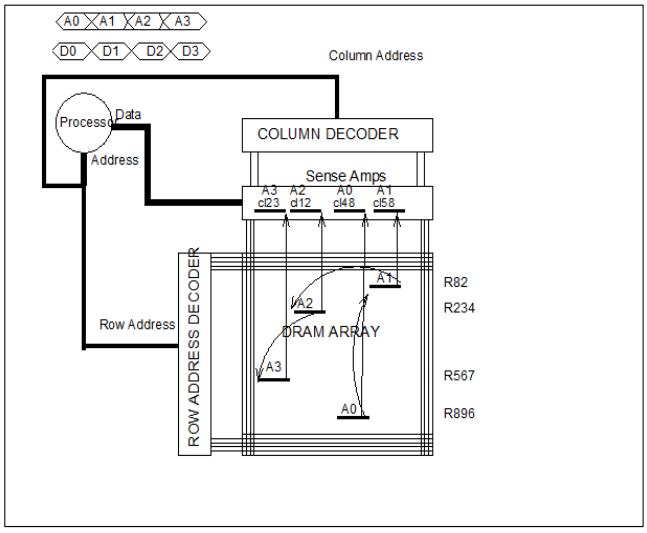

Figure 2: Mapping accesses in adaptable row DRAM

Figure 2 shows the adaptable row system in accessing the four addresses A0, A1, A2 , A3 for the above example. Although these addresses map to different four rows R896, R82, R234 and R567, the logical row is formed to contain only four partial rows. The other row segments are not included in the activated logical row, therefore leaving 12 partial rows available and precharged for the next group of accesses. It uses only a portion of the total sense amps leaving the rest available for next access. It should also be noted that in the reconfigurable row, because a large portion of DRAM array will be ready and precharged, latching new rows, and columns for the next accesses could be completely overlapped and interleaved with data transfer to the processor.

The latching of row segments allows the row address to change and be decoded, while accessing present row segments. This is a pipelining of accessing DRAM in which segments are used and accessed in parallel by overlapping their delay time of precharging, latching, decoding and access as explained below in system operation.

5 The Reconfigurable ROW DRAM "RRDRAM" System

5.1 The Basic System

The basic reconfigurable row for DRAM maintains the same interface to the outside system. It uses the same DRAM core of an array of columns and rows and the same storage elements of single capacitor. The main difference is in using multiple row segment latches, with each latch having a number of flip flops equal to the number of rows in the DRAM array. For example, if the DRAM array is 1024x1024, then these row segment latches have 1024 flip-flops each. Each physical row in DRAM array is divided into multiple segments, each segment is connected to one output of the corresponding segment row latch [6].

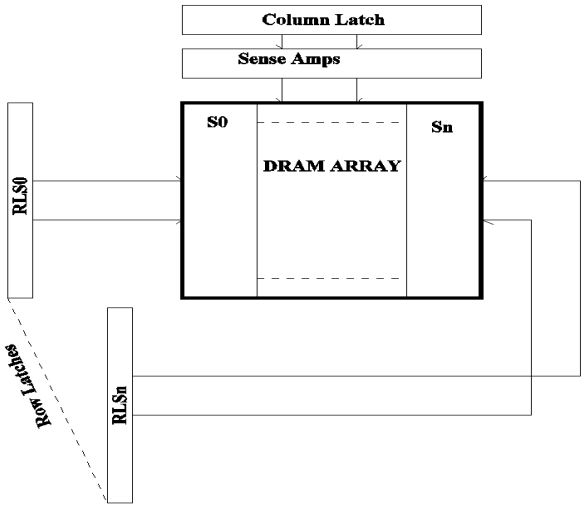

Figure 3: Block diagram for reconfigurable row DRAM

Figure 3 shows a block diagram for a reconfigurable row for DRAM. Multiple row segment latches. RLSn are used to latch the decoded row of an accessed segment, each time there is a request to access DRAM.

The outputs of these RLS latches control the word lines of a specific row segment and each output is connected to 1 word line of a segment. For 1024x1024 and 16 segments, there are 1024 outputs from each of RLS and each output is connected to 64 of word lines to activate the segment for 64 cells.

Figure 4 shows a more detailed schematic diagram to the RRDRAM for each segment of the different rows being latched by its corresponding segment row latch using k flip-flop, where k is the number of rows in DRAM array and n is the number of segments in each row, thus requiring nxk Flip-Flops in total per the DRAM array. The flip-flops use RlClk to activate one flip-flop that has its row = 1. ROW signals are activated by the row decoder when a valid access is requested. The column decoder shown is used to generate the proper RlClk for one segment at a time. RlClk is generated to latch 1 at flip flop for the segment in the intersection of accessed row and accessed column in DRAM array.

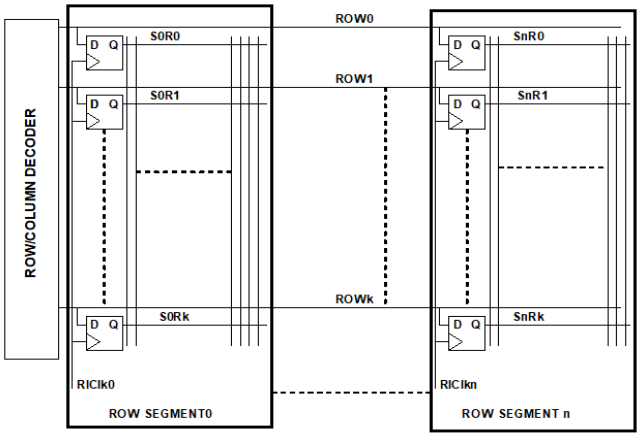

Figure 4: Block diagram for row segments of reconfigurable row DRAM

6 System Operation

6.1 Pipelined Operation

Our RRDRAM uses the same known DRAM technology, therefore it requires the same timing to access data from one location. In conventional DRAM the following are the basic operations that must be performed for a DRAM operation (read or write):- 1-Precharge operation wait for Tpr then 2-Latch a valid row address wait for Trcd then 3-Latch a valid column wait for Tcac then 4-access data In RRDRAM, the use of different latches for row address, column address and latches for the decoded rows in a segment (RL) and the latch for multiple decoded columns (Column decoder Latch) allows the system to overlap multiple accesses to different DRAM rows similar to processor pipelining of instruction execution.

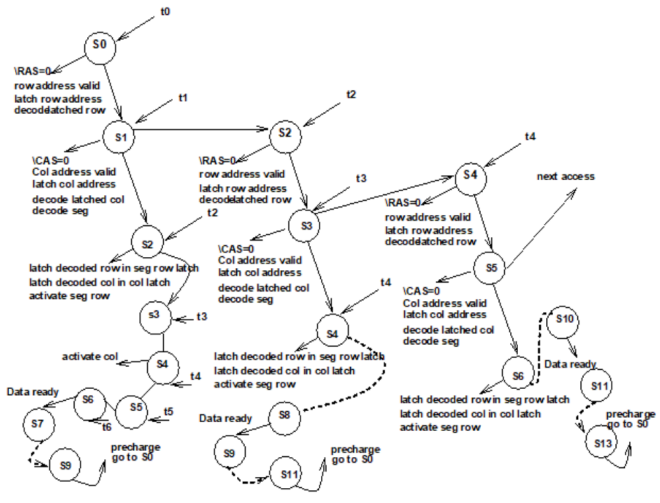

Figure 5: State diagram for pipelined operation in reconfigurable row DRAM

Figure 5 shows a state diagram for the pipelined operation of RRDRAM using the system of latches given above. The following are the different states for multiple DRAM accesses to different rows in RRDRAM:

- S0 (time t0): The row address of the first access is applied to the DRAM multiplexed address and is latched by row address latch at time t0. The row decoder immediately starts to decode this row address.
- S1 (time t1): The memory controller applies the column address of first access to DRAM multiplexed bus and is latched by the column address latch at time t1. The column decoder starts immediately decoding this column address. The segment decoder decodes a portion of that address corresponding to the segment selection and makes one output active that corresponds to the accessed row segment.
- S2 (time t2) and S0 of second access: The system starts the second access by applying the row address of the second access similar to the first access in S0. It also continues with the first access by latching the decoded row from the row decoder in the corresponding RL latch by one of RlClk generated from the segment decoder circuit. It also latches the decoded column for the first access in the column decoder latch and keeps previous accessed columns in the active latch by the feedback.
- S3 (time t3) and S1 of second access: The first access is waiting for Trcd and the accessed row is being active and applied to the word line of the selected row segment from the corresponding RL latch. The second access is continuing in S1 similar to access 1 to latch its column address. It is important to note that the first access column has been captured and stored in the column decoder latch in S2.
- S4 (time t4), S2 of second access and S0 of third access: The first access asserts its active column to the bit line to access the data. The second access latches the decoded row in RL latch similar to first access in S2 and also latches the decoded column. The third access starts in applying a row address to the multiplexed bus similar to first access in S0.
- S5 (time t5) ,S3 of second access and S1 of third access: The first access waits for Tcac to get data. The second access waits for Trcd and accessed row is applied to word line as in S3 for first access. Third access is in S1 to latch column address and decode segment.
- S6 (time t6), S4 of second access and S2 of third access: First access has its data ready and valid. The second access asserts its active column to the bit line to access the data. The third access is latching the decoded row from the row decoder in the corresponding RL latch by one of RlClk generated from the segment decoder circuit. Also it latches the decoded column for the first access in the column decoder latch and keeps previous accessed columns in the active latch by the feedback.

At S6 data is ready for access 1, after two cycles data is ready for access 2, after two cycles data is ready for access 3. When

access 1 delivers its data at S6, RRDRAM starts immediately precharging the segment of access1. In this way the precharge time is hidden and pipelined with other accesses. The second and third accesses follow the same method making all accesses pipelined in precharge, latching, decoding, access time and data delivery.

6.2 Timing for System Operation

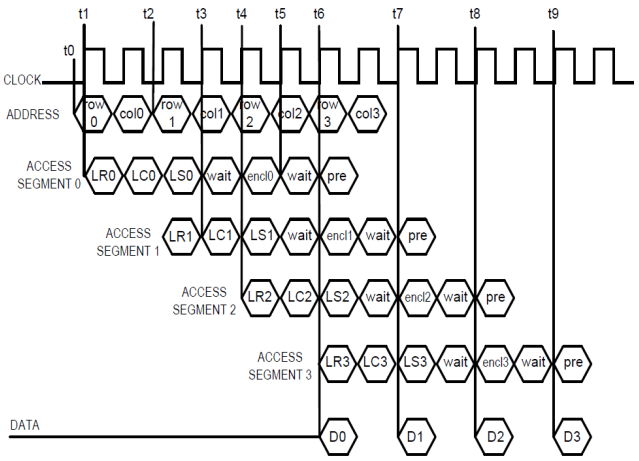

Figure 6: The timing of pipelined operation in RRDRAM

Figure 6 shows the timing of the RRDRAM operation given above with the state diagram. The address bus supplies row and column address for each access through the use of a multiplexer in the memory controller as in any conventional DRAM. The next access address is supplied to the address bus without the waiting for data to be accessed from DRAM for the first access. The memory controller keeps supplying the RRDRAM address bus by new accesses every two clock cycles. The bus speed could be increased compared to conventional DRAM because of the use of latches in RRDRAM. Each address is latched immediately and stored in the latch to be decoded.

The access to first address starts by latching the row in LR0 assuming that LR0 is used for access 0. LC0 is used to latch column address for first access 0. The partial row is then activated through applying the output of its RL latch to the word line of the segment in LS0. The system waits, then uses the output of column latch to activate bit line and access the data after waiting for one cycle as shown in data bus as D0. After each segment delivers its data it goes through precharge individually without the need to precharge the whole bank as in conventional DRAM.

Other accesses, access 1 and access 2 time is overlapped with each other in a pipeline fashion to deliver data at a rate of 1 access every two cycles regardless of row number, therefore more than one physical row could be accessed in parallel.

7 Performance Evaluation of RRDRAM

7.1 Conventional Single Processor Model

$$Ts = Tp \times Ni + M \times Ni \times (Ta + Tf) \quad (1)$$

Ts is the execution time for single processor using conventional DRAM
Tp= processor cycle time and could be less than 1 for superscalar
Ni=number of instructions in an application
M=Cache Miss rate
Ta=main memory access time to activate DRAM Row and access first location
Tf=time to transfer 1 block of cache using memory bus

7.2 Conventional Multiprocessor Model Using DRAM

$$Tm = Tp \times Ni + Np \times M \times Ni \times (Ta + Tf) \quad (2)$$

Tm is the time to execute multiple parallel processes on multiprocessor system using conventional DRAM where Np is number of processors. The total number of misses to external shared main memory will be proportional to Np as they have to be serviced in a serial fashion from one shared memory.

The processor execution time is overlapped among multiple processors working in parallel, but accesses to shared memory must be serialized.

$$Scalability = Np \times Ts \div Tm \quad (3)$$

The scalability is calculated by dividing the total time that a single processor spends to execute same Np number of processes in serial fashion by the time that multiprocessor takes to execute same number of processes in parallel as given by Equation (2).

7.3 Conventional Multiprocessor Model Using RRDRAM

$$Tm(R) = Tp \times Ni + M \times Ni \times (Ta + Np \times Tf) \quad (4)$$

Tm(R) is the time to execute multiple processes on a multiprocessor system with RRDRAM where Np is the number of processors.

$$Scalability = Np \times Ts \div Tm(R) \quad (5)$$

The total time to access the external main memory RRDRAM will be the first access time Ta to DRAM then overlap the rest of access times with Np transfer times to fill Np caches.

The scalability is calculated by dividing the total time that a single processor spends to execute same Np number of processes in serial fashion by the time that multiprocessor takes to execute the same number of processes in parallel as given by Equation (4).

7.4 Parameters of the System

Assuming a SPEC CPU2000 [10] benchmark application in a processor running at 3.3 GHz.
IPC = 2 from using superscalar.
Tp = .15n average.
Ni= depends on the workload and changes from .774 to 14.6 billion instructions for SPEC CPU2000. We assume Ni = 1 billion.
M depends on the application and the cache. It can vary from .0002 to .001, We assume M=.0006 for SPEC CPU2000 , 64 KB L1 caches and is the same for all systems.
We assume Ta= 30 ns for a typical DRAM, this is the access time to get first data element.
We can calculate the cache block transfer time as: similar to Intel multi core that has about 10 GB/S (1.33 GHz and bus = 8 B) Tf = 128/10 = 12.8 ns.

7.5 Scalability of DRAM and RRDRAM Systems Results

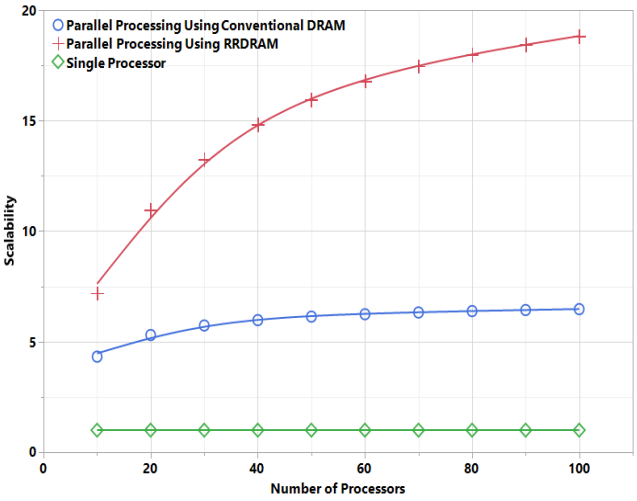

Figure 7: Scalability of multiprocessor systems using DRAM versus RRDRAM

Figure 7 shows the results of multiprocessor scalability when using conventional DRAM and RRDRAM versus the number of parallel processors.

The systems parameters are: processor speed is 3.3 GHz, DRAM and RRDRAM access time Ta is 30 ns, Global Cache miss rate is .0006 and memory bus bandwidth is 10 GB/s.
The effect of DRAM and RRDRAM is significant to the performance and scalability of multiprocessors limiting its scalability for a large number of processors. This shows that the memory gap is still an issue that should be dealt with for parallel computing. This confirms the Amdahl's law that the portion of time that cannot be improved remains the system bottleneck to overall performance gain.

The multiprocessor using RRDRAM significantly improves scalability of the system compared to the system using conventional DRAM up to 300%.
Scalability of multiprocessor using DRAM reaches its maximum at a low number of processors while the multiprocessor using RRDRAM reaches its maximum performance at a higher number of processors.

7.6 Scalability of DRAM versus RRDRAM Systems Using different Processor Speed

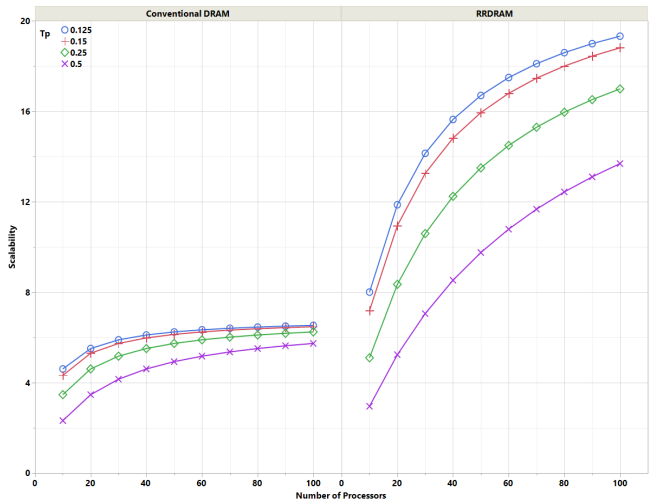

Figure 8: Scalability of multiprocessor systems using DRAM versus RRDRAM with different processor speed

Figure 8 shows the scalability of both systems when changing processor speed from 1 GHz to 4 GHz. The multiprocessor system using conventional DRAM gains much less than the system using RRDRAM for improving processor speed. The system using DRAM only improves its scalability by 20% for four times increase in processor speed. The system using RRDRAM improves by 40% for same processor speed increase. This indicates that RRDRAM gains more from increasing processor speed and technology could use both improvements in the number of transistors and improvements speed of transistors to continue in improving the gain of multiprocessor scalability.

The results also show that RRDRAM system using a faster processor gives the same scalability when using 20 faster processors as the same system using 70 processors if using slower processors. The DRAM based system with slower processor speed needs to use 100 processors to get the same scalability of 20 faster processors. This indicates that the system using RRDRAM gains more from increasing processor speed than the system using DRAM.

7.7 Scalability of DRAM versus RRDRAM Systems Using different Miss Rate

Figure 9 shows the scalability of both systems when changing global cache miss rate. The miss rate changes by eight times from .0003 to .0024.

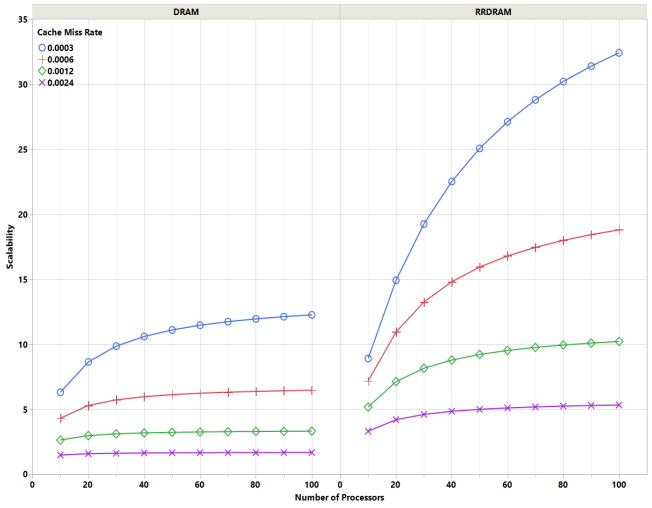

Figure 9: Scalability of multiprocessor systems using DRAM versus RRDRAM with different miss rate

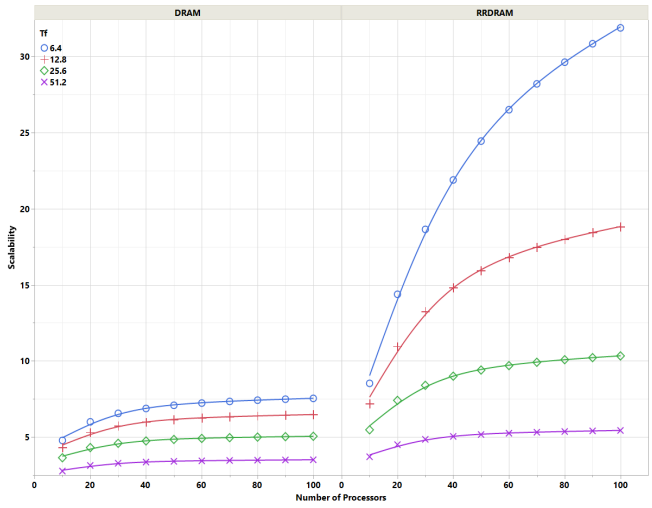

Figure 10: Scalability of multiprocessor systems using DRAM versus RRDRAM with different transfer time

The multiprocessor system using conventional DRAM improves its scalability by 200% from 6 to 12 when miss rate improves by eight times.

The multiprocessor system using RRDRAM improves its scalability by 180% from 18 to 32 when miss rate improves by eight times.

This indicates that both systems are more dependent on the performance of their DRAM and RRDRAM memory and that the memory gap between processor speed and memory speed is still a bottleneck for the performance of computer systems.

The result shows that if global miss rate is high, the scalability of both systems is drastically reduced, for the system with DRAM its scalability decreases from 12 to 1 by 1200%, when miss rate increases by eight times. The scalability of the system using RRDRAM deceases from 32 to 5 by 600% and is less dependent on miss rate.

7.8 Scalability of DRAM versus RRDRAM Systems Using different Memory Bus Bandwidth

Figure 10 shows the scalability of both systems when using different memory to processor bandwidth. Bus bandwidth changes from 2.5 GB/S to 20 GB/S by eight times. This causes transfer time Tf of a 8 B block to the cache from 6.4 ns to 51.2 ns.

The system using RRDRAM improves its scalability from 5 to 32 by about 600%. The system using DRAM improves its scalability from 2 to 7 by 350%. This indicates that RRDRAM system gains more from improving the memory bus bandwidth.

8 Conclusions

Multi-core based system performance depends on using suitable memory system able to provide it with low latency and high bandwidth. The conventional DRAM with one row per bank is not suitable to handle the requirements of a multi core that access memory with different addresses simultaneously. High bandwidth obtained by fast interface cannot help the performance of a multi core system when using conventional DRAM.

Our proposed RRDRAM is able to offer multi-core better scalability for data with irregular access patterns that have different addresses mapped to different physical rows and accessed in a pipelined fashion as if it is in one physical row. The scalability of the system using RRDRAM is multiple fold that of the same system using DRAM. Using RRDRAM also enables multiprocessor system for benefiting from faster processors and higher bandwidth of memory bus. The speed gap of processor memory becomes less severe when the system uses RRDRAM.

References

[1] D. Burger, J. R. Goodman, and A. Kagi, Memory Bandwidth of Future Microprocessors, *Proc. 23rd Ann. Int'l Symp. On Computer Architecture*, ACM Press, New York, 1996, pp. 78-89.

[2] J. Hennessy, and D.A. Patterson, *Computer Architecture: A Quantitative Approach*, Morgan Kaufmann Publishers, Inc, San Francisco, CA, 2003.

[3] Kohji Hosokawa, Toshio Sunaga, and Shinpei Watanabe, DRAM with Multiple Virtual Bank Architecture for Random Row Access, US Patent No. 6,925,028 B2 Aug. 2, 2005

[4] Bruce L. Jacob, Synchronous DRAM Architectures, Organizations, and Alternative Technologies 2002-12-10, Electrical & Computer Engineering Dept. University of Maryland College Park, MD 20742 http://www.ece.umd.edu/ blj/

[5] Sally A. McKee and Robert H. Klenke, Smarter Memory: Improving Bandwidth for streamed References, *IEEE Computer* , pp 54-63, July 1998.

[6] Nagi Mekhiel, Reconfigurable Row DRAM, US Patent 9,734,889 B2, Aug 15, 2017.

[7] Samuel Moore, Multicore Is Bad News For Supercomputers, *IEEE* Spectrum, Nov 2008.

[8] R. Murphy, On the Effects of Memory Latency and Bandwidth on Supercomputer Application Performance Workload Characterization, 2007. IISWC 2007. IEEE 10th International Symposium, pp.35-43, Sept 27-29, 2007.

[9] RAMBUS XDR DRAM, www.rambus.com, 2007 .

[10] SPEC CPU2000 Benchmark https://www.spec.org/cpu2000

[11] Wm. A. Wulf, and Sally A. McKee, Hiting the Memory Wall: Implication of the Obvious, *ACM Computer Architecture News* Vol. 23, No. 1, pp. 2024, 1995..

Nagi Mekhiel is a Professor in the Department of Electrical, Computer and Biomedical Engineering, Ryerson University, Toronto. His research interests are computer architecture, parallel processing, high performance memory systems, advanced processors, VLSI, and performance evaluation of computer systems. He holds many U.S. and World patents in memory and multiprocessors. He is conducting research to solve the fundamental problems facing computer industry, including scalability of parallel processors, and processor/memory speed gap.

A Survey of Data Clustering for Cancer Subtyping

Yan Yan*, Frederick C Harris, Jr.*
University of Nevada, Reno,
Reno, Nevada, USA. *

Abstract

Cancer subtyping remains a challenging task in microarray data analysis. The major goals of a successful cancer subtyping system are accuracy and reliability. Cluster analysis techniques have proven to be effective in this area. To facilitate further development in cancer subtyping based on microarray data, we provide a comprehensive review of the major cluster analysis algorithms from the clinical and computational domains that have been applied on microarray mRNA expression data and miRNA expression data for cancer subtyping, as well as other clustering algorithms with potential application in cancer subtyping.

Key Words: Algorithms, cancer subtype detection, data clustering, microarrays

1 Introduction

Clustering is an interdisciplinary research topic and is also known by researchers in different fields as unsupervised learning, exploratory data analysis, grouping, clumping, taxonomy, typology, and Q-analysis [138]. Cluster analysis is defined as 'a statistical classification technique for discovering whether the individuals of a population fall into different groups by making quantitative comparisons of multiple characteristics' and its first known use was in 1948 (Merriam-Webster Online Dictionary, 2013). The clustering algorithm was first developed by biologists in numerical taxonomy study in 1963 before being utilized by statisticians [134]. Clustering is used for class discovery, *i.e.* exploration or discovery of the underlying patterns of a dataset by separating the dataset into groups, with little or no prior knowledge [86, 136, 254, 255]. Clustering is also used for natural classification, *i.e.* identifying the degree of similarity among organisms, and compression, *i.e.* organizing and summarizing data using cluster prototypes [138]. Clustering has become increasingly popular as society increasingly generates an overwhelming amount of data, and it is often used as the first step in data analysis or as a preparation step for experimental work [163, 256].

There is no universally agreed upon definition of clusters [86]. A cluster is a set of objects that are compact (or similar to each other) and isolated (or dissimilar) from other clusters. In reality, cluster definition is subjective, and its significance and interpretation requires related domain knowledge [138]. Similarity measure is used by clustering methods to calculate the similarity between two objects. Different similarity measures will have different clustering results, as some objects may be similar to one another using one measure but dissimilar using another. Similarity between two objects can be measured in different ways, and the three dominant methods are distance measures, correlation measures, and association measures [134]. Common similarity measures include Euclidean distance, Manhattan distance, Maximum norm, Mahalanobis distance, Pearson coefficient, Spearman's rank correlation coefficient, angle between two vectors, and the Hamming distance.

Since the process of clustering is subjective, judging the relative efficacy of clustering methods is difficult [20, 139]. Cluster validity is used to assess clustering results and can be classified into three categories: a) Internal validities formulate quality as a function of the given data set [130]. Examples include Dunn's Validity Index, Silhouette Value, Hubert Gamma Statistic, Entropy, Xie-Beni, Normalized Mutual Information. b) External validities assess quality by additional external information such as category labels [130]. Examples include Jaccard Index, Rand Index, Adjusted Rand Index, Variation of Information, Kappa Statistic, CA. c) Relative validities evaluate a clustering result by comparing it to results from other clustering methods.

The procedure of cluster analysis includes four steps [254]: Step one is feature selection or extraction. Feature selection selects a subset of all features, and feature extraction generates novel features from the original ones by using some transformations [31, 135, 139, 254]. Step two is clustering algorithm design or selection. Since clustering algorithms group objects based on some proximity measure, this step usually includes choosing an appropriate proximity measure and construction of a clustering criterion function, creating an optimization problem that has been well studied in the literature. Step three is cluster validation. This step calculates a confidence level for the clustering results. Step four is results interpretation. This step provides meaningful insights from the data.

There is no single clustering algorithm that performs best across all problems or data sets [152, 254]. Therefore, it is important to study the characteristics of the problem and use an appropriate clustering strategy [254].

Properties to be considered in choosing a clustering algorithm include [28]: a) feature type (numeric and non-numeric), b) scalability (large datasets), c) handling high dimensional data, d) finding clusters of irregular shape, e) handling outliers, f) time complexity of the algorithm, g) data order dependency, h) assignment type (hard or strict vs. soft or fuzzy), i) prior

*Department of Computer Science and Engineering. Email: fred.harris@cse.unr.edu

knowledge and user defined parameters dependency, and j) interpretability and visulization of results.

Despite many examples of successful applications of cluster analysis, there still remain many challenges due to the existence of many inherent uncertain factors [254]. The following fundamental challenges in clustering [136, 138] are relevant even today [138]: a) definition of a cluster, b) selection of features, c) normalization of the data, d) outlier detection, e) definition of pair-wise similarity, f) number of clusters, g) selection of clustering method, h) existence of clustering tendency, and i) validity of the clusters.

Some recent trends in clustering include [138]: semi-supervised clustering utilizing external or side information; interactive clustering, where a user can specify or change program parameters based on domain knowledge or results from previous clustering iterations; clustering ensembles, where the partitions resulting from different algorithms (or the same algorithm with different parameters) are combined; multi-objective clustering, where the clustering algorithm optimizes multiple specific objectives; large-scale clustering, which handles very large databases; multi-way clustering, which extends the bi-clustering framework and simultaneously clusters heterogeneous components of the data objects [26]; and heterogeneous data clustering for data comprising multiple types, such as rank data, dynamic data, graph data, and relational data [134].

Clustering techniques can be organized into categories. Different criteria may result in different categories of clustering algorithms [254]. Furthermore, categorization of clustering algorithms is not straightforward or canonical, and categories can overlap [28]. For convenience, in this review we use the following taxonomy, which is also widely used in the literature: hierarchical clustering (Section 2), partitioning clustering (Section 3), graph-based clustering (Section 4), distribution-based clustering (Section 5), density-based clustering (Section 6), grid-based clustering (Section 7), clustering big data (Section 8), clustering high dimensional data (Section 9), and other clustering techniques (Section 10).

2 Hierarchical Clustering

Hierarchical clustering algorithms organize a data set into a hierarchical structure according to a similarity measure [254]. It is based on the belief that nearby objects are more related than objects that are farther away [183]. These algorithms connect objects based on their similarity to form clusters, which is usually represented using a dendrogram. Hierarchical clustering algorithms differ in the choice of similarity measures, the linkage criterion (distance between clusters), and whether the process is agglomerative (bottom-up) or divisive (top-down). Agglomerative hierarchical clustering starts with singleton clusters and then recursively merges appropriate clusters, and divisive hierarchical clustering starts with one cluster containing all objects and recursively splits appropriate clusters [28].

Divisive clustering is very expensive in computation [86] and is not commonly used in practice [254]. We focus on the agglomerative clustering first and then mention two divisive clustering algorithms named MONA and DIANA [146, 254].

There are many agglomerative hierarchical clustering algorithms based on different linkage criterion. The single linkage method or nearest neighbor method [110, 136, 215, 220, 221, 254] uses the distance between two closest objects in different clusters, and the shortest distance determines the merge of two clusters. The complete linkage method or farthest neighbor method [67, 149, 223, 254] uses the distance between two farthest objects in different clusters, and the shortest distance determines the merge of two clusters. These two methods are the simplest and most popular [254]. Average linkage methods include UPGMA (Unweighted Pair-Group Method using Arithmetic averages), WPGMA (Weighted Pair-Group Method using Arithmetic averages), UPGMC (Unweighted Pair Group Method using Centroids), and WPGMC (Weighted Pair Group Method using Centroids). UPGMA and UPGMC use a simple average, while WPGMA and WPGMC use a weighted average where the weight is the inverse of cluster size. UPGMA [63, 87, 136, 220, 222] uses average distance between two objects in different clusters, and the shortest average distance determines the merge of two clusters. WPGMA or weighted average linkage method [182] uses weighted average distance between two objects in different clusters, and the shortest average distance determines the merge of two clusters. UPGMC or centroid linkage method [220] uses Euclidean distance between unweighted centroids (calculated by arithmetic mean) of different clusters, and the shortest distance determines the merge of two clusters. WPGMC or median linkage method [220] uses Euclidean distance between weighted centroids of different clusters, and the shortest distance determines the merge of two clusters. Minimum-variance method or Ward's method [245] considers the relationship of all objects in a cluster. Its objective is to form clusters such that the increase of variance within each group is minimized [247]. Further readings about these methods include [86, 254, 259].

What follows are examples of divisive hierarchical clustering algorithms. DIANA [146] (DIvisive ANAlysis Clustering) selects in each dividing step the cluster with the largest diameter and divides it into two new clusters. MONA [146] (MONothetic Analysis Clustering of Binary Variables) divides clusters based on a single well-chosen variable (or feature), whereas most other hierarchical methods use all variables (or features).

Advantages of hierarchical clustering are a) Good visualization with dendrogram representation [136, 231, 254, 256], b) Very informative descriptions with dendrogram representation [136, 231, 254, 256], and c) Flexibility regarding the number of clusters, since the clustering results can be obtained by cutting the dendrogram at different levels.

Disadvantages of hierarchical clustering are [254, 256]: a) Lacking of robustness and sensitivity to noise and outliers. b) High computational complexity, which limit their application on large scale data. c) Tendency to form clusters with

spherical shapes instead of natural shapes. d) Prone to reversal phenomenon [189].

BIRCH [269] (Balanced Iterative Reducing and Clustering using Hierarchies) clusters incoming data objects incrementally and dynamically. It first builds a CF (Clustering Feature) tree dynamically as new data objects are inserted and then applies an agglomerative hierarchical clustering algorithm to the nodes represented by their CF vectors. After obtaining a centroid for each cluster, it assigns each data object to its nearest centroid. CURE [112] (Clustering Using REpresentatives) uses a number of representative data points in a cluster to evaluate the distance between clusters. Closest cluster pair are merged at each step of its hierarchical clustering process. ROCK [113] (RObust Clustering using linKs) uses links and not distances when merging clusters for boolean and categorical data. DISMEA [224] uses the k-means algorithm to divide a cluster into two clusters. The Edwards and Cavalli-Sforza Method [79] divides all available clusters at each step. Minimum Spanning Tree-based clustering algorithms [80, 190, 266] construct an MST (Minimum Spanning Tree) [156, 185, 200] from a data set and produce a group of clusters by removing selected edges. Figure 1 shows an example of hierarchical clustering.

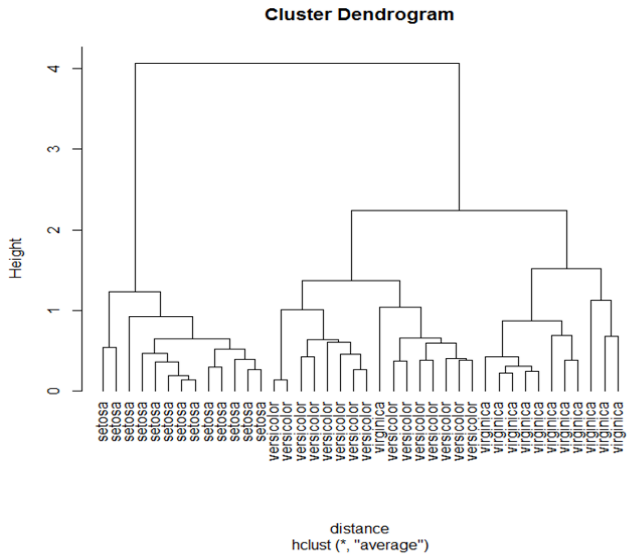

Figure 1: Hierarchical clustering [124]

3 Partitioning Clustering

Partitioning clustering algorithms divide objects into clusters without hierarchical structure. Clusters are represented by a central vector. Given the number of clusters, partitioning clustering assigns the objects to the closest cluster center. Partitioning algorithms can be grouped into k-means methods and k-medoids methods. k-means methods use the centroid of objects within a cluster as center. k-medoids methods use the most appropriate object within a cluster as center.

K-means clustering [28, 93, 120, 121, 167, 226, 254, 256] is very simple, but one of the best known and popular clustering algorithms. There are many variations of the basic k-means clustering. Classic k-means reassigns data objects based on optimization of the objective function. If a reassigning has a positive effect, the data object is reassigned and the cluster centers are updated. ISODATA [19] (Iterative Self-Organizing Data Analysis Technique) splits and merges intermediate clusters based on a user-defined threshold and iterates until the threshold is reached. FORGY [93] reassigns objects to nearest centroids and recomputes centroids. It iterates until a stopping criterion is achieved. Fuzzy c-means [29, 77] assigns fuzzy cluster membership to each data object, and updates cluster centers and membership after each iteration. Methods to speed up k-means and fuzzy C-means such as brFCM (bit reduction by Fuzzy C-Means) [83] replace similar data objects with their centroid before clustering.

Variations of k-medoid [146] methods are as follows. PAM (Partitioning Around Medoids) assigns each data object to the closest medoid and iteratively reassigns objects and updates medoids to optimize the objective function. CLARA (Clustering LARge Applications) [146] applies PAM on multiple subsets or samples of the data set, and selects the best clustering as output. CLARANS (Clustering Large Applications based upon RANdomized Search) [187] searches a graph where each node is a set of medoids. It selects a node randomly in search for a local minimum among its neighbor nodes through iterations and outputs the best node to form clustering results.

Advantages of partitioning clustering are a) simple, straightforward and easy implementation, b) fast execution with computation complexity of $O(n)$, c) very suitable for compact and hyperspherical clusters, d) computational rigor (firm foundation of analysis of variances).

Disadvantages of partitioning clustering are a) they are still subjective processes that are sensitive to assumptions, b) they require the number of clusters to be specified in advance, c) they prefer clusters of approximately similar size, as they will always assign an object to the nearest center, often leading to incorrectly cut borders in between clusters, d) they are subject to easy trapping in local minima and sensitivity to the initial partition (hill-climbing optimization method).

Other developments are as follows. Bisecting k-means [225] recursively partitions a cluster into two. KD-trees k-means [195] uses the KD-Tree data structure to speed up the assignment of data objects to their closest cluster by reducing the number of nearest-neighbor queries in the traditional algorithm. Scaling k-means [37] retains important data objects and summarizes or discards other objects. Centroids of the resulting data set are then used on the whole data set. X-means [196] finds the number of clusters K automatically by optimizing a criterion function such as AIC (Akaike Information Criterion) or BIC (Bayesian Information Criterion). Kernel k-means [209] enhances k-means by using a kernel function that nonlinearly maps the original feature space to

a higher dimensional one, where clusters are more separable. Weighted kernel k-means [71] further extends kernel k-means by assigning a weight for each cluster. The weight is defined as the reciprocal of the number of data objects in the cluster. GA k-means [16] applies a genetic algorithm to improve cluster centers initialization for k-means. Simulated annealing [7, 132, 151, 211] uses simulated annealing optimization to avoid local optima and find the global minimum solution. Soft assignment [267] assigns data objects to different clusters with appropriate weights to improve the optimization process. It uses Harmonic Averages of the distances from the data object to all the centers. Mahalanobis distance [170] is used to detect clusters with hyperellipsoidal shapes. Maximum of intra-cluster variances [109] can be used as the objective function instead of the sum to obtain good clustering results. K-prototypes [131] incorporates categorical data as a generalization approach. Accelerated k-means by triangle inequality [81] avoids unnecessary distance calculations by using the triangle inequality and keeping track of lower and upper bounds for distances between data objects and cluster centers. K-means++ [12] improves the speed and the accuracy of k-means by using a simple randomized seeding technique. Figure 2 shows an example of partitioning clustering.

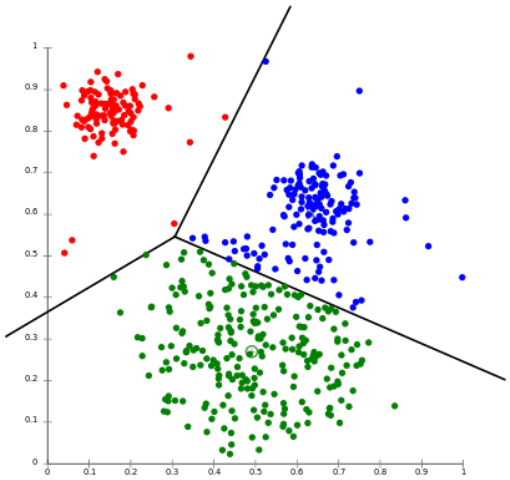

Figure 2: Partitioning clustering [57]

4 Graph-based Clustering

Graph-based clustering algorithms construct a graph/hypergraph from the data and then partition the graph/hypergraph into subgraphs/subhypergraphs or clusters. Each vertex represents a data object, and the edge weight represents the similarity of two vertices [50]. The edges in the same subgraph/subhypergraph should have high weights, and the edges between different subgraphs/subhypergraph should have low weights [50]. It is also called spectral clustering [138].

Representative algorithms are as follows. Chameleon [143] uses a connectivity graph and graph partitioning to build small clusters, followed by the agglomerative hierarchical clustering process. Its key feature is that it considers both interconnectivity and closeness when merging clusters. CACTUS (Clustering Categorical Data Using Summaries) [100] detects candidate clusters based on the summary of the data set and determines the actual clusters through a validation process against the candidate clusters. It uses a similarity graph to represent the inter-attribute and intra-attribute summaries [98]. A Dynamic System-based Approach or STIRR (Sieving Through Iterated Relational Reinforcement) [106] represents each attribute value as a weighted vertex in a graph. It iteratively assigns and propagates weights until a fixed point is reached. Different weight groups correspond to different clusters on the attribute. ROCK (Robust Clustering algorithm for Categorical Data) [113] repeatedly merges two clusters until the specified number of clusters is reached, and it uses data sampling to improve complexity. It uses a connectivity graph to calculate the similarities between data objects [98].

The advantages of graph-based clustering are [50]: a) A graph is an elegant data structure that can model many real applications. b) It is based on solid mathematical foundations, including spectral theory and Markov stochastic process. c) It produces optimal clustering (optimizing a quality measure instead of acting greedily toward the final clustering).

The major disadvantage of graph-based clustering is that it may be slow when working on large scale graphs [50].

Other developments are as follows. The Ratio Cut algorithm [117] adopts a cluster size constraint, which is the number of data points in a cluster. The Normalized Cut (NCut) algorithm [214] is an approximate graph-cut based clustering algorithm with a cluster size constraint, which is the volume of the cluster or sum of edge weights within a cluster. It also has a multiclass version [264]. The MNCut (Modified Normalized Cut) algorithm [174] gives a new interpretation to the NCut algorithm in the framework of a Markov Random Walk. Ng's method [186] derives a new data representation from normalized eigenvectors of a kernel matrix simultaneously and in a particular manner. Laplacian Eigenmap [27] uses the eigenvectors of the graph Laplacian to represent data. Pairwise Data Clustering by Deterministic Annealing [126] uses proximity measures between the data objects to represent data. Dominant Sets Pairwise Clustering [191] relates clusters to maximal dominant sets [180] in pair-wise clustering. Fast approximate spectral clustering [260] applies a distortion-minimizing local transformation to the data to speed up conventional spectral clustering. Active spectral clustering [243] follows the concept of constrained clustering and uses pairwise relations. Its constraints are specified in a incremental manner. Locally-scaled spectral clustering using empty region graphs [60] employs β-skeleton (a subset of empty region graphs) and non-linear diffusion to define a locally adapted affinity matrix which

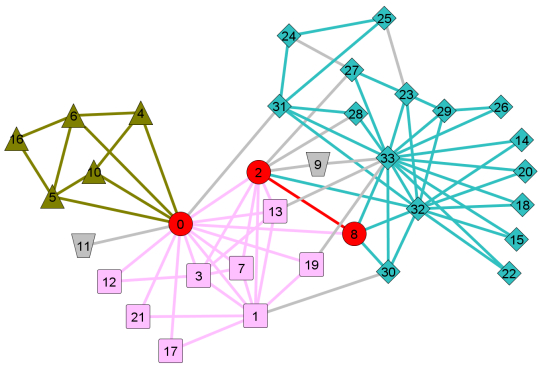

Figure 3: Graph-based clustering [85]

defines the similarity of two data objects. Figure 3 shows an example of graph-based clustering.

5 Distribution-based Clustering

Distribution-based clustering views or assumes that the data are generated by a mixture of probability distributions, each of which represents a different cluster [99, 172]. This way, a cluster can be seen as objects generated by the same distribution. Thus, a particular clustering method can be expected to produce good results when the data conform to the method's distribution model [99]. It is also called model-based clustering. There are usually two approaches to form the model: the classification likelihood approach and the mixture likelihood approach [99].

Distribution-based clustering has a long history. Early works include [30, 65, 210, 249]. A survey of cluster analysis in a probabilistic and inferential framework is presented in [33].

Representative algorithms are as follows. The EM (Expectation-Maximization) clustering algorithm [69] is the most popular method in distribution-based clustering. It tries to fit the data set into the assumed number of Gaussian distributions by moving the means of Gaussian distributions toward the cluster centers. COOLCAT (reducing the entropy, or COOLing of the CATegorical data clusters)[22] uses entropy to cluster categorical data. It consists of data sampling and incremental assignment. STUCCO (Search and Testing for Understandable Consistent Contrasts) [25] uses tree searching and significant contrast-sets to find clusters. GMDD (Gaussian Mixture Density Decomposition) [271] uses a recursive approach and identifies each Gaussian component in the mixture successively. Autoclass [49] is based on the classic distribution-based approach and uses a Bayesian method to determine the optimal clusters. P-AutoClass [198] is a parallel version of Autoclass and can be used on large data sets.

The advantages of distribution-based clustering are as follows [28]: a) It can be modified to handle complex data, b) It has a solid theoretical foundation, c) Its results are easily interpretable, d) It not only provides clusters, but also produce complex models that capture relationships among attributes, e) Results are independent of the timing of consecutive batches of data, f) It is good for online learning since the intermediate mixture model can be used to cluster objects, g) the Mixture model can be naturally generalized to cluster heterogeneous data.

The disadvantage of distribution-based clustering is the difficulty in choosing the appropriate model complexity (since a more complex model will usually be able to explain the data better but may cause an overfitting problem from excessive parameter set).

Other developments are as follows. Latent Dirichlet Allocation (LDA) [32] uses a hierarchical Bayesian model that has three levels. Each data object is modeled as a finite mixture over an underlying set of groups (or clusters) of objects. Each group (or cluster) is modeled as an infinite mixture over a set of group (or cluster) probabilities. Pachinko Allocation Model (PAM) [161] uses a Directed Acyclic Graph (DAG) to model cluster correlations. The leaves of the DAG represent data objects, and the interior nodes represent correlations. Undirected graphical model for data clustering [246] is based on exponential family distributions and the semantics of undirected graphical models. It uses the technique of minimizing contrastive divergence to speed up the process. Robust cluster analysis via mixture models method [173] uses the mixtures of multivariate t distributions approach to the clustering. It also uses the t distribution to cluster high-dimensional data via mixtures of factor analyzers. Online learning for LDA method [125] is an online Variational Bayes (VB) algorithm for LDA. It uses natural gradient step in online stochastic optimization, which converges to a local optimum of the VB objective function. Figure 4 shows an example of distribution-based clustering.

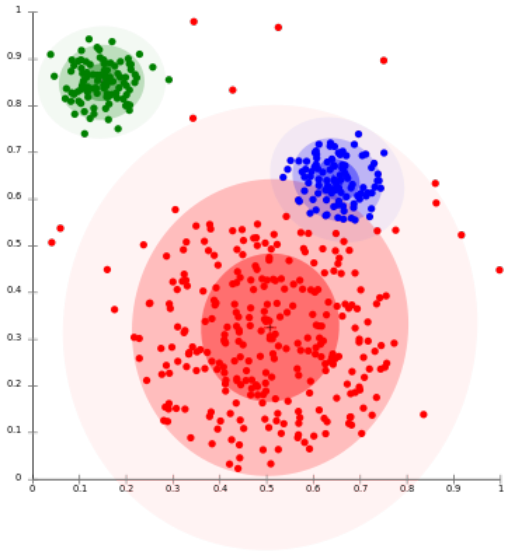

Figure 4: Distribution-based clustering [56]

6 Density-based Clustering

Density-based clustering defines clusters as dense regions of data objects separated by low-density regions. A cluster is a connected dense component and grows in any direction that density leads [99]. Objects in low-density areas which separate clusters are usually considered to be noise and border points. There are two major approaches for density-based clustering [28]: the connectivity approach pins density to a training data point; the density function approach pins density to a point in the attribute space.

Representative algorithms for the connectivity approach are as follows. DBSCAN (Density-Based Spatial Clustering of Applications with Noise) [84] starts by selecting a data object and tries to find all data objects density-reachable from it to form a cluster. If none are found, the algorithm selects a new data point and repeats. GDBSCAN (Generalized DBSCAN) [205] generalizes the concept of neighborhood by permitting the use of any distance function besides Euclidian distance and allows other measures besides simply counting the objects to define the cardinality of that neighborhood. OPTICS (Ordering Points To Identify the Clustering Structure) [11] is like an extended DBSCAN algorithm. It does not assign cluster memberships but stores the order in which the data objects are processed as well as the core-distance and a reachability-distance for each data object. An extended DBSCAN is used to assign cluster memberships. DBCLASD (Distribution Based Clustering of LArge Spatial Databases) [257] uses the notion of clusters based on the distance distribution and incrementally augments an initial cluster by its neighboring points as long as the nearest neighbor distance set of the resulting cluster still fits the expected distance distribution.

Representative algorithms for the density function approach are as follows. DENCLUE (DENsity-based CLUstEring) [122] calculates the impact of each data object within its neighborhood (*i.e.* influence function) and determines clusters mathematically by identifying local maxima of the overall density function (*i.e.* density-attractors).

The advantages of density-based clustering are as follows [28, 99]: a) They can find clusters of arbitrary shapes, in contrast to many other methods. b) Time complexity is low (linear or $O(n)$). c) It is deterministic for core and noise points (but not for border points), therefore there is no need to run it multiple times. d) It can handle noise well. e) The number of clusters is not required, since it finds clusters and the number of clusters automatically. f) Results are independent of data ordering. g) There are no limitations on the dimension or attribute types.

The disadvantages of density-based clustering are as follows: a) It is often difficult to detect cluster boarders when the cluster density decreases continuously (*i.e.* arbitrary borders). b) For a mixtures of Gaussians data set, distribution-based clustering (*e.g.* EM) usually outperforms density-based clustering. c) Limitations in processing high-dimensional data, since it is difficult to distinguish high-density regions from low-density regions when the data is high-dimensional [138]. d) Most density-based clustering algorithms were developed for spatial data [99].

Other developments are as follows. BRIDGE [64] integrates the k-means algorithm and the DBSCAN algorithm. K-means is first performed, and then DBSCAN is used on each partition. Finally, results are improved by removing the noise found by DBSCAN. Jarvis-Patrick algorithm [94] partitions the data set into clusters based on the number of shared nearest neighbors. It first identifies the k nearest neighbors of each data object and then merges two data objects at a time. C-DBSCAN (Constrained-DBSCAN) [204] enhances the DBSCAN algorithm with pairwise constraints. SCAN (Structural Clustering Algorithm for Networks) [258] can detect hubs and outliers, in addition to clusters in networks (or graphs). It uses a structural similarity measure to cluster vertices. Figure 5 shows an example of density-based clustering.

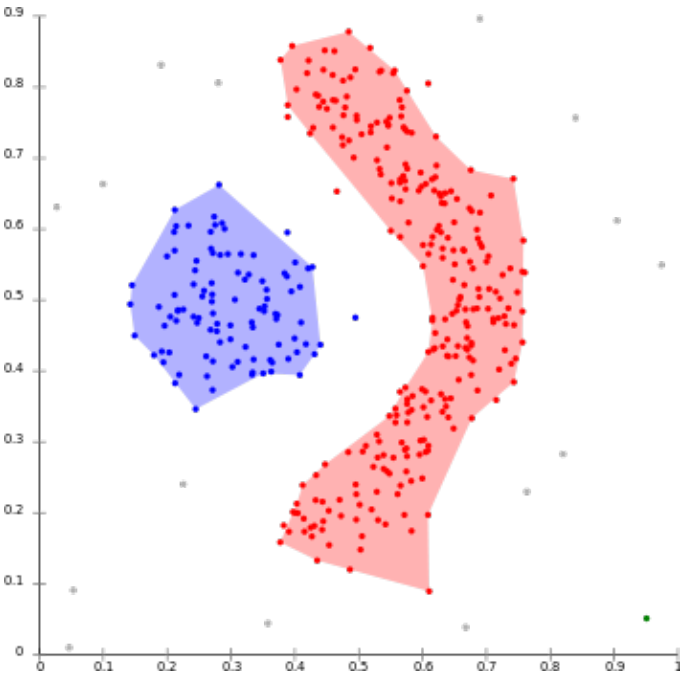

Figure 5: Density-based clustering [55]

7 Grid-based Clustering

Grid-based clustering operates on space partitioning instead of data partitioning to produce clusters [28]. It first creates the grid structure by partitioning the data space into cells (or cubes) and then clusters the cells based on their densities.

Representative algorithms are as follows. BANG-clustering [28, 207] uses a multi-dimensional grid data structure to organize or partition the data. It uses the cell information in the grid and clusters the cells. STING (A STatistical INformation Grid approach) [241] uses a hierarchical structure of grid cells with a top-down approach. It labels a cell to be relevant or not at a specified confidence level. Then, it finds all the regions formed by relevant cells. STING+ [28,

242] uses a similar hierarchical cell structure as STING and introduces an active spatial data mining approach. OptiGrid (Optimal Grid) [123] constructs an optimal grid partitioning of the data by finding the best partitioning hyperplanes for each dimension with projections of the data. GRIDCLUS (GRID-CLUStering) [206] organizes the space surrounding the clusters with a grid data structure. It uses a topological neighbor search to cluster the grid cells. GDILC (Grid-based Density-IsoLine Clustering) [261] is based on the idea that the density-isoline figure reflects the distribution of data. It uses a grid-based approach to calculate the density and finds dense regions. WaveCluster (Wavelet-based clustering) [212] transforms the original feature space by applying wavelet transform and then finds the dense regions in the new space. It yields sets of clusters at different resolutions and scales, which can be chosen based on the user's needs. FC (Fractal Clustering) [21] adds one data object at a time to one cluster in such a way that the fractal dimension changes the least after adding the data object.

The advantages of grid-based clustering are as follows [28, 99]: a) It is fast and works well with large data sets (since speed is independent of the number of objects in the data), b) It handles noise well, c) It is independent of data ordering, d) It can handle attributes of different types, e) It can be used as an intermediate step in many other algorithms such as CLIQUE and MAFIA.

The disadvantages of grid-based clustering are as follows: a) Most algorithms need the user to specify grid size or density thresholds, which can be difficult (fine grid sizes result in high computational time, while coarse grid sizes result in low quality of clusters) [99]. b) Some grid-based clustering algorithms (*e.g.* STING, WaveCluster) are not good at high dimensional data [99].

Other developments are as follows. AMR (Adaptive Mesh Refinement clustering) [162] creates grids at multiple resolutions where higher resolution grids are applied to the localized denser regions. O-Cluster (Orthogonal partitioning CLUSTERing) [175] is a variant of OptiGrid. It creates a hierarchical grid-based structure by making axis-parallel (orthogonal) partitions on the input data. It operates recursively, and the final irregular grid frames the data into clusters. CBF (Cell-Based Filtering) [47] splits each dimension into a set of partitions using a filtering-based index. It then creates cells based on the overlapping regions of the partitions. PGMCLU (Parallel Grid-based CLUstering algorithm for Multi-density datasets) [251] consists of parallel data partitioning, local clustering, and merging local clusters. It introduces a new measure called grid compactness for the degree of tightness between data objects within the grid, and the notion of grid feature for summarizing the information about a grid. Figure 6 shows an example of grid-based clustering.

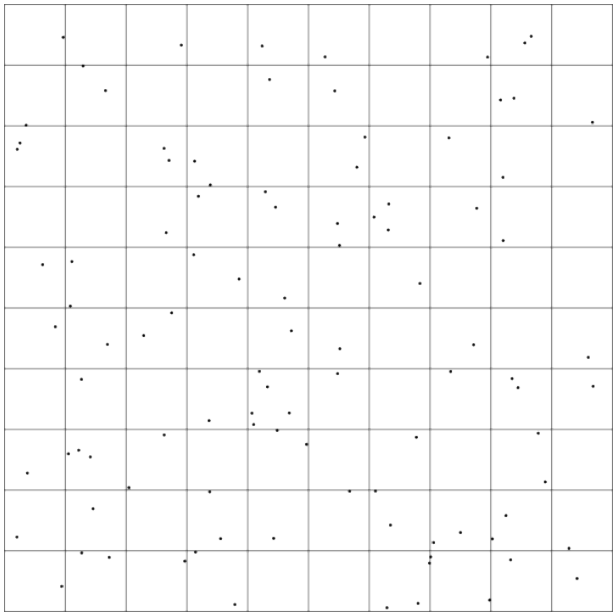

Figure 6: Grid-based clustering [263]

8 Clustering Big Data

Big data clustering refers to clustering on millions of data objects [138]. These algorithms need to have good scalability and process big data within reasonable computing time and memory space [28]. A high computational complexity would dramatically limit an algorithm's application to big data. The strategies used for big data clustering can be categorized into sampling, data summarization, distributed computing, and incremental learning.

8.1 Sampling

Sampling methods select a sample of the original large data set and perform clustering over the sample data. Old-fashioned sampling methods may or may not use rigorous statistical reasoning. Newer sampling methods use special uniform checks to control their adequacy [28]. Advantages are that it is simple to implement and can screen out most outliers. However, small clusters may be missed.

Examples are as follows. CURE (Clustering using REpresentatives) [112] and ROCK (RObust Clustering using linKs) [113] were covered in Section 2. CLARA (Clustering LARge Applications) [145] draws several samples from the data set, runs PAM on each of them, and selects the best result. CLARANS (Clustering Large Applications based on RANdomized Search) [187] starts with a new randomly-selected node (a set of k potential medoids) in the graph in search of the local optimum. It repeats if a local optimum is found.

8.2 Data Summarization

Data summarization methods calculate data summary statistics and perform clustering on the summaries instead of the original data. The advantage is that the requirement for the storage of and frequent operations on the large amount of data are greatly reduced, saving both computational time and storage

space. The disadvantage is reduced cluster quality.

Examples are as follows. BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies) was covered in Section 2. BUBBLE [101] instantiates generalized BIRCH for data in a distance space. BUBBLE-FM (BUBBLE-FastMap) [101] improves upon BUBBLE by reducing the computation time using FastMap [88]. EMADS (EM Algorithm for Data Summaries) [141] directly generates a Gaussian mixture model from simplified data summaries. bEMADS (BIRCH's EMADS) [141] uses data summarization procedures in the BIRCH algorithm.

8.3 Distributed Computing

Distributed computing methods divide a large data set into smaller data sets and perform clustering on each smaller data set. The advantage is that clusterings on each smaller data set can be done in parallel to reduce the overall computation time [138]. The disadvantage is the overhead and complexities due to the dividing and combining steps.

Examples are as follows. Parallel k-means [70] is a parallel implementation of the k-means clustering algorithm. DBDC (Density Based Distributed Clustering) [140] clusters distributed data locally and extracts suitable representatives from these local clusters to send to a global site where the complete clusters are restored based on the local representatives. It uses a density-based clustering algorithm for both local and global clustering. Parallel spectral clustering in distributed systems [51] makes the dense similarity matrix sparse by retaining nearest neighbors using a parallel approach.

8.4 Incremental Learning

Incremental learning methods process one data object at a time and may discard it. They require only one single pass over all data objects, in contrast to most clustering methods that require multiple passes over data objects before identifying the cluster centers [138]. Advantages are: improved clustering efficiency in terms of data storage and processing time (they can admit new data objects without learning from scratch [256]); handling outliers well [28]; resumable processing which makes it very suitable for dynamic big data sets [28]. Disadvantages are that results depend on data order and may not be stable [43, 178, 256], and can result in lower quality clusters [28].

Examples are as follows. DIGNET [232, 244] moves cluster centers toward a new data point with each new addition. Hartigan's leader algorithm [120] uses a distance/similarity threshold to decide if a data point should be added to the cluster or used for a new cluster center. ART (Adaptive Resonance Theory) family [42, 256] simulates neural circuits that are believed to trigger fast learning. It includes a large family of neural network variants such as ART1 [43], ART2 [42], Gaussian ART [248], Bayesian ART [236], Ellipsoid ART [9], ART tree [45, 250], ARTMAP [44], Q-learning ART [38], Fuzzy ART [41]. Charikar's incremental clustering [48] maintains a clustering of the data objects so that the maximum cluster diameter is minimized as new data objects are added. GenIC (Generalized Incremental algorithm for Clustering) [114] divides the data stream into chunks or windows, updating each cluster center with each new data object addition and merging clusters at the end of a window of data. Cobweb [91] is an incremental system for hierarchical clustering, which enables bi-directional hill-climbing search through the space of hierarchical schemes.

9 Clustering High Dimensional Data

High Dimensional Data clustering refers to clustering on data objects that represent from a few dozen to thousands or more features. Such high dimensional data are often seen in areas such as medicine (*e.g.* microarray experiments), and text documents (*e.g.* word-frequency vector methods [46]). Clustering high dimensional data is tremendously difficult. One problem is that increased irrelevant features eliminate the likelihood of clustering tendency [28]. Another problem is the 'curse of dimensionality', or lack of data separation, in high dimensional space (the problem becomes severe for dimensions greater than 15) [28]. Performing feature selection before applying clustering can improve the first problem. Principal Component Analysis (PCA) [193] is commonly used. However, the dimension may still be high after feature selection. In this review, we discuss techniques that have been developed to address such situations: projected clustering, subspace clustering, bi-clustering (or co-clustering), tri-clustering, hybrid approaches, and correlation clustering.

9.1 Projected Clustering

Projection techniques map data objects from a high dimensional space to a low dimensional space, while maintaining some of the original data's characteristics [13].

Examples are as follows. PreDeCon [34] finds subsets of feature vectors that have low variance along subsets of attributes. PROCLUS [3] finds the candidate clusters and dimensions by using medoids. For each medoid, the subspace is determined based on attributes with low variance. Random projections for k-means clustering [36] implements a dimensionality reduction technique for k-means clustering based on random projections.

9.2 Subspace Clustering

Subspace clustering algorithms identify clusters in appropriate subspaces of the original data space.

Examples are as follows. CLIQUE (CLustering In QUEst) [5] partitions the data space into units and then finds the maximum sets of connected dense units. SUBCLU (density-connected Subspace Clustering) [155] adopts the notion of density-connectivity introduced in DBSCAN (Section 6) and uses the monotonicity of density-connectivity to prune subspaces. CACTUS (Clustering Categorical Data Using

Summaries) is covered in Section 4. ENCLUS (ENtropy-based CLUStering) [53] finds clusters in subspaces based on entropy values of subspaces. Subspaces with lower entropy values typically have clusters. It then applies CLIQUE or other clustering algorithms to such subspaces. MAFIA (Merging of Adaptive Finite Intervals) [108] uses adaptive grids in each dimension and then merges them to find clusters in higher dimensions. OptiGrid (Optimal Grid) is covered in Section 7. MrCC (Multi-resolution Correlation Cluster detection) [58] constructs a novel data structure based on multi-resolution and detects correlation clusters by identifying initial clusters as axis-parallel hyper-rectangles with high data densities, followed by merging overlapping initial clusters. Figure 7 shows an example of subspace clustering.

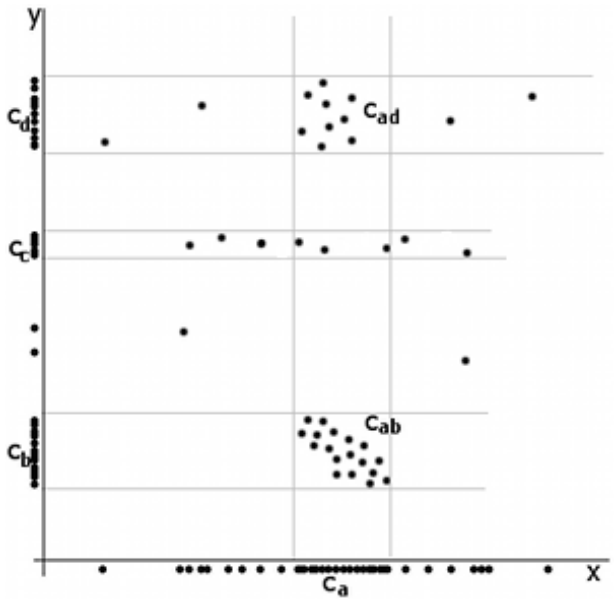

Figure 7: Subspace clustering [216]

9.3 Hybrid Approaches

Hybrid approaches find overlapping clusters. Some of them find only potentially interesting subspaces and use full-dimensional clustering algorithms to obtain the final clusters.

Examples are as follows. DOC (Density-based Optimal projective Clustering) [201] uses a global density threshold to compute an approximation of an optimal projective cluster. FIRES (FIlter REfinement Subspace clustering) [154] first computes one-dimensional clusters and then merges them by applying 'clustering of clusters' based on the number of intersecting points between clusters. P3C (Projected Clustering via Cluster Cores) [176, 177] first computes intervals matching or approximating higher-dimensional subspace clusters on every dimension and then aggregates those intervals into cluster cores. The cluster cores are refined and used to assign data objects.

9.4 Bi-clustering

Bi-clustering is also called bi-dimensional clustering [54], co-clustering, coupled clustering, or bimodal clustering. Bi-clustering is popular in bioinformatics research, especially in gene or sample clustering. For gene expression data, there are experimental conditions in which the activity of genes is uncorrelated. This causes limitations for results obtained by standard clustering methods. So bi-clustering algorithms that can perform simultaneous clustering on the genes and conditions are developed to find subgroups of genes and subgroups of conditions in which the genes exhibit highly correlated activities for every condition [168].

Examples are as follows. CTWC (Coupled Two-Way Clustering) [104] generates submatrices by an iterative process and considers only those submatrices whose rows and columns belong to genes and samples/conditions that were in a stable cluster in a previous iteration. ITWC (Interrelated Two-Way Clustering) [230] clusters the rows and then clusters the columns, based on each row cluster. It keeps the cluster pairs that are most dissimilar. Block Clustering [120] sorts the data by row mean or column mean and splits the rows or columns such that the variance within each 'block' is reduced. It then repeats and splits rows or columns differently. δ-biclusters [54] or CC algorithm (Cheng and Church's) finds biclusters whose rows and conditions show coherent values, using mean-squared residue. SAMBA (Statistical-Algorithmic Method for Bicluster Analysis) [229] uses probabilistic modeling and graph theoretic techniques to find subsets of rows whose values are very different in a subset of columns. Plaid Models [159] allows biclusters to overlap, *i.e.* a gene or a sample/condition can belong to more than one cluster. Information-theoretic co-clustering [72] intertwines the row and column clusterings to increase mutual information.

9.5 Correlation Clustering

Correlation clustering uses the correlations among attributes to guide the clustering process. These correlations may be different and exist in different clusters and cannot be reduced to uncorrelated ones by traditional global decorrelation techniques. Such correlations create clusters with different spatial shapes, and local correlation patterns are used to define the similarity between data objects. Correlation clustering is closely related to biclustering.

Examples are as follows. ORCLUS (ORiented projected CLUSter generation) [4] is similar to k-means but uses a distance function based on an eigensystem, *i.e.* the distance in the projected subspace. The eigensystem is adapted during iterations and close pairs of clusters are merged. 4C (Computing Correlation Connected Clusters) [34] takes a density-based approach and uses a density criterion to grow clusters. The density criterion is the minimal number of data objects within the neighborhood of a data object. The neighborhood is based on distance between two data objects in the eigensystems. HiCO (Hierarchical COrrelation clustering) [2] defines the

similarity between two data objects based on their local correlation dimensionality and subspace orientation. It takes a hierarchical density-based approach to obtain correlation clusters. CASH (Clustering in Arbitrary Subspaces based on the Hough transform) [1] is based on the Hough transform [129], which maps the data space into parameter space. It then uses a grid-based approach to find dense regions in the parameter space and corresponding data subsets in the original data space. It recursively applies itself on such corresponding data subsets.

10 Other Clustering Techniques

10.1 Neural Network-Based Clustering

The neural network approach has been studied intensively by mathematicians, statisticians, physicists, engineers, and computer scientists [157]. A neural network is an interconnected group of artificial neurons and an adaptive system for information processing. Neural-network-based clustering is competitive-learning-based clustering, not statistical model-identification based clustering. For competitive-learning-based clustering, the first phase is learning where the algorithmic parameters are adjusted, and the second phase is generalization [74]. Competitive learning can be implemented using a two-layer neural network: the input layer and the output layer [74].

Examples are as follows. A SOM (Self-Organizing Map) [153] consists of nodes or neurons, each of which is associated with a weight vector and a position in the map space. It creates a mapping from a higher dimensional input space to a lower dimensional output space. SOM clustering computes the distance of the input pattern to each neuron and finds the winning neuron. LVQ (Learning Vector Quantization) or VQ (Vector Quantization) [39, 102] is a classical quantization technique for signal processing. It models the probability density functions by using the distribution of prototype vectors. It divides a set of vectors into groups that have approximately the same number of vectors closest to them. Basic VQ is k-means clustering, and LVQ is a precursor to self-organizing maps (SOM) [102]. Neural gas [171] is inspired by SOM. It is a simple algorithm and finds optimal data representations based on feature vectors. During the adaptation process, the feature vectors distribute themselves dynamically like a gas within the data space. ART model is covered in Section 8.4.

10.2 Evolutionary Clustering

Evolutionary computation has many applications in computer science, bioinformatics, pharmacometrics, engineering, physics, and economics. Evolutionary computation is inspired by the biological mechanisms of evolution, and uses iterative processes such as growth or development followed by selection in a population of candidate solutions. Clustering methods that use local search techniques including hill-climbing approach-based k-means suffer from local minima problems. The recent advancements in evolutionary computational technologies [92] provide an alternate and effective way to find the global or approximately global optimum [256]. PSO (Particle Swarm Optimization) simulates social behavior in nature, such as bird flocking or fish schooling [148]. ACO (Ant Colony Optimization) algorithms model the behaviors of ants in nature [73]. GAs (Genetic Algorithms) [127] mimic natural selection and use evolutional mechanisms such as crossover, mutation and selection to generate solutions.

Examples are as follows. PPO (Particle-Pair Optimizer) [75] is a modification of the Particle Swarm Optimizer. It uses two particle pairs to search for the global optima in parallel and uses k-means for efficient clustering. Niching genetic k-means [213] modifies Deterministic Crowding [169], one of the niching genetic algorithms, and incorporates one step of k-means into its regeneration steps [213]. EvoCluster algorithm [166] encodes cluster structure in a chromosome, in which one gene represents one cluster or the objects belonging to one cluster. Reproduction operators are used between chromosomes. GenClust [103] is a simple algorithm and proceeds in stages. It uses genetic operators and a fitness function to compute partitions in a new stage based on partitions in the previous stage.

10.3 Kernel Clustering

Kernel-based learning such as Support Vector Machines (SVMs) [61, 208, 199] has had successful applications in pattern recognition and machine learning and is becoming increasingly important [199]. Kernel methods [62] perform a nonliner mapping of the low dimensional input data into a high dimensional space, which becomes linearly separable. To improve efficiency, they avoid explicitly defining the nonlinear mapping by using kernel functions, such as polynomial kernels, sigmoid kernels, and Gaussian radial basis function (RBF) kernels. This is the known as the *kernel trick*.

Examples are as follows. SVC (Support Vector Clustering) [239, 265] uses SVM training to find the cluster boundaries and an adjacency matrix to assign a cluster label to each data object [256]. Variations of SVC include Iterative One-Class SVC [40], and rough Set SVC [192]. Kernel k-means [107] uses a kernel method to calculate the distance between items in a data set, instead of using the Euclidean distance as in regular k-means. Variations include Incremental Kernel-k-means [209]. Kernel deterministic annealing clustering [262] uses an adaptively selected Gaussian parameter and a Gaussian kernel to determine the nonlinear mapping. Kernel fuzzy clustering [164, 268, 270] applies kernel techniques to fuzzy clustering algorithms by replacing the original Euclidean distance with a kernel-induced distance. Kernel Self-Organizing Maps [10, 35, 158] perform self-organizing between an input data object and the corresponding prototype in the mapped high dimensional feature space or in the mapped space completely.

10.4 Sequential Data Clustering

Sequential data are sequences of numerical data or non-numerical symbols and can be generated from speech processing, video analysis, text mining, gene sequencing, and medical diagnosis. Time series data or temporal data are a type of sequential data, which, unlike static data, contain feature values that change over time. Since sequential data usually have variable length, dynamic behaviors, and time constraints [116, 228], they cannot be represented as points in the multi-dimensional feature space and thus cannot be analyzed using any of the clustering techniques we have mentioned thus far [256]. Clustering techniques targeting sequential data have been developed, and they commonly use three strategies: proximity-based approaches, feature-based approaches, and model-based approaches.

Proximity-based approaches use proximity information such as the distance or similarity between pairs of sequences. They then use hierarchical or partitional clustering algorithms to group the sequences into clusters [256]. Examples are as follows. The Needleman-Wunsch algorithm [78, 184] uses basic dynamic programming and is a global optimal alignment algorithm. The Smith-Waterman algorithm [78, 217] is based on Needleman-Wunsch algorithm, and also uses dynamic programming. It compares multi-lengthed sequence segments using character-to-character pair-wise comparisons. FASTA (FAST-All) [194] first finds segments of the two sequences that have some degree of similarity and marks these potential matches. It then performs a more time-consuming optimized search approach such as the Smith-Waterman algorithm. BLAST (Basic Local Alignment Search Tool) [8] searches for short alignment matches between two sequences using a heuristic approach, which approximates the Smith-Waterman algorithm. GeneRage [82] automatically clusters sequence datasets by using Smith-Waterman dynamic programming alignment and single-linkage clustering. SEQOPTICS (SEQuence clustering with OPTICS) [52] implements Smith-Waterman algorithms as the distance measurement and uses OPTICS [11] to perform sequence clustering.

Feature-based approaches map sequences onto multi-dimensional data points using feature extraction methods and then use vector-based clustering algorithms on the data points [256]. Examples are as follows. Scalable sequential data clustering [115] uses a k-means based clustering algorithm which has near-linear time complexity to improve the scalability problem. Pattern-oriented hierarchical clustering [179] uses a hierarchical algorithm, which can generate the clusters as well as the clustering models based on sequential patterns found in the database. The wavelet-based anytime algorithm [237] combines a novel k-means based clustering algorithm and the multi-resolution property of wavelets. It repeatedly uses coarse clustering to obtain a clustering at a slightly finer level of approximation.

Model-based approaches assume sequences that belong to one cluster are generated from one probabilistic model [256]. Examples are as follows. Autoregressive moving average (ARMA) models [18, 253] derive an EM algorithm to learn the mixing coefficients and the parameters of the component ARMA models. They use the Bayesian information criterion (BIC) to determine the number of clusters. The Markov chain approach [202, 219] models dynamics as Markov chains and then applies an agglomerative clustering procedure to discover a set of clusters that best capture different dynamics. The Polynomial models approach [17, 97] assumes the underlying model is a mixture of polynomial functions. It uses an EM algorithm to estimate the cluster membership probabilities, using weighted least squares to fit the models. The Hidden Markov Model (HMM) [188, 218] is a probabilistic model-based approach. It uses HMMs, which have shown capabilities in modeling the structure of the generative processes underlying real-world time series data.

10.5 Ensemble Clustering

Clustering ensembles have emerged to improve robustness, stability and accuracy of clustering results [105]. A cluster ensemble combines the results of multiple clustering algorithms to obtain a consensus result [197]. It can produce better average performance and avoid worst case results. Other usages of clustering ensembles include improving scalability by performing clustering on subsets of data in parallel and then combining the results, and data integration when data is distributed across multiple sources [137].

There are two main steps in a clustering ensemble: generation and consensus. In the generation step, several approaches are used [235]: different clustering algorithms, a single algorithm with different parameter initializations, different object representations, different object projections, and different subsets of objects.

In the consensus step, several approaches are used: relabeling and voting, Mutual Information (MI), co-association based functions, finite mixture models, a graph/hypergraph partitioning approach, and others.

The relabeling and voting approach is also called the direct approach. It finds the correspondence of the cluster labels among different clustering results and then uses a voting method to determine the final cluster label for a data object. Examples are as follows. BagClust1 [76] applies a clustering procedure to each bootstrap sample and obtains the final partition by plurality voting so that the majority cluster label for each data object determines the final cluster membership. BagClust2 [76] introduces a new dissimilarity matrix which contains the proportion of time each pair of data objects were clustered together in the bootstrap clusters. It then performs clustering on the dissimilarity matrix to obtain the final partition.

The MI approach uses MI to measure and quantify the statistical information shared between a pair of clusterings. It can automatically select the best clustering method from several algorithms. Examples are as follows. A Genetic Algorithm (GA) clustering ensemble [15] uses a GA to obtain the best partition and the co-association function as the consensus

function. It determines fitness function parameters based on co-association function values. The information theory based GA clustering ensemble [165] uses a GA to find a combined clustering by minimizing an information-theoretical criterion function. The generalized MI clustering ensemble [233] introduces a new consensus function using a generalized mutual information definition. The consensus function is related to the classical intraclass variance criterion.

The co-association based functions approach is also called the pair-wise approach. It uses a co-association matrix in the consensus step. Examples are as follows. Clusterfusion [147] first generates an agreement matrix with each cell containing the number of agreements amongst clustering methods and then uses the matrix to cluster data objects. Voting-k-Means [95] transforms data partitions into a co-association matrix with coherent association mappings. It then extracts underlying clusters from this matrix. Evidence accumulation-based clustering [96] maps data partitions created by each individual clustering into a new similarity matrix, based on voting. It then uses the single link algorithm to extract clusters from this matrix.

Finite mixture model approach assumes that the probability of assigning a label to a data object is based on a finite mixture model or that the labels are 'modeled as random variables drawn from a probability distribution described as a mixture of multivariate component densities' [235]. It obtains the consensus clustering result by solving a maximum likelihood estimation problem. Mixture model clustering ensemble [234] uses a probabilistic model of consensus based on a finite mixture of multinomial distributions in a space of clusterings. It finds a combined partition by solving the corresponding maximum likelihood problem with the EM algorithm.

The graph/hypergraph partitioning approach considers the combination problem as a graph or hypergraph partitioning problem. Methods taking this approach differ in how they build a (hyper)graph from the clusterings, as well as how they define the cuts on the graph to obtain the consensus partition [235]. Examples are as follows. METIS [144] is a multi-level graph partitioning system. It collapses vertices and edges of the graph, partitions the resulting coarsened graph, and then refines the partitions. SPEC (spectral graph partitioning algorithm) [186] tries to optimize the normalized cut criterion. It treats the rows of the largest eigenvalues matrix as multiple dimensional embeddings of the vertices of the graph and then uses k-means to cluster the embedded points. CSPA (Cluster based Similarity Partitioning Algorithm) [227] first creates a graph based on a co-association matrix, and then performs METIS clustering on the graph. HGPA (Hypergraph Partitioning Algorithm) [227] uses a hyperedge in a graph to represent each cluster. It then uses minimal cut algorithms such as HMETIS [142] to find good hypergraph partitions. MCLA (Meta Clustering Algorithm) [227] determines soft cluster membership values for each data object by using hyperedge collapsing operations. HBGF (Hybrid Bipartite Graph Formulation) [90] constructs a bipartite graph where data objects and clusters are both modeled as vertices. It later partitions the bipartite graph with an appropriate graph partitioning method.

Other approaches are as follows. The cumulative voting consensus method [14] solves the cluster label alignment problem by using cumulative voting, where a probabilistic mapping between labels is computed. Bipartite Merger and Metis merger [128] are approaches for merging an ensemble of clustering solutions using sets of cluster centers. They are highly scalable and provide competitive results. Weighted consensus clustering [160] weights each input clustering. It determines weights in a way so that the clusters are better separated. Bayesian Cluster Ensembles [240] takes a Bayesian approach to combine clusterings. It uses a variational approximation based algorithm for learning. This way, it is able to avoid the cluster label correspondence problems. Figure 8 shows an example of ensemble clustering.

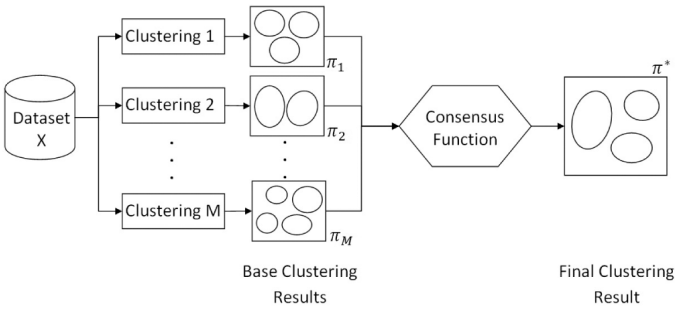

Figure 8: Ensemble clustering [133]

10.6 Multi-objective Clustering

Conventional clustering algorithms use a single clustering objective function only, which may not be appropriate for the diversities of the underlying data structures. Multi-objective clustering uses multiple clustering objective functions simultaneously. Such methods consider clustering as a multi-objective optimization problem [89].

Examples are as follows. FCPSO (Fuzzy Clustering-based Particle Swarm Optimization) [6] uses an external repository to save nondominated particles during the search process and a fuzzy clustering technique to manage the size of the repository. It also uses a fuzzy-based iterative feedback mechanism to determine the compromised solution among conflicting objectives. Evolutionary Multiobjective Clustering [118] and MOCK (MultiObjective Clustering with automatic k-determination) [119] use an evolutionary approach to solve the multi-objective problem in clustering. They are based on a multi-objective evolutionary algorithm named PESA-II (Pareto Envelope-based Selection Algorithm version 2) [59] to optimize two complementary clustering objectives. Multi-objective real coded genetic fuzzy clustering [181] aims to optimize multiple validity measures simultaneously. It encodes the cluster centers in its chromosomes while optimizing the fuzzy compactness within a cluster and fuzzy separation among

clusters. EMO-CC (Evolutionary MultiObjective Conceptual Clustering) [203] combines evolutionary algorithms with multi-objective optimization techniques and relies on the NSGA-II multi-objective genetic algorithm [66]. It can discover less obvious but informative data associations.

10.7 Semi-supervised Clustering

Semi-supervised clustering provides limited supervision to unsupervised clustering. There are many cases when some knowledge about the data is available such as the constraints between data objects or cluster labels for some data objects. Such knowledge can be used to guide the clustering process. There are several approaches for semi-supervised clustering: similarity-adapting methods, search-based methods, and other methods.

Similarity-adapting methods use a similarity measure which is adapted to make the available constraints more easily satisfied [111]. Examples are as follows. Distance metric learning based clustering [252] learns a distance metric based on examples of similar pairs of data objects in the input space using convex optimization. Space-level constraints based clustering [150] exploits space-level implications based on instance-level constraints. It uses an all-pairs-shortest-paths algorithm to adjust the distance metric.

Search-based methods modify the clustering algorithm itself to use the available constraints or labels to guide the search for an appropriate clustering [111]. Examples are as follows. Seeded-K Means and Constrained-K Means [23] generate initial seed clusters based on labeled data. The latter also generates constraints from labeled data and guides the clustering process using those constraints. Semi-Supervised Clustering Using Genetic Algorithms [68] modifies k-means clustering to minimize within-cluster variance and a measure of cluster impurity. Clustering with Instance-level Constraints [238] incorporates hard constraints using a modified version of Cobweb (covered in Section 8.4) which partitions the data.

Other methods include the probabilistic semi-supervised clustering with constraints method [24], which derives an objective function from the joint probability defined over the Hidden Markov Random Field model and performs semi-supervised clustering by minimizing this object function.

11 Conclusions

We have presented a survey of the literature on clustering techniques. For convenience, in this review we used the following taxonomy, which is also widely used in the literature: hierarchical clustering (Section 2), partitioning clustering (Section 3), graph-based clustering (Section 4), distribution-based clustering (Section 5), density-based clustering (Section 6), grid-based clustering (Section 7), clustering big data (Section 8), clustering high dimensional data (Section 9), and other clustering techniques (Section 10).

References

[1] Elke Achtert, Christian Böhm, Jörn David, Peer Kröger, and Arthur Zimek. “Global Correlation Clustering Based on the Hough Transform”. *Statistical Analysis and Data Mining*, 1(3):111–127. 2008.

[2] Elke Achtert, Christian Böhm, Peer Kröger, and Arthur Zimek. “Mining Hierarchies of Correlation Clusters”. In “Scientific and Statistical Database Management, 2006. 18th International Conference on”, IEEE, pp. 119–128. 2006.

[3] Charu C. Aggarwal, Joel L. Wolf, Philip S. Yu, Cecilia Procopiuc, and Jong Soo Park. “Fast Algorithms for Projected Clustering”. In “SIGMOD ’99: Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data”, ACM, New York, NY, USA, pp. 61-72. doi:10.1145/304182.304188. 1999.

[4] Charu C. Aggarwal and Philip S. Yu. “Finding Generalized Projected Clusters in High Dimensional Spaces”. *SIGMOD Rec.*, 29(2):70–81. doi:10.1145/335191.335383. May 2000.

[5] Rakesh Agrawal, Johannes Gehrke, Dimitrios Gunopulos, and Prabhakar Raghavan. “Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications”. *SIGMOD Rec.*, 27(2):94–105. doi:10.1145/276305.276314. Jun. 1998.

[6] Shubham Agrawal, B. K. Panigrahi, and Manoj Kumar Tiwari. “Multiobjective Particle Swarm Algorithm with Fuzzy Clustering for Electrical Power Dispatch.” *IEEE Trans. Evolutionary Computation*, 12(5):529–541. 2008.

[7] Khaled S. Al-Sultan and Shokri Z. Selim. “A Global Algorithm for the Fuzzy Clustering Problem.” *Pattern Recognition*, 26(9):1357–1361. 1993.

[8] Stephen F. Altschul, Warren Gish, Webb Miller, Eugene W. Myers, and David J. Lipman. “Basic Local Alignment Search Tool”. *Journal of Molecular Biology*, 215(3):403 – 410. doi:http://dx.doi.org/10.1016/S0022-2836(05)80360-2. 1990.

[9] G.C. Anagnostopoulos and M. Georgiopoulos. “Ellipsoid ART and ARTMAP for Incremental Clustering and Classification”. In “IJCNN’01. International Joint Conference on Neural Networks. Proceedings (Cat. No.01CH37222)”, pp. 1221–1226 vol.2. doi:10.1109/IJCNN.2001.939535. 2001.

[10] Peter Andras. “Kernel-Kohonen Networks.” *Int. J. Neural Syst.*, 12(2):117–135. 2002.

[11] Mihael Ankerst, Markus M. Breunig, Hans-Peter Kriegel, and Jörg Sander. “OPTICS: Ordering Points to Identify the Clustering Structure”. *SIGMOD Rec.*, 28(2):49–60. doi:10.1145/304181.304187. Jun. 1999.

[12] David Arthur and Sergei Vassilvitskii. “k-means++: The Advantages of Careful Seeding”. In “Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA ’07)”, Society for Industrial

and Applied Mathematics, Philadelphia, PA, USA, pp. 1027–1035. 2007.

[13] Roberto Avogadri and Giorgio Valentini. "Fuzzy Ensemble Clustering Based on Random Projections for DNA Microarray Data Analysis". *Artificial Intelligence in Medicine*, 45(2):173–183. 2009.

[14] Hanan G Ayad and Mohamed S Kamel. "Cumulative Voting Consensus Method for Partitions with Variable Number of Clusters". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(1):160–173. 2008.

[15] J. Azimi, M. Mohammadi, A. Movaghar, and M. Analoui. "Clustering Ensembles Using Genetic Algorithm". In "International Workshop on Computer Architecture for Machine Perception and Sensing, 2006. CAMP 2006.", pp. 119–123. doi: 10.1109/CAMP.2007.4350366. 2007.

[16] G. Phanendra Babu and M. Narasimha Murty. "A Near-Optimal Initial Seed Value Selection in K-means Means Algorithm Using a Genetic Algorithm." *Pattern Recognition Letters*, 14(10):763–769. 1993.

[17] A. Bagnall, G. Janacek, B. Iglesia, and M. Zhang. "Clustering Time Series from Mixture Polynomial Models with Discretized Data". In "Proc. 2nd Australasian Data Mining Workshop", pp. 105–120. 2003.

[18] Anthony J. Bagnall and Gareth J. Janacek. "Clustering Time Series from ARMA Models with Clipped Data." In Won Kim, Ron Kohavi, Johannes Gehrke, and William DuMouchel, editors, "KDD", ACM, pp. 49–58. 2004.

[19] G. H. Ball and D. J. Hall. "ISODATA, an Iterative Method of Multivariate Analysis and Pattern Classification". *IFIPS Congress*. 1965.

[20] A. Baraldi and E. Alpaydin. "Constructive Feedforward ART Clustering Networks – Part I and II". *IEEE Trans. Neural Netw.*, 13(3). May 2002.

[21] D. Barbara and P. Chen. "Using the Fractal Dimension to Cluster Datasets". In "Proc. of the 6th International Conference on Knowledge Discovery and Data Mining", ACM, pp. 260–264. 2000.

[22] Daniel Barbara, Julia Couto, and Yi Li. "COOLCAT: An Entropy-Based Algorithm for Categorical Clustering". In "In Proceedings of the Eleventh International Conference on Information and Knowledge Management", ACM, pp. 582–589. 2002.

[23] S. Basu, A. Banerjee, and R. Mooney. "Semi-Supervised Clustering by Seeding". In "Proceedings of the International Conference on Machine Learning", pp. 27–34. 2002.

[24] Sugato Basu, Mikhail Bilenko, Arindam Banerjee, and Raymond J Mooney. "Probabilistic Semi-Supervised Clustering with Constraints". *Semi-Supervised Learning*, pp. 71–98. 2006.

[25] Stephen D. Bay and Michael J. Pazzani. "Detecting Change in Categorical Data: Mining Contrast Sets". In "Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining", ACM, New York, NY, USA, KDD '99, pp. 302–306. doi: 10.1145/312129.312263. 1999.

[26] Ron Bekkerman, Ran El-Yaniv, and Andrew McCallum. "Multi-way Distributional Clustering via Pairwise Interactions". In "Proceedings of the 22nd International Conference on Machine learning", ACM, pp. 41–48. 2005.

[27] Mikhail Belkin and Partha Niyogi. "Laplacian Eigenmaps for Dimensionality Reduction and Data Representation". *Neural Computation*, 15(6):1373–1396. 2003.

[28] Pavel Berkhin. "Survey Of Clustering Data Mining Techniques". Tech. rep., Accrue Software, San Jose, CA. 2002.

[29] J. Bezdek, C. Coray, R. Gunderson, and J. Watson. "Detection and Characterization of Cluster Substructure I. Linear Structure: Fuzzy c-Lines". *SIAM Journal on Applied Mathematics*, 40(2):339–357. doi:10.1137/0140029. 1981.

[30] D. A. Binder. "Bayesian Cluster Analysis". *Biometrika*, 65(1):31–38. 1978.

[31] C. Bishop. *Neural Networks for Pattern Recognition*. Oxford Univ. Press. 1995.

[32] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. "Latent Dirichlet Allocation". *J. Mach. Learn. Res.*, 3:993–1022. Mar. 2003.

[33] Hans H. Bock. "Probabilistic Models in Cluster Analysis". *Computational Statistics & Data Analysis*, 23(1):5–28. Nov. 1996.

[34] Christian Böhm, Karin Kailing, Hans-Peter Kriegel, and Peer Kröger. "Density Connected Clustering with Local Subspace Preferences." In "ICDM", IEEE Computer Society, pp. 27–34. 2004.

[35] Romain Boulet, Bertrand Jouve, Fabrice Rossi, and Nathalie Villa. "Batch Kernel SOM and Related Laplacian Methods for Social Network Analysis". *Neurocomputing*, 71(7-9):1257–1273. 2008.

[36] Christos Boutsidis, Anastasios Zouzias, and Petros Drineas. "Random Projections for K-Means Clustering." In "Advances in Neural Information Processing Systems", pp. 298–306. 2010.

[37] Paul S. Bradley, Usama M. Fayyad, and Cory Reina. "Scaling Clustering Algorithms to Large Databases". In "Knowledge Discovery and Data Mining", pp. 9–15. 1998.

[38] Nathan Brannon, John Seiffertt, Timothy Draelos, and Donald C. Wunsch II. "Coordinated Machine Learning and Decision Support for Situation Awareness." *Neural Networks*, 22(3):316–325. 2009.

[39] D. Burton, J. Shore, and J. Buck. "A Generalization of Isolated Word Recognition Using Vector Quantization". In "Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP '83.", vol. 8, pp. 1021–1024. doi:10.1109/ICASSP.1983.1171915. Apr 1983.

[40] Francesco Camastra and Alessandro Verri. "A Novel Kernel Method for Clustering." *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27(5):801–804. 2005.

[41] G. A. Carpenter, S. Grossberg, N. Markuzon, J.H. Reynolds, and D.B. Rosen. "Fuzzy ARTMAP: A Neural Network Architecture for Incremental Supervised Learning of Analog Multidimensional Maps". *IEEE Transactions on Neural Networks*, 3(5):698–713. September 1992.

[42] Gail A. Carpenter and Stephen Grossberg. "ART 2: Self-Organization of Stable Category Recognition Codes for Analog Input Patterns". *Applied Optics*, 26(23):4919–4930. 1987.

[43] Gail A. Carpenter and Stephen Grossberg. "A Massively Parallel Architecture for a Self-Organizing Neural Pattern Recognition Machine". *Computer Vision, Graphics, and Image Processing*, 37(1):54–115. 1987.

[44] Gail A. Carpenter, Stephen Grossberg, and John H. Reynolds. "ARTMAP: Supervised Real-Time Learning and Classification of Nonstationary Data by a Self-Organizing Neural Network". *Neural Networks*, 4(5):565–588. 1991.

[45] Thomas P Caudell, Scott DG Smith, G Craig Johnson, and Donald C Wunsch II. "Application of Neural Networks to Group Technology". In "Proceedings of SPIE 1469, Applications of Artificial Neural Networks II", SPIE–The International Society for Optical Engineering, pp. 612–621. Jan 1991.

[46] Soumen Chakrabarti. *Mining the Web: Discovering Knowledge from Hypertext Data*. Morgan Kaufmann. 2003.

[47] Jae-Woo Chang and Du-Seok Jin. "A New Cell-Based Clustering Method for Large, High-Dimensional Data in Data Mining Applications". In "Proceedings of the 2002 ACM Symposium on Applied Computing", ACM, pp. 503–507. 2002.

[48] Moses Charikar, Chandra Chekuri, Tomás Feder, and Rajeev Motwani. "Incremental Clustering and Dynamic Information Retrieval". In "Proceedings of the Twenty-ninth Annual ACM Symposium on Theory of Computing", ACM, New York, NY, USA, STOC '97, pp. 626–635. doi:10.1145/258533.258657. 1997.

[49] Peter Cheeseman and John Stutz. "Advances in Knowledge Discovery and Data Mining". In Usama M. Fayyad, Gregory Piatetsky-Shapiro, Padhraic Smyth, and Ramasamy Uthurusamy, editors, "Bayesian Classification (AutoClass): Theory and Results", American Association for Artificial Intelligence, Menlo Park, CA, USA, pp. 153–180. 1996.

[50] Jiun-Rung Chen. *Efficient Biclustering Methods for Microarray Databases*. Ph.D. thesis, National Sun Yat-sen University. 2010.

[51] Wen-Yen Chen, Yangqiu Song, Hongjie Bai, Chih-Jen Lin, and Edward Y Chang. "Parallel Spectral Clustering in Distributed Systems". *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(3):568–586. 2011.

[52] Yonghui Chen, Kevin D Reilly, Alan P Sprague, and Zhijie Guan. "SEQOPTICS: A Protein Sequence Clustering System". *BMC Bioinformatics*, 7(Suppl 4):S10. 2006.

[53] Chun Hung Cheng, Ada Wai-Chee Fu, and Yi Zhang. "Entropy-Based Subspace Clustering for Mining Numerical Data." In Usama M. Fayyad, Surajit Chaudhuri, and David Madigan, editors, "KDD", ACM, pp. 84–93. 1999.

[54] Yizong Cheng and George M. Church. "Biclustering of Expression Data". In "Proc. of the 8th Intelligent Systems for Molecular Biology", AAAI Press, pp. 93–103. 2000.

[55] Chire. "Density-Based Clustering with DBSCAN". URL http://commons.wikimedia.org/wiki/File:DBSCAN-density-data.svg. Accessed Oct. 2015. 2011.

[56] Chire. "Expectation-Maximization (EM) Clustering Examples". URL http://commons.wikimedia.org/wiki/File:EM-Gaussian-data.svg. Accessed Oct. 2015. 2011.

[57] Chire. "k-Means Clustering Examples". URL http://commons.wikimedia.org/wiki/File:KMeans-Gaussian-data.svg. Accessed Oct. 2015. 2011.

[58] Robson Leonardo Ferreira Cordeiro, Agma J. M. Traina, and Christos Faloutsos. "Finding Clusters in Subspaces of Very Large, Multi-Dimensional Datasets". In "IEEE 26th International Conference on Data Engineering (ICDE), 2010", IEEE, pp. 625–636. 2010.

[59] David W. Corne, Nick R. Jerram, Joshua D. Knowles, and Martin J. Oates. "PESA-II: Region-Based Selection in Evolutionary Multiobjective Optimization". In "Proceedings of the 3rd Annual Conference on Genetic and Evolutionary Computation", Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, GECCO'01, p. 283–290. 2001.

[60] Carlos D Correa and Peter Lindstrom. "Locally-Scaled Spectral Clustering Using Empty Region Graphs". In "Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining", ACM, pp. 1330–1338. 2012.

[61] Corinna Cortes and Vladimir Vapnik. "Support-Vector Networks". *Machine learning*, 20(3):273–297. doi:10.1023/a:1022627411411. Sep. 1995.

[62] Nello Cristianini and John Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press, 1st edn. Mar. 2000.

[63] R. D'Andrade. "U-Statistic Hierarchical Clustering". *Psychometrika*, 4. 1978.

[64] M. Dash, H. Liu, and Xiaowei Xu. "'1+1 > 2': Merging Distance and Density Based Clustering". In "Database Systems for Advanced Applications, 2001. Proceedings. Seventh International Conference on", IEEE Computer Society, pp. 32 –39. doi:10.1109/DASFAA.2001.916361. april 2001.

[65] N. E. Day. "Estimating the Components of a Mixture of Normal Distributions". *Biometrika*, 56(3):463–474. doi:10.1093/biomet/56.3.463. 1969.

[66] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. "A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II". *IEEE Transactions on Evolutionary Computation,*, 6(2):182–197. 2002.

[67] D. Defays. "An Efficient Algorithm for a Complete Link Method". *The Computer Journal (British Computer Society)*, 20(4):364–366. 1977.

[68] Ayhan Demiriz, Kristin Bennett, and Mark J. Embrechts. "Semi-Supervised Clustering Using Genetic Algorithms". In "Artificial Neural Networks in Engineering (ANNIE-99)", ASME Press, pp. 809–814. 1999.

[69] A. P. Dempster, N. M. Laird, and D. B. Rubin. "Maximum Likelihood From Incomplete Data via the EM Algorithm". *Journal of the Royal Statistical Society, Series B*, 39(1):1–38. 1977.

[70] I. Dhillon and D. Modha. "A Data Clustering Algorithm on Distributed Memory Multiprocessors". In "5th ACM SIGKDD, Large-scale Parallel KDD Systems Workshop", pp. 245–260. 1999.

[71] Inderjit S. Dhillon, Yuqiang Guan, and Brian Kulis. "Kernel K-Means: Spectral Clustering and Normalized Cuts". In "Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining", ACM, pp. 551–556. 2004.

[72] Inderjit S Dhillon, Subramanyam Mallela, and Dharmendra S Modha. "Information-Theoretic Co-Clustering". In "Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining", ACM, pp. 89–98. 2003.

[73] Marco Dorigo and Thomas Stützle. "The Ant Colony Optimization Metaheuristic: Algorithms, Applications, and Advances". In "Handbook of Metaheuristics", Springer, pp. 250–285. 2003.

[74] K.-L. Du. "Clustering: A Neural Network Approach." *Neural Networks*, 23(1):89–107. 2010.

[75] Zhihua Du, Yiwei Wang, and Zhen Ji. "PK-means: A New Algorithm for Gene Clustering." *Computational Biology and Chemistry*, 32(4):243–247. 2008.

[76] Sandrine Dudoit and Jane Fridlyand. "Bagging to Improve the Accuracy of a Clustering Procedure". *Bioinformatics*, 19(9):1090–1099. 2003.

[77] Joseph C. Dunn. "A Fuzzy Relative of the ISODATA Process and its use in Detecting Compact Well-Separated Clusters". *Journal of Cybernetics*, 3:32–57. 1973.

[78] Richard Durbin, Sean R. Eddy, Anders Krogh, and Graeme Mitchison. *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press. 1998.

[79] A. W. F. Edwards and L. L. Cavalli-Sforza. "A Method for Cluster Analysis". *Biometrics*, 21:362–375. 1965.

[80] C. Eldershaw and M. Hegland. "Cluster Analysis Using Triangulation". *Computational Techniques and Applications*, pp. 201–208. 1997.

[81] Charles Elkan. "Using the Triangle Inequality to Accelerate k-means". In "Proceedings of the Twentieth International Conference on Machine Learning (ICML)", vol. 3, pp. 147–153. 2003.

[82] A. Enright and C. Ouzounis. "GeneRAGE: A Robust Algorithm for Sequence Clustering and Domain Detection". *Bioinformatics*, 16(5):451–457. 2000.

[83] S. Eschrich, Jingwei Ke, L.O. Hall, and D.B. Goldgof. "Fast Accurate Fuzzy Clustering Through Data Reduction". *IEEE Transactions on Fuzzy Systems*, 11(2):262–270. doi:10.1109/TFUZZ.2003.809902. 2003.

[84] Martin Ester, Hans P. Kriegel, Jorg Sander, and Xiaowei Xu. "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise". In Evangelos Simoudis, Jiawei Han, and Usama Fayyad, editors, "Second International Conference on Knowledge Discovery and Data Mining", AAAI Press, Portland, Oregon, pp. 226–231. 1996.

[85] Tim Evans. "Zachary Karate Club Network Clustered Using Clique Graph methods". URL http://netplexity.org/?p=1261. Accessed Oct. 2015. 2014.

[86] B. Everitt, S. Landau, and M. Leese. *Cluster Analysis*. London: Arnold. 2001.

[87] B. S. Everitt. "Cluster Analysis". In "Cluster Analysis, Second Edition", Heineman Educational Books Ltd. 1980.

[88] Christos Faloutsos and King-Ip Lin. "FastMap: A Fast Algorithm for Indexing, Data-mining and Visualization of Traditional and Multimedia Datasets". *SIGMOD Rec.*, 24(2):163–174. doi:10.1145/568271.223812. May 1995.

[89] A. Ferligoj and V. Batagelj. "Direct Multicriterion Clustering Algorithms". *Journal of Classification*, 9:43–61. 1992.

[90] Xiaoli Zhang Fern and Carla E. Brodley. "Random Projection for High Dimensional Data Clustering: A Cluster Ensemble Approach." In Tom Fawcett and Nina

Mishra, editors, "International Conference on Machine Learning", AAAI Press, pp. 186–193. 2003.

[91] Douglas H Fisher. "Knowledge Acquisition via Incremental Conceptual Clustering". *Machine Learning*, 2(2):139–172. 1987.

[92] David B Fogel. *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence*. John Wiley & Sons. 2006.

[93] E. W. Forgy. "Cluster Analysis of Multivariate Data: Efficiency Versus Interpretability of Classifications". *Biometrics*, 21:768–769. 1965.

[94] Ildiko E. Frank and Roberto Todeschini. *The Data Analysis Handbook*. Elsevier Science Inc. 1994.

[95] Ana L. N. Fred. "Finding Consistent Clusters in Data Partitions." In Josef Kittler and Fabio Roli, editors, "Multiple Classifier Systems", Springer, vol. 2096 of *Lecture Notes in Computer Science*, pp. 309–318. 2001.

[96] Ana L. N. Fred and Anil K. Jain. "Data Clustering Using Evidence Accumulation". In "Proceedings 16th International Conference on Pattern Recognition", IEEE, vol. 4, pp. 276–280. 2002.

[97] Scott Gaffney and Padhraic Smyth. "Trajectory Clustering with Mixtures of Regression Models". In "Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining", ACM, pp. 63–72. 1999.

[98] Guojun Gan. *Data Clustering in C++: An Object-Oriented Approach*. Chapman & Hall/CRC Data Mining and Knowledge Discovery Series. Taylor & Francis. 2011.

[99] Guojun Gan, Chaoqun Ma, and Jianhong Wu. *Data Clustering - Theory, Algorithms, and Applications*. SIAM. 2007.

[100] Venkatesh Ganti, Johannes Gehrke, and Raghu Ramakrishnan. "CACTUS - Clustering Categorical Data Using Summaries". In "Proceedings of the fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining", ACM, New York, NY, USA, KDD '99, pp. 73–83. doi:10.1145/312129.312201. 1999.

[101] Venkatesh Ganti, Raghu Ramakrishnan, Johannes Gehrke, Allison Powell, and James French. "Clustering Large Datasets in Arbitrary Metric Spaces". In "Data Engineering, 1999. Proceedings., 15th International Conference on", IEEE, pp. 502–511. 1999.

[102] A. Gersho and B. Ramamurthi. "Image Coding Using Vector Quantization". *International Conference on Acoustics, Speech, and Signal Processing*, 1:428–431. April 1982.

[103] Vito Di Gesù, Raffaele Giancarlo, Giosuè Lo Bosco, Alessandra Raimondi, and Davide Scaturro. "GenClust: A Genetic Algorithm for Clustering Gene Expression Data". *BMC Bioinformatics*, 6(1):289–289. 2005.

[104] Gad Getz, Erel Levine, and Eytan Domany. "Coupled Two-Way Clustering Analysis of Gene Microarray Data". *Proceedings of the National Academy of Sciences*, 97(22):12079–12084. doi:10.1073/pnas.210134797. 2000.

[105] Reza Ghaemi, Md. Nasir Sulaiman, Hamidah Ibrahim, and Norwati Mustapha. "A Survey: Clustering Ensembles Techniques". *World Academy of Science, Engineering and Technology*, 50:636–645. 2009.

[106] David Gibson, Jon Kleinberg, and Prabhakar Raghavan. "Clustering Categorical Data: An Approach Based on Dynamical Systems". *The International Journal on Very Large Data Bases*, 8(3-4):222–236. doi:10.1007/s007780050005. Feb. 2000.

[107] M. Girolami. "Mercer Kernel-Based Clustering in Feature Space". *Neural Networks, IEEE Transactions on*, 13(3):780–784. doi:10.1109/tnn.2002.1000150. Aug. 2002.

[108] Sanjay Goil, Harsha Nagesh, and Alok Choudhary. "MAFIA: Efficient and Scalable Subspace Clustering for Very Large Data Sets". In "Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining", pp. 443–452. 1999.

[109] Teofilo F Gonzalez. "Clustering to Minimize the Maximum Intercluster Distance". *Theoretical Computer Science*, 38(2-3):293–306. 1985.

[110] J. C. Gower and G. J. S. Ross. "Minimum Spanning Trees and Single Linkage Cluster Analysis". *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 18(1):54–64. 1969.

[111] Nizar Grira, Michel Crucianu, and Nozha Boujemaa. "Semi-Supervised Fuzzy Clustering with Pairwise-Constrained Competitive Agglomeration". In "Fuzzy Systems, 2005. FUZZ'05. The 14th IEEE International Conference on", IEEE, pp. 867–872. 2005.

[112] Sudipto Guha, Rajeev Rastogi, and Kyuseok Shim. "CURE: An Efficient Clustering Algorithm for Large Databases". *SIGMOD Record*, 27(2):73–84. doi:10.1145/276305.276312. Jun. 1998.

[113] Sudipto Guha, Rajeev Rastogi, and Kyuseok Shim. "ROCK: A Robust Clustering Algorithm for Categorical Attributes". *Information Systems*, 25(5):345 – 366. doi:10.1016/S0306-4379(00)00022-3. 2000.

[114] Chetan Gupta and Robert Grossman. "GenIc: A Single Pass Generalized Incremental Algorithm for Clustering". In "Proceedings of the Fourth SIAM International Conference on Data Mining", SIAM, pp. 147–153. 2004.

[115] Valerie Guralnik and George Karypis. "A Scalable Algorithm for Clustering Sequential Data." In Nick Cercone, Tsau Young Lin, and Xindong Wu, editors, "ICDM", IEEE Computer Society, pp. 179–186. 2001.

[116] Dan Gusfield. *Algorithms on Strings, Trees and Sequences: Computer Science and Computational Biology*. Cambridge University Press. 1997.

[117] Lars Hagen and Andrew B. Kahng. "New Spectral Methods for Ratio Cut Partitioning and Clustering". *IEEE Tansactions on Computer-aided Design*, 11(9):1074–1085. September 1992.

[118] Julia Handl and Joshua Knowles. "Evolutionary Multiobjective Clustering". In "Parallel Problem Solving from Nature-PPSN VIII", Springer, pp. 1081–1091. 2004.

[119] Julia Handl and Joshua Knowles. "An Evolutionary Approach to Multiobjective Clustering". *Evolutionary Computation, IEEE Transactions on*, 11(1):56–76. 2007.

[120] J. A. Hartigan. "Clustering algorithms". In "Clustering Algorithms", John Wiley & Sons, Inc. 1975.

[121] J. A. Hartigan and M. A. Wong. "Algorithm AS 136: A k-Means Clustering Algorithm". *Journal of the Royal Statistical Society, Series C (Applied Statistics)*, 28(1):100–108. 1979.

[122] Alexander Hinneburg, Er Hinneburg, and Daniel A. Keim. "An Efficient Approach to Clustering in Large Multimedia Databases with Noise". In "SIGKDD Conference on Knowledge Discovery and Data Mining", AAAI Press, vol. 98, pp. 58–65. 1998.

[123] Alexander Hinneburg and Daniel A. Keim. "Optimal Grid-Clustering: Towards Breaking the Curse of Dimensionality in High-Dimensional Clustering." In Malcolm P. Atkinson, Maria E. Orlowska, Patrick Valduriez, Stanley B. Zdonik, and Michael L. Brodie, editors, "Proceedings of the 25th International Conference on Very Large Data Bases", Morgan Kaufmann, pp. 506–517. 1999.

[124] Ricky Ho. "Machine Learning in R: Clustering". URL http://horicky.blogspot.com/2012/04/machine-learning-in-r-clustering.html. Accessed Oct. 2015. 2012.

[125] Matthew D. Hoffman, David M. Blei, and Francis R. Bach. "Online Learning for Latent Dirichlet Allocation". In "Advances in Neural Information Processing Systems", pp. 856–864. 2010.

[126] Thomas Hofmann and Joachim M. Buhmann. "Pairwise Data Clustering by Deterministic Annealing". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19:1–14. 1997.

[127] J. Holland. *Adaption in Natural and Artificial Systems*. Ann Arbor, MI: Univ. Michigan Press. 1975.

[128] Prodip Hore, Lawrence O Hall, and Dmitry B Goldgof. "A Scalable Framework for Cluster Ensembles". *Pattern recognition*, 42(5):676–688. 2009.

[129] P. V. C. Hough. "Methods and Means for Recognizing Complex Patterns". US Patent 3069654. December 1962.

[130] Xiaohua Hu and Illhoi Yoo. "Cluster Ensemble and its Applications in Gene Expression Analysis". In "Proceedings of the Second Conference on Asia-Pacific Bioinformatics", Australian Computer Society, Inc., Darlinghurst, Australia, vol. 29 of *APBC '04*, pp. 297–302. 2004.

[131] Z. Huang. "Extensions to the k-Means Algorithms for Clustering Large Data Sets with Categorical Values". *Data Mining and Knowledge Discovery*, 2(3):283–304. 1998.

[132] Christopher L. Huntley and Donald E. Brown. "A Parallel Heuristic for Quadratic Assignment Problems". *Computers & Operations Research*, 18(3):275–289. 1991.

[133] N. Iam-On, T. Boongeon, S. Garrett, and C. Price. "A Link-Based Cluster Ensemble Approach for Categorical Data Clustering". *Knowledge and Data Engineering, IEEE Transactions on*, 24(3):413–425. doi:10.1109/TKDE.2010.268. March 2012.

[134] V. Ilango, R. Subramanian, and V. Vasudevan. "Cluster Analysis Research Design Model, Problems, Issues, Challenges, Trends and Tools". *International Journal on Computer Science and Engineering*, 3(8):2926–2934. 2011.

[135] A. Jain, R. Duin, and J. Mao. "Statistical Pattern Recognition: A Review". *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(1):4–37. 2000.

[136] A. K. Jain and R. C. Dubes. "Algorithms for Clustering Data". In "Prentice-Hall Advanced Reference Series", Prentice-Hall, Inc. 1988.

[137] Anil K. Jain. "Data Clustering: User's Dilemma". In "Proceedings of the 5th International Conference on Machine Learning and Data Mining in Pattern Recognition", Springer-Verlag, Berlin, Heidelberg, MLDM '07, pp. 1–1. doi:10.1007/978-3-540-73499-4_1. 2007.

[138] Anil K Jain. "Data clustering: 50 Years Beyond K-Means". *Pattern Recognition Letters*, 31(8):651–666. 2010.

[139] Anil K. Jain, M. Narasimha Murty, and Patrick J. Flynn. "Data Clustering: A Review". *ACM Computing Surveys (CSUR)*, 31(3):264–323. 1999.

[140] Eshref Januzaj, Hans-Peter Kriegel, and Martin Pfeifle. "DBDC: Density Based Distributed Clustering". In "Advances in Database Technology-EDBT 2004", Springer, pp. 88–105. 2004.

[141] H. D. Jin. *Scalable Model-Based Clustering Algorithms for Large Databases and Their Applications*. Ph.D. thesis, The Chinese University of Hong Kong. Aug. 2002.

[142] G. Karypis, R. Aggarwal, V. Kumar, and Shashi Shekhar. "Multilevel hypergraph partitioning: Applications in VLSI domain". *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 7(1):69–79. doi:10.1109/92.748202. March 1999.

[143] G. Karypis, E. Han, and V. Kumar. "Chameleon: Hierarchical Clustering Using Dynamic Modeling". *IEEE Computer*, 32(8):68–75. Aug. 1999.

[144] George Karypis and Vipin Kumar. *METIS: A Software Package for Partitioning Unstructured Graphs, Partitioning Meshes, and Computing Fill-Reducing Orderings of Sparse Matrices*. University of Minnesota, Department of Computer Science. September 1998.

[145] L. Kaufman and P. Rousseeuw. *Clustering by Means of Medoids*. North-Holland. 1987.

[146] Leonard Kaufman and Peter J. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*, vol. 344. John Wiley & Sons. 2009.

[147] Paul Kellam, Xiaohui Liu, Nigel Martin, Christine Orengo, Stephen Swift, and Allan Tucker. "Comparing, Contrasting and Combining Clusters in Viral Gene Expression Data". In "Proceedings of 6th Workshop on Intelligent Data Analysis in Medicine and Pharmocology", pp. 56–62. 2001.

[148] James F. Kennedy, James Kennedy, and Russell C. Eberhart. *Swarm Intelligence*. Morgan Kaufmann. 2001.

[149] Benjamin King. "Step-Wise Clustering Procedures". *Journal of the American Statistical Association*, 62(317):86–101. 1967.

[150] D. Klein, S. Kamvar, and C. Manning. "From Instance-Level Constraints to Space-Level Constraints: Making the Most of Prior Knowledge in Data Clustering". In "Proceedings of the 19th International Conference on Machine Learning", pp. 307–314. 2002.

[151] R. W. Klein and R. C. Dubes. "Experiments in Projection and Clustering by Simulated Annealing". *Pattern Recognition*, 22(2):213–220. 1989.

[152] J. Kleinberg. "An Impossibility Theorem for Clustering". *Proceeding Conference Advances in Neural Information Processing Systems*, 15:463–470. 2002.

[153] Teuvo Kohonen. "Self-Organized Formation of Topologically Correct Feature Maps". *Biological Cybernetics*, 43(1):59–69. 1982.

[154] Hans-Peter Kriegel, Peer Kröger, Matthias Renz, and Sebastian Wurst. "A Generic Framework for Efficient Subspace Clustering of High-Dimensional Data." In "Data Mining, Fifth IEEE International Conference on", IEEE Computer Society, pp. 250–257. 2005.

[155] Peer Kröger, Hans-Peter Kriegel, and Karin Kailing. "Density-Connected Subspace Clustering for High-Dimensional Data." In Michael W. Berry, Umeshwar Dayal, Chandrika Kamath, and David B. Skillicorn, editors, "SDM", SIAM, vol. 4. 2004.

[156] 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. 1956.

[157] N. Kumar and R. S. Joshi. "Data Clustering Using Artificial Neural Networks". In "Proceedings of National Conference on Challenges & Opportunities in Information Technology (COIT-2007)", pp. 197–200. 2007.

[158] K. W. Lau, H. Yin, and S. Hubbard. "Kernel Self-Organising Maps for Classification". *Neurocomputing*, 69(16-18):2033 – 2040. doi:http://dx.doi.org/10.1016/j.neucom.2005.10.003. 2006.

[159] Laura Lazzeroni and Art Owen. "Plaid Models for Gene Expression Data". *Statistica Sinica*, 12(1):61–86. 2000.

[160] Tao Li and Chris H. Q. Ding. "Weighted Consensus Clustering". In "SDM'08", pp. 798–809. 2008.

[161] Wei Li and Andrew McCallum. "Pachinko Allocation: DAG-Structured Mixture Models of Topic Correlations". In "Proceedings of the 23rd International Conference on Machine Learning", ACM, New York, NY, USA, ICML '06, pp. 577–584. doi:10.1145/1143844.1143917. 2006.

[162] Wei-keng Liao, Ying Liu, and Alok Choudhary. "A Grid-Based Clustering Algorithm Using Adaptive Mesh Refinement". In "7th Workshop on Mining Scientific and Engineering Datasets of SIAM International Conference on Data Mining", 2004.

[163] Ricardo Linden. "Clustering Techniques". *Revista de Sistemas de Informação da FSMA*, 4:18–36. 2009.

[164] Jingwei Liu and Meizhi Xu. "Kernelized Fuzzy Attribute C-Means Clustering Algorithm." *Fuzzy Sets and Systems*, 159(18):2428–2445. 2008.

[165] Huilan Luo, Furong Jing, and Xiaobing Xie. "Combining Multiple Clusterings using Information Theory based Genetic Algorithm". In "2006 International Conference on Computational Intelligence and Security", vol. 1, pp. 84–89. doi:10.1109/ICCIAS.2006.294095. 2006.

[166] P. C. H. Ma, K. C. C. Chan, Xin Yao, and D. K. Y. Chiu. "An Evolutionary Clustering Algorithm for Gene Expression Microarray Data Analysis". *Evolutionary Computation, IEEE Transactions on*, 10(3):296 – 314. doi:10.1109/TEVC.2005.859371. June 2006.

[167] J. B. MacQueen. "Some Methods for Classification and Analysis of Multivariate Observations". *Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability. University of California Press.*, 1(14):281–297. 1967.

[168] Sara C. Madeira and Arlindo L. Oliveira. "Biclustering Algorithms for Biological Data Analysis: A Survey". *IEEE/ACM Trans. Comput. Biol. Bioinformatics*, 1(1):24–45. doi:http://dx.doi.org/10.1109/TCBB.2004.2. 2004.

[169] S. W. Mahfoud. *Niching Methods for Genetic Algorithms*. Ph.D. thesis, University of Illinois at Urbana-Champaign. 1995.

[170] Jianchang Mao and Anil K. Jain. "A Self-Organizing Network for Hyperellipsoidal Clustering (HEC)". *IEEE Transactions on Neural Networks*, 7(1):16–29. 1996.

[171] Thomas M. Martinetz and Klaus J. Schulten. "A "Neural Gas" Network Learns Topologies". In Teuvo Kohonen, Kai Mäkisara, Olli Simula, and Jari Kangas,

editors, “Proceedings of the International Conference on Artificial Neural Networks (Espoo, Finland)”, Amsterdam; New York: North-Holland, pp. 397–402. 1991.

[172] G. J. McLachlan and K. E. Basford. *Mixture Models: Inference and Applications to Clustering*. Marcel Dekker Inc, New York / Basel. 1988.

[173] Geoffrey J. McLachlan, Shu-Kay Ng, and Richard Bean. “Robust Cluster Analysis via Mixture Models”. *Austrian Journal of Statistics*, 35(2):157–174. 2006.

[174] Marina Meila and Jianbo Shi. “Learning Segmentation by Random Walks”. In “In Advances in Neural Information Processing Systems”, MIT Press, pp. 873–879. 2001.

[175] Boriana L. Milenova and Marcos M. Campos. “O-cluster: Scalable Clustering of Large High Dimensional Data Sets”. In “Data Mining, 2002. ICDM 2003. Proceedings. 2002 IEEE International Conference on”, IEEE, pp. 290–297. 2002.

[176] Gabriela Moise, Jörg Sander, and Martin Ester. “P3C: A Robust Projected Clustering Algorithm”. In “ICDM”, IEEE Computer Society, pp. 414–425. 2006.

[177] Gabriela Moise, Jörg Sander, and Martin Ester. “Robust Projected Clustering”. *Knowledge and Information Systems*, 14(3):273–298. 2008.

[178] B. Moore. “ART1 and Pattern Clustering”. *Proceedings Connectionist Models Summer School*, pp. 174–185. 1989.

[179] Tadeusz Morzy, Marek Wojciechowski, and Maciej Zakrzewicz. “Pattern-Oriented Hierarchical Clustering”. In “Proceedings of the third East-European Symposium on Advances in Databases and Information Systems - ADBIS-99, Slovenia, LNCS 1691”, pp. 179–190. 1999.

[180] T. S. Motzkin and E. G. Straus. “Maxima for Graphs and a New Proof of a Theorem of Turán”. *Canadian Journal of Mathematics*, 17(4):533–540. 1965.

[181] Anirban Mukhopadhyay and Ujjwal Maulik. “A Multiobjective Approach to MR Brain Image Segmentation.” *Applied Soft Computing*, 11(1):872–880. 2011.

[182] Fionn Murtagh. “A Survey of Recent Advances in Hierarchical Clustering Algorithms”. *The Computer Journal*, 26(4):354–359. 1983.

[183] Megha Nangia. “Partitional Clustering”. *ACM student chapter, SIGKDD Presentation*. February 2012.

[184] S. B. Needleman and C. D. Wunsch. “A General Method Applicable to the Search for Similarities in the Amino Acid Sequence of Two Proteins”. *Journal of Molecular Biology*, 48(3):443–453. Mar. 1970.

[185] Jaroslav Nešetřil, Eva Milková, and Helena Nešetřilová. “Otakar Boruvka on Minimum Spanning Tree Problem Translation of Both the 1926 Papers, Comments, History”. *Discrete Mathematics*, 233(1):3–36. 2001.

[186] Andrew Y. Ng, Michael I. Jordan, and Yair Weiss. “On Spectral Clustering: Analysis and an Algorithm”. In “Advances in Neural Information Processing Systems”, MIT Press, pp. 849–856. 2001.

[187] Raymond T. Ng and Jiawei Han. “Efficient and Effective Clustering Methods for Spatial Data Mining”. In “Proceedings of the 20th International Conference on Very Large Data Bases”, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, VLDB ’94, pp. 144–155. 1994.

[188] Tim Oates, Laura Firoiu, and Paul R. Cohen. “Using Dynamic Time Warping to Bootstrap HMM-Based Clustering of Time Series.” In Ron Sun and C. Lee Giles, editors, “Sequence Learning”, Springer, vol. 1828 of *Lecture Notes in Computer Science*, pp. 35–52. 2001.

[189] Atsuyuki Okabe and Kokichi Sugihara. *Spatial Analysis Along Networks: Statistical and Computational Methods*. John Wiley & Sons. 2012.

[190] Niina Päivinen. “Clustering with a Minimum Spanning Tree of Scale-Free-Like Structure”. *Pattern Recognition Letters*, 26(7):921–930. 2005.

[191] Massimiliano Pavan and Marcello Pelillo. “Dominant Sets and Pairwise Clustering”. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(1):167–172. doi:10.1109/TPAMI.2007.10. Jan. 2007.

[192] Z. Pawlak. *Rough Sets: Theoretical Aspects of Reasoning About Data*. Mathematics and Its Applications. Soviet Series. Springer Netherlands. 1991.

[193] Karl Pearson. “LIII. On Lines and Planes of Closest Fit to Systems of Points in Space”. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572. 1901.

[194] W. R. Pearson and D. J. Lipman. “Improved Tools for Biological Sequence Comparison”. *Proceedings of the National Academy of Sciences of the USA*, 85(8):2444–2448. doi:10.1073/pnas.85.8.2444. Apr. 1988.

[195] Dan Pelleg and Andrew Moore. “Accelerating Exact K-Means Algorithms with Geometric Reasoning”. In “Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining”, ACM, pp. 277–281. 1999.

[196] Dan Pelleg and Andrew W. Moore. “X-Means: Extending K-Means with Efficient Estimation of the Number of Clusters”. In “Proceedings of the Seventeenth International Conference on Machine Learning”, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, ICML ’00, p. 727–734. 2000.

[197] Harun Pirim, Dilip Gautam, Tanmay Bhowmik, Andy D. Perkins, and Burak Ekioglu. “Performance of an Ensemble Clustering Algorithm on Biological Data Sets”. *Mathematical and Computational Applications*, 16(1):87–96. 2011.

[198] Clara Pizzuti and Domenico Talia. "P-AutoClass: Scalable Parallel Clustering for Mining Large Data Sets". *Knowledge and Data Engineering, IEEE Transactions on*, 15(3):629–641. doi:10.1109/TKDE.2003.1198395. Mar. 2003.

[199] J. Platt. *Advances in Kernel Methods: Support Vector Learning*, MIT press, Cambridge, MA, chap. Fast training of SVMs using sequential minimal optimization, pp. 185–208. 1999.

[200] R. C. Prim. "Shortest Connection Networks and Some Generalizations". *Bell System Technology Journal*, 36(6):1389–1401. 1957.

[201] Cecilia Magdalena Procopiuc, Michael Jones, Pankaj K. Agarwal, and T. M. Murali. "A Monte Carlo Algorithm for Fast Projective Clustering." In Michael J. Franklin, Bongki Moon, and Anastassia Ailamaki, editors, "SIGMOD Conference", ACM, pp. 418–427. 2002.

[202] Marco Ramoni, Paola Sebastiani, and Paul R. Cohen. "Bayesian Clustering by Dynamics." *Machine Learning*, 47(1):91–121. 2002.

[203] Rocío Romero-Záliz, Cristina Rubio-Escudero, J. P. Cobb, Francisco Herrera, Oscar Cordón, and Igor Zwir. "A Multiobjective Evolutionary Conceptual Clustering Methodology for Gene Annotation Within Structural Databases: A Case of Study on the Gene Ontology Database." *Evolutionary Computation, IEEE Transactions on*, 12(6):679–701. 2008.

[204] Carlos Ruiz, Myra Spiliopoulou, and Ernestina Menasalvas. "C-DBSCAN: Density-Based Clustering with Constraints". In "Rough Sets, Fuzzy Sets, Data Mining and Granular Computing", Springer, pp. 216–223. 2007.

[205] Jörg Sander, Martin Ester, Hans-Peter Kriegel, and Xiaowei Xu. "Density-Based Clustering in Spatial Databases: The Algorithm GDBSCAN and Its Applications". *Data Mining and Knowledge Discovery*, 2(2):169–194. doi:10.1023/A:1009745219419. Jun. 1998.

[206] E. Schikuta. "Grid-Clustering: An Efficient Hierarchical Clustering Method for Very Large Data Sets". *Pattern Recognition, 1996., Proceedings of the 13th International Conference on*, 2:101–105. doi: 10.1109/ICPR.1996.546732. Aug. 1996.

[207] Erich Schikuta and Martin Erhart. "The BANG-Clustering System: Grid-Based Data Analysis". In "Advances in Intelligent Data Analysis Reasoning about Data", Springer, pp. 513–524. 1997.

[208] Bernhard Schölkopf, Chris Burges, and Vladimir Vapnik. "Incorporating Invariances in Support Vector Learning Machines". In "Artificial Neural Networks ICANN 96", Springer, pp. 47–52. 1996.

[209] Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. "Nonlinear Component Analysis as a Kernel Eigenvalue Problem". *Neural Computation*, 10(5):1299–1319. doi:10.1162/089976698300017467. Jul. 1998.

[210] A. J. Scott and Michael J Symons. "Clustering Methods Based on Likelihood Ratio Criteria". *Biometrics*, pp. 387–397. 1971.

[211] Shokri Z. Selim and K. Alsultan. "A Simulated Annealing Algorithm for the Clustering Problem." *Pattern Recognition*, 24(10):1003–1008. 1991.

[212] Gholamhosein Sheikholeslami, Surojit Chatterjee, and Aidong Zhang. "WaveCluster: A Multi-Resolution Clustering Approach for Very Large Spatial Databases". In "Proceedings of the 24rd International Conference on Very Large Data Bases", Morgan Kaufmann Publishers Inc., pp. 428–439. 1998.

[213] Weiguo Sheng, Allan Tucker, and Xiaohui Liu. *Clustering with Niching Genetic K-Means Algorithm*, Springer Berlin / Heidelberg, pp. 162–173. Lecture Notes in Computer Science, Genetic and Evolutionary Computation – GECCO 2004. 2004.

[214] Jianbo Shi and Jitendra Malik. "Normalized Cuts and Image Segmentation". *IEEE. Reprinted from IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905. August 2000.

[215] R. Sibson. "SLINK: An Optimally Efficient Algorithm for the Single-Link Cluster Method". *The Computer Journal (British Computer Society)*, 16(1):30–34. 1973.

[216] Slowmo. "Example 2D Space with Subspace Clusters". URL http://commons.wikimedia.org/wiki/File:SubspaceClustering.png. Accessed Oct. 2015. 2010.

[217] Temple F. Smith and Michael S. Waterman. "New Stratigraphic Correlation Techniques". *The Journal of Geology*, 88(4):451–457. Jul. 1980.

[218] Padhraic Smyth. "Clustering Sequences with Hidden Markov Models". In Michael Mozer, Michael I. Jordan, and Thomas Petsche, editors, "Advances in Neural Information Processing", MIT Press, pp. 648–654. 1996.

[219] Padhraic Smyth. "Probabilistic Model-Based Clustering of Multivariate and Sequential Data". In "Proceedings of Artificial Intelligence and Statistics", Morgan Kaufmann, pp. 299–304. 1999.

[220] P. Sneath and R. Sokal. "Numerical Taxonomy". In "Numerical Taxonomy", W. H. Freeman and Company. 1973.

[221] Peter H. A. Sneath. "The Application of Computers to Taxonomy". *Journal of general microbiology*, 17(1):201–226. 1957.

[222] R. R. Sokal and C. D. Michener. "A Statistical Method for Evaluating Systematic Relationships". *The University of Kansas Scientific Bulletin*, 38:1409–1438. 1958.

[223] T. Sorensen. "A Method of Establishing Groups of Equal Amplitude in Plant Sociology Based on Similarity of

Species Content and its Application to Analyzes of the Vegetation on Danish Commons". *Biologiske Skrifter*, 5:1–34. 1948.

[224] H. Spath. "Cluster Analysis Algorithms". In "Cluster Analysis Algorithms", West Sussex, Ellis Horwood Limited. 1980.

[225] Michael Steinbach, George Karypis, and Vipin Kumar. "Efficient Algorithms for Creating Product Catalogs". Tech. rep., DTIC Document. 2000.

[226] H. Steinhaus. "Sur la Division des corp Materiels en Parties". *Bulletin of Acad. Polon. Sci.*, 4(12):801–804. 1956.

[227] Alexander Strehl and Joydeep Ghosh. "Cluster Ensembles – A Knowledge Reuse Framework for Combining Multiple Partitions". *Journal of Machine Learning Research*, 3:583–617. 2002.

[228] Ron Sun and C. Lee Giles. *Sequence Learning: Paradigms, Algorithms, and Applications*, vol. 1828. Springer. 2001.

[229] Amos Tanay, Roded Sharan, Martin Kupiec, and Ron Shamir. "Revealing Modularity and Organization in the Yeast Molecular Network by Integrated Analysis of Highly Heterogeneous Genomewide Data". *PNAS*, 101(9):2981–2986. 2004.

[230] Chun Tang, Li Zhang, Aidong Zhang, and Murali Ramanathan. "Interrelated Two-way Clustering: An Unsupervised Approach for Gene Expression Data Analysis". In "Bioinformatics and Bioengineering Conference, 2001. Proceedings of the IEEE 2nd International Symposium on", pp. 41–48. 2001.

[231] S. Theodoridis and K. Koutroumbas. *Pattern Recognition, Third Edition*. Academic Press, Inc., Orlando, FL, USA. 2006.

[232] S. C. A. Thomopoulos, D. K. Bougoulias, and Chin-Der Wann. "Dignet: An Unsupervised-Learning Clustering Algorithm for Clustering and Data Fusion". *Aerospace and Electronic Systems, IEEE Transactions on*, 31(1):21–38. doi:10.1109/7.366289. Jan 1995.

[233] Alexander P. Topchy, Anil K. Jain, and William F. Punch. "Combining Multiple Weak Clusterings." In "Proceedings of the IEEE International Conference on Data Mining", IEEE Computer Society, pp. 331–338. 2003.

[234] Alexander P. Topchy, Anil K. Jain, and William F. Punch. "A Mixture Model for Clustering Ensembles." In "Proceedings SIAM International Conference on Data Mining", SIAM. 2004.

[235] Sandro Vega-Pons and José Ruiz-Shulcloper. "A Survey of Clustering Ensemble Algorithms". *International Journal of Pattern Recognition and Artificial Intelligence*, 25(3):337–372. 2011.

[236] Boaz Vigdor and Boaz Lerner. "The Bayesian ARTMAP". *IEEE Transactions on Neural Networks*, 18(6):1628–1644. 2007.

[237] Michail Vlachos, Jessica Lin, Eamonn Keogh, and Dimitrios Gunopulos. "A Wavelet-Based Anytime Algorithm for K-Means Clustering of Time Series". In "In Proceedings Workshop on Clustering High Dimensionality Data and Its Applications", pp. 23–30. 2003.

[238] K. Wagstaff and C. Cardie. "Clustering with Instance-level Constraints". *Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000)*, pp. 1103–1110. 2000.

[239] Haiying Wang, Huiru Zheng, and Francisco Azuaje. "Poisson-Based Self-Organizing Feature Maps and Hierarchical Clustering for Serial Analysis of Gene Expression Data". *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 4(2):163–175. doi:http://0-dx.doi.org.innopac.library.unr.edu/10.1109/TCBB.2007.070204. 2007.

[240] Hongjun Wang, Hanhuai Shan, and Arindam Banerjee. "Bayesian Cluster Ensembles". *Statistical Analysis and Data Mining*, 4(1):54–70. doi:10.1002/sam.10098. 2011.

[241] Wei Wang, Jiong Yang, and Richard R. Muntz. "STING: A Statistical Information Grid Approach to Spatial Data Mining". In Matthias Jarke, Michael J. Carey, Klaus R. Dittrich, Frederick H. Lochovsky, Pericles Loucopoulos, and Manfred A. Jeusfeld, editors, "VLDB'97, Proceedings of 23rd International Conference on Very Large Data Bases, August 25-29, 1997, Athens, Greece", Morgan Kaufmann, pp. 186–195. 1997.

[242] Wei Wang, Jiong Yang, and Richard R. Muntz. "STING+: An Approach to Active Spatial Data Mining." In Masaru Kitsuregawa, Michael P. Papazoglou, and Calton Pu, editors, "Data Engineering, 1999. Proceedings., 15th International Conference on", IEEE Computer Society, pp. 116–125. 1999.

[243] Xiang Wang and Ian Davidson. "Active Spectral Clustering". In "Data Mining (ICDM), 2010 IEEE 10th International Conference on", IEEE, pp. 561–568. 2010.

[244] Chin-Der Wann and Stelios C. A. Thomopoulos. "A Comparative Study of Self-organizing Clustering Algorithms Dignet and ART2." *Neural Networks*, 10(4):737–753. 1997.

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

[246] Max Welling. "Learning in Markov Random Fields with Contrastive Free Energies". In "Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics", pp. 397–404. 2005.

[247] O. Wildi. *Data Analysis in Vegetation Ecology*. John Wiley & Sons. 2010.

[248] James R. Williamson. "Gaussian ARTMAP: A Neural Network for Fast Incremental Learning of Noisy

Multidimensional Maps." *Neural Networks*, 9(5):881–897. 1996.

[249] John H. Wolfe. "Pattern Clustering by Multivariate Mixture Analysis". *Multivariate Behavioral Research*, 5(3):329–350. 1970.

[250] Donald C. Wunsch, Thomas P. Caudell, C. David Capps, Robert J. Marks, and R. Aaron Falk. "An Optoelectronic Implementation of the Adaptive Resonance Neural Network." *IEEE Transactions on Neural Networks*, 4(4):673–684. 1993.

[251] Chen Xiaoyun, Chen Yi, Qi Xiaoli, Yue Min, and He Yanshan. "PGMCLU: A Novel Parallel Grid-Based Clustering Algorithm for Multi-Density Datasets". In "Web Society, 2009. SWS'09. 1st IEEE Symposium on", IEEE, pp. 166–171. 2009.

[252] Eric P. Xing, Andrew Y. Ng, Michael I. Jordan, and Stuart Russell. "Distance Metric Learning with Application to Clustering with Side-Information". *Advances in Neural Information Processing Systems*, pp. 521–528. 2003.

[253] Yimin Xiong and Dit-Yan Yeung. "Time Series Clustering with ARMA Mixtures". *Pattern Recognition*, 37(8):1675–1689. 2004.

[254] Rui Xu and D. Wunsch. "Survey of Clustering Algorithms". *Neural Networks, IEEE Transactions on*, 16(3):645 –678. doi:10.1109/TNN.2005.845141. may 2005.

[255] Rui Xu and D. Wunsch. *Clustering*. IEEE/Wiley. 2009.

[256] Rui Xu and D. C. Wunsch. "Clustering Algorithms in Biomedical Research: A Review". *Biomedical Engineering, IEEE Reviews in*, 3:120. 2010.

[257] Xiaowei Xu, Martin Ester, Hans-Peter Kriegel, and Jörg Sander. "A Distribution-Based Clustering Algorithm for Mining in Large Spatial Databases". In "Proceedings of the Fourteenth International Conference on Data Engineering", IEEE Computer Society, Washington, DC, USA, ICDE '98, pp. 324–331. 1998.

[258] Xiaowei Xu, Nurcan Yuruk, Zhidan Feng, and Thomas A. J. Schweiger. "SCAN: A Structural Clustering Algorithm for Networks". In "Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining", ACM, pp. 824–833. 2007.

[259] Ronald R. Yager. "Intelligent Control of the Hierarchical Agglomerative Clustering Process". *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 30(6):835–845. 2000.

[260] Donghui Yan, Ling Huang, and Michael I Jordan. "Fast Approximate SpectralCclustering". In "Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining", ACM, pp. 907–916. 2009.

[261] Zhao Yanchang and Song Junde. "GDILC: a Grid-Based Density-Isoline Clustering Algorithm". In "2001 International Conferences on Info-Tech and Info-Net. Proceedings (Cat. No.01EX479)", vol. 3, pp. 140–145. doi:10.1109/ICII.2001.983048. 2001.

[262] Qiang Yang and Xindong Wu. "10 Challenging Problems in Data Mining Research". *International Journal of Information Technology and Decision Making (IJITDM)*, 05(04):597–604. 2006.

[263] James Yolkowski. "The Clustering Illusion". URL http://mathlair.allfunandgames.ca/clustering.php. Accessed Oct. 2015. 2014.

[264] Stella X. Yu and Jianbo Shi. "Multiclass Spectral Clustering". In "Proceedings of the Ninth IEEE International Conference on Computer Vision - Volume 2", IEEE Computer Society, Washington, DC, USA, ICCV '03, pp. 313–319. 2003.

[265] Stefanos Zafeiriou and Nikolaos A. Laskaris. "On the Improvement of Support Vector Techniques for Clustering by Means of Whitening Transform." *Signal Processing Letters, IEEE*, 15:198–201. 2008.

[266] Charles T Zahn. "Graph-Theoretical Methods for Detecting and Describing Gestalt Clusters". *Computers, IEEE Transactions on*, 100(1):68–86. 1971.

[267] B. Zhang. "Generalized K-Harmonic means – Dynamic Weighting of Data in Un-supervised Learning". *Proceedings of the 1st SIAM ICDM, Chicago, IL, USA*, pp. 1–13. 2001.

[268] Dao-Qiang Zhang and Song-Can Chen. "A Novel Kernelized Fuzzy C-means Algorithm with Application in Medical Image Segmentation". *Artificial Intelligence in Medicine*, 32(1):37–50. 2004.

[269] Tian Zhang, Raghu Ramakrishnan, and Miron Livny. "BIRCH: An Efficient Data Clustering Method for Very Large Databases". *SIGMOD Record*, 25(2):103–114. doi:10.1145/235968.233324. Jun. 1996.

[270] Shangming Zhou and John Q. Gan. "An Unsupervised Kernel Based Fuzzy C-means Clustering Algorithm with Kernel Normalisation". *International Journal of Computational Intelligence and Applications*, 04(04):355–373. doi:10.1142/S1469026804001379. 2004.

[271] H. Zhuang, Y. Huang, K. Palaniappan, and Y. Zhao. "Gaussian Mixture Density Modeling, Decomposition, and Applications". *IEEE Trans. Image Processing*, 5(9):1293–1302. Sep. 1996.

Yan Yan received her MS degree in Computer Science and Engineering from the University of Nevada, Reno. After this she worked in industry as a Software Engineer. She then came back to the University of Nevada, Reno and received her Ph.D in Computer Science and Engineering in 2019.

She is currently working as a Software Engineering Consultant. Her research interests are in Cancer Subtyping and Clustering of large data sets.

Frederick C. Harris Jr. received his BS and MS degrees in Mathematics and Educational Administration from Bob Jones University, Greenville, SC, USA in 1986 and 1988 respectively. He then went on and received his MS and Ph.D. degrees in Computer Science from Clemson University, Clemson, SC, USA in 1991 and 1994 respectively.

He is currently a Professor in the Department of Computer Science and Engineering and the Director of the High Performance Computation and Visualization Lab at the University of Nevada, Reno, USA. He is also the Nevada State EPSCoR Director and the Project Director for Nevada NSF EPSCoR. He has published more than 290 peer-reviewed journal and conference papers along with several book chapters. He has had 14 PhD students and 80 MS Thesis students finish under his supervision. His research interests are in parallel computation, simulation, computer graphics, and virtual reality. He is also a Senior Member of the ACM, and a Senior Member of the International Society for Computers and their Applications (ISCA).

Clustered Particle Swarm Optimization Using Self-Organizing Maps

Yongwon Park*, Harika Kilari* and Sanjeev Baskiyar*
Auburn University, Auburn, AL 36849

Abstract

Premature convergence of particles is known to be the main cause of local optima in particle swarm optimization (PSO). Such premature convergence limits the use of PSO for the multimodal function optimization problems which may have more than one optimum of a function and many local minima. In this paper, we use a novel approach called clustered PSO or CPSO, which clusters particles periodically around sample vectors (particles) using self-organizing maps. Such clustering provides sufficient diversity providing CPSO with the opportunity to explore other solutions while proactively escaping local optima. Furthermore, it enables each cluster of particles to search on the solution space concurrently for multimodal optimization problems being readily able to escape many local minima to reach the minimum. The abilities of the proposed approach in escaping local optima and finding a global solution were evaluated through simulations on the test problems used by other researchers. The results of simulations conclusively demonstrate that CPSO is highly effective in avoiding local minima for multimodal function optimization problems as well as reducing the particle population with every iteration.

Key Words: Clustering; particle swarm optimization; local minima; self-organizing maps

1 Introduction

Image synthesis of dynamic objects such as clouds, smoke, water, and fire are known to be difficult to simulate due to their fuzziness. Researchers have studied particle systems in which particles have their behavior [11] and have tried to simulate such natural dynamic objects. Many scientists have also been interested in the movement of a flock of birds or a school of fish to discover underlying rules that make possible their aggregate motion. The aggregate motion enables them to find food quickly and protects them from predators through early detection and the spread of information. Intrigued by such social behavior, Kennedy and Eberhart [4] developed the particle swarm optimization (PSO) method to apply their natural behavior to problem-solving. In PSO, particles simulate the social behavior of a flock of birds, which cooperate to find food (goal), in a way that each one remembers its personal best location ever visited called "*personal best*," and shares the local information with their neighbor to identify "*local best*" location within a group.The "*local best*" of a group is shared with other groups to identify the "*global best*" value. The local information refers to individual cognition and the global information to social interaction. Using the local and global information, a swarm of particles is able to cooperate and explore the solution space effectively to find an optimal solution. One of the advantages of PSO is the particle's prompt convergence on solutions by exploring the solution space at a fast speed like a flock of birds moving fast, but in astonishingly perfect harmony. However, the solution by early convergence may cause a local solution if done prematurely. This drawback of PSO can be attributed to its lack of ability to indicate premature convergence. The premature convergence phenomenon is commonly observed in evolutionary methods such as Genetic Algorithms (GA). In the case of a GA, it is known that high selection pressure in choosing only superior offsprings for new generations results in premature convergence. This is because the high selection pressure restricts the diversity of the new population, searching for solution space limited to local areas. Therefore, a search may get stuck in a local optimum. Riget and Vesterstrlm attributed the premature convergence of PSO to the fast information flow between particles, which causes particles to cluster early around local minima [12]. Since little diversity among the population is considered the main cause of premature convergence, many researchers [6] [18] [1] have tried to avoid premature convergence in a way that provides sufficient diversity for particles at the indication of premature convergence. The earlier methods like k-means clustering had an overhead of how to decide an optimum value of k, and also could not consider the dynamic factor in particle population update by using fixed pre-defined parameters. Later complex hierarchical and graphical clustering techniques have been devised which had low applicability due to structural limitations in data analysis. However, it remains a very difficult problem to identify the sign of premature convergence. Furthermore, using such a reactive approach may be difficult to escape local minima for optimization problems. In this research, we discuss a novel Cluster-PSO (CPSO) that we developed in 2011, using self-

*Department of Computer Science and Software Engineering Emails: yongwonpark@gmail.com, hzk0041@tigermail.auburn.edu, and baskiyar@eng.auburn.edu

organizing maps (SOM), which is known as an unsupervised learning method called "*Kohonen vector*" [5]. Alkazemi and Mohan [1], and Seo et al.[13] used clustered particles to search the solution space concurrently. However, the search may not be adaptive since the population of the group is static. On the other hand, CPSO using the SOM technique is able to make the search adaptive to the solution space by dynamically updating the population of clustered particles. The unsupervised learning method of SOM is able to cover large solution space effectively by periodically clustering particles around randomly created *Kohonen* vectors, thereby preventing local optima. The SOM helps to simulate the high dimensional topology information of particle swarm into $2D$ or $3D$ vectors by using mathematical quantification and projection.The preprocessing of raw data into clusters by SOM helps in efficient local search, which in turn helps in maintaining diversity in the optimization process.

The remainder of this paper has been organized as follows. Section 2 describes the related works, Section 3, the PSO model, and Section 4 the CPSO algorithm. In Section 5, we discuss the results and in Section 6 we make concluding remarks.

2 Related work

The PSO model simulates a swarm of particles moving in an m-dimensional solution space where a particle corresponds to a candidate solution characterized by m attributes. It is represented in the solution space by its position vector $\vec{x}_i$, and its velocity is represented by a velocity vector $\vec{v}_i$.

The velocity of the i^{th} particle of the swarm and its projected position in the d^{th} dimension are defined by the following two equations:

$$\begin{aligned}\vec{v}_{id}(t+1) = \vec{v}_{id}(t) + c_1 \cdot rand() \cdot (\vec{l}_{id} - \vec{x}_{id}(t)) \\ + c_2 \cdot rand() \cdot (\vec{g}_{id} - \vec{x}_{id}(t))\end{aligned} \tag{1}$$

$$\vec{x}_{id}(t+1) = \vec{x}_{id}(t) + \vec{v}_{id}(t+1) \tag{2}$$

where:

- n is the size of the swarm
- m is the number of dimensions in the solution space
- $i = 1, \ldots, n$
- $d = 1, \ldots, m$
- $\vec{l}_{id}$ is the local best position of particle i on the d^{th} dimension
- $\vec{g}_i$ is the global best position of particle i in the d^{th} dimension
- c_1 is the learning rate of particles for individual cognition
- c_2 is the learning rate of particles for social interaction
- $rand()$ is the random function with the output in the range $(0 \ldots 1)$

Shi and Eberhart [16] introduced a parameter *inertia weight* ω into the basic PSO:

$$\begin{aligned}\vec{v}_{id}(t+1) = \omega \cdot \vec{v}_{id}(t) + c_1 \cdot rand() \cdot (\vec{l}_{id} - \vec{x}_{id}(t+1)) \\ + c_2 \cdot rand() \cdot (\vec{g}_{id} - \vec{x}_{id}(t+1))\end{aligned} \tag{3}$$

where ω weights the magnitude of the old velocity $\vec{v}_{id}(t)$. They found the range of $(0.9 \ldots 1.2)$ a good area to choose ω from.

Many researchers have tackled the premature convergence problem in PSO. They tried to overcome it in a reactive way that provides diversity to particles at the indication of premature convergence to escape a local optimum. Krink and Riget [6] provided diversity for particles upon indication of a collision. The indication of the collision was determined based on the distance between particles, and subsequently, diversity was provided in a way that particles bounce away randomly or make a U-turn by increasing their velocity to collide against the boundary of the solution space. The tailored PSO outperformed the basic PSO for several benchmark functions. However, the reactive method may not be sufficient for complex optimization problems. Once converged at a local optimum, clustered on local best by their nature, particles would struggle to escape the local optimum without substantial diversity. On the other hand, CPSO explores solution space by explicitly clustering particles around randomly created sample vectors, thus being able to escape local optima for optimization problems. Wei, Guangbin and Dong [19] presented Elite Particle Swarm with Mutation (EPSM). EPSM tried to take advantage of best fit particles to avoid wasting time visiting the solution space with poor fitness values. To do this, particles with poor fitness were substituted by *elite particles* with better fitness. But such elitism decreases the diversity of particles. To provide diversity EPSM employed a mutation operator so that the global best particle may be mutated to generate a new particle. EPSM outperformed the Standard Particle Swarm Optimization [16] with respect to the quality of the solution. In contrast to the elitism, Wang and Qiu [18] tried to give opportunities to inferior particles to search solution space. Their approach was motivated by the observation that a search process is very likely to be dominated by several super particles, which is often not good in the long term. In order to alleviate the dominance by super particles, the selection probability of a particle was set inversely proportional to its original fitness. Next, a particle is selected in the roulette wheel manner to explore the solution space. Such a procedure is expected to mitigate the high selection pressure by super particles. Their approach outperformed other known algorithms in terms of solution quality but took additional computational time for the fitness scaling and roulette selecting process. Veeramachaneni and Osadciw [17] claimed that particles by nature oscillate between local optima and a global optimum, wasting time moving in the same direction to converge at a global optimum. Therefore, they made particles attract toward the best positions visited by their neighbors. In other words, particles are influenced by successful neighbors to explore the solution space. This algorithm was further improved by concurrent PSO implementation [2] in which two particle groups worked concurrently, with each group tracing particles independently and sharing the information about the best particle. Similarly, the Multi-Phase Particle Swarm Optimization algorithm (MPSO) employed multiple groups of particles, each changing a search direction in every

phase to increase population diversity [1]. Seo et al. [13] presented multi-grouped particle swarm optimization (MGPSO) algorithm in which each group searches its own best solution independently. They prevented each group from interfering with other groups by regulating the radius of each group. Both MPSO and MGPSO provide concurrent search through clusters of particles. However, since early grouped particles search the solution space throughout the search, it may not be adaptive in a pathological environment. On the other hand, CPSO can be more adaptive to the varying environments by periodically updating the population of a group randomly. Sha and Yang [14] proposed APSO K-means clustering for speaker recognition where ant colony algorithm and PSO algorithm were combined with K-means clustering. In K-means clustering the number of groups is fixed, whilst the number of groups in CPSO is updated dynamically throughout the search of the solution space. Ratnaweera, Watson and Saman [10] used random dimension and time-varying coefficients to compute particle velocities during mutation.This technique has a drawback of loss of valuable data in some dimensions. It also has an overhead of data re-computation from scratch in case of any malfunction due to the hierarchical structure. The other major concern is outdated data, which results in diversity loss. To handle it, Li and Yang [7] devised a new dynamic optimization technique. It conducts a detailed search by dividing search space into sub-swarms. But, it faces the "*two-step forward, one-step backward*" phenomenon, as weakness in one dimension affects the overall fitness of a particle. Another dynamic technique proposed by Daniel and Xiaodong [9], Species-based Particle Swarm Optimization is effective in dealing with multimodal optimization functions in both static and dynamic environments. A networked structured PSO, called NS-PSO [8] has been proposed in which adjacent particles are connected in the neighborhood of a topological space and share the information of their best positions. These connections have been used to enhance the local search and increase diversification.

3 The CPSO Model

The CPSO is a modified version of PSO with an additional process of clustering particles. In the CPSO model, particles are periodically clustered around sample vectors using SOM to provide particles with enough diversity to prevent them from prematurely converging to local optima. The CPSO procedure has been described in Procedure 1. In Eq. 4, $\vec{V}_p(t+1)$ is the new velocity of particles, $\alpha(t)$ controls the learning rate where t is the generation number of particles and $\Phi(p,t)$ is the neighborhood function which determines the degree of the neighborhood between BMP and particle p. We took a Gaussian function as the neighborhood function for particles which denotes the lateral particle interaction and the degree of excitation of the particle. The Gaussian function which returns values between 0 and 1 is a commonly used simple model for simulating a large number of random values. Gaussian function for particles returns a value close to 1 if the particle is close

Procedure 1: Procedure CPSO

Step 1. Initialize particles

Step 2. Randomly create sample particles s in the solution space, with velocity $\vec{V}_s(t)$, where $1 \leq s \leq k$, k being the maximum number of sample vectors

Step 3. Traverse each particle p, $0 \leq p < n$, where n is the swarm size, and find the best matching particle (BMP) using the fitness value.

Step 4. Update the velocity of particles in the neighborhood of BMP by drawing them closer to sample vectors using the following formulas:

$$\vec{V}_p(t+1) = \vec{V}_p(t) + \Phi(p,t)\alpha(t)(\vec{V}_s(t) - \vec{V}_p(t)) \quad (4)$$

$$\vec{x}_p(t+1) = \vec{x}_p(t) + \vec{v}_p(t+1) \quad (5)$$

Step 5. Evolve particles using PSO.

Step 6. Go to Step 2, if $t <$ MaxGeneration and $F > \theta$ (where F is the gross increment in particle fitness on objective optimization functions, and $\theta \doteq 0$ is a very small value)

to BMP (neighbors of BMP). The neighborhood function is defined [15] as:

$$\Phi(p,t) = \exp\left(-\frac{(p-b)^2}{2\alpha(t)^2}\right) \quad (6)$$

where:

- p is the current particle
- b is the best matching particle (BMP)
- t is the Time/Generation
- $\alpha(t) \in (0 \ldots 1)$ is the learning rate

Diversity is required during early phases of optimization when local optima are computed, but as generations exhaust, the learning rate should decrease to allow convergence at global optima. Initially, the learning rate will be close to 1 and gradually it will decrease. The number of neighbors is reduced as the generation number grows. From Step 1 through Step 4, the learning process chooses the best particle from each sample vector based on fitness value and clusters the particles around the vectors by updating the positions of particles in the neighborhood. This process gives a chance to explore a new possible solution space that may contain an optimal or near-optimal solution. In Step 5, each cluster of particles is evolved by recomputing the sample vectors based on the updated particle distribution. These processes of clustering and evolving particles are iterated until the generation number is exhausted or a stopping condition that identifies no more improvements in fitness on objective functions is met. Figure 1 shows particles (clear circles) moving toward three randomly generated sample vectors (dark circles) to form clusters.

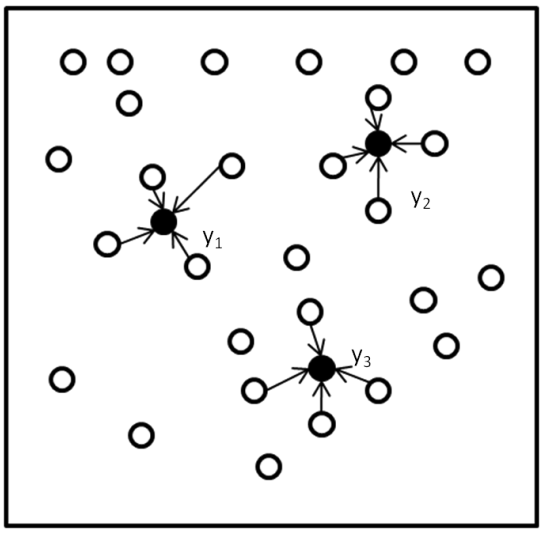

Figure 1: Clustering particles using self-organizing maps.

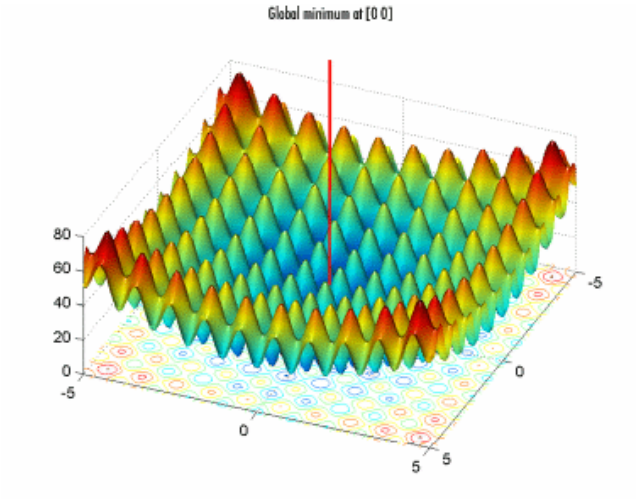

Figure 2: Rastrigin's function

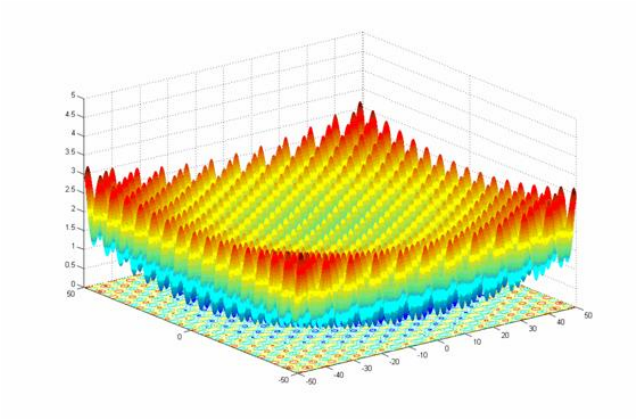

Figure 3: Griewank function

4 Simulations and Results

To run the tests, we implemented PSO and CPSO using the Java programming language. It is very cumbersome to conduct simulations on high-dimensional functions due to large computational overhead. For instance, if we choose a mixed range of dimension functions, it is very difficult to compare their results. The SOM uses vector projection to convert complex particle swarm topology into low-dimensional vectors. So it is economical to run simulations on a set of low-dimensional functions with different cluster sizes and generations as input. In the simulations, PSO and CPSO were run for four well-known objective functions namely DeJong's F2, Schaffer F6, Rastrigin, and Griewank. These are the same functions that were used by Kennedy [3]. The optimization performances of PSO and CPSO were compared. The corresponding objective functions have been described as follows:

$$f(x,y) = 100(x^2 - y)^2 + (1 - x)^2 \tag{7}$$

where $-2.048 < x, y < 2.048$

$$f(x,y) = 0.5 + \frac{(sin^2(\sqrt{x^2 - y^2} - 0.5)}{(1 + 0.001(x^2 + y^2)^2} \tag{8}$$

where $-100 \leq x, y \leq 100$

$$f(x) = 100 + \sum_{i=1}^{10} x_i^2 - 10cos(2\pi x_i) \tag{9}$$

$$f(x) = \sum_{i=1}^{100} \frac{x_i^2}{4000} - \prod_{i=1}^{100} cos(\frac{x_i}{\sqrt{i}}) + 1 \tag{10}$$

where $-600 \leq x_i \leq 600$

De Jong's F2 function, represented by Eq. (7), is a two-dimensional function with a deep valley with the shape of a parabola. The Schaffer F6 function represented by Eq. (8) is known to be very difficult to optimize, having infinite local minima and one global minimum at $(x,y) = (0,0)$. Rastrigin represented by Eq. (9) and Griewank represented by Eq. (10) are multimodal functions that have many local minima. Figures. 2 and 3 show the Rastrigin's function and the Griewank's function, respectively, which have many local minima shown by the "valleys." Both have the global minimum at $(0,0,\ldots,0)$. Figures. 4-7 show minimum fitness values found by particles exploring the solution space under the objective functions described above. Figure 4 shows the results of PSO and CPSO on the F2 function. For the F2 function, both CPSO and PSO perform well, early finding a minimum. Both converge early to a minimum, but CPSO continues to search for a better solution (which in this case does not exist). In Figure 5, it is shown that PSO prematurely converges to a solution, whereas CPSO escapes several local optima to reach global optima. Again, Figure 6 shows that for the Rastrigin's optimization problem (9), CPSO enables particles to find a global solution, oscillating between local minima and global minimum, whereas

particles of PSO converge at a local minimum. Finally, Figure 7 shows CPSO outperforms PSO dramatically for a very complex multimodal optimization problem. PSO converges early at local minima, whereas CPSO quickly finds a global minimum. The results clearly demonstrate that our approach is very effective for highly complex multimodal optimization problems.

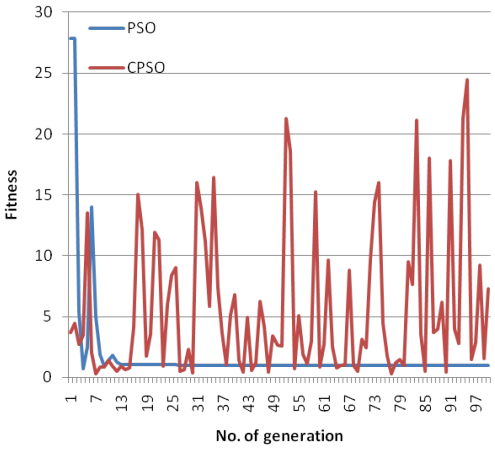

Figure 4: CPSO vs. PSO for De Jong's F2

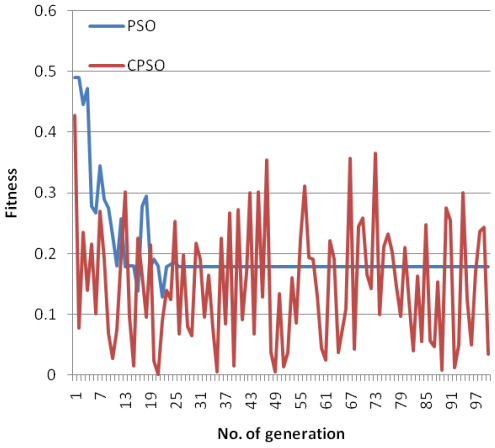

Figure 5: CPSO vs. PSO for Schaffer F6

5 Conclusions

We addressed the problem of premature convergence observed in PSO. In this research, we focused on providing enough diversity for particles to escape local minima. CPSO explicitly clusters particles around sample vectors to enable particles to escape local minima, and explore new possible solution space which may contain better solutions. Simulation results show that CPSO outperforms PSO significantly for complex optimization problems and avoids local minima yielding global solutions. Although CPSO makes an extensive

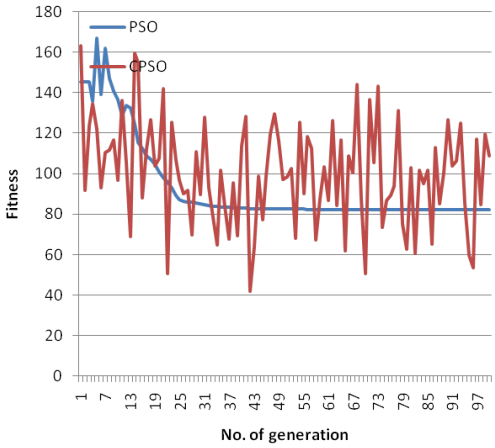

Figure 6: CPSO vs. PSO for Rastrigin F1

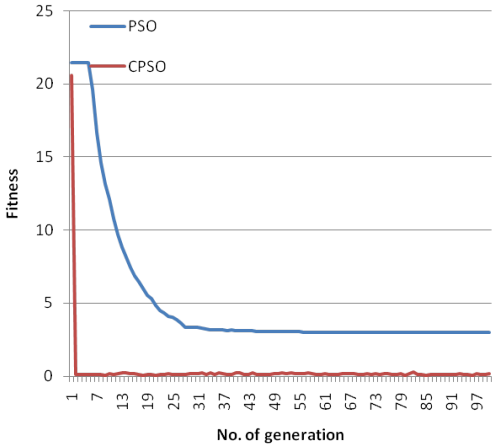

Figure 7: CPSO vs. PSO for Griewank

search within the solution space, it limits the search time by limiting the particles which explore the solution space using self-organizing maps. The research strongly suggests that CPSO is very effective for complex multimodal problems. In future work, we will study finding early signs of premature convergence for preventing such premature convergence.

Acknowledgments

We wish to thank the anonymous reviewers for their feedback which made the paper much better.

References

[1] B. Al-kazemi and C. K. Mohan, "Training Feedforward Neural Networks using Multi-phase Particle Swarm Optimization," *Proceedings of the 9th International Conference on Neural Information Processing*, 5:2615-2619, Nov. 2002.

[2] S. Baskar and P.N. Suganthan, "A Novel Concurrent Particle Swarm Optimization," *Proceedings of the Congress on Evolutionary Computation*, 1:792-796, June 2004.

[3] J. Kennedy, "Stereotyping: Improving Particle Swarm Performance With Cluster Analysis," *Proceedings of the 2000 Congress on Evolutionary Computation*, 2:1507-1512, 2000.

[4] J. Kennedy and R. Eberhart, "Particle Swarm Optimization," *Proceedings of IEEE International Conference on Neural Networks*, 4:1942-1948, Nov-Dec 1995.

[5] T. Kohonen, "The Self-Organizing Map," *Neurocomputing*, 2(2):1-6, Nov. 1993.

[6] T. Krink, J.S. Vesterstrom and J. Riget, "Particle swarm Optimization with spatial particle extension," *Proceedings of the Congress on Evolutionary Computation*, 2:1474-1479, 2002.

[7] C. Li and S. Yang, "A Clustering Particle Swarm Optimizer For Dynamic Optimization," *IEEE Congress on Evolutionary Computation*, pp. 439-446, May 2009.

[8] H. Matshushita, Y. Nishio and C.K. Tse, "Network-Structured Particle Swarm Optimizer that Considers Neighborhood Distances and Behaviors," *Journal of Signal Processing*, 18(6):291-302, Nov. 2014.

[9] D. Parrott and X. Li, "Locating And Tracking Multiple Dynamic Optima By A Particle Swarm Model Using Speciation," *IEEE Transactions on Evolutionary Computation*, 10(4):440-458, Aug. 2006.

[10] A. Ratnaweera, S. K. Halgamuge and H. C. Watson, "Self-Organizing Hierarchical Particle Swarm Optimizer With Time-Varying Acceleration Coefficients," *IEEE Transactions on Evolutionary Computation*, 8(3):240-255, June 2004.

[11] W. T. Reeves, "Particle Systems–a Technique for Modeling a Class of Fuzzy Objects," *ACM Trans. Graph.*, 2(2):91-108, 1983.

[12] J. Riget and J.S. Vesterstrom, *A Diversity-Guided Particle Swarm Optimizer - the ARPSO*, Tech Report - Dept. of Computer Science, University of Aarhus, Denmark, 2002.

[13] J.-H. Seo, C.-H. Im, C.-G. Heo, J.-K. Kim, H.-K. Jung and C.-G. Lee, "Multimodal Function Optimization Based On Particle Swarm Optimization," *IEEE Transactions on Magnetics*, 42(4):1095-1098, 2006.

[14] M. Sha and H. Yang, "Speaker Recognition Based on APSO-K-means Clustering Algorithm," *IEEE International Conference on Artificial Intelligence and Computational Intelligence*, 2:440-444, Nov. 2009.

[15] A. Sharma and C. W. Omlin, "Performance Comparison of Particle Swarm Optimization with Traditional Clustering Algorithms used in Self-Organizing Map," *World Academy of Science, Engineering and Technology*, 3(3):718-729, Mar. 2009.

[16] Y. Shi and R. Eberhart, "A Modified Particle Swarm Optimizer," *Proceedings of IEEE International Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence*, pp. 69-73, May 1998.

[17] K. Veeramachaneni, T. Peram, C.K. Mohan and L. A. Osadciw, "Optimization using Particle Swarms with Near Neighbor Interactions," *Lecture Notes in Computer Science - from Proceedings of the Genetic and Evolutionary Computation Conference*, 2723:110-121, 2003.

[18] F. Wang and Y. Qiu, "A Modified Particle Swarm Optimizer With Roulette Selection Operator," *Proceedings of IEEE International Conference on Natural Language Processing and Knowledge Engineering*, pp. 765-768, Oct. 2005.

[19] J. Wei, L. Guangbin and L. Dong, "Elite Particle Swarm Optimization with Mutation," *Asia Simulation Conference - Proceedings of the 7th IEEE International Conference on System Simulation and Scientific Computing*, pp. 800-803, Oct. 2008.

Yong-Won Park received the bachelor's degree in Computer Science from Kwangwoon University, Seoul, Korea and the master's and Ph.D. degrees in Computer Science and Software Engineering from Auburn University. He is a senior researcher in Samsung S1, Seoul. His research interests are in machine learning and data mining.

Harika Kilari received the B.Tech. (Honours) degree in Computer Software Engineering from IIIT, Hyderabad and the MS degree in Computer Science from Auburn University. She has worked as a Systems Software Engineer in NVIDIA, India and is currently a Senior Camera Engineer in Qualcomm, San Diego, CA.

Sanjeev Baskiyar is a Full Professor in the Department of Computer Science and Software Engineering at Auburn University, Auburn, Alabama. He received the Ph.D. and MSEE degrees from the University of Minnesota, Minneapolis, and the B.E. degree in Electronics and Communications from the Indian Institute of Science, Bangalore. His current research interests are in the areas of computer architecture, edge computing, optimization, scheduling, and real-time and embedded computing. In addition to academic appointments, he also worked in the hardware and software industry.

Journal Submission

The International Journal of Computers and Their Applications is published four times a year with the purpose of providing a forum for state-of-the-art developments and research in the theory and design of computers, as well as current innovative activities in the applications of computers. In contrast to other journals, this journal focuses on emerging computer technologies with emphasis on the applicability to real world problems. Current areas of particular interest include, but are not limited to: architecture, networks, intelligent systems, parallel and distributed computing, software and information engineering, and computer applications (e.g., engineering, medicine, business, education, etc.). All papers are subject to peer review before selection.

A. Procedure for Submission of a Technical Paper for Consideration

1. Email your manuscript to the Editor-in-Chief, Dr. Wenying Feng. Email: wfeng@trentu.ca.
2. Illustrations should be high quality (originals unnecessary).
3. Enclose a separate page (or include in the email message) the preferred author and address for correspondence. Also, please include email, telephone, and fax information should further contact be needed.
4. **Note**: Papers shorter than 10 pages long will be returned.

B. Manuscript Style:

1. **WORD DOCUMENT**: The text should be **double-spaced** (12 point or larger), **single column** and **single-sided** on 8.5 X 11 inch pages. Or it can be single spaced double column.

   LaTex DOCUMENT: The text is to be a double column (10 point font) in pdf format.
2. An informative abstract of 100-250 words should be provided.
3. At least 5 keywords following the abstract describing the paper topics.
4. References (alphabetized by first author) should appear at the end of the paper, as follows: author(s), first initials followed by last name, title in quotation marks, periodical, volume, inclusive page numbers, month and year.
5. The figures are to be integrated in the text after referenced in the text.

C. Submission of Accepted Manuscripts

1. The final complete paper (with abstract, figures, tables, and keywords) satisfying Section B above in **MS Word format** should be submitted to the Editor-in-Chief. If one wished to use LaTex, please see the corresponding LaTex template.
2. The submission may be on a CD/DVD or as an email attachment(s). **The following electronic files should be included:**
   - Paper text (required).
   - Bios (required for each author).
   - Author Photos are to be integrated into the text.
   - Figures, Tables, and Illustrations. These should be integrated into the paper text file.
3. Reminder: The authors photos and short bios should be integrated into the text at the end of the paper. All figures, tables, and illustrations should be integrated into the text after being mentioned in the text.
4. The final paper should be submitted in (a) pdf AND (b) either Word or LaTex. For those authors using LaTex, please follow the guidelines and template.
5. Authors are asked to sign an ISCA copyright form (http://www.isca-hq.org/j-copyright.htm), indicating that they are transferring the copyright to ISCA or declaring the work to be government-sponsored work in the public domain. Also, letters of permission for inclusion of non-original materials are required.

Publication Charges

After a manuscript has been accepted for publication, the contact author will be invoiced a publication charge of **$500.00 USD** to cover part of the cost of publication. For ISCA members, publication charges are **$400.00 USD** publication charges are required.

Revised 2020