



**COMPARING BICLUSTERING ALGORITHMS USING DATA
ENVELOPMENT ANALYSIS TO CHOOSE THE BEST PARAMETERS**

**A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
GAZİ UNIVERSITY**

BY

Ammar HOMAIDA

**IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
STATISTICS**

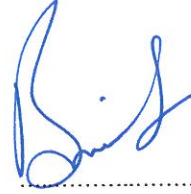
JANUARY 2019

The thesis study titled "COMPARING BICLUSTERING ALGORITHMS USING DATA ENVELOPMENT ANALYSIS TO CHOOSE THE BEST PARAMETERS" is submitted by Ammar HOMAIDA in partial fulfillment of the requirements for the degree of Master of Science in the Department of Statistics, Gazi University by the following committee.

Supervisor: Assoc. Prof. Dr. Bülent ALTUNKAYNAK

Department of Statistics, Gazi University


I certify that this thesis is a graduate thesis in terms of quality and content.



Chairman: Prof. Dr. Özgür YENİAY

Department of Statistics, Hacettepe University

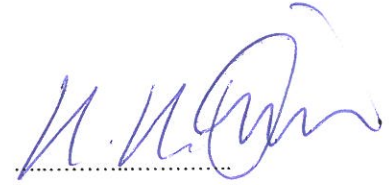
I certify that this thesis is a graduate thesis in terms of quality and content.



Member: Assoc. Prof. Dr. H. Hasan ÖRKÜ

Department of Statistics, Gazi University

I certify that this thesis is a graduate thesis in terms of quality and content.



Date: 11/01/2019

I certify that this thesis, accepted by the committee, meets the requirements for being a Master of Science Thesis.

.....

Prof. Dr. Sena YAŞYERLİ

Dean of Graduate School of Natural and Applied Sciences

ETHICAL STATEMENT

I hereby declare that in this thesis study I prepared in accordance with thesis writing rules of Gazi University Graduate School of Natural and Applied Sciences;

- All data, information and documents presented in this thesis have been obtained within the scope of academic rules and ethical conduct,
- All information, documents, assessments and results have been presented in accordance with scientific ethical conduct and moral rules,
- All material used in this thesis that are not original to this work have been fully cited and referenced,
- No change has been made in the data used,
- The work presented in this thesis is original,

or else, I admit all loss of rights to be incurred against me.



Ammar HOMAIDA

11/01/2019

COMPARING BICLUSTERING ALGORITHMS USING DATA ENVELOPMENT
ANALYSIS TO CHOOSE THE BEST PARAMETERS

(M. Sc. Thesis)

Ammar HOMAIDA

GAZİ UNIVERSITY

GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

January 2019

ABSTRACT

Biclustering method is one of the most important methods of the data mining techniques. Biclustering can be used to discover similar patterns in datasets especially gene expression datasets or any datasets that can be presented as a matrix. Starting with Block clustering algorithm in 1972 until now a good number of the biclustering algorithms have been introduced. Each one of these algorithms was proposed to discover specific things in the data. In addition, each one of the introduced algorithms has some features differ from other algorithms. So far, we can say that there is no clear user manual to help on choosing the best algorithms. Another problem is to choose the parameters for each algorithm. Many works have been done aiming to compare the biclustering algorithms according to some evaluating measures and no appropriate effort has been made to determine how to choose the best parameters under certain conditions. In this work, a two-stage comparison study is introduced. In the first stage, data envelopment analysis (DEA) is used to choose the best parameters for each algorithm according to some measures. In the second stage, using the results of the first stage some of the introduced algorithms were compared according to the size and some different variance measures.

Science Code : 20514

Keywords : Biclustering, Clustering, Gene Expression Data, Data Envelopment Analysis.

Number of pages : 138

Supervisor : Assoc. Prof. Dr. Bülent ALTUNKAYNAK

İKİLİ KÜMELEME ALGORİTMALARININ KARŞILAŞTIRILMASINDA VE PARAMETRELERİNİN SEÇİMİNDE VERİ ZARFLAMA ANALİZİNİN KULLANIMI

(Yüksek Lisans Tezi)

Ammar HOMAIDA

GAZİ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

Ocak 2019

ÖZET

İkili kümeleme yöntemi, veri madenciliğinde en önemli yöntemlerden biridir. İkili kümeleme, veri kümelerinde, özellikle gen ifade veri kümelerinde veya matrisler olarak sunulabilecek herhangi bir veri kümesinde benzer kalıpları keşfetmek için kullanılabilir. 1972'de Blok Kümeleme Algoritması ile başlayarak bugüne kadar çok sayıda algoritma tanıtılmıştır. Bu algoritmaların her biri verideki belli yapıları keşfetmek üzere önerilmiştir. Buna ilaveten, tanıtılan algoritmaların özellikleri birbirlerinden farklıdır. Şimdiye kadar, en iyi algoritmayı seçmek için yardımcı olacak bir kılavuzun olmadığı söylenebilir. Diğer bir problem, her algoritma için en uygun parametrelerin seçilmesidir. Bazı değerlendirme ölçütlerine göre çok aşamalı algoritmaların karşılaştırılmasını amaçlayan birçok çalışma yapılmıştır. Ancak, belirli koşullar altında, algoritmaların en iyi parametrelerinin nasıl seçileceğini belirlemek için çalışma yapılmamıştır. Bu çalışmada, ikili kümeleme algoritmalarının karşılaştırılması için iki aşamalı bir yaklaşım önerilmiştir. Birinci aşama, her bir algoritma için en iyi parametreleri seçmek amacıyla Veri Zarflama Analizinin (DEA) kullanılmasıdır. İkinci aşama, ilk aşamadan elde edilen en iyi parametrelere sahip algoritmaların boyut ve homojenlik ölçümlerine göre karşılaştırılmasıdır.

Bilim Kodu : 20514

Anahtar Kelimeler : İkili Kümeleme, Kümelenme, Gen Açıklama Verisi, Veri Zarflama Analizi.

Sayfa Adedi : 138

Danışman : Doç. Dr. Bülent ALTUNKAYNAK

ACKNOWLEDGEMENTS

This thesis is the result of an intensive work of nearly two-years. There are many people whom I owe thanks for helping me reach this stage. First of all, I would like to thank and my supervisor Assoc. Prof. Bülent ALTUNKAYNAK for all of his patience and the great ideas and guidance, which helped me finish my thesis.

Even if the distance between us is so far, I also want to thank my parents, my brothers, my sisters and my wife for the non-stopping support, which was the most important support made me finish this work.

I would like to thank the Presidency for Turks Abroad and Related Communities and Turkey for their hospitality and their support during the study period, which was the cornerstone of the completion of the Master's degree.

Finally, I dedicate this work to my family, my relatives, my Syrian teachers from my first day in the school to the last day in my university life at the University of Aleppo and all of my friends from Syria, Turkey and all of the other countries who I was lucky to know them.

I dedicate this work to my homeland with love Syria,

Left our hopes back at home,
Couldn't pick up the broken pieces,
Or shed tears, that we haven't already.
Left our dreams back home,
However, there is going back.

TABLE OF CONTENT

	Page
ABSTRACT.....	iv
ÖZET.....	v
ACKNOWLEDGEMENTS.....	vi
TABLE OF CONTENT.....	vii
TABLE OF TABLES.....	ix
TABLE OF FIGURES.....	x
LIST OF ABBREVIATIONS.....	xiii
1. INTRODUCTION.....	1
2. BICLUSTERING.....	3
2.1. Previous Works.....	5
2.2. Types of Biclusters.....	11
2.3. Biclustere Structure.....	13
2.4. Search Approaches.....	15
3. BICLUSTERING ALGORITHMS.....	19
3.1. Block Clustering.....	19
3.2. δ -Biclustere (CC) Algorithm.....	24
3.3. FLOC Algorithm.....	30
3.4. The Plaid Model Algorithm.....	36
3.5. Coupled Two-Way Clustering Algorithm (CTWC).....	42
3.6. Interrelated Two Way Clustering (ITWC) Algorithm.....	45
3.7. δ -pCluster Algorithm.....	50
3.8. SAMBA Biclustere Algorithm.....	55

	Page
3.9. xMotif Algorithm.....	58
3.10. ROBA Algorithm.....	61
3.11. Bimax Algorithm.....	68
3.12. RMSBE Algorithm.....	71
3.13. QUBIC Algorithm.....	75
3.14. CPB Algorithm.....	80
4. COMPARISON OF THE BICLUSTERING ALGORITHMS.....	87
4.1. First Stage: Applying Data Envelopment Analysis.....	87
4.1.1. Data envelopment analysis (DEA).....	87
4.1.2. Results.....	91
4.2. Second Stage: The Quality Comparison.....	99
4.3. Discussion and Recommendations.....	119
REFERENCES.....	127
APPENDICES.....	133
APPENDIX-1. Installing and loading packages in R software.....	134
APPENDIX-2. Using the biclustering libraries in R software.....	135
APPENDIX-3. Chia and Karuturi function and Coherence measures in R software..	136
CURRICULUM VITE.....	137

LIST OF TABLES

Table	Page
Table 2.1. Gene expression data matrix.....	3
Table 2.2. Classifications for some of the biclustering Algorithms according to the type, structure and search approach	17
Table 4.1. DEA results for CC algorithm	92
Table 4.2. DEA results for FLOC algorithm	93
Table 4.3. DEA results for Plaid Model algorithm.....	95
Table 4.4. DEA results for xMotif algorithm	96
Table 4.5. DEA results for Bimax algorithm.....	97
Table 4.6. DEA results for Qubic algorithm.....	99
Table 4.7. Test of normality.....	107
Table 4.8. The number of accepted values for the detected values	108
Table 4.9. Multiple comparisons table for the number of rows.....	108
Table 4.10. Multiple comparisons table for the number of columns	110
Table 4.11. Multiple comparisons table for the biclusters sizes	111
Table 4.12. Multiple comparisons table for the constant-variance values.....	113
Table 4.13. Multiple comparisons table for the additive-variance values	114
Table 4.14. Multiple comparisons table for the multiplicative-variance values.....	116
Table 4.15. Multiple comparisons table for the sign-variance values	118
Table 4.16. The results of the two-stage comparison study.....	124

LIST OF FIGURES

Figure	Page
Figure 2.1. The main differences between using clustering (left) and biclustering (right) methods.....	4
Figure 2.2. Examples of different types of biclusters	11
Figure 2.3. Bicluster structure.....	14
Figure 3.1. Direct clustering models.....	20
Figure 3.2. Scheme of splitting algorithm	21
Figure 3.3. Direct Clustering Algorithm.....	24
Figure 3.4. Algorithm 0 (Brute-Force Deletion and Addition).....	26
Figure 3.5. Algorithm 1 (Single Node Deletion)	27
Figure 3.6. Algorithm 2 (Multiple Node Deletion).....	28
Figure 3.7. Algorithm 3 (Node Addition).....	29
Figure 3.8. Algorithm 4 (Finding a Given Number of Biclusters)	30
Figure 3.9. Two Examples of Missing values in biclusters	31
Figure 3.10. FLOC algorithm	33
Figure 3.11. FLOC Algorithm	36
Figure 3.12. A fitted model for yeast data	37
Figure 3.13. Plaid Model Algorithm.....	42
Figure 3.14. CTWC Algorithm.....	44
Figure 3.15. The structure of ITWC algorithm.....	47
Figure 3.16. Clustering results combination when $k=2$	48
Figure 3.17. Pairwise Clustering Algorithm	52
Figure 3.18. An example of generating gene-pair MDSs	53
Figure 3.19. An example of the prefix tree used in the pClusteringmethod.....	53

Figure	Page
Figure 3.20. pClusters Algorithm	54
Figure 3.21. SAMBA model.....	55
Figure 3.22. MaxBoundBiCligue(U,V,E,d).....	57
Figure 3.23. SAMBA(U,V,E,w,d,N ₁ ,N ₂ ,k)	58
Figure 3.24. Find Motif.....	61
Figure 3.25. Algorithm 1	63
Figure 3.26. ROBA Algorithm: finding biclusters with constant values.....	65
Figure 3.27. ROBA Algorithm: finding biclusters with constant values on Columns ..	66
Figure 3.28. ROBA Algorithm: finding biclusters with constant values on Rows	67
Figure 3.29. An example of Bimax algorithm	69
Figure 3.30. Bimax Algorithm.....	70
Figure 3.31. The MSB algorithm.....	73
Figure 3.32. Additive MSBE Algorithm.....	75
Figure 3.33. Overview of CPB algorithm.....	80
Figure 3.34. CPB Algorithm.....	82
Figure 3.35. Relationship between PCC and RMSE on random vectors.....	84
Figure 4.1. Heatmap figure for the original dataset	100
Figure 4.2. Heatmap figures for CC algorithm	100
Figure 4.3. Heatmap figures for FLOC algorithm	101
Figure 4.4. Heatmap figures for Plaid Model algorithm.....	102
Figure 4.5. Heatmap figures for xMotif algorithm	103
Figure 4.6. Heatmap figures for Bimax algorithm:.....	103
Figure 4.7. Heatmap figures for Qubic algorithm.....	104
Figure 4.8. The sizes of the detected biclusters	105

Figure	Page
Figure 4.9. The boxplot figures for the different variances types.....	106
Figure 4.10. Line chart for the mean ranks of the number of rows for each algorithm..	109
Figure 4.11. Line chart for the mean ranks of the number of columns for each algorithm.....	110
Figure 4.12. Line chart for the mean ranks of the sizes of the detected biclusters for each algorithm	112
Figure 4.13. Line chart for the mean ranks of the constant-variance values for each algorithm.....	113
Figure 4.14. Line chart for the mean ranks of the additive-variance values for each algorithm.....	115
Figure 4.15. Line chart for the mean ranks of the multiplicative-variance values for each algorithm	116
Figure 4.16. Line chart for the mean ranks of the sign-variance values for each algorithm.....	118

LIST OF ACRONYMS/ABBREVIATIONS

The symbols and abbreviations used in this study are given below with their explanations.

Symbols	Explanation
---------	-------------

α	Alpha
β	Beta
δ	Delta
ε	Epsilon
η	Eta
θ	Theta
λ	Lambda
μ	Mu
χ	Chi
π	Pi
ρ	Rho
σ	Sigma
ϕ	Phi
ξ	X

Abbreviations	Explanation
---------------	-------------

ANOVA	Analysis of Variance
BCC	Banker, Charnes, and Cooper
CC	Cheng and Church
CCR	Charnes, Cooper and Rhodes
CPB	Correlated Pattern Biclusters
CS	Correlation Score
CTWC	Coupled Two-way Clustering
DC	Direct Clustering
DEA	Data Envelopment Analysis

Abbreviations	Explanation
df	Degrees of Freedom
DMUs	Decision Making Units
FLOC	Flexible Overlapping biClustering
ITWC	Interrelated Two Way Clustering
MDS	Maximum Dimension Set
MSB	Maximum Similarity Bi-Cluster
MSR	Mean Squared Residue
OS	Overlap Score
PCC	Pearson Correlation Coefficient
Qubic	QUalitative BIClustering
RMSBE	Randomized Maximum Similarity Bi-Cluster algorithm
RMSE	Root Mean Squared Error
ROBA	Robust Biclustering Algorithm
SAMBA	Statistical-Algorithmic Method for Bicluster Analysis
Sig.	Significance Probability
SSQ	Sum of the Squares

1. INTRODUCTION

Every moment the amount of data in databases is exploding. Data collecting process is everywhere like point-of-sale transactions, communications companies, social media and search engines etc.... To make these huge and wide spread data sets meaningful, some techniques can be used according to what we need to answer; discovering or predicting for example.

Some methods can be used to analyze such datasets. The most important methods to analyze big datasets are the data mining techniques and statistical methods. Data mining techniques may be preferable to be used with big datasets instead of statistics in most of the cases, because the fact of data mining techniques uses statistics, artificial intelligence and machine learning techniques to obtain the information from the datasets. In addition, data mining methods do not require most of the assumptions, which are very important to have the ability of using the statistical methods. Formally, data mining or knowledge discovery techniques are used to discover patterns in large datasets, which includes some methods like Classification, Clustering, and some other methods [1].

There are various types of datasets according to the study or work filed. Some well-known examples of studies fields where there are big datasets are Market Basket Analysis, Education, Customer Segmentation and Future Healthcare etc... [2-4]. One of the most used and known datasets are the Gene Expression Datasets. These datasets are presented as a matrix where each element in this matrix is representing an expression value for one gene under a specific experimental condition. In these datasets, the researchers are interested in discovering patterns for different genes under the experimental conditions, which can be used to study diseases especially cancers, tumors and genetic diseases [5-7].

Clustering method is known as one of the most important methods in the data mining techniques, and can be used with the gene expression data to discover patterns. However, clustering can be applied to one dimension of a data matrix, which gives us global results. Local results mean that some of the genes in the data may have a specific pattern under some experimental conditions. Clustering methods will include all of the conditions in the detected gene clusters [8].

To overcome this problem and some other problems, which may happen if the clustering method is being used, in 1974, a generalized clustering method was introduced to be used instead of clustering method and opened the door for introducing many other algorithms.

This method is known as Biclustering or Block Clustering, which applies the clustering method on the two dimensions, simultaneously [2].

A good number of the biclustering algorithms have been introduced until now. However, until now the biggest problems for using these algorithms are: (a) Which one of these algorithms is better to be used; and (b) which parameters are the best to use for different datasets types. Many works have been introduced in the literature to compare these algorithms [3, 6, 9-22].

In this thesis, a two-stage comparison study is introduced to compare some of the well-known biclustering algorithms. In the first stage of the comparison study, Data Envelopment Analysis (DEA) [23, 24] will be used to make an order (rank) for the performances of each algorithm with different parameters according to some conditions. In the second stage, the ensemble method [3] will be used to compare the performance of some of the best performances algorithms, which were ranked by the DEA method according to some criteria. The chosen variable in the DEA stage and the chosen criteria in the second stage were chosen just to present an example of this two-stage comparison method.

This thesis consists of three main chapters: In the first chapter, the biclustering method concept is reviewed in details and how its algorithms can be classified. In addition, the first chapter includes some of the most important previous works in the biclustering field. In the second chapter, a big number of the biclustering algorithms were discussed in a detailed theoretical manner. Finally, the third chapter, which consists of two sections contains a detailed view of the two-stage comparison method using one of the most widely used databases with the biclustering algorithms in a big number in the previous studies.

2. BICLUSTERING

The widespread availability of information technology has made it possible to inflate the volume of information proactively that history has not seen before, making the issue of large data controversial, in terms of the usefulness of its existence in this random way. This has led to an increasing need for the development of powerful tools for data analysis and extraction of information and knowledge. Traditional and statistical methods cannot deal with this huge quantity, so advanced methods are used to process this data. Data mining techniques have emerged as a technique to extract knowledge from vast amounts of data, based on mathematical algorithms that are the basis of data mining and are derived from many disciplines such as statistics, mathematics, logic, learning science, artificial intelligence, expert systems, machine science and other sciences, which are considered as intelligent and non-traditional sciences.

One of the most important databases are the gene expression data. These data is being generated by measuring the expression level of a large number of genes under different experimental samples or environmental conditions. Gene expression data as presented in Table 2.1 can be ordered in a data matrix. In the following table every row presents one gene and every column is presenting a specific condition[25].

Table 2.1. Source [25]: Gene expression data matrix

	Condition ₁	Condition _j	Condition _m
Gene ₁	a_{11}	a_{1j}	a_{1m}
.....
Gene _i	a_{i1}	a_{ij}	a_{im}
.....
Gene _n	a_{n1}	a_{nj}	a_{nm}

Clustering methods have been used to deal with gene expression data. The use of clustering methods may give good results with uncomplicated structural data. Even with gene expression data, good results can be obtained. Clustering algorithms aim to partition objects into clusters to maximize within-cluster similarity, based on a similarity measure. Sometimes in gene expression data, we may be interested in grouping genes under specific conditions instead of grouping them according to overall conditions. However, clustering cannot do this

mission because of the idea that clustering methods cannot do clustering on the rows and columns at the same time[26].

To overcome this problem, new algorithms were developed and are being known as biclustering algorithms. Biclustering, block clustering, co-clustering, or two-mode clustering is a data mining technique, which allows simultaneous clustering of the rows and columns of a matrix. The term was first introduced by Boris Mirkin[27] to name a technique introduced many years earlier, in 1972, by J. A. Hartigan[2]. Cheng and Church [28] were the first to apply biclustering to gene expression datasets. Although biclustering methods have been developed for gene expression data, these algorithms can be applied to all other fields of science whose data can be ordered in a data matrix.

The main differences between the clustering and the biclustering methods as presented in Figure 2.1, can be summarized as follows[8]:

- While clustering can be applied to either the rows or the columns of the data matrix separately, biclustering performs clustering to the two dimensions simultaneously.
- Clustering produces clusters of rows or clusters of columns. However, biclustering seeks blocks of rows and columns that are interrelated.
- The clusters, which are obtained using clustering methods, are exhaustive, but in the case of biclustering methods, it is not necessary to be exclusive and/or exhaustive.

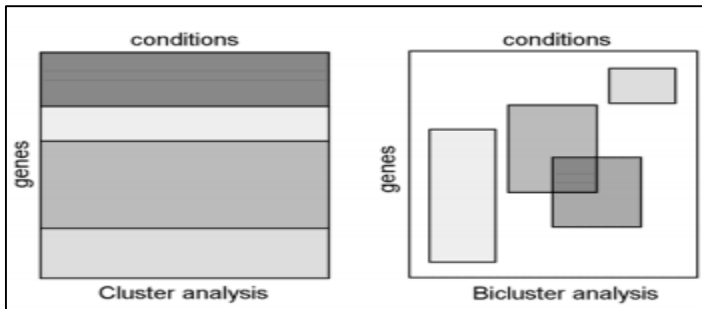


Figure 2.1. Source [9]: The main differences between using clustering (left) and biclustering (right) methods.

2.1. Definition: Let A be a data matrix with n rows and m columns as being defined by its sets of rows, $X = \{x_1, \dots, x_n\}$ and its sets of columns, $Y = \{y_1, \dots, y_m\}$. The data matrix A is denoted as (X, Y) . $A_{IJ} = (I, J)$ is defined as submatrix that contains only the objects

a_{ij} corresponding to the rows I and columns J , such that $i \in I$ and $j \in J$, where $I \subseteq X$ and $J \subseteq Y$. A bicluster is a submatrix $A_{IJ} = (I, J)$, where $I = \{i_1, \dots, i_k\} \subseteq X$, and $J = \{j_1, \dots, j_s\} \subseteq Y$ [29].

From the previous definition, the following can be obtained[29]:

- The mean of the i^{th} row in the bicluster (I, J) is given by

$$a_{iJ} = \frac{1}{|J|} \sum_{j \in J} a_{ij} \quad (2.1)$$

- The mean of the j^{th} column in the bicluster (I, J) is given by:

$$a_{IJ} = \frac{1}{|I|} \sum_{i \in I} a_{ij} \quad (2.2)$$

- The mean of all elements in the bicluster (I, J) is given by:

$$a_{IJ} = \frac{1}{|I| |J|} \sum_{i \in I, j \in J} a_{ij} = \frac{1}{|I|} \sum_{i \in I} a_{iJ} = \frac{1}{|J|} \sum_{j \in J} a_{IJ} \quad (2.3)$$

2.1. Previous Works

One of the earliest approaches for biclustering data is the Block Clustering algorithm, which was introduced by J. A. Hartigan [2] in 1972. Block Clustering or Direct Clustering algorithm is the first known try for the biclustering algorithms to overcome the limitations of the traditional clustering methods. Block algorithm was not applied to a gene expression dataset. It was applied to the UN vote results to find some patterns in the data. However, when we come to talk about biclustering, the gene expression datasets also comes. Most of the biclustering algorithms were introduced to analyze the gene expression datasets. CC algorithm, which was introduced by Cheng and Church [28], is a greedy iterative algorithm and is the first known biclustering algorithm, which dealt with the Gene Expression data and opened the door for introducing many other biclustering algorithms.

A large number of the biclustering algorithms were introduced to analyze the gene expression datasets or any dataset can be presented as a matrix. However, algorithms have different ways of work and different results types and structures and some other things.

Many works were introduced to compare the biclustering algorithms. However, in some of the studies, the comparison was done using simulated datasets and comparing the results according to some measures like size or variance. In addition, until now the step of choosing the best biclustering algorithm to use according to the data type is not an easy job.

One of the most important works is the one, which was introduced by Madeira and Oliveira [9]. In their work, they analyzed a large number of the existed algorithms. In addition, they classified these algorithms according to the type of biclusters, which can be detected, and the number and the structure of the biclusters in the datasets. Finally, they also made another classification for the methods, which are used to perform the search process to detect the biclusters.

Another important work was introduced by Prelic et al. [10]. In this work, they introduced a good known biclustering algorithm, which is known as Bimax algorithm. In addition, they evaluate the performance of some well-known biclustering algorithms with the Bimax algorithm and a hierarchical clustering method using simulated datasets. In this work, they showed some of the advantages of using the biclustering methods instead of clustering method. In addition, they showed that there are noticeable differences between the compared algorithms and how much the reference algorithm (Bimax) is able to deliver relevant patterns within all considered settings.

Vicente R. S. [6] focused in their work on compiling the biclustering algorithms and the studying the methods of bicluster visualization. The importance of studying visualization methods comes from being an easy way to interpret the results. In this work, they proposed a metric applicable to the computation of the best parameter setting for a biclustering algorithm the first step towards biclustering benchmarking. In addition, they introduced a new visualization method. With this technique, the biclustering results are being represented making emphasis in conveying the special properties of biclusters.

Tchagang et al. [11] reviewed in their work some of the most-known biclustering algorithms. In addition, they made a review of some of the important biological evaluation methods and some other methods.

Chia and Karuturi [30], introduced a differential co-expression framework and a differential co-expression scoring function. These frameworks and functions can be used to quantify the quality or the goodness of detected biclusters. That will be done based on the observation that genes in a bicluster are co-expressed in the conditions belonged to the bicluster and not co-expressed in the other conditions. In addition, they introduced a scoring function, which can be used to classify the detected biclusters into three types: T-type co-expression (strong gene only effects), B-type co-expression (strong condition only effects) and μ -type co-expression (strong gene as well as strong condition effects).

Another one of the best works the work done by Kaiser S. [3]. In his work, he made a review contains some of the well-known biclustering algorithms. In addition, he introduced an ensemble method in the biclustering technique by using different parameters for the same biclustering algorithm and combine the results so the output will be overlapped biclusters. Then, he made a review of some data types. Finally, they discussed the software, which can be used for biclustering methods like R and other programs.

Pontes et al. [13] analyzed a big number of the most used quality measures for biclusters. In addition, they did a comparative study of the quality measures. The comparison was based on the capability to recognize different expression patterns in biclusters. Pontes et al. [31] presented a survey of biclustering method also in another paper. In addition, they made a classification for the biclustering algorithms into two group according to whether or not to use evaluation metrics within the search method.

Padilha and Campello [14], made a big comparative study between 17 algorithms using 3 synthetic datasets with five different scenarios and 2 real data sets. The results of this study like most of the presented works said, “Each algorithm achieved satisfactory results in part of the biclustering tasks in which they were investigated. The choice of the best algorithm for some application thus depends on the task at hand and the types of patterns that one wants to detect”.

There are 2 other important works for our study (which give us the idea of using DEA techniques), the two works were done by Lu C. C. [32, 33]. In these works, DEA was used to evaluate outputs according to their inputs. However, in these works, he works to evaluate some algorithms with different parameters, which is not an easy job in most of the cases.

One of the most popular and used dataset in the biclustering studies is the Yeast *Saccharomyces Cerevisiae* cell cycle expression dataset [34]. Many different results obtained using this dataset with different algorithms, some of these results and some of the results of using some simulated data are presented as follows (the results will be summarized for just some algorithms, which will be used in this study. These algorithms are CC, FLOC, Plaid model, xMotif, Bimax and Qubic algorithms [10, 15, 28, 35-37]):

The first study that can be mentioned, the work of Cheng and Church [28]. In this work, they introduced the first known biclustering algorithm, which was tested with a gene expression dataset. This dataset was the Yeast *Saccharomyces Cerevisiae* Cell Cycle expression dataset. The threshold value was set to 300 in that work. In addition, the value is used with this dataset in most of the comparison works. With this threshold value, the detected biclusters were able to cover about 91.12% of rows and 100% of the columns in the dataset, which can be considered as a good value if we are looking to cover the data.

Another study that used the yeast dataset set is the one, which was done by Yang et al. [15], who introduced an improved algorithm for the CC algorithm, which known as FLOC algorithm. In that work, as it was mentioned before the yeast dataset was used to compare the performance of CC and FLOC algorithms. The same threshold value was used for both algorithms. The comparison was based on the sizes of the detected biclusters, the run-time, and the residue values. In this comparison study, FLOC algorithm showed better performance in all of the previously mentioned measures.

Another study was done by Ayadi et al. [16] using the yeast dataset in order to compare some biclustering algorithm. Two of these algorithms were the CC and Bimax algorithms, which we are more interested in their results in our study. The comparison was based on the biological relevance of the detected biclusters. In this study, Bimax algorithm was better than CC algorithm in having more biological significant results.

In addition, another work that used yeast dataset is the one which compared some of the introduced biclustering algorithms including CC, Bimax and xMotif algorithms [10, 28, 36], which was done by Nepomuceno et al. [17]. Algorithms had been evaluated based on biologically with the percentage of biclusters enriched by any Gene Ontology Consortium [38] category at different levels of significance. By using GO, a group of genes that belong

to the detected biclusters will be instigated to see if it has any significant enrichment with respect to a specific GO term. These types of evolution method will be the better way most of the times to detect the benefits of using biclustering algorithms in this research field. xmotif algorithm was able to detect the bigger number of biclusters compared with the CC and Bimax algorithms. However, the numbers of rows that founded in the detected biclusters by CC were bigger than both xMotif and Bimax algorithms, respectively. The same thing was for the numbers of conditions (columns). However, the overall sizes of the founded biclusters were bigger with the CC algorithm then Bimax algorithm, and in the end the xMotif algorithm. For the biological evaluation, the order of the performance was the best with the CC algorithm then Bimax, and the last with nearly no statistically significant results with the xMotif algorithm.

Other datasets were used also to compare the performance of the biclustering algorithms like the work of Oghabian et al. [18], which used data from the GeneSapiens database [39]. In this study, 13 biclustering algorithms and 2 clustering algorithms were compared. CC, FLOC, Qubic and Bimax algorithms were included in this study. The evaluation was based on the sample-based and gene-based. Sample-based evolution depending on assessing the set of samples included in generated biclusters for each algorithm. In other words, it shows how is the bicluster method is good to distinguish between different types of samples using some measures. In addition, the gene-based benchmarks estimate the quality of the biclusters by assessing the genes included in them using some measures. The results showed that for the biclustering algorithms that we are interested in the Plaid Model algorithm had the best performance. FLOC was able to give some good results also. Bimax almost did not have any good results. FLOC and Bimax algorithms required long run-time to obtain the results while the Qubic algorithm was so fast compared with the other algorithms.

Pontes et al. [19] also compared 5 biclustering algorithms including CC, Bimax and xMotif algorithms using yeast dataset and 3 other datasets. The comparison was based on the number of the detected biclusters, numbers of rows and columns in the detected biclusters and the mean row variance. In the yeast data, CC and Bimax have been used. CC algorithm was able to detect more biclusters than the Bimax algorithm. The numbers of rows and columns in the detected biclusters using Bimax algorithm were bigger. The mean average variance was bigger with the CC algorithm. In the second dataset, both of CC and xMotif algorithm were used. xMotif algorithm was able to detect more biclusters than the CC

algorithm with bigger number of rows and less numbers of columns within the detected biclusters. The mean row variance was bigger with the xMotif algorithm.

Yin and Liu [20] also made a comparison study between some of the well-known biclustering algorithms including CC, FLOC, and Bimax using yeast dataset and 2 other datasets. The comparison was based on numbers of rows and columns in the detected biclusters, the mean square residue (MSR), virtual error (VE), the average correlation value (ACV), average Spearman's Rho (ASR) and the time. FLOC algorithm was able to detect biclusters with bigger numbers of rows while Bimax detected smaller numbers of rows. For the numbers of the columns, CC showed the best performance while Bimax were able to detect biclusters with a smaller numbers of columns. The biclusters that detected by FLOC algorithm have the biggest MSR and VE values while Bimax was the best with smaller values. For ACV and ASR, CC algorithm had the best performance in this study while the FLOC algorithm was the worst. (The above-presented results was summarized and had been cut off because we are interested in some algorithms not all in our study).

Previous studies have not limited the process of comparing algorithms with just real data. In many of them, the comparison was applied using simulated datasets. Gu and Liu [21] compared some of the biclustering algorithms, which includes CC and Plaid Model algorithm. The evaluation was made based on the sensitivity, specificity, overlapping rate and the number of the detected biclusters. Sensitivity has values range between 0 and 1. When sensitivity value goes higher, that indicates that more true members of the biclusters have been identified by the used algorithm. Specificity also has the same range of values, and when it goes higher that indicates more background data points are excluded from the biclusters. Overlapping rate with 0 value means that there is no overlap between the detected biclusters. In addition, the maximum value is 1. The simulated data was generated using the plaid model. In the results, CC and Plaid Model algorithms have the same value of sensitivity equal to one. The plaid model had 1 as a value for the specificity while CC had a value equal to 0. The overlapping rate in CC algorithm was 0.02 and for the Plaid Model algorithm, it was equal to 0. CC algorithm was able to detect 10 biclusters while Plaid Model algorithm was able to detect just 1 bicluster.

Another study was introduced by Eren et al. [22]. In this study, they used 20 generated datasets. The datasets were generated with the following model: constant, constant-

upregulated, shift, scale, shift-scale and plaid models. Different models because each algorithm has its own way to detect different types of biclusters. 12 algorithms were used in this study including Bimax, CC, Plaid Model, Qubic and xMotif algorithms. For the constant biclusters, xMotif was the best at detecting them, and then the CC algorithm comes in the second place. The plaid Model algorithm was the best in case of constant upregulated. xMotif also had a good performance with this type. The plaid biclusters were detected in a good way comparing with the other algorithms by the Plaid model algorithm. CC algorithm was the best for the scale type. For the shift type, the Plaid model algorithm was the best. Finally, for the shift-scale type CC and Plaid Model deal the best comparing with the performance of the other algorithms.

2.2. Types of Biclusters

Biclustering algorithms can be classified according to the type of the biclusters, which can be detected by each algorithm. According to Madeira and Oliveira [9], as presented in Figure 2.2, there are four different categories of biclusters:

1. Biclusters with constant values.
2. Biclusters with constant values on rows or columns.
3. Biclusters with coherent values.
4. Biclusters with coherent evolutions.

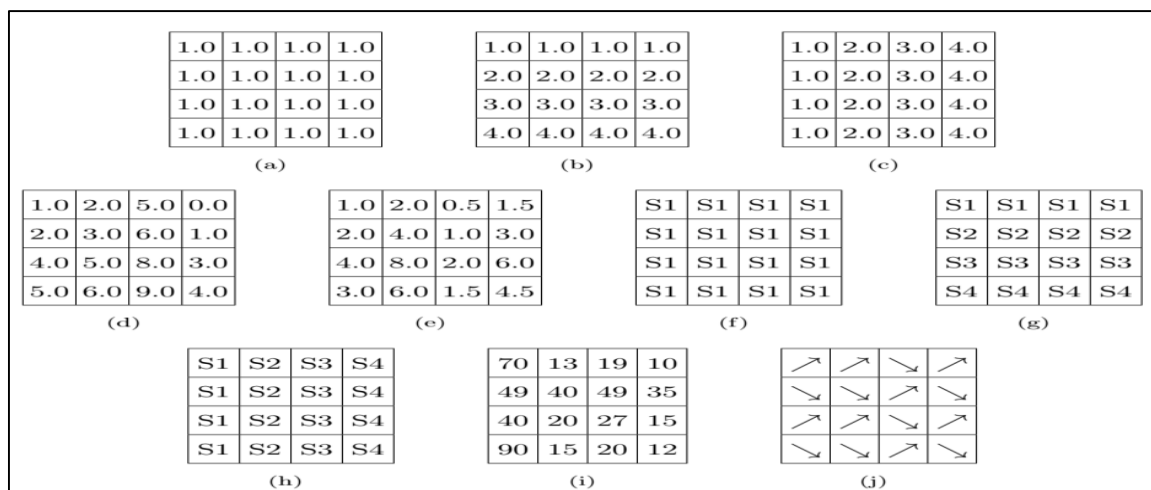


Figure 2.2. Source [9]: Examples of different types of biclusters: (a) constant bicluster, (b) constant rows, (c) constant columns, (d) coherent values (additive model), (e) coherent values (multiplicative model), (f) overall coherent evolution, (g) coherent evolution on the rows, (h) and (i) coherent evolution on the columns, and (j) coherent sign changes on rows and columns.

A constant bicluster is a submatrix (I, J) , where all the values in this submatrix have the same value. Figure 2.2(a) is an example of this type, where each value a_{ij} for all $i \in I$ and all $j \in J$ can be obtained by:

$$a_{ij} = \mu \quad (2.4)$$

where μ is a constant. This type is usually uncommon in real data because of the noise[9]. Figure 2.2(b) and Figure 2.2(c) are giving examples of biclusters with constant values on rows or columns, respectively. The first type is identifying a subset of genes (rows) with similar expression values under a subset of conditions (columns). In other words, the expression levels differ from each other by some adjustment value, α_i , associated with each row $i \in I$, so every value a_{ij} in this type can be obtained using one of the followings equations:

$$a_{ij} = \mu + \alpha_i \quad (2.5)$$

$$a_{ij} = \mu \times \alpha_i \quad (2.6)$$

where μ is the common value in the bicluster. This adjustment can be either an additive (equation 2.5) or multiplicative (equation 2.6) model. In the same way, a bicluster with constant columns, which is presented in Figure 2.2(c), is a subset of conditions (columns) with similar expression values under a subset of genes (rows) have similar expression values, and every value a_{ij} can be obtained by using one of the following equations:

$$a_{ij} = \mu + \beta_j \quad (2.7)$$

$$a_{ij} = \mu \times \beta_j \quad (2.8)$$

where μ is the typical value within the bicluster and β_j is the adjustment value for column $j \in J$.

Another approach proposing that interest may be focused on finding biclusters with coherent values on both rows and columns. The researcher may be interested in identifying biclusters

where a subset of genes and a subset of conditions have coherent values on both rows and columns in case of gene expression data. This type is illustrated in Figure 2.2(d) and Figure 2.2(e). In this type, each value a_{ij} in the bicluster can be obtained using either an additive model (equation 2.9) or a multiplicative model (equation 2.10):

$$a_{ij} = \mu + \alpha_i + \beta_j \quad (2.9)$$

$$a_{ij} = \mu \times \alpha'_i \times \beta'_j \quad (2.10)$$

where μ is the typical value within the bicluster; α_i and α'_i are the adjustment values for a row $i \in I$ in the additive model and the multiplicative model, respectively; while β_j and β'_j are the adjustment values for the column $j \in J$ in the additive model and the multiplicative model, respectively.

While some of the biclustering algorithms aim to discover biclusters with coherent values in the other hand, some of them address the problem of finding biclusters with coherent evolutions values across the rows and/or columns of the data matrix regardless of their exact value. Examples of this type are in Figure 2.2(f) to Figure 2.2(j). These coherent evolutions are described by a subset of rows and/or columns where the values within it change in the same direction. In the case of gene expression data, we may be want to find evidence that a subset of genes is up-regulated or down-regulated across a subset of conditions without focusing on their actual expression values in the data matrix. Figure 2.2(f) is giving an example of bicluster with coherent evolutions property on the both of rows and columns. While Figure 2.2(g) is presenting an example of bicluster with coherent evolutions on the rows, and Figure 2.2(h) and Figure 2.2(i) are examples of biclusters with coherent evolutions on the columns.

2.3. Bicluster Structure

Madeira and Oliveira [9], also made another classification to help in choosing the best biclustering algorithm according to the data structure. This classification assumed one of the following situations: either there is only one bicluster in the data matrix as presented in

Figure 2.3(a) or there are K biclusters in the data matrix. According to these approaches, the following structures can be obtained (Figure 2.3):

1. Exclusive row and column biclusters.
2. Non-Overlapping biclusters with a checkerboard structure.
3. Exclusive-rows biclusters.
4. Exclusive-columns biclusters.
5. Non-Overlapping biclusters with tree structure.
6. Non-Overlapping non-exclusive biclusters.
7. Overlapping biclusters with hierarchical structure.
8. Arbitrarily positioned overlapping biclusters.

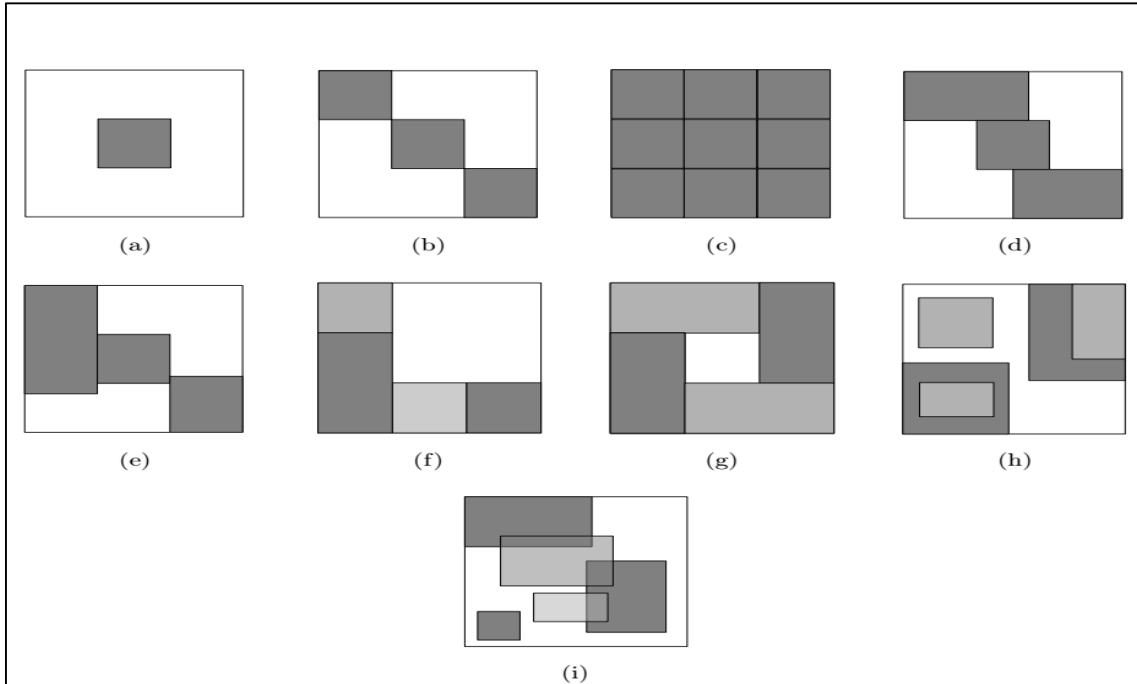


Figure 2.3. Source [9]: Bicluster structure: (a) Single bicluster, (b) Exclusive row and column biclusters, (c) Checkerboard structure, (d) Exclusive rows biclusters, (e) Exclusive columns biclusters, (f) Non-overlapping biclusters with tree structure, (g) Non-overlapping nonexclusive biclusters (h) Overlapping biclusters with a hierarchical structure, and (i) Arbitrarily positioned overlapping biclusters.

Madeira and Oliveira[9], as they had mentioned: to achieve the goal of identifying several biclusters in a data matrix A the natural starting point to create a color image for the data according to their values a_{ij} . After making the color image, the data reordered in special

ways according to the similarity between rows and the columns then there are one of the following structures:

Figure 2.3(b) is an example of exclusive row and column biclusters. In this structure, every row and every column in the data matrix must belong just to one of the K biclusters, which exist in the data matrix. On the other hand, some biclustering algorithms assumes that every row and column belong to the K biclusters in a data matrix as illustrated in Figure 2.3(c). This structure is known as non-overlapping biclusters with a checkerboard structure.

Other biclustering algorithms assume that row can belong to one bicluster, while columns can belong to more than one bicluster as presented in Figure 2.3(d), which known as exclusive-rows biclusters. In addition, other biclustering algorithms are working on the opposite approaches, which allow to the rows to be in more than one bicluster while columns belong just to one bicluster. This structure is known as exclusive-columns biclusters, which is presented in Figure 2.3(e).

The previous structures assume that every row and every column in the data matrix belongs at least to one bicluster. Figure 2.3(f) is presenting non-overlapping biclusters with tree structure and Figure 2.3(g) is presenting non-overlapping non-exclusive biclusters also every row and every column belongs at least to one bicluster.

The previous structures assume non-overlapping between biclusters. However, in real data, some rows and columns do not belong to any bicluster and some biclusters overlap in some places. Figure 2.3(h) is giving an example of overlapping biclusters with a hierarchical structure. In this case, either the biclusters are disjoint or one includes the other. The last structure showed in Figure 2.3(i) allows the existence of overlapping, non-exclusive and nonexhaustive biclusters positioned arbitrarily.

2.4. Search Approaches

Like traditional clustering, most of the biclustering algorithms are also heuristic in nature. After defining what biclustering type we search for, we have to define how we do it. There are many ways of how biclustering algorithms work. Following the classification of Madeira

and Oliveira [9], the biclustering algorithms may fall into one or more of the following categories:

1. Iterative Row and Column Clustering Combination: Biclustering algorithms apply clustering methods to rows and columns, separately, to get clusters. To build biclusters from the result of clustering algorithms some sort of iterative procedure are used.
2. Divide and Conquer: According to this approach, instead of working directly, several biclustering algorithms break the problem into several subproblems. The difference between the original problem and the subproblems is just the size. The result of the original problem is obtained by solving the subproblems and combining their results.
3. Greedy Iterative Search: The biclustering algorithms, which work according to this category, choose a locally optimal solution and hope that this choice will lead to a globally good solution.
4. Exhaustive Bicluster Enumeration: According to this method, the best biclusters are only possible if an exhaustive search of all the possible biclusters of the data matrix can be made. Working in this way will take a long time for having the results.
5. Distribution Parameter Identification: This method assumes that the data structure follows a statistical model. They try to fit its parameters to the data by minimizing a certain criterion through an iterative approach.

In this chapter, three different classifications were discussed, which help in choosing which biclustering algorithm is the best to be used according to the type of the biclusters that can be found in the data matrix, the number and the position of the biclusters in the data matrix, and the ways of how biclustering methods work. The following table showing some of the used biclustering methods according to the previous classifications, which will be discussed in details in the next chapter [6, 9, 22, 31, 40-42]:

Table 2.2. Classifications for some of the biclustering Algorithms according to the type, structure and search approach

Biclustering Algorithm	Type	Structure	Approach
Block Clustering[2]	Constant	Non-overlapping biclusters with a tree structure	Divide and Conquer:
δ -biclusters[28]	Coherent Values	Arbitrarily positioned overlapping biclusters	Greedy Iterative Search
FLOC[15]	Coherent Values	Arbitrarily positioned overlapping biclusters	Greedy Iterative Search
Plaid Models[35]	Coherent Values	Arbitrarily positioned overlapping biclusters	Distribution Parameter Identification
CTWC[43]	Constant Columns	Arbitrarily positioned overlapping biclusters	Iterative Row and Column Clustering Combination
ITWC[44]	Coherent Values	Exclusive rows biclusters, or Exclusive columns biclusters	Iterative Row and Column Clustering Combination
pClusters[45]	Coherent Values	Non-overlapping nonexclusive biclusters	Exhaustive Bicluster Enumeration
SAMBA[46]	Coherent Evolution	Arbitrarily positioned overlapping biclusters	Exhaustive Bicluster Enumeration
xMOTIFs[36]	Coherent Evolution	Single bicluster, or Arbitrarily positioned overlapping biclusters	Greedy Iterative Search
ROBA[47]	Coherent Evolution	Arbitrarily positioned overlapping biclusters	Matrix algebra
Bimax[10]	Coherent Evolution	Checkerboard structure	Divide and Conquer
RMSBE[48]	Coherent values	Arbitrarily positioned overlapping biclusters	Greedy Iterative Search
QUBIC[37]	Coherent Values	Non exhaustive, or Non exclusive	Distribution Parameter Identification
CPB[49]	Shift and Scale Patterns	Arbitrarily positioned overlapping biclusters	Greedy Iterative Search

3. BICLUSTERING ALGORITHMS

3.1. Block Clustering

Direct Clustering (DC), which is known also as Block Clustering, was introduced by Hartigan [2] in 1972. DC algorithm is the first known biclustering algorithms. According to Hartigan[2], using clustering methods involves the following problems:

- **Expensive computations:** In general, clustering methods depending on finding a distance matrix. This process is required in some clustering algorithms a lot of computation especially in case of having a huge number of objects. For example, if we have a number of objects $n = 1000$, $(n(n-1)/2) = 499500$ distance must be calculated to obtain the distance matrix.
- **Weighting decisions:** Decisions must be made about the relative weight to be given to each variable, which led to the problem in choosing the best distance function according to the type of the data.
- **Remoteness from data:** the results of the clustering methods will be interpreted according to the closeness in the distance. Sometimes may give us good results. However, in real data will give us meaningless information in general.

Hartigan [2] also defined three families of clusters that can be obtained from the data in his work: (a) the cluster of response values, (b) the marginal cluster of cases, and (c) the marginal cluster of variables. In the case of a cluster of response values, all responses within the cluster are equal. If responses are not comparable across variables, the model specifies that each variable in the marginal cluster is constant over the cases in the marginal case cluster. The last structure will be appropriate for different forms of data like ANOVA model. The previous structures are illustrated in Figure 3.1, respective.

		Case	Variable				
			1	2	3	4	5
A. Responses comparable over variables	1		4	4	4	4	7
	2		4	4	4	4	2
	3		4	4	4	4	1
	4		6	9	3	2	1
	5		5	4	1	4	6
	6		7	8	1	6	2
B. Responses not comparable between variables	1		4	7	A	K	1
	2		4	7	A	K	2
	3		4	7	A	K	3
	4		7	1	B	L	4
	5		8	2	C	L	7
	6		9	6	E	F	1
C. ANOVA model	1		4	5	2	8	1
	2		7	8	5	11	6
	3		11	12	9	15	1
	4		2	4	9	8	1
	5		1	7	2	6	7
	6		3	1	4	3	4

Figure 3.1. Source [2]: Direct clustering models

Hartigan [2] gave the three-tree structures name for the previous structures. According to the three tree structures, the clusters in the data are either non-overlap, or they are disjoint or one includes the other.

The complete algorithm

The direct clustering algorithm is a partition-based algorithm that allows the division of the data in submatrices (biclusters). Hartigan [2] introduced his algorithm in the following way: Let A be a data matrix. In this matrix, rows are referring to the cases and columns to the variables. Hartigan [2] used the variance to evaluate the quality of the biclusters. However, using the variance will lead to useless result because the fact of every single-row, single-column matrix is an ideal bicluster since variance will be equal to zero. Thus, he proposed that every data matrix would include a user-defended K number of biclusters.

The data matrix A will be divided into sub-matrixes of responses B_1, B_2, \dots, B_k measured by the sum of the square:

$$SSQ = \sum_{i,j} (A_{ij} - A_{ij}^*)^2 \quad (3.1)$$

where A_{ij}^* is the best data matrix closest to A_{ij} . In addition, A_{ij}^* will be constant within each cluster of the partition, and it is defined by:

$$\sum_{i,j \in B_p} (A_{ij}^* - A_{ij}) = 0, \quad p = 1, 2, \dots, k \quad (3.2)$$

Equivalently,

$$SSQ = \sum_p \sum_{i,j \in B_p} (A_{ij} - b_p)^2 \quad (3.3)$$

where b_p is the average value of A_{ij} in the cluster B_p . By computing SSQ for every partition B_1, B_2, \dots, B_p , every partition can be evaluated. However, previous presses will not be easy in the case of big datasets.

The split algorithm for this method is presented in Figure 3.2. According to Hartigan [2], there will be a partition into k blocks at the k^{th} step of the algorithm. (R_p, C_p) will refer to the number of the rows and the columns in the cluster B_p , respectively. The number of rows in R_p is r_p , and the number of columns in C_p is c_p .

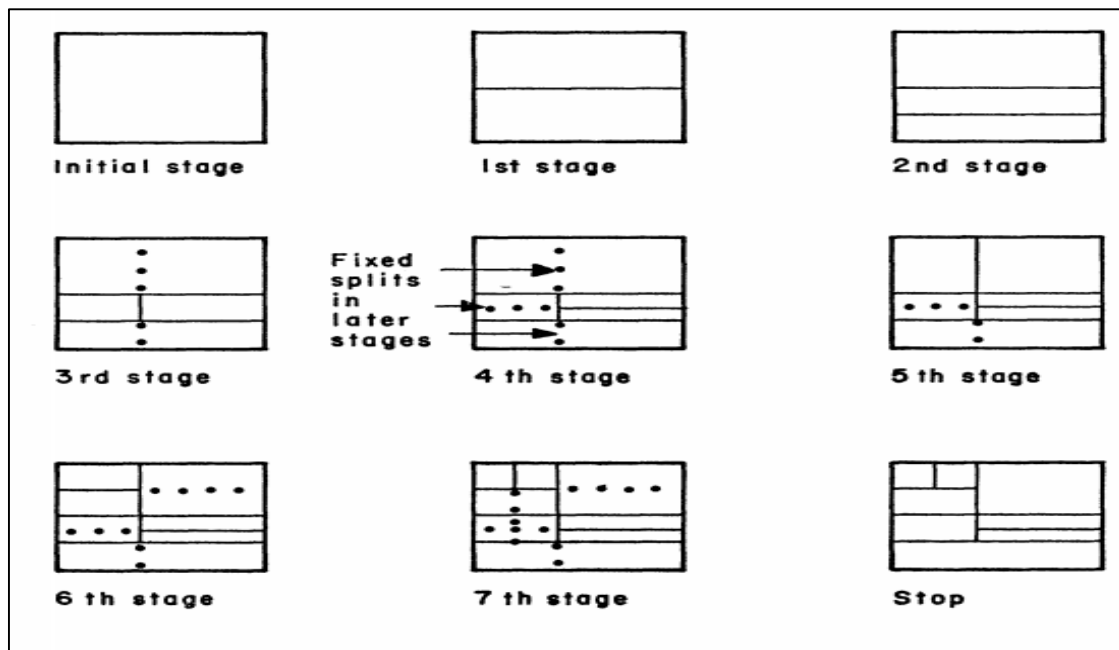


Figure 3.2. Source [2]: Scheme of splitting algorithm

The split of B_p either by rows or by columns will give two submatrices $B'_p = (R'_p, C_p)$ and $B''_p = (R''_p, C_p)$ where R'_p and R''_p is a partition of R_p . Thus, with algorithm, starting with the

partition of a single set, at the k^{th} step, the partition changes from B_1, B_2, \dots, B_k to $B_1, B_2, \dots, B_{p-1}, B'_p, B''_p, \dots, B_k$. Due to the splitting process, SSQ will be:

$$SSQR = c_p r'_p (A(B'_p) - A(B_p))^2 + c_p r''_p (A(B''_p) - A(B_p))^2 \quad (3.4)$$

where $A(B)$ denotes the average of A over the block B . The splitting process will be according to the rows or columns mean, which mean each one of the R'_p is less than the row means in R''_p . Choosing the best split row requires testing $(r_p - 1)$ divisions with the ordered data matrix by means to find the best split row B_p .

To solve the problem of splitting matrix in this method Hartigan [2] proposed of using one of the following methods: (a) free split, which will maximize the SSQ reduction overall divisions into two disjoint sets of rows, or (b) fixed split, which divides the rows into two predetermined disjoint sets in the following way:

Let A_{ij} be the values in the cluster where $i = 1, 2, \dots, m$, and $j = 1, 2, \dots, n$. In the data matrix $A_{ij} = \mu + \xi_{ij}$, where $\xi_{ij} \sim N(0,1)$ and independents. If the actual value of SSQ reduction is not high compared to the expected reduction under the null model, then the splitting will not be executed.

\bar{X}_1 will be the mean of all observations A_{ij} where $i < k$, and \bar{X}_2 is the mean of all observations A_{ij} where $i > k$, in case that fixed split that has been applied to the k^{th} row. That will give:

$$SSQ_{reduction} = (\bar{X}_1 - \bar{X}_2)^2 nk(m-k) / m \sim \sigma^2 \chi_1 \quad (3.5)$$

In case of the free split, let $X_{(1)}, X_{(2)}, \dots, X_{(m)}$ be the ordered row means, then:

$$SSQ_{reduction} = \max_k \left(\frac{X_{(1)} + \dots + X_{(k)}}{k} - \frac{X_{(k+1)} + \dots + X_{(m)}}{m-k} \right)^2 \quad (3.6)$$

where $X_{(1)}, X_{(2)}, \dots, X_{(m)}$ are order statistics from normal distribution $N(\mu, \sigma^2 / n)$. According to L. J. Savage [2], for m large the SSQ reductions for all k within $O(\sqrt{m})$ of $\frac{1}{2}m$ differ by $O(1)$. Thus, he proposed the best splitting would be near of the median. Using k as splitting point where $X_{(k)} \leq \mu \leq X_{(k+1)}$, SSQ will be obtained using the following equation:

$$SSQ_{reduction} = (\bar{X}_1 + \bar{X}_2)^2 nk(m-k) / m + O(1) \quad (3.7)$$

Then he approved that:

$$SSQ_{reduction} = 2\sigma^2 \chi^2_{m/r} + O(1) \quad (3.8)$$

Using this distribution MSQ can be used in every stage of the algorithm by using the smallest value of MSQ . For free splits of more than two rows, MSQ can be obtained by:

$$MSQ = (SSQ_{reduction})\pi / 2m \quad (3.9)$$

where m is the number of rows. For other splits, MSQ is obtained by:

$$MSQ = SSQ_{reduction} \quad (3.10)$$

DC algorithm will work in each iteration to find the best split point with the smallest MSQ . Algorithm will stop and the result will be obtained after considering all free splits of more than two rows or columns, with total sum of squares reductions $SS1 \sim \sigma^2 \chi^2_{N_1/\pi}$ (N_1 is the total number of rows or columns freely split). In addition, all other splits, with total sum of squares reduction $SS2 \sim \sigma^2 \chi^2_{N_2}$ (N_2 is the number of other splits), and considering the sum of squares within blocks, $SS3 \sim \sigma^2 \chi^2_{N_3}$ (N_3 is the total number of data points, less the number of blocks). That means algorithm stops when $(SS3 / N_3) > ((0.5SS1 + SS2) / (N_1 / \pi) + N_2)$, which means there is no other split can reduce the error.

Finally, DC algorithm practically works in the following way[9]:

<p>Input: Data matrix, k number of the biclusters and a threshold value.</p> <p>Output: k biclusters.</p> <p>Steps:</p> <ol style="list-style-type: none"> 1. Start with the entire data matrix. 2. Reorder the data matrix by row or column according to their means. 3. Find the best split row or column in the data matrix, which reduces within block-variance. 4. Repeat the step 3 alternately on the rows dimension and columns dimension. 5. Stop when we have k biclusters and no more splitting can be done to reduce the within block-variance.

Figure 3.3. Direct Clustering Algorithm

3.2. δ -Biclusters (CC) Algorithm

Cheng and Church [28] introduced the δ -Biclusters, which also commonly referred to by Cheng and Church (CC) algorithm. CC algorithm is one of the most popular biclustering algorithms, which is considered as the first biclustering algorithm that deals with gene expression data.

Cheng and Church [28] defined the bicluster as a subset of genes (rows) and a subset of conditions (columns) with a high similarity score. CC algorithm seeks to find submatrices in data that have low mean squared residue scores. CC algorithm allows biclusters to overlap so more biologically patterns could be discovered in the data.

3.1. Definition: Let X be a set of rows and Y a set of columns. The value a_{ij} of the data matrix A is the value corresponding to the i^{th} row (gene) and j^{th} column (condition). In addition, a_{ij} is taken the logarithm of the original values so multiplicative changes are represented as an additive increment. Let $I \subset X$ and $J \subset Y$ be a subset of rows and columns. The pair (I, J) specifies a submatrix A_{IJ} with the following mean squared residue score:

$$H(I, J) = \frac{1}{|I||J|} \sum_{i \in I, j \in J} (a_{ij} - a_{iJ} - a_{IJ} + a_{IJ})^2 \quad (3.11)$$

where:

$$a_{iJ} = \frac{1}{|J|} \sum_{j \in J} a_{ij} \quad (3.12)$$

$$a_{IJ} = \frac{1}{|I|} \sum_{i \in I} a_{ij} \quad (3.13)$$

$$a_{IJ} = \frac{1}{|I||J|} \sum_{i \in I, j \in J} a_{ij} = \frac{1}{|I|} \sum_{i \in I} a_{iJ} = \frac{1}{|J|} \sum_{j \in J} a_{IJ} \quad (3.14)$$

Here a_{iJ} , a_{IJ} and a_{IJ} are the row mean, column mean and the total mean in the submatrix (I, J) , respectively. According to Cheng and Church [28], the submatrix A_{IJ} is a δ -bicluster if $H(I, J) \leq \delta$ for some $\delta \geq 0$. In addition, they mentioned that if $H(I, J) = 0$, the bicluster that has been found is a constant bicluster, which indicates that the gene expression levels fluctuate in unison. The row variance may also be used to prevent obtaining trivial biclusters and reject them using the following equation:

$$V(I, J) = \frac{1}{|J|} \sum_{j \in J} (a_{ij} - a_{IJ})^2 \quad (3.15)$$

The CC algorithm can be considered as a three steps procedure, including single node deletion algorithm, multiple node deletion algorithm, and node addition algorithm. They also proved that the problem of finding the largest square δ -bicluster $(|I| = |J|)$ is NP-hard.

Node deletion algorithm

Every data matrix contains submatrices with score $H(I, J) = 0$, because every single value can be considered as a submatrix in the data matrix. However, according to Cheng and Church [28], the biclusters that should have a specific size according to the number of rows I and the columns J in the bicluster.

Node deletion algorithm starts with the whole matrix as a bicluster, then try to find a submatrix with a low H score. CC algorithm is a greedy method that aims to decrease the H score by removing rows and columns, which requires the computation of the scores of all the submatrices that may be the consequences of any row, or column removal, before each choice of removal can be made. The process that presented in Algorithm 0 (Brute-Force Deletion and Addition) will find one bicluster in $O((n+m)nm)$ where n and m the row and columns sizes in the data matrix.

Input: A , a matrix of real numbers, and $\delta \geq 0$, the maximum acceptable mean squared residue score.

Output: A_{IJ} , a δ -bicluster that is a submatrix of A with row set I and column set J , with a score no larger than δ .

Initialization: I and J are initialized to the row and column sets in the data and $A_{IJ} = A$

Iteration:

Compute the score H for each possible row/column addition/deletion and choose the action that decreases H the most. If no action will decrease H , or if $H \leq \delta$, return A_{IJ}

Figure 3.4. Algorithm 0 (Brute-Force Deletion and Addition)

Because of a long time for having one bicluster using the previous algorithm, Cheng and Church [28] proposed Algorithm 1 which known as Single Node Deletion with time complexity in $O(nm)$, as follows:

Input: A , a matrix of real numbers, and $\delta \geq 0$, the maximum acceptable mean squared residue score.

Output: A_{IJ} , a δ -bicluster that is a submatrix of A with row set I and column set J , with a score no larger than δ .

Initialization: I and J are initialized to the row and column sets in the data and $A_{IJ} = A$

Iteration:

1. Compute a_{iJ} for all $i \in I$, a_{iJ} for all $j \in J$, a_{IJ} , and $H(I, J)$. If $H(I, J) \leq \delta$, return

A_{IJ} .

2. Find the row $i \in I$ with the largest

$$d(i) = \frac{1}{|J|} \sum_{j \in J} (a_{ij} - a_{iJ} - a_{iJ} + a_{IJ})^2$$

and the column $j \in J$ with the largest

$$d(j) = \frac{1}{|I|} \sum_{i \in I} (a_{ij} - a_{iJ} - a_{iJ} + a_{IJ})^2$$

remove the row or column whichever with the larger d value by updating either I or J

Figure 3.5. Algorithm 1 (Single Node Deletion)

The data matrix has a finite number of rows and columns to remove, so the maximum number of iteration will be no more than $n + m$ where n is the number of rows and m is the number of columns. During the process of the algorithm may be all of $d(i)$ and $d(j)$ are equal to $H(I, J)$ for $i \in I$ and $j \in J$. In this case, removing one of them may decrease the score unless it is equal to zero.

Multiple node deletion

In Single Node Deletion algorithm, every time a row or column is being deleted all of the biclustering parameters have to be recomputed. This will cause large running time for big or high-dimensional data. Cheng and Church [28] proposed a Multiple Node Deletion with time complexity in $O(m \log n)$. Algorithm 2 (Multiple Node Deletion) will delete multiple rows or columns before recomputing the parameters.

Input: A , a matrix of real numbers, $\delta \geq 0$, the maximum acceptable mean squared residue score, and $\alpha > 1$, a threshold for multiple node deletion.

Output: A_{IJ} , a δ -bicluster that is submatrix of A with row set I and column set J , with a score no larger than δ .

Initialization: I and J are initialized to the row and columns sets in the data and $A_{IJ} = A$

Iteration:

1. Compute a_{iJ} for all $i \in I$, a_{IJ} for all $j \in J$, a_{IJ} , and $H(I, J)$. If $H(I, J) \leq \delta$, return A_{IJ} .
2. Remove the rows $i \in I$ with

$$\frac{1}{|J|} \sum_{j \in J} (a_{ij} - a_{iJ} - a_{IJ} + a_{IJ})^2 > \alpha H(I, J)$$
3. Recompute a_{IJ} , a_{IJ} , and $H(I, J)$.
4. Remove the columns $j \in J$ with

$$\frac{1}{|I|} \sum_{i \in I} (a_{ij} - a_{iJ} - a_{IJ} + a_{IJ})^2 > \alpha H(I, J)$$
5. If nothing has been removed in the iterate, switch to Algorithm 1 (Single Node Deletion).

Figure 3.6. Algorithm 2 (Multiple Node Deletion)

Node addition algorithm

After applying algorithm 1 or algorithm 2 by deleting rows and columns, some of the deleted nodes may gain some importance for the bicluster. Because of that, Cheng and Church [28] proposed the Node Addition Algorithm, which will work on adding rows and columns to the bicluster that obtained from the previous process without increasing the mean squared residue score H and sometimes adding some rows or columns may decrease the score too. This algorithm with time complexity in $O(nm)$.

Input: A , a matrix of real numbers, I and J signifying a δ -bicluster.

Output: I' and J' such that $I \subset I'$ and $J \subset J'$ with the property that $H(I', J') \leq H(I, J)$

Iteration:

1. Compute a_{iJ} for all i , a_{iJ} for all j , a_{IJ} , and $H(I, J)$.

2. Add the columns $j \notin J$ with

$$\frac{1}{|I|} \sum_{i \in I} (a_{ij} - a_{iJ} - a_{iJ} + a_{IJ})^2 \leq H(I, J)$$

3. Recompute a_{iJ} , a_{IJ} , and $H(I, J)$.

4. Add the rows $i \notin I$ with

$$\frac{1}{|J|} \sum_{j \in J} (a_{ij} - a_{iJ} - a_{iJ} + a_{IJ})^2 \leq H(I, J)$$

5. For each row i still not in I , add its inverse if

$$\frac{1}{|J|} \sum_{j \in J} (-a_{ij} + a_{iJ} - a_{iJ} + a_{IJ})^2 \leq H(I, J)$$

6. If nothing is added in the iterate, return the final I and J as I' and J' .

Figure 3.7. Algorithm 3 (Node Addition)

The complete algorithm

By using algorithm 1 or algorithm 2 with algorithm 3, one bicluster can be detected every time, Cheng and Church [28] proposed algorithm 4, which combine the previous algorithm so we can have n biclusters. In every iteration random numbers, using the same way that used to generate the missed values in the data matrix will be replaced using this algorithm the submatrix that represents the bicluster that founded in the data matrix. Algorithm 4 (Finding a Given Number of Biclusters) is summarized below:

Input: A , a matrix of real numbers with possible missing elements, $\alpha \geq 1$, a parameter for multiple node deletion, $\delta \geq 0$, the maximum acceptable mean squared residue score, and n , the number of δ -biclusters to be found.

Output: n δ -bicluster in A .

Initialization: Missing elements in A are replaced with random numbers from a range covering the range of non-null values. A' is a copy of A .

Iteration for n times:

1. Apply Algorithm 2 on A' , δ , and α . If the row (column) size is small, do not perform multi node deletion on rows (columns). The matrix after multiple node deletion is B .
2. (Step 5 of Algorithm 2) Apply Algorithm 1 on B and δ and the matrix after single node deletion is C .
3. Apply Algorithm 3 on A and C and the result is the bicluster D .
4. Report D , and replace the elements in A' that are also in D with random numbers.

Figure 3.8. Algorithm 4 (Finding a Given Number of Biclusters)

3.3. FLOC Algorithm

Yang et al. [15] introduced the FLOC (Flexible Overlapping biClustering) as an alternative to the Cheng and Church algorithm [28] to find δ -biclusters. FLOC algorithm has the same CC algorithm's goal. The goal is to find biclusters with low mean squared residue MSR . Yang et al. [15] showed that the use of random interference that used in CC algorithm would affect the biclustering result. They defined a δ -bicluster as a subset of rows and a subset of columns exhibiting coherent values on the specified (non-missing) value of the rows and columns considered. Thus, they proposed a generalized model for that is used in CC algorithm and proposed FLOC algorithm that can find a set of possibly overlapping biclusters simultaneously.

Yang et al. [15] proposed their algorithm in 2 phases. Before moving to the algorithm phases, they redefined some terms that proposed by Cheng and Church [28] as follows:

They renamed the biclusters, which will be obtained using the FLOC algorithm as generalized biclusters. Let $\mathfrak{I} = \{A_1, A_2, \dots, A_N\}$ be a set of columns (Conditions) and $\mathfrak{R} = \{O_1, O_2, \dots, O_M\}$ be a set of rows (genes). The data matrix D is $M \times N$ matrix of real

numbers. Every value d_{ij} corresponds to the row O_i and column A_j . A bicluster essentially corresponds to a submatrix that exhibits some coherent tendency. In other words, each bicluster in the data matrix can be uniquely identified by the set of relevant rows and columns. In addition, Yang et al. [15] mentioned in their work the number of missing entries in a bicluster should be limited to some extent to avoid trivial cases. To overcome the previous problem they proposed the using of a value α that belong to $[0,1]$ to control the amount of missing values for each row and each column in a bicluster. For example in the following Figure let $\alpha = 0.65$, so the bicluster in the left is not a valid bicluster because it has less than α amount of missing values while the one on the left is considered as a valid bicluster.

	cond 1	cond 2	cond 3	cond 4
gene 1	1		3	
gene 2		4		5
gene 3		3	4	
(a) not a valid bicluster				

	cond 1	cond 2	cond 3	cond 4
gene 1	1		3	3
gene 2	3	4		5
gene 3		3	4	4
(b) a bicluster				

Figure 3.9. Source[15]: Two Examples of Missing values in biclusters

3.2. Definition: For a given matrix $\mathcal{R} \times \mathcal{C}$ and occupancy threshold α , a bicluster (of α occupancy) can be represented by a pair (I, J) where $I \subseteq \{1, \dots, M\}$ is a subset of rows and $J \subseteq \{1, \dots, N\}$ is a subset of columns. For each row $i \in I$, $|J'_i| / |J| > \alpha$ where $|J'_i|$ and $|J|$ are the number of specified columns for a row i in the bicluster and the number of columns in the bicluster, respectively. In the same way, for each column $j \in J$, $|I'_j| / |I| > \alpha$ where $|I'_j|$ and $|I|$ are the number of specified rows under the column j in the bicluster and the number of rows in the bicluster, respectively.

3.3. Definition: The volume of a bicluster $(I, J)_{v_{IJ}}$ is defined as the number of specified entries d_{ij} such that $i \in I$ and $j \in J$.

3.4. Definition: For a given bicluster (I, J) , the base of the row O_i is defined as the average value of O_i for all specified columns in J :

$$d_{iJ} = \frac{\sum_{j \in J'_i} d_{ij}}{|J'_i|} \quad (3.16)$$

where $J'_i \subseteq J$ is the set of specified columns in J for row O_i . Similarly, the base of a column A_j is the average specified value of A_j taken by all rows in I :

$$d_{IJ} = \frac{\sum_{i \in I'_j} d_{ij}}{|I'_j|} \quad (3.17)$$

where $I'_j \subseteq I$, is the set of rows whose value is specified in the column A_j . The base of the bicluster is the average value of all specified entries of the submatrix defined by (I, J) :

$$d_{IJ} = \frac{\sum_{i \in I, j \in J} d_{ij}}{v_{IJ}} \quad (3.18)$$

where v_{IJ} is the volume of the bicluster.

Using the previous definitions Yang et al. [15] introduced the residue of an entry d_{ij} (equation 3.19), the residue of a bicluster (I, J) (equation 3.20), and the row variance (equation 3.21), as followings:

$$r_{ij} = \begin{cases} d_{ij} - d_{iJ} - d_{IJ} + d_{IJ} & , d_{ij} \text{ is specied} \\ 0 & , \text{otherwise} \end{cases} \quad (3.19)$$

$$r_{IJ} = \frac{\sum_{i \in I, j \in J} r_{ij}^2}{v_{IJ}} \quad (3.20)$$

where r_{ij} is the residue of the entry d_{ij} and v_{IJ} is the volume of the bicluster.

$$\text{var}_{I,J} = \frac{\sum_{i \in I, j \in J} (d_{ij} - d_{iJ})^2}{v_{IJ}} \quad (3.21)$$

According to Yang et al.[15] by extending the model that is used in the FLOC algorithm the user can have some extra features as followings:

1. Some researchers may require some degree of overlap while some may not require. FLOC algorithm gives us the power to control the amount of overlap between biclusters.
2. By using the FLOC algorithm, the appearance of some rows in the biclusters can be controlled.
3. Algorithm helps in obtaining balanced biclusters by controlling the ratio of the rows and columns.
4. The user can control the volume of the final biclusters.

The complete algorithm

The FLOC algorithm works in two phases as presented in Figure 3.4 described as follows:

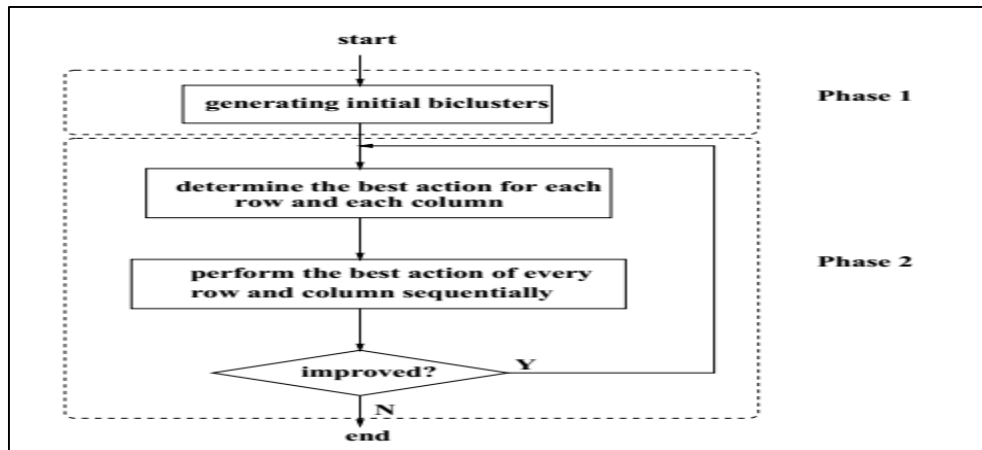


Figure 3.10. Source [15]: FLOC algorithm

The FLOC algorithm is a probabilistic move-based algorithm that can find k biclusters with low mean squared residues. The data is represented in a matrix form. The FLOC algorithm will start with a set of initial biclusters and work to improve the quality of these biclusters. In each iteration, each row and columns will be moved from bicluster to another bicluster according to their mean squared residues to improve the quality of the biclusters. FLOC

algorithm will stop when there is no rows or columns movement can be done to improve the quality of the biclusters.

As presented in Figure 3.4 algorithm has two phases. In the first phase, the initial biclusters are constructed. Every bicluster contains a set of rows and a set of columns. Let ρ be a controlling parameter that used to control the size of bicluster. Each row and column is included in the bicluster with probability ρ . The initial biclusters will contain $M \times \rho$ rows and $N \times \rho$ columns. If some biclusters have less than α a threshold percentage of specified values, then we must generate new initial biclusters until the condition is satisfied. α value can be chosen as $|D|/N \times M$ where $|D|$ is the volume of the original data matrix D that has N rows and M columns.

After the process of choosing the initial biclusters according to α threshold value, the FLOC algorithm will start its second stage: This phase will apply an iterative process to improve the quality of the biclusters. The goal of this phase is to reduce the overall mean squared residue by examination every row and column to choose the best actions and apply them. Because the goal is to detect k bicluster using this algorithm, so for every row or column there is k action can be applied.

Let x be referring to one of the rows (or columns) and let c refers to a bicluster. The action (x, c) is defined as the change of membership of x with respect to c . Thus, the action (x, c) can be either moving x from bicluster c (x is in the c bicluster) or add x to the c bicluster (x is not in the c bicluster). Because the fact that there are k actions can be made, Yang et al. [15] proposed the using of the following new concept.

3.5. Definition: Given a residue threshold r , the gain of an $Action(x, c)$ is defined as:

$$Gain(x, c) = \frac{r_c - r_{c'}}{r^2} + \frac{v_{c'} - v_c}{v_c} \quad (3.22)$$

where r_c , $r_{c'}$ are the residues of bicluster c and the bicluster c' , obtained by performing $Action(x, c)$ on c , respectively. Similarly v_c and $v_{c'}$ are the volumes of c and c' , respectively.

The goal is to find biclusters with low residue ($< r$). The Gain measurement works to increase the volume of the biclusters even if they have smaller residue than the threshold r . Another advantage of using the Gain measurement is to control the residue of the c bicluster when it has residue larger than r . Applying the action $Action(x, c)$ with positive Gain value will improve the quality of the bicluster c , and vice versa. Therefore, to obtain k biclusters using the FLOC algorithm the action with the highest gain must be applied for each row (or column). Sometimes the best action may be with negative Gain value, and applying it may improve the biclusters in later iterations.

In every iteration, the best action for each row and column is identified and performed. The best actions with higher gain scores are performed earlier until we obtain the best k biclusters from the data. Some actions may be blocked temporarily if applying it may cause violates constraints that defined by the user or may lead to obtaining trivial bicluster. The actions will be performed a random weighted order. That means the actions with higher positive gain values will be with high probability values to be executed earlier. A random weighted order will be generated for the action list using a swap process according to their probability values. The overall time complexity to obtain the final biclusters is $O((N+M)^2 \times k \times p)$

where N is the number of rows and M is the number of the columns in the data matrix, k is the number of the biclusters, and p the number of the iterations to termination. Finally, FLOC algorithm steps are summarized by Chan W. [50] as follows:

Input: ρ : biclusters size controlling parameter, r : user-defined *MSR* threshold, D : data matrix, K : number of the biclusters, and any user-defined constraints.

Output: K biclusters.

1. Initialize:
 - a. K biclusters from the data matrix D that met r value, volume, and the other conditions.
 - b. best_bicluster_volume = 0.
 - c. best_biclustering = initialized biclusters of step (1a)
 - d. curr_best_biclustering_MSR = $+\infty$
 - e. Set curr_best_biclustering to empty biclusters
2. While (improved != true)
 - a. Compute action for each element e , $1 \leq e \leq M + N$.
 - b. Put actions on the action list.
 - c. Perform weighted Random Re-arrangement on the action list.
 - d. For each action f , from beginning to end of action list,
 - i. Block f if resulting bicluster violates constraints.
 - ii. Otherwise, if the average volume of biclustering f is greater than best_biclustering_volume:
 1. curr_best_biclustering = biclustering f , if average *MSR* of biclustering f exceeds curr_best_biclustering_MSR.
 2. ensure improved is true.
 - e. if (improved = true) best_biclustering = curr_best_biclustering.
3. return best_biclustering.

Figure 3.11. FLOC Algorithm

3.4. The Plaid Model Algorithm

The plaid model algorithm is an additive biclustering method, which was proposed by Lazzeroni and Owen [35], and was improved by Turner et al. [51]. The plaid model algorithm allows a gene (row) to be in more than one bicluster or in none at all. Let Y be a data matrix with n rows and p columns. The value Y_{ij} means the value in row i and column j where the number of values in the data matrix is equal to np .

The plaid model algorithm starts with creating a color image for the data according to their values as presented in Figure 3.5, where the red color is referring to high response and the blue color for a low response obtained from yeast gene expression data for example.

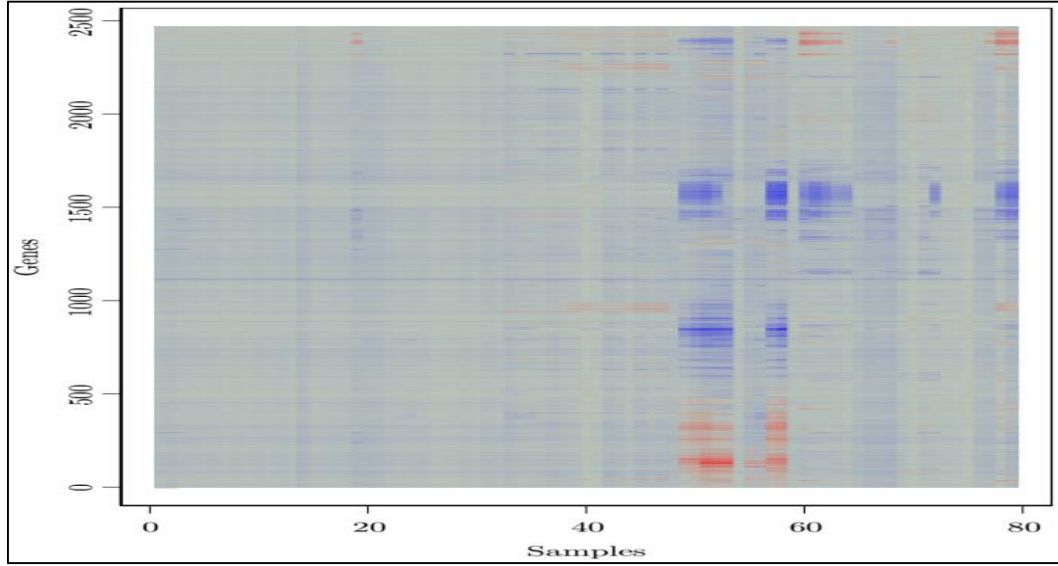


Figure 3.12. Source [35]: A fitted model for yeast data

The colored image will be ordered according to the color to make blocks with nearly the same color as Figure 3.5, which has been ordered using hierarchical clustering on the rows. If the reordering is ideal, it will produce an image with K rectangle block on the diagonal and every block would be nearly uniformly colored. In this case, the obtained blocks will be K mutually expulsive and exhaustive bicluster. Every value in the data matrix according to Lazzeroni and Owen [35] can be expressed as a linear model of the form:

$$Y_{ij} = \mu_0 + \sum_{k=1}^K \mu_k \rho_{ik} k_{jk} \quad (3.23)$$

where μ_0 is a background color (the color that is given to the values, which not belong to any one of the K , blocks), μ_k describes the color in the block k . In addition, ρ_{ik} is 1 if row i is in the k^{th} row-block and 0 if not, and k_{jk} is 1 if column j is in the k^{th} column-block and 0 if not. In addition, the conditions that make every row and every column be just in one bicluster are $\sum_k \rho_{ik} = 1$ for all i , and $\sum_k k_{jk} = 1$ for all j , respectively. However, non-overlap biclusters are rare in real data; they modified the previous conditions so overlap can

occur in some places. The modified conditions are $\sum_k \rho_{ik} \geq 2$ for some i , and $\sum_k k_{jk} \geq 2$ for some j . In addition, some rows or columns do not fit well into any bicluster, so for them, the conditions will be $\sum_k \rho_{ik} = 0$ for some i , and $\sum_k k_{jk} = 0$ for some j , respectively. Lazzeroni and Owen [35] modified the model (3.23) to generalized model, which give us to power to obtain biclusters with the identical response for set or rows (genes) under a set of columns (conditions), or a set of columns with common expression patterns for a set of rows as follows:

$$Y_{ij} = \mu_0 + \sum_{k=1}^K (\mu_k + \alpha_{ik} + \beta_{jk}) \rho_{ik} k_{jk} \quad (3.24)$$

where each $\rho_{ik} \in \{0,1\}$, and $k_{jk} \in \{0,1\}$. The last model will give the users more flexibility with the properties of the biclusters that they are looking for. The model (3.23) a layer describes a response μ_k that is shared by all rows in it for all columns in it. In model (3.24) when β_{jk} is not used that will give us biclusters with a set of rows that had an identical response to set of columns. At the same time, if β_{jk} just used in the model that means we are interested in finding biclusters that had a set of columns with common values levels for a set of rows.

The last model can be written using θ_{ijk} (θ_{ijk} describe the background layer) to represent either μ_k , or $\mu_k + \alpha_{ik}$, or $\mu_k + \beta_{jk}$, or $\mu_k + \alpha_{ik} + \beta_{jk}$ as needed as follows:

$$Y_{ij} = \sum_{k=0}^K \theta_{ijk} \rho_{ik} k_{jk} \quad (3.25)$$

This model gives one of the following situations:

1. If $\rho_{ik} = 1$ for all i , but k_{jk} is not 1 for all j , then layer describes a cluster of columns.
2. If $k_{jk} = 1$ for all j , but ρ_{ik} is not 1 for all i , then layer describes a cluster of rows.

3. If the layer for a bicluster of rows contains a term β_{jk} , then the bicluster of rows is a set of p-vectors centered near the vector $(\mu_k \beta_{1k}, \dots, \mu_k \beta_{pk})$
4. If that layer also contains a term α_{ik} then the rows bicluster along a line segment through this center.

Parameters estimation and updating

Working with this algorithm requires a long time to have the results that because for each layer k , there are $(2^n - 1)(2^p - 1)$ ways to select the participating rows and columns. To simplify the work, Lazzeroni and Owen [35] supposed that we have $K - 1$ layers, and algorithm will look for the K^{th} layer to minimize the sum of squared errors. Let:

$$Q = \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^p (Z_{ij} - \theta_{ijK} \rho_{iK} k_{jK})^2 \quad (3.26)$$

Where

$$Z_{ij} = Y_{ij} - \theta_{ij0} - \sum_{k=1}^{K-1} \theta_{ijk} \rho_{ik} k_{jk} \quad (3.27)$$

is the residue from the first $K - 1$ layers. In every iteration θ , ρ and k values are being updated where $\theta^{(s)}$, $\rho^{(s)}$ and $k^{(s)}$ will refer to the values of θ_{iK} , ρ_{iK} and k_{jK} at iterative s .

To update the values of θ_{ij} with given ρ_i and k_j the following must be minimized:

$$Q = \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^p (Z - (\mu + \alpha_i + \beta_j) \rho_i k_j)^2 \quad (3.28)$$

with respect to the following conditions:

$$\sum_{i=1}^n \rho_i^2 \alpha_i = \sum_{j=1}^p k_j^2 \beta_j = 0 \quad (3.29)$$

In addition, straightforward Lagrange multiplier arguments show that:

$$\mu = \frac{\sum_i \sum_j \rho_i k_j Z_{ij}}{(\sum_i \rho_i^2)(\sum_j k_j^2)} \quad (3.30)$$

$$\alpha_i = \frac{\sum_j (Z_{ij} - \mu \rho_i k_j) k_j}{\rho_i \sum_j k_j^2} \quad (3.31)$$

$$\beta_j = \frac{\sum_i (Z_{ij} - \mu \rho_i k_j) \rho_i}{k_j \sum_i \rho_i^2} \quad (3.32)$$

Given values for θ_{ij} and k_j , to update the values for ρ_i that minimize Q are:

$$\rho_i = \frac{\sum_j \theta_{ij} k_j Z_{ij}}{\sum_j \theta_{ij}^2 k_j^2} \quad (3.33)$$

In addition, for k_j with given θ_{ij} and ρ_i are:

$$k_j = \frac{\sum_i \theta_{ij} \rho_i Z_{ij}}{\sum_i \theta_{ij}^2 \rho_i^2} \quad (3.34)$$

Lazzeroni and Owen also make control of the values of ρ_i and k_j so they do not move quickly towards 0 or 1. Thus, at iteration s , the two values are replaced by $0.5 + s / (2S)$ if they are larger than 0.5 and by $0.5 - s / (2S)$ otherwise. According to Lazzeroni and Owen's work, the starting point with a value near 0.5 for all of the parameters.

The importance of a layer k is measured by:

$$\sigma_k^2 = \sum_{i=1}^n \sum_{j=1}^p \rho_{ik} K_{jk} \theta_{ijk}^2 \quad (3.35)$$

The distribution of σ_k^2 is unknown and the layer will be accepted if σ_k^2 is significantly larger than what would be found in the noise. Let Z_{ij} be the residue matrix for layer k . Let \tilde{Z}_{ij}^r be a residual matrix that obtained by randomly permuting every row and every column of the result for $r = 1, \dots, R$. There are $(n + p)R$ independently and uniformly distributed permutations. Let $\tilde{\sigma}_k^{2,r}$ be the importance or the size of the founded layer during the process of the algorithm in the randomized data \tilde{Z}_{ij}^r . The stopping condition is: $\sigma_k^2 > \max_{1 \leq r \leq R} \tilde{\sigma}_k^{2,r}$ and $k < K_{\max}$ add the new layer k to the model, otherwise stop where K_{\max} defines the number of layers in the model.

The complete algorithm

In the following, the pseudo code for the plaid model, which was presented by Turner et al. [51]:

1. Compute \hat{Z} , the matrix of residuals from the model so far
2. Compute starting values $\hat{\rho}_i^{(0)}$ and $\hat{k}_j^{(0)}$
3. For $s=1:S$ do
 - a. Compute $\hat{\mu}^{(s)}, \hat{\alpha}_i^{(s)}$ and $\hat{\beta}_j^{(s)}$ using *OLS* estimates with $\hat{\rho}_i^{(s-1)}$ and $k_j^{(s-1)}$
 - b. Compute $\hat{\rho}_i^{(s)}$ using *OLS* estimate with $\hat{\mu}^{(s)}, \hat{\alpha}_i^{(s)}, \hat{\beta}_j^{(s)}$ and $k_j^{(s-1)}$
 - c. Compute $k_j^{(s)}$ using *OLS* estimate with $\hat{\mu}^{(s)}, \hat{\alpha}_i^{(s)}, \hat{\beta}_j^{(s)}$ and $\hat{\rho}_i^{(s-1)}$
 - d. Shift any $\hat{\rho}_i^{(s)}$ and $k_j^{(s)}$ to $0.5 + s / (2 - (S - T))$ if greater than 0.5
 - e. Shift any $\hat{\rho}_i^{(s)}$ and $k_j^{(s)}$ to $0.5 - s(2(S - T))$ if less than 0.5
4. End for
5. Compute $\hat{\mu}^{(s+1)}, \hat{\alpha}_i^{(s+1)}$ and $\hat{\beta}_j^{(s+1)}$
6. Calculate candidate layer sum of squares $LLS_c = \sum_{i,j} (\hat{\mu} + \hat{\alpha}_i + \hat{\beta}_j) \hat{\rho}_i \hat{k}_j$
7. For $m=1:M$ do
 - a. Permute \hat{Z}
 - b. Repeat search for bicluster
 - c. Calculate layer sum of squares LSS_m
8. End for
9. If $LLS_c > \max(LSS_1, \dots, LSS_M)$ Then
 - a. Accept bicluster and search for next layer
10. Else
 - a. Stop

Figure 3.13. Plaid Model Algorithm

3.5. Coupled Two-Way Clustering Algorithm (CTWC)

Getz et al. [43, 52] introduced the Couplet Two-way Clustering algorithm, which is known as CTWC algorithm. CTWC algorithm is based on iterative clustering, which performs a search for stable and significant partitions emerge subsets of rows and subset of columns. Like most of the biclustering algorithm is originally was developed to deal with data expression data where the rows represent genes under deferent conditions that are represented by the columns.

CTWC algorithm uses clustering algorithms to find biclusters. In addition, any clustering method can be used. Getz et al. [52] by using CTWC algorithm, they were able to identify correlated biologically partitions in an unsupervised way and to find new partitions may contain important information.

Let \mathcal{A} be the data matrix where every row of this matrix corresponds to a single gene and each column represents a particular sample. The main goal is to find a pairs of subsets $(\mathcal{O}_j, \mathcal{F}_i)$ where \mathcal{F}_i could be a subset of features of rows or columns and \mathcal{O}_j a subset of objects that can be rows or columns. Algorithm will start using a clustering algorithm to identify all stable clusters of rows or columns. Testing all submatrices will be impossible in large data, so CTWC will work in iterative process by using the previous founded clusters candidates that founded in the previous iteration.

Let \mathcal{F} be the feature set, which is obtained from the values levels of the rows in each cluster. \mathcal{F} representing object set. In addition, let \mathcal{O} be object sets, which contain either all the columns or any column cluster. \mathcal{F} may can identified using columns clusters. ν^g and ν^s refer to all of the stable clusters of both rows and columns, respectively. The rows clusters are accumulated in a list V^g and the column clusters in V^s .

The complete algorithm

In every iteration in the CTWC algorithm a subset of objects (either rows or columns), using a subset of features (rows or columns) will be clustered, and if a new cluster is found they will be used in the following iteration. Algorithm will stop when there is no new information is being generated. Finally, algorithm will give us the final sets of V^g, V^s and the pointers that identify the way of how all stable clusters of row and columns were generated, as shown below:

Input: Full data matrix.

Output: A set V^g of stable row clusters and a set V^s of stable column clusters.

Algorithm:

Step 1. Initialization

- 1a. Let v_0^g be the cluster of all rows, and v_0^s be the cluster of all columns.
- 1b. Initialize sets of rows clusters, V^g , and column clusters, V^s , such that $V^g = \{v_0^g\}$ and $V^s = \{v_0^s\}$.
- 1c. Add each known class of rows as a member of V^g , and each known class of columns as a member of V^s .
- 1d. Define a new set $W = \emptyset$. This set is needed to keep track of clustering analyses that have already been performed.

Step 2. For each pair $(v^g, v^s) \in (V^g \times V^s) \setminus W$:

- 2a. Apply the clustering algorithm on the genes of v^g using the columns of v^s as its features and vice versa.
- 2b. Add all the robust row clusters generated by Step 2a to V^g , and all the robust column clusters to V^s .
- 2c. Add (v^g, v^s) to W .

Step 3.

For each new robust cluster u in either V^g or V^s define and store a pair of labels $P_u = (u_o, u_f)$. Of these, u_o is the cluster of objects which were clustered to find u , and u_f is the cluster of features used in that clustering.

Step 4.

Repeat Step 2 until no new clusters are added to either V^g or V^s .

Figure 3.14. CTWC Algorithm

According to Getz et al. [52], the output from the CTWC algorithm will provide a wide list of rows and columns clusters, and for each cluster, which subset was clustered to find it is also known. In addition, for every cluster C , which other clusters can be found C as the feature set.

The columns in the data matrix will be classified using C into two classes c_1 and c_2 . C is a known classification of samples. CTWC give a unique method to test the list of the candidate rows clusters that will be used to make the partition of the columns. This method has two steps. In the first step, for each cluster of columns v^s in V^s must be evaluated according to purity and efficiency. This evaluation will reflect the extent to which assignment of the columns to v^s correspond to the classification C . For example, the purity and efficiency for c_1 are computed by:

$$purity(v^s | c_1) = \frac{|v^s \cap c_1|}{|v^s|} \quad (3.36)$$

$$efficiency(v^s | c_1) = \frac{|v^s \cap c_1|}{|c_1|} \quad (3.37)$$

When a high purity cluster S is founded, the saved pointers is saved to read off clusters of rows that used as features set to obtain S . Algorithm will merge the small stable clusters or divide the big ones into smaller according to the user-specified parameters.

3.6. Interrelated Two Way Clustering (ITWC) Algorithm

Tang et al. [44] developed the unsupervised method Interrelated Two-Way Clustering (ITWC) method following a similar strategy to CTWC algorithm [52]. The goal of this algorithm is to identify important gene patterns and perform cluster discovery on samples.

Data will be arranged in a data matrix where in case of the gene expression data each row refers to one gene and columns represent samples. Let $L = \{g_1, \dots, g_i, \dots, g_n\}$ be the set of all genes (rows), $S = \{s_1, \dots, s_j, \dots, s_m\}$ be the set of all samples (columns), and w_{ij} be the intensity value associated with each gene g_i and sample s_j in the data matrix. Thus, the data matrix is $W = \{w_{ij} | 1 \leq i \leq n, 1 \leq j \leq m\}$ where $n \gg m$.

According to Tang et al. [44] to obtain meaning classes from columns, the vector space must be reduced into smaller one using clustering methods. That means it is better to reduce the

number of rows on the data matrix to the number of the columns then performing the clustering.

To solve this problem they proposed the ITWC algorithm, which can find a subset of genes, which are highly related to the experiment conditions, and to cluster the columns into different groups.

In gene expression data, different genes have different ranges of intensity values, which may not have an important meaning alone. Working with algorithm requires the normalization of the values in the data matrix according to the following formula, which was mentioned in Tang et al. [44] work:

$$w'_{ij} = \frac{w_{ij} - \mu_i}{\mu_i} \quad (3.38)$$

$$\mu_i = \frac{\sum_{j=1}^m w_{ij}}{m} \quad (3.39)$$

This step helps to reduce the amount of noise in the data. Each row vector after normalization is denoted by $g_i = \{w'_{i1}, w'_{i2}, \dots, w'_{im}\}$ where $i = 1, \dots, n$. To test whether a row intensity value varies much among columns vector-cosine between each row vector and a pre-defined stable pattern $E = \{e_1, e_2, \dots, e_m\}$ is used, as follows:

$$\cos(\theta) = \frac{\langle \vec{g}_i, \vec{E} \rangle}{\|\vec{g}_i\| \cdot \|\vec{E}\|} = \frac{\sum_{j=1}^m w'_{ij} \times e_j}{\sqrt{\sum_{j=1}^m w'^2_{ij}} \times \sqrt{\sum_{j=1}^m e_j^2}} \quad (3.40)$$

where θ is the angle between two vectors \vec{g}_i and \vec{E} in m-dimensional space. When the vector-cosine is close to one that means the two vector patterns are similar. After the process of computing the vector-cosine values, the user chooses a threshold that used to remove rows matching pattern E . According to Tang et al. [44] in every step twenty to thirty percent of rows can be removed with the condition of the genes vector-cosine values with E are higher

than the threshold. Thus, if the genes vector-cosine values are higher than the threshold that means these rows changes little during the experiment.

The distance measure, which will be used during the use of ITWC algorithm, must be carefully chosen. Any distance measure can be used like the Euclidean distance. However, if we are dealing with data like gene expression, Tang et al. [44] have recommended using a correlation coefficient, which measures the strength of the linear relationship between two vectors. The formula for the correlation coefficient between two vectors $X = \{x_1, x_2, \dots, x_k\}$ and $Y = \{y_1, y_2, \dots, y_k\}$ is:

$$\rho_{X,Y} = \frac{k \left(\sum_{i=1}^k x_i \times y_i \right) - \left(\sum_{i=1}^k x_i \right) \times \left(\sum_{i=1}^k y_i \right)}{\sqrt{\left[k \sum_{i=1}^k x_i^2 - \left(\sum_{i=1}^k x_i \right)^2 \right] \left[k \sum_{i=1}^k y_i^2 - \left(\sum_{i=1}^k y_i \right)^2 \right]}} \quad (3.41)$$

By using the previous formula, the relationship between row clustering and column clustering will be used to reduce the vector of columns to a reasonable level and perform class discovery.

The complete algorithm

Tang et al. [44] proposed the ITWC as shown in the following figure:

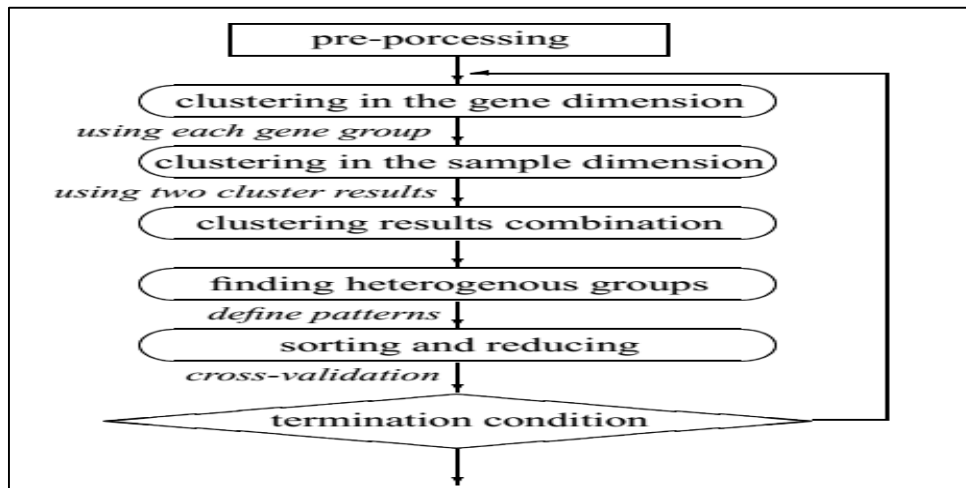


Figure 3.15. Source [44]: The structure of ITWC algorithm

The iterative procedure based on G with n_1 rows (genes) after the pre-processing. ITWC algorithm will start with clustering the row dimension, which will cluster n_1 into k groups G_1, G_2, \dots, G_k . These groups are an exclusive subset of G . It is obvious, which can give us the control to choose how many clusters we want.

In the second step, the process of clustering will be applied to the column (samples) dimension based on the results from the first step into 2 clusters $S_{i,a}$ and $S_{i,b}$. After that in the third step, the clustering results from the step 1 and step 2 will be combined, and the following four groups will be obtained by dividing the columns where $k = 2$ for example:

- C_1 : all columns clustered into $S_{1,a}$ based on G_1 and clustered into $S_{2,a}$ based on G_2 .
- C_2 : all columns clustered into $S_{1,a}$ based on G_1 and clustered into $S_{2,b}$ based on G_2 .
- C_3 : all columns clustered into $S_{1,b}$ based on G_1 and clustered into $S_{2,a}$ based on G_2 .
- C_4 : all columns clustered into $S_{1,b}$ based on G_1 and clustered into $S_{2,b}$ based on G_2 .

The previous step is presented in the following figure:

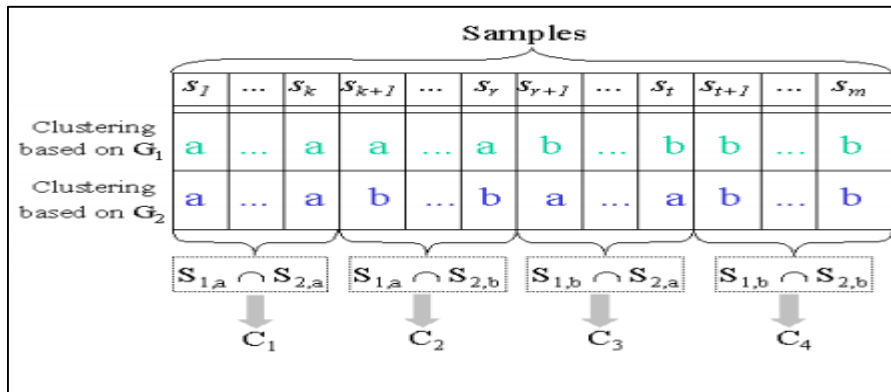


Figure 3.16. Source [44]: Clustering results combination when $k=2$. s_1, s_2, \dots, s_m represent samples and the second and third lines are the cluster results on samples based on gene groups G_1 and G_2 independently.

In the Figure 3.7, there are 4 situations, and in general, there is 2^k number of possible sample groups according to the value of 2^k . The 4th step will try to found heterogeneous groups. To do that two groups will be chosen from C_1, C_2, C_3, C_4 in case of $k = 2$, and let them be C_s and C_t where $(1 \leq s, t \leq 2^k \text{ in general})$. These two groups should satisfy the following

condition: For $\forall u \in C_s, \forall v \in C_t$, where u and v are column if $u \in S_{i,r_1}, v \in S_{i,r_2}$, then $r_1 \neq r_2$ ($r_1, r_2 \in \{a, b\}$) for all i ($1 \leq i \leq k$). When the previous condition is true (C_s, C_t) called heterogeneous group.

The last step is sorting and reducing. For each heterogeneous group, two patterns are introduced. Vector-cosine (equation 3.40) will be used to compute each pattern with each row vector. To the computations for example from the Figure 3.7, (C_1, C_4) patterns are $(0, 0, \dots, 0, 1, 1, \dots, 1)$ and $(1, 1, \dots, 1, 0, 0, \dots, 0)$. Where for example $(1, 1, \dots, 1, 0, 0, \dots, 0)$ is a number of columns in C_1 ones followed by a number of columns in C_4 zeros. After the computations are done all rows will be sorted according to the similarity values in descending order, keep the first one-third of the sorted row sequence by cutting off the other two-thirds of the row sequence. That will give us reduced row sequence G' from the remaining sorted row sequences from two patterns.

By doing the same process with (C_2, C_3) , reduced row sequence G'' will be obtained. Then to choose which one of the G' and G'' will be used in the next iteration. The cross-validation method is used to evaluate each group. In each heterogeneous group for each column in the group, the remaining columns of the group are used to select important rows, and predict the class of the withheld columns. The cumulative error rate is calculated and the heterogeneous group with a lower error rate is selected. Let \hat{G} with n_2 rows be the selected heterogeneous group.

The previous five steps will be repeated by clustering n_2 rows, and so on until the termination conditions, which will be described in the following, are satisfied.

3.6. Definition: Let χ be all heterogeneous groups. The occupancy ratio between columns in heterogeneous groups and all columns is giving by:

$$Occratio = \max \left(\left\{ \frac{|C_i| + |C_j|}{m} \right\} \right) \quad (3.42)$$

where $(C_i, C_j) \in \chi (1 \leq i, j \leq 2^k)$, m is the total number of columns, $|C_i|$ is the number of columns in C_i . If $k = 2$ then the *occratio* value will be between 0.5 and 1, and the sum of the number of columns in all heterogeneous groups is equal to m . In addition, if the row clustering based on G_1 and G_2 are the same, then either $C_1 \cup C_4 = S$ or $C_2 \cup C_3 = S$, which give us evidence that \hat{G} is good to be used for column clustering. The iterations can be stopped when *occratio* reached a user defended value. Another way to terminate the process is used when the threshold cannot be reached. The alternative termination condition is used when the remaining genes number n_2 is very small.

3.7. δ -pCluster Algorithm

Wang et al. [45] introduced δ -pCluster algorithm, which is related to mean squared residues to find biclusters in the gene expression data. They aimed in their work to identify subspace clusters in high-dimensional data sets, and to find a new similarity model that can capture the pattern similarity among the objects.

The pCluster model can capture not only the nearness of objects but also the similarity of the patterns exhibited by the objects. δ -pCluster algorithm can detect multiple overlapping clusters and can deals in a good way with the outliers in the data. To discuss the δ -pCluster algorithm first the following terms will be given [45]:

The data matrix has a set of objects \mathcal{D} (genes), where everyone is defined by a set of attributes \mathcal{A} (conditions). Algorithm will try to find objects that exhibit a coherent pattern on a subset of attributes of \mathcal{A} . They defined the pScore of 2×2 matrix as the following:

$$pScore \left(\begin{bmatrix} d_{xa} & d_{xb} \\ d_{ya} & d_{yb} \end{bmatrix} \right) = \left| (d_{xa} - d_{xb}) - (d_{ya} - d_{yb}) \right| \quad (3.43)$$

where $x, y \in \mathcal{O} \subset \mathcal{D}$ and $a, b \in \mathcal{T} \subset \mathcal{A}$ (\mathcal{T} is a subset of attributes) and d_{uv} is the expression value. The pair $(\mathcal{O}, \mathcal{T})$ is a δ -pCluster if any 2×2 submatrix X in $(\mathcal{O}, \mathcal{T})$ we have $pScore(X) \leq \delta$ for some value for $\delta \geq 0$.

Every submatrix in a bicluster that is obtained by using this algorithm is also a bicluster. This property is not founded in many biclustering algorithms, which makes δ -pCluster algorithm a powerful method to be used.

Wang et al. [45] redefined the mean squared residue for 2×2 matrix where $I = \{x, y\}$ and $J = \{a, b\}$ as follows:

$$\begin{aligned}
 H(I, J) &= \frac{1}{|I||J|} \sum_{i \in I} \sum_{j \in J} (d_{ij} - d_{Ij} - d_{iJ} + d_{IJ})^2 \\
 &= \frac{\left((d_{xa} - d_{xb}) - (d_{ya} - d_{yb}) \right)^2}{4} \\
 &= (pScore(X) / 2)^2
 \end{aligned} \tag{3.44}$$

Wang et al. [45] defined δ -bicluster as $(\delta / 2)^2 - pCluster$ for 2-objects/2-attribute matrix. However, the biclusters that obtained by using δ -pCluster is more homogeneous because the obtained biclusters require every 2 objects and every 2 attributes conform to the inequality. The volume of pCluster is the size of \mathcal{O} and the size of \mathcal{T} , which can be controlled by the user using a threshold value.

To find all pairs $(\mathcal{O}, \mathcal{T})$ where $(\mathcal{O}, \mathcal{T})$ is pCluster, the user must provide a cluster threshold δ , a minimal number of columns n_c , and n_r a minimal number of rows with $|\mathcal{O}| \geq n_r$, $|\mathcal{T}| \geq n_c$ conditions.

With the previous parameters, Algorithm will work to detect multiple clusters. Wang et al. [45] presented Pairwise Clustering, which will be used in the δ -pCluster's steps. The pairwise clustering algorithm will work to find two-object pClusters as following:

Input: X, Y : two objects, \mathcal{T} : set of columns, n_c , δ .

Output: All δ -pClusters with more than n_c .

1. $s \leftarrow d_x - d_y$; (i.e., $s_i \leftarrow d_{xi} - d_{yi}$ for each i in \mathcal{T})
2. sort array s ;
3. start $\leftarrow 0$; end $\leftarrow 1$;
4. new $\leftarrow \text{TRUE}$; (a Boolean variable, if TRUE, indicates an untested column in [start,end]
5. repeat
 - $v \leftarrow S_{end} - S_{start}$;
 - If $|v| \leq \delta$ then (expands δ -pCluster to include one more columns)
 - end $\leftarrow \text{end} + 1$;
 - new $\leftarrow \text{TRUE}$;
 - else
 - Return δ -pCluster if end-start $\geq n_c$ and new = TRUE;
 - Start $\leftarrow \text{start} + 1$;
 - New $\leftarrow \text{FALSE}$;
6. Until end $\geq |\mathcal{T}|$;
7. Return δ -pCluster if end - start $\geq n_c$ and new = TRUE;

Figure 3.17. Pairwise Clustering Algorithm

The complete algorithm

The main δ -pCluster has 3 main steps. In the first step, the data will be scanned to find column-pair *MDSs* for every column-pair, and object-pair *MDSs* for every object-pair where *MDS* concept defined as following:

3.8. Definition: Let $c = (\mathcal{O}, \mathcal{T})$ be a δ -pCluster. Column set \mathcal{T} is a Maximum Dimension Set (*MDS*) of c if there does not exist $\mathcal{T}' \supset \mathcal{T}$ such that $(\mathcal{O}, \mathcal{T}')$ is also a δ -pCluster.

The *MDSs* will be combined later to create pCluster consists of more than two genes and more than two conditions. An Example of generating gene-pair *MDS* in Figure 3.8:

condition \ gene	a	b	c	d	e
1	5	9	6	7	5
2	2	4	3	1	2
differences	3	5	3	6	3

(a)

differences	3	3	3	5	6
condition	a	c	e	b	d

(b)

Figure 3.18. Source [53]: An example of generating gene-pair MDSs: (a) the differences between gene 1 and 2; (b) the sorted differences.

For each group, the difference between the largest one and the smallest one must be less than or equal to the threshold δ . In Figure 3.8, for example, δ was taken as 2, which means for gene 1 and 2 the corresponding *MDS* is $\{\{a, c, e, b\}, \{b, d\}\}$.

In the next step, object-pair *MDSs* and column-pair *MDSs* are pruned. The process of pruning is done by counting the number \mathcal{O}_{ab} that contain $\{x, y\}$ for any dimension a in a *MDS* \mathcal{T}_{xy} for example. Then a will be removed from \mathcal{T}_{xy} if the number of such \mathcal{O}_{ab} is less than $n_c - 1$. In addition, if the removal of a makes $|\mathcal{T}_{xy}| < n_c$, \mathcal{T}_{xy} is removed also.

The final stage is known as Tree Constructing and Traversing. To generate pClusters prefix tree will be build using gene-pair *MDSs*. Each edge in the prefix tree corresponds to one condition of a gene-pair *MDS*. At the left node along one path, it records the two genes of the gene-pair *MDS*. An example of a prefix tree for two gene-pair *MDSs* like $(\{1, 2\}, \{a, b, c, d\})$ and $(\{1, 3\}, \{b, e\})$.

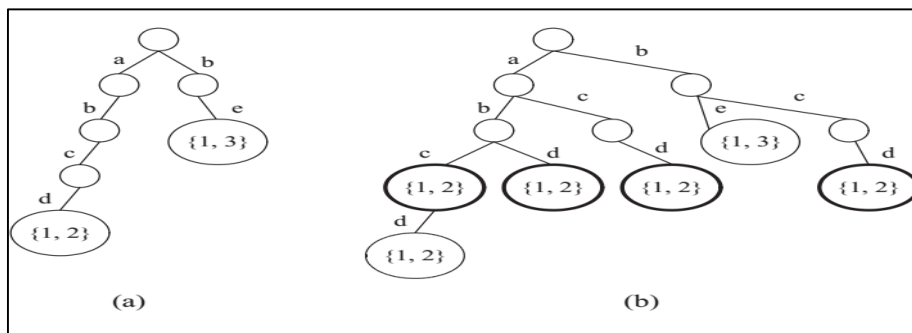


Figure 3.19. Source [53]: An example of the prefix tree used in the pClustering method: (a) the prefix tree; (b) duplicating the gene information.

Algorithm will apply a post-order traversal of the prefix tree. For each traversing node, pClusters contained within it will be detected first, and then the gene pairs at the current node will be duplicated to the nodes that represent subsets with size $(k-1)$ of the k conditions of the current node.

According to Wang et al. [45] the in the following algorithm, which describes how δ -pCluster algorithm the initial generation of *MDSs* has time complexity $O(M^2 N \log N + N^2 M \log M)$, and worst case for pruning is $O(kM^2 N^2)$ where M is the number of columns, and N is the number of objects.

Input: \mathcal{D} : data set, δ : pCluster threshold, n_c : minimal number of columns, n_r : minimal number of rows.

Output: All pClusters with size $\geq n_r \times n_c$.

- 1- For each $(a, b \in \mathcal{A}, a \neq b)$ do
 - Find column-pair *MDSs*: pairCluster(a, b, \mathcal{D}, n_d)
- 2- For each $(x, y \in \mathcal{D}, x, y)$ do
 - Find object-pair *MDSs*: pairCluster(x, y, \mathcal{A}, n_c)
- 3- Repeat
 - For each (object-pair pCluster($\{x, y\}, T$)) do
 - Use column-pair *MDSs* to prune columns in T
 - Eliminate *MDS* ($\{x, y\}, T$) if $|T| < n_c$
 - For each (column-pair pCluster($\{a, b\}, O$)) do
 - Use object-pair *MDSs* to prune objects in O
 - Eliminate *DS* ($\{a, b\}, O$) if $|O| < n_r$
- 4- Until
 - (no pruning takes place)
- 5- Insert all object-pair *MDSs* ($\{x, y\}, T$) into the prefix tree: insertTree(x, y, T)
- 6- Make a post-order traversal of the tree
- 7- For each (node n encountered in the post-order traversal) do
- 8- $O :=$ objects in node n
- 9- $T :=$ columns represented by node n
- 10- For each $(a, b \in T)$ do
 - Find column-pair *MDSs*: $C = \text{pairCluster}(a, b, O, n_r)$
 - Remove from O those objects not contained in any *MDS* $c \in C$
- 11- Output (O, T)
- 12- Add objects in n to nodes which has no less column than n

Figure 3.20. pClusters Algorithm

3.8. SAMBA Biclustering Algorithm

Tanay et al. [46] introduced a Statistical-Algorithmic Method for Bicluster Analysis, which is known as the SAMBA algorithm. They defined the biclusters as objects, which have homogeneous rows and columns. In this algorithm, they combined graph-theoretic and statistical considerations to find biclusters in the dataset.

Statistical data modeling

SAMBA algorithm models the input data as bipartite graphs (for more details see [54]). This graph has two parts the first one corresponds to the columns and the second one to the rows. Tanay et al. [46] in their work presented two statistical models of the resulting graph. An example in Figure 3.10 shows how to build the bipartite $G = (U, V, E)$ for a given dataset. U is the set of columns and V is a set of rows in the data matrix. In addition, the edge $(u, v) \in E$ indicates the response of gene v in condition u that means the value level of u has a significant change in v condition. The next step is to develop the graph so it will include the direction of change. For example, in Figure 3.10 we can see that there is effect on the genes *gal7* in the *tup1* condition while there is no effect on *ecm11*.

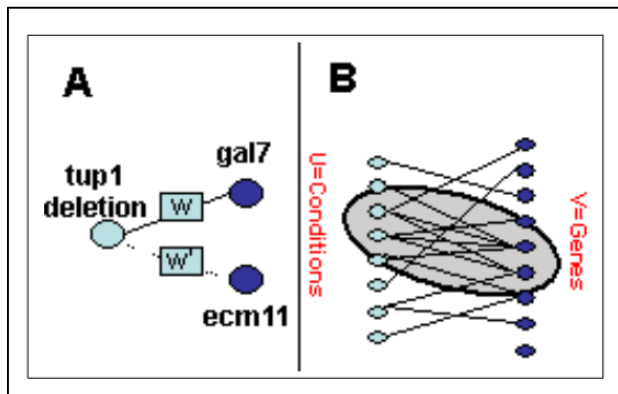


Figure 3.21. Source [46]: SAMBA model: Gene expression data is modeled using bipartite graph. An edge (u, v) indicates the response of gene v in condition u . Weights are assigned using a statistical model for the edges and non-edges of the graph. (A): part of the graph showing the condition *tup1* deletion and its effect on the genes *gal7* while there is no effect on *ecm11*. (B) a heavy subgraph representing a significant bicluster.

Thus, the bicluster is connected with the subgraph $H = (U', V', E')$ of G as presented for example in Figure 3.10(b) where the shaded area presenting a bicluster. The bipartite

subgraph a subset of rows V' , which co-regulated under a subset of columns U' . In addition, the weight that used in the bicluster is the sum of the weights of row-column pairs in it for both of the edges and non-edges.

Simple and refined models:

Tanay et al. developed [46] a simple model and a refined model for the bipartite graph to reduce the biclusters-finding problem by finding heavy subgraphs in a bipartite graph, like the followings:

The simple model assumes the occurrence of edges is independent with the same probability p where $p = |E| / (|U| |V|)$ for a given bipartite graph G . Let $H = (U', V', E')$ be a subgraph of G . In addition, $|U'| = m'$, $|V'| = n'$ and $|E'| = k'$. Thus, the probability of detecting k or more time of success in n trails has a binominal tail distribution $BT(k, n, p)$. The goal of this model is to find a subgraph H with the lowest $p(H) = BT(k', p, n', m')$.

The second model is developed to take into account the validity of the degrees in G . The model assumes that the occurrence of each edge (u, v) is an independent Bernoulli variable with $p_{u,v}$ as a parameter. Tanay et al. [46] assumed that the score of H is simply its weight by finding the results of the following:

$$\log L(H) = \sum_{(u,v) \in E'} \log \frac{p_c}{p_{u,v}} + \sum_{(u,v) \in \bar{E}'} \log \frac{1-p_c}{1-p_{u,v}} \quad (3.45)$$

where $\bar{E}' = (U' \times V') \setminus E'$ and by setting the weight of each edge (u, v) to $\log(p_c / p_{u,v}) > 0$ and the weight of each non-edge (u, v) to $\log((1-p_c) / (1-p_{u,v})) < 0$, which take into account the direction of expression change for each edge.

The complete algorithm

SAMBA algorithm has 3 phases to find high-quality biclusters: In the first phase, a bipartite graph is formed, then using one of two models that described above we calculate vertex pair

weights. The row is considered up-regulated in a column if its standardized level with mean 0 and variance 1 is above 1. In the same way, it will be down-regulated if the standardized level is below -1. In addition, depending on the data type, we could work with a signed graph that takes in account the direction of change for each edge or to use unsigned graph so the likelihood score will be computed in the same way for each one.

In the second stage, the hashing technique will be applied to find the heaviest bicliques in the graph. The main idea in this phase that by finding the heaviest subgraphs in the bipartite graph will help to find the most significant biclusters in the data. To solve this problem they proposed the following algorithm, which can deals with this problem when the degree of every gene vertex is bounded, as follows:

```

1- Initialize a hash table weight:  $weight_{best} = 0$ 
2- For all  $v \in V$  do
    For all  $S \subseteq N(v)$  do
         $weight[S] \leftarrow weight[S] + \max\{0, w(S, \{v\})\}$ 
        If( $weight[S] > weight_{best}$ )
             $U_{best} \leftarrow S$ 
             $weight_{best} \leftarrow weight[S]$ 
3- Compute  $V_{best} = \cap_{u \in U_{best}} N(u) \rightarrow \rightarrow \rightarrow \rightarrow$  Output( $U_{best}, V_{best}$ )

```

Figure 3.22. MaxBoundBiClique(U,V,E,d)

By using the previous algorithm, for every given column or row it will look to find the k best bicliques that intersecting them.

In the last phase of the SAMBA algorithm, in each heap, a local improvement procedure will be applied on the founded biclusters. This procedure includes will add or delete a single vertex iteratively until no improvement could be applied so the bicluster is better. During the process of algorithm, some biclusters that differ from each other with a few numbers of vertex set. Therefore, a filtering process will be applied on such biclusters with a specific threshold value. Finally, the SAMBA as presented by Tanay et al. [46, 55] is as following:

Input: U : conditions, V : genes, E : graph edges, w : edge/non-edge weights, N_1, N_2 : hashed set size limits, k : max biclusters per gene/condition.

- 1- Initialize a hash table weight.
- 2- For all $v \in V$ with $|N(v)| \leq d$ do
 - For all $S \subseteq N(v)$ with $N_1 \leq |S| \leq N_2$ do $\text{weight}[S] \leftarrow \text{weight}[S] + w(S, \{v\})$
- 3- For each $v \in V$ set $\text{best}[v][1..k]$ to the k heaviest S such that $v \in S$
- 4- For each $v \in V$, $i \in \{1..k\}$
 - $S = \text{best}[v][i]$
 - $V' \leftarrow \bigcap_{u \in S} N(u)$
 - $B \leftarrow S \cup V'$
 - Do {
 - $a = \arg \max_{x \in V \cup U} (w(B \cup x))$
 - $b = \arg \max_{x \in B} (w(B - x))$ $b = \arg \max_{x \in B} (w(B - x))$
 - If $w(B \cup a) > w(B - b)$ then $B = B \cup a$ Else $B = B - b$
 - } While improving
 - Store B
- 5- Post process to filter overlapping bicluster

Figure 3.23. SAMBA($U, V, E, w, d, N_1, N_2, k$)

3.9. xMotif Algorithm

Murali and Kasif [36] introduced a nondeterministic greedy algorithm, which deals with a discretized dataset to find biclusters with conserved gene expression motifs that known as xMotifs. An xMotif (bicluster) is a subset of rows (genes) that is simultaneously conserved across a subset of columns (samples). Conserved means that the row's levels in the same abundance across a subset of columns. Thus, xMotif algorithm's goal is finding the largest conserved gene motifs that cover all the samples and classes in the data.

According to Murali and Kasif [36], gene expression levels could be up-regulated or down-regulated. In addition, xMotifs has the following properties:

1. Each motif should be matched by a large fraction of the samples in that class.
2. Each motif should contain as many conserved genes as possible.
3. Motif should not have many genes in general because that may cause no sample may match the motif.

3.9. Definition: For a given set of rows with values levels that measured across a set of conditions, and with user-defined parameters $0 < \alpha, \beta < 1$, xMotif is a $pair(C, G)$, where C is a subset of columns and G is a subset of rows and satisfies the following properties:

- C should contain a number of columns that at least α -fraction from all columns.
- The values levels are in the same bounds across the columns in C .
- For every row not in G , the row is conserved in at most a β -fraction of the columns in C .

From the previous definition, any dataset may have more than one bicluster and xMotif algorithm will work to find the largest one. However, k biclusters can be detected by repeating algorithm k times after taking off the samples that are contained in the previously founded biclusters from previous repetitions. Using xMotif algorithm allows a row to be founded in more than one motif. In addition, after finding one bicluster by not deleting the matched columns will allow them to appear in other biclusters. Another good feature in this algorithm that it is not necessary to specify the number of the biclusters, so we can find all of the biclusters in the data.

According to Murali and Kasif [36], a state is a range of expression values that are statistically significant. If there are columns (samples), then there are $\binom{n}{2}$ possible states for each row (gene) that not all of them may be interesting according to the study filed. They assumed that the data has been generated by a uniform distribution and the state $[a, b]$ will be interesting if the expression values in it are unlikely to have been generated by a uniform distribution. To make a decision that a state is interesting or not will be by computing p-value like the following:

$$\sum_{k \leq i \leq n} \binom{n}{i} (b-a)^i (1-(b-a))^{n-i} \quad (3.46)$$

which will be used to test the null hypothesis, which says that the state is not interesting where k is the number of values that lie in the interval $[a, b]$.

The complete algorithm

Murali and Kasif [36] introduced the xMotif algorithm as follow: xMotif Algorithm takes a set of rows, a set of columns, and an expression value for each row-column pair, and for each row, a list of intervals representing the states in which the rows are expressed in the columns. The data is in a matrix form. To find an xMotif (bicluster), the set G of the conserved row, the states that these rows are in, and the set C of columns that match the motif must be computed. By having the set G , the states of the conserved rows, and one column c that matches this motif, then the remaining columns in C can be computed also by checking for each column c' whether the rows in G are in the same state in c and c' . c is known as a seed that used to compute the entire motif.

With a known column c , and with a given set D of columns, which have the following properties: For every column c' in the set D and for every row in the largest motif, there is exactly one state such that the row is in that state in columns c and c' . For every row g , which does not belong to the largest motif, there exists a column c' in D such that row g is not in the same state in columns c and c' . D is known as a Discriminating Set. Finding the larger xMotif in the data is as the following[36]:

Input: n_s : number of the columns (samples) that will be selected at random, n_d : the number of columns, which will be selected at random for every seed, s_d : number of the element in the d^{th} set, α : scaling factor.

Output: One Biclustor for every repetition.

Steps:

1- For $i = 1$ to n_s do

Choose a sample c uniformly at random

For $j = 1$ to n_d do

Choose a subset D of the samples of size s_d uniformly at random.

For each gene g , if g is in the state s in c and all samples in D , include the pair (g, s) in the set G_{ij}

C_{ij} = set of samples that agree with c in all the gene-states in G_{ij}

Discard (C_{ij}, G_{ij}) if C_{ij} contains less than αn samples

2- Return the motif (C^*, G^*) that maximizes $|G_{ij}|, 1 \leq i \leq n_s, 1 \leq j \leq n_d$

Figure 3.24. Find Motif

Practically, xMotif algorithm is a probabilistic algorithm that assumes for each row, the intervals corresponding to that row's states are disjoint. The process is done by selecting n_s column (samples) uniformly at random from the set of all columns. The selected items are the seeds. Then, for each random seed, n_d sets of columns will be selected at random from all of the columns with s_d elements in each set. These sets act as candidates for the discriminating set. Now, seed-discriminating set pairs are obtained, and for each one, the corresponding xmotif will be computed as explained above. If the motif has less than an α -fraction of the columns match it will discard the motif. At the end of the whole process, we obtain the largest motif as a bicluster.

3.10. ROBA Algorithm

Tchagang and Tewfix [47] introduced Robust Biclustering Algorithm, which is known as ROBA algorithm, which uses basic linear algebra and arithmetic tools to detect the

biclusters. The goal of this algorithm is to find submatrices that contain a subset of rows (genes) and a subset of columns (conditions) with high correlation to each other. This method is easy to apply because it uses basic linear algebra and arithmetic tools as mentioned by Tchagang and Tewfix [47]. ROBA algorithm is able to find the following bicluster types:

1. Biclusters with constant values.
2. Biclusters with constant values on rows.
3. Biclusters with constant values on columns.
4. Biclusters with coherent values.

The data will be arranged in a matrix where each row represents one gene and each column represents one condition, as follows:

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1M} \\ a_{21} & a_{22} & \cdots & a_{2M} \\ \vdots & \vdots & \vdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nM} \\ \vdots & \vdots & \vdots & \vdots \\ a_{N1} & a_{N2} & \cdots & a_{NM} \end{bmatrix} = \begin{bmatrix} r_1 \\ r_2 \\ \vdots \\ r_n \\ \vdots \\ r_N \end{bmatrix} = [c_1 \quad c_2 \quad \cdots \quad c_m \quad \cdots \quad c_M] \quad (3.47)$$

Where a_{ij} is the expression value in row i and under the column j ,

$$r_n = [a_{n1} \quad a_{n2} \quad \cdots \quad a_{nm} \quad \cdots \quad a_{nM}], c_m = [a_{1m} \quad a_{2m} \quad \cdots \quad a_{nm} \quad \cdots \quad a_{Nm}]^T.$$

ROBA algorithm will work to find a bicluster with maximum size and it gives the user to identify all qualified biclusters in each type. The ROBA algorithm has three main parts: (1) manipulating the data to deal with noise and missing values, (2) decomposing the data matrix A into its elementary matrices, and (3) extracting any type of biclusters defined by the user.

Data conditioning

Most of the data in real life contain some noise and amount of missing values. Before detecting biclusters ROBA algorithm conditioning data to find more high-quality biclusters. There are many methods to deal with missing values, but they proposed replacing the

missing values by zeros. On another hand, to deal with the noise they identified the number L of distinct values a_l that constitute the gene expression matrix A . Then they redefined a_l as follow:

$$\alpha_l = (b_l + b_{l-1}) / 2 \quad (3.48)$$

Where $b_l = b_0 + le$, with $l = 1$ to L , $e = (b_L - b_0) / L$, $b_0 = \min([a_{nm}])$ and $b_L = \max([a_{nm}])$.

Then the interval $[b_0, b_L]$ will be divided into L equal intervals, as follow:

$$[b_0, b_L] = [b_0, b_1[\cup \dots \cup [b_{l-1}, b_l[\cup \dots \cup [b_{L-1}, b_L] \quad (3.49)$$

Finally, a new data matrix will be obtained for the next phase. These steps are shown in the following algorithm:

Input: A data matrix with non-missing values replaced by zeros

Output: A matrix where the noise was dealt with.

Steps:

1- Compute: $L, b_L, b_0, e, b_l, \alpha_l$

2- For $l = 1$ to L

 For $n = 1$ to N

 For $m = 1$ to M

 If $a_{nm} \in [b_{l-1}, b_l[$

$a_{nm} = \alpha_l$

 End

 End

 End

End

Figure 3.25. Algorithm 1

Gene expression matrix decomposition

In this phase, the matrix will be decomposed into its elementary matrices using the following equation:

$$A = \sum_{l=1}^L \alpha_l A_l = \alpha_1 A_1 + \dots + \alpha_L A_L \quad (3.50)$$

Where

$$A_l = \begin{bmatrix} r_1^l & r_2^l & \dots & r_n^l & \dots & r_N^l \end{bmatrix}^T = \begin{bmatrix} c_1^l & c_2^l & \dots & c_m^l & \dots & c_M^l \end{bmatrix} \quad (3.51)$$

$$r_n = \sum_{l=1}^L \alpha_l r_n^l, \text{ and } c_m = \sum_{l=1}^L \alpha_l c_m^l \quad (3.52)$$

From equation (3.50) A_l are binary matrices with N row and M column, r_n^l are binary $l \times M$ vectors, and c_m^l are binary $N \times l$ vectors.

Biclusters identification

As it was mentioned before, ROBA algorithm four type of biclusters as follows:

Biclusters with constant values

This type of biclusters have the same constant value in it, and let it be μ that means:

$$B = [a_{ij}] = \mu, i = 1, \dots, I; j = 1, \dots, J \quad (3.53)$$

That means the expression values in rows do not change across the columns (conditions). From equation (3.50) a submatrix with constant values can be obtained by analyzing each A_l separately. In addition, the matrix A_l is a binary matrix, and because of the fact, the number of rows in data is bigger from columns in gene expression data especially the number of the biclusters with constant values is defined as follow:

$$N_b = \sum_{l=1}^L P_l \quad (3.54)$$

where P_l is the number of distinct rows r_i^l of each A_l with a sum that is greater than 0. In addition, every distinct row of A_l constitutes the principal row element of the i^{th} bicluster B_i^l of the matrix A_l considered. Then, to check if any other row of A_l belongs to i^{th} bicluster or not the following equation is used:

$$r_i^l * r_n^l = r_i^l, \quad i = 1, \dots, p_l, \quad n = 1, \dots, N, \quad l = 1, \dots, L \quad (3.55)$$

Finally, the following algorithm shows how practically biclusters with constant expression level α_l can be detected:

```

1- Compute:  $P_l, r_i^l, r_n^l$ 
2- For  $l = 1$  to  $L$ 
  For  $i = 1$  to  $P_l$ 
     $B_i^l = []$ ;
    For  $n = 1$  to  $N$ 
      If  $r_i^l * r_n^l == r_i^l$ 
         $B_i^l = [B_i^l; [Genes(n) \alpha_l r_i^l]]$ 
      End
    End
  End
End;  $B_i^l = [[0 \text{ Conditions}]; B_i^l]$ ;

```

Figure 3.26. ROBA Algorithm: finding biclusters with constant values

Biclusters with constant values on columns

A submatrix $B(I, J)$ is a bicluster with constant values on the column if its values a_{ij} are either in form of additive model $a_{ij} = \mu + \beta_j$ or as a multiplicative model $a_{ij} = \mu \cdot \beta_j$ and the general form is:

$$B = \begin{bmatrix} \cdot & \cdot & \cdots & \cdot \\ \mu_1 & \mu_2 & \cdots & \mu_j \\ \cdot & \cdot & \cdots & \cdot \end{bmatrix} \quad (3.56)$$

Using equation (3.50) the number of biclusters with constant values on columns is $N_b = P_c$ where P_c is the number of distinct columns c_j of the whole A_i that his sum is greater than 0. In addition, each distinct column c_j of the entire A_i constitutes the principal column element of the i^{th} bicluster B_j . After that, to find the other columns of any A_i that belong to the i^{th} bicluster the following equation is used:

$$c_j * c_m^l = c_j, \quad J = 1, \dots, P_c, m = 1, \dots, M, l = 1, \dots, L \quad (3.57)$$

Finally, ROBA algorithm that finds biclusters with constant values on columns is:

```

1- Compute:  $P_c, c_j, c_m^l$ 
2- For  $j = 1$  to  $P_c$ 
     $B_j = []$ ;
    For  $l = 1$  to  $L$ 
        For  $m = 1$  to  $M$ 
            If  $c_j * c_m^l = c_j$ 
                 $B_j = [B_j [Conditions(m); \alpha_l c_j]]$ 
            End
        End
    End;  $B_j = [[0 \text{ Genes}] \ B_j]$ ;
End

```

Figure 3.27. ROBA Algorithm: finding biclusters with constant values on Columns

Biclusters with constant values on rows

A submatrix $B(I, J)$ is a bicluster with constant values on rows if its values a_{ij} are either in form of additive model $a_{ij} = \mu + \alpha_i$ or as a multiplicative model $a_{ij} = \mu \cdot \alpha_i$ and the general form is:

$$B = \begin{bmatrix} \cdots & \mu_1 & \cdots \\ \cdots & \mu_2 & \cdots \\ \cdots & \cdots & \cdots \\ \cdots & \mu_l & \cdots \end{bmatrix} \quad (3.58)$$

From equation (3.50) the number of biclusters with constant values on rows is $N_b = P_r$ where P_r representing the number of distinct rows of the entire A_l with $\text{sum}(r_i) > 0$. Each distinct row r_i of the entire A_l constitutes the principal row element of the i^{th} bicluster B_i .

In the same way above, to detect other rows that belong to the i^{th} bicluster the following equation is used:

$$r_i * r_n^l = r_i; \quad i = 1, \dots, P_r, n = 1, \dots, N, l = 1, \dots, L \quad (3.59)$$

Thus, the final algorithm to find biclusters with constant values on rows is:

```

1- Compute:  $P_r, r_i, r_n^l$ 
2- For  $i = 1$  to  $P_r$ 
     $B_i = []$ ;
    For  $l = 1$  to  $L$ 
        For  $n = 1$  to  $N$ 
            If  $r_i * r_n^l = r_i$ 
                 $B_i = [B_i; [\text{Genes}(n) \ \alpha_l r_i]]$ 
            End
        End
    End;  $B_i = [[0 \ \text{Conditions}]; B_i]$ ;
End

```

Figure 3.28. ROBA Algorithm: finding biclusters with constant values on Rows

Biclusters with coherent values

A bicluster $B = [a_{ij}]$ with I rows and J columns is a bicluster with coherent values if its values have either an additive model $a_{ij} = \mu + \alpha_i + \beta_j$ or a multiplicative model $a_{ij} = \mu \cdot \alpha_i \cdot \beta_j$ form. In Tchagang and Tewfix's work [47], they worked just with the additive model $B = [\mu + \alpha_i + \beta_j] = [\mu] + [\alpha_i] + [\beta_j]$, which is the sum of three matrices. The first matrix is a matrix with constant values; the second is with constant values on rows, and the third on columns. Let the three matrices be B_1, B_2 , and B_3 , respectively.

To find the biclusters, the data matrix A will be written as a sum of three matrices: Z_1 a matrix with constant values, Z_2 a matrix with constant values on columns and the last one $Z_3 = A - (Z_1 + Z_2)$. After this step, an algorithm that used to find biclusters with constant values on rows will be applied on Z_3 . After that, they will be added back to their corresponding matches into Z_1 and Z_2 . Finally, obtain subgroups of the gene with coherent values.

3.11. Bimax Algorithm

Prelic et al. [10] introduced Bimax algorithm, which works with binary data type and follows the divide and conquers strategy to detect biclusters in the data. The main goal of this algorithm to make it as a reference algorithm when applying biclustering on data. Bimax algorithm is considered as a fast algorithm, which can find all optimal groupings. Algorithm uses a simple data model reflecting the fundamental idea of biclustering while allowing determining all optimal biclusters in a reasonable time.

Algorithm deals with binary data, which can be obtained from the data matrix, by observing whether there is change or there is not under different conditions. Let $E_{n \times m}$, be the data matrix with n genes (rows) under m microarray experiments (columns). Thus, each value e_{ij} will have the value 1 if there is a change and 0 in the other state. A bicluster (G, C) with a subset of rows G that has response across a subset of columns C . In the Bimax algorithm, a bicluster is a submatrix of E for which all elements equal 1.

3.10. Definition: The pair $(G, C) \in 2^{\{1, \dots, n\}} \times 2^{\{1, \dots, m\}}$ is called an inclusion-maximal bicluster if and only if:

- 1) $\forall i \in G, j \in C: e_{ij} = 1$
- 2) $\nexists (G', C') \in 2^{\{1, \dots, n\}} \times 2^{\{1, \dots, m\}}$ with:
 - a. $\forall i' \in G', j' \in C': e_{i'j'} = 1$
 - b. $G \subseteq G' \wedge C \subseteq C' \wedge (G', C') \neq (G, C)$

The Bimax algorithm will work to find biclusters that are inclusion-maximal.

The complete algorithm

The Bimax algorithm has running-time complexity equal to $O(nm\beta \log \beta)$, where n is the number of rows, m the number of columns, and β is the number of all inclusion-maximal biclusters in E .

Algorithm, as illustrated in Figure 3.11, will divide E into 3 submatrices, one of them have only zeros that can be ignored later. Let the other two submatrices be U and V , and the Bimax algorithm will be applied many times to these submatrices until obtaining bicluster that has only ones.

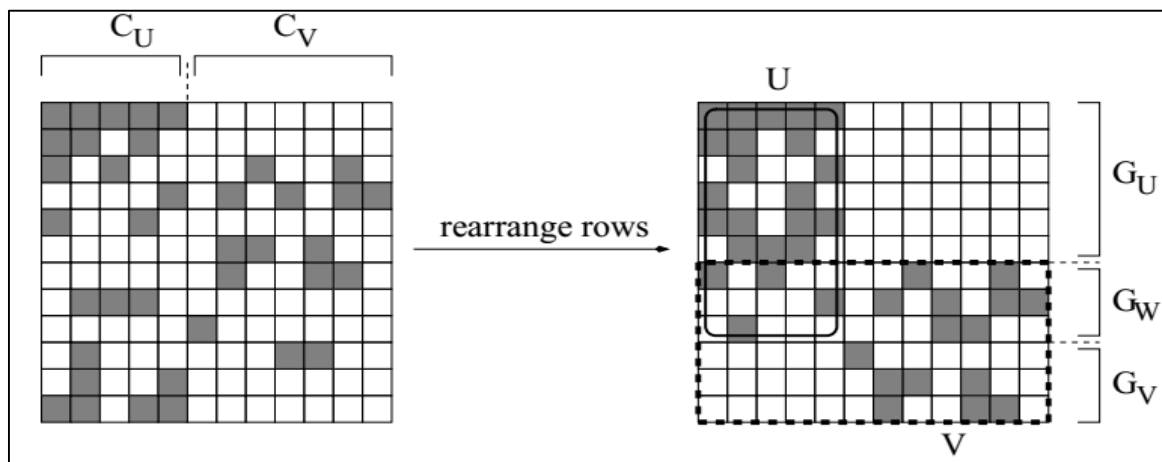


Figure 3.29. Source [10]: An example of Bimax algorithm

Bimax algorithm made of some procedures as follows [41]:

<pre> 1- Procedure Bimax(E) $Z \leftarrow \phi$ $Z \leftarrow \phi \ M \leftarrow conquer(E, (\{1, \dots, n\}, \{1, \dots, m\}), Z)$ Return M End Procedure 2- Procedure $conquer(E, (G, C), Z)$ if $\forall i \in G, j \in C : e_{ij} = 1$ then return $\{(G, C)\}$ end if $(G_U, G_V, G_W, C_U, C_V) = divide(E, (G, C), Z)$ $M_U \leftarrow \phi, M_V \leftarrow \phi$ if $G_U \neq \phi$ then $M_U \leftarrow conquer(E, (G_U \cup G_V, C_U), Z)$ end if if $G_V \neq \phi \wedge G_W \neq \phi$ then $M_V \leftarrow conquer(E, (G_V, C_V), Z)$ else if $G_W \neq \phi$ then $Z' \leftarrow Z \cup \{C_V\}$ $M_V \leftarrow conquer(E, (G_W \cup G_V, C_U \cup C_V), Z')$ end if return $M_U \cup M_V$ End procedure 3- Procedure $divide(E, (G, C), Z)$ $G' \leftarrow reduce(E, (G, C), Z)$ choose $i \in G'$ with $0 < \sum_{j \in C} e_{ij} \triangleleft C$ if such an $i \in G'$ exists then $C_U \leftarrow \{j \mid j \in C \wedge e_{ij} = 1\}$ </pre>	<pre> Else $C_U = C$ end if $C_V \leftarrow C \setminus C_U$ $G_U \leftarrow \phi, G_V \leftarrow \phi, G_W \leftarrow \phi$ for each $i \in G'$ do $C^* \leftarrow \{j \mid j \in C \wedge e_{ij} = 1\}$ if $C^* \subseteq C_U$ then $G_U \leftarrow G_U \cup \{i\}$ else if $C^* \subseteq C_V$ then $G_V \leftarrow G_V \cup \{i\}$ else $G_W \leftarrow G_W \cup \{i\}$ end if end for return $(G_U, G_V, G_W, C_U, C_V)$ End Procedure 4- Procedure $reduce(E, (G, C), Z)$ $G' \leftarrow \phi$ For each $i \in G$ do $C^* \leftarrow \{j \mid j \in C \wedge e_{ij} = 1\}$ if $C^* \neq \phi \wedge \forall C^+ \cap C^* \neq \phi$ then $G' = G' \cup \{i\}$ end if end for return G' End Procedure </pre>
----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Figure 3.30. Bimax Algorithm

The input to the Bimax algorithm is a discretized gene expression data matrix. The data will be converted using a threshold to a binary data where each expression value will have a value equal to one if it is bigger than the threshold and zero otherwise. Then, using the previous procedures above will work to detect all of the possible biclusters that contain only ones. In every iteration, the rows and the columns in the data matrix will be rearranged to concentrate ones in the upper right of the matrix. Then, the matrix will be divided into submatrices. Finally, when a matrix that only has ones in it, this submatrix will be returned as showed in Figure 3.11[42].

3.12. RMSBE Algorithm

Wang and Liu [48] introduced Randomized MSB Extension algorithm, which is known as RMSBE algorithm. This algorithm is able to detect optimal square biclusters with the maximum similarity score. RMSBE algorithm is the first algorithm that admits a polynomial time algorithm for optimal solutions. Algorithm has the following features: Discretization procedure is not necessary, work well with overlap biclusters, and works well for additive biclusters. Algorithm uses a similarity score that has been defined in their work, which measures the similarity between two genes and another similarity score for a sub-matrix.

The data will be in a matrix shape $A(I, J)$ with n number of rows (genes) and m number of columns. Every entry a_{ij} in the data matrix represents the value level in the gene (row) i under the condition (column) j . The goal of this algorithm is to find submatrix (bicluster) that related to a reference gene. However, if the reference gene is not known, the references will be chosen from data by random.

3.11. Definition (Similarity score between genes): To compute the difference between an element a_{ij} and a reference gene $i^* \in I$ in the data matrix as follow:

$$d_{ij} = |a_{ij} - a_{i^*j}| \quad (3.60)$$

If we are looking to find constant biclusters the average distance value of all elements in $A(I, J)$ must first be computed as follow:

$$d_{avg} = \frac{\sum_{i \in I, j \in J} d_{ij}}{|I| |J|} \quad (3.61)$$

Then if $d_{ij} \geq \alpha d_{avg}$ that means the two elements a_{ij} and a_{i^*j} are different from each other and the similarity s_{ij} is 0, where α is a user-defined threshold to control the amount of the similarity. Otherwise, the similarity score is:

$$s_{ij} = 1 - \frac{d_{ij}}{\alpha d_{avg}} + \beta \quad (3.62)$$

where β is a bonus value for small d_{ij} that used to enlarge the similarity score for small d_{ij} so we ignore d_{ij} values that are greater than the threshold.

3.12. Definition (Similarity score for a bicluster): Let $S(I, J)$ be a data matrix with n number of rows and m number of columns, and $S(I', J')$ be a bicluster. The similarity score for a row i in the bicluster is:

$$s(i, J') = \sum_{j \in J'} s_{ij} \quad (3.63)$$

In the same way, the similarity score for a column j in the bicluster is:

$$s(I', j) = \sum_{i \in I'} s_{ij} \quad (3.64)$$

In addition, the similarity score for $S(I', J')$ is:

$$s(I', J') = \min \left\{ \min_{i \in I'} s(i, J'), \min_{j \in J'} s(I', j) \right\} \quad (3.65)$$

In the constant bicluster state, if the similarity score for a row $i \in I'$ in a bicluster $S(I', J')$ is high, then the expression values of gene i is similar to the reference row i^* under the column subset J' . If we are looking to detect constant biclusters, that means we are looking to find a submatrix with the highest similarity score. Formally maximum similarity bicluster is defined as follow:

3.13. Definition: For a given $n \times m$ similarity matrix $S(I, J)$, MSB (maximum similarity bi-cluster) problem is to find a bicluster $S(I', J')$ with $I' \subseteq I$ and $J' \subseteq J$ such that $s(I', J')$ is maximized.

The complete algorithm

The problem of finding MSB in $n \times m$ similarity matrix $S(I, J)$ has a polynomial time, which is a greedy algorithm. Algorithm will start with the whole matrix as a bicluster. Then the rows or columns with the smallest similarity score will be deleted. The previous steps will be applied until we obtain one element in the current bicluster. The result of this process will be $n + m - 1$ bicluster where n is the number of rows and m is the number of the columns. Finally, the submatrix $S(I_{k'}, J_{k'})$ with the maximum similarity score $s(I_{k'}, J_{k'})$ will be chosen. The previous process practically is as follow, which has time in complexity equal to $O((n + m)^2)$

Input: $n \times m$ similarity matrix $S(I, J)$

Output: A maximum similarity bicluster $S(I_A, J_A)$

1. Set the first bicluster $S(I_1, J_1) = S(I, J)$ and compute the similarity score for all rows and columns of $S(I_1, J_1)$.
2. For $k = 1$ to $n + m - 2$ do
 - a. Find row $i' \in I_k$ such that $s(i', J_k) = \min_{i \in I_k} s(i, J_k)$
 - b. Find column $j' \in J_k$ such that $s(I_k, j') = \min_{j \in J_k} s(I_k, j)$
 - c. If $s(i', J_k) < s(I_k, j')$ then set $I_{k+1} = I_k - \{i'\}$ and $J_{k+1} = J_k$ Else set $I_{k+1} = I_k$ and $J_{k+1} = J_k - \{j'\}$
3. Let $S(I_{k'}, J_{k'}), 1 \leq k' \leq n + m - 1$, be the bicluster such that

$$s(I_{k'}, J_{k'}) = \max_{1 \leq k \leq n + m - 1} s(I_k, J_k).$$
4. Output $S(I_A, J_A) = S(I_{k'}, J_{k'})$

Figure 3.31. The MSB algorithm

The output of this algorithm may not be ideal because it may contain some element with low similarity values. To solve this problem, Wang and Liu [48] also proposed of using a second similarity score to control the quality of the biclusters. This scale is the average similarity score, which is a follow for a given bicluster $S(I', J')$ is:

$$s_{avg}(I', J') = \frac{\sum_{i \in I'} \sum_{j \in J'} s_{ij}}{|I'| |J'|}$$

(3.66)

The goal is to detect biclusters with average similarity score that is not less than a threshold γ . To improve the previous algorithm according to the γ threshold the third step in the algorithm will be modified as follow:

Let $S(I_{k'}, J_{k'}), 1 \leq k' \leq n+m-1$, be the bicluster such that:

$$s(I_{k'}, J_{k'}) = \max_{1 \leq k \leq n+m-1 \& s_{avg}(I_k, J_k) \geq \gamma} s(I_k, J_k) \quad (3.67)$$

The modified algorithm is called γ -MSB algorithm that works to detect approximately squared biclusters. In case that the data matrix has a number of rows very bigger from the number of columns the results biclusters that obtained by not take in computation some rows. This problem is solved by using the γ -MSB algorithm then the bicluster will be extended by adding rows with high similarity scores in the subset of columns to the bicluster.

In an additive bicluster $A(I', J')$ from data matrix $A(I, J)$, the expression values of the rows fluctuate in the same way as the reference row. In an error-free additive bicluster $A(I', J')$ for reference row i^* , we have $\forall i \in I'$ and $\forall j \in J'$, $a_{ij} = a_{i^*j} + c_i$ where c_i is a constant for row i . In addition, if reference column j^* is known a new matrix $B(I, J)$ can be obtained by setting $b_{ij} = a_{ij} - (a_{ij^*} - a_{i^*j^*})$, which is an error-free constant bicluster for error-free additive bicluster of the reference gene i^* .

Most of times, the reference column (condition) is unknown, so all of the columns will be tried as a reference. The complete algorithm for computing additive MSBE as presented by Wang and Liu [48] as follows:

Input: $A(I, J)$ data matrix with size $n \times m$.

Output: A set of additive biclusters.

1. For every column j^* in J do
 - a. Compute $B(I, J)$ using $b_{ij} = a_{ij} - (a_{ij^*} - a_{i^*j^*})$
 - b. For each row i^* in I do
 - i. Convert $B(I, J)$ into $S(I, J)$ based on row i^*
 - ii. Use γ -MSB algorithm to compute the bicluster $S(I_A, J_A)$
 - iii. Use the extension algorithm to get extended bicluster $S(I_E, J_E)$ from $S(I_A, J_A)$
 - iv. Output the bicluster $A(I_E, J_E)$

Figure 3.32. Additive MSBE Algorithm

According to Wang and Liu [48] when dealing with a big data set additive MSBE algorithm requires a long time equal to $O(nm(n+m)^2)$, so they developed a randomized algorithm, which known as RMSBE (Randomized MSBE) algorithm, which will speed up the process as follows:

Instead of trying to use all of the rows as a reference, a part of the rows will be randomly selected as the reference rows. If there are a $b \times c$ bicluster from $n \times m$ bicluster, n/b rows can be randomly selected. The expectation of the number of selected genes that are in the $b \times c$ bicluster is 1. To obtain good results while applying biclustering if more than one reference row is used in the bicluster and select the best result. In addition, if the column reference is not known, set of columns can be also selected. Using RMSBE algorithm will give faster and better results depending on the randomly selected row and column sets.

3.13. QUBIC Algorithm

Li et al. [37] introduced a Qualitative BIClustering algorithm, which is known as Qubic algorithm, which is employing a combination of qualitative measures of gene expression data and a combinatorial optimization technique to detect all statistically significant biclusters including biclusters with the so-called ‘scaling patterns’. Algorithm has another

important feature that it can deal with very big data sets more efficiently from many biclustering algorithms. By using Qubic, all statistically significant biclusters can be detected, which could be overlapped. In addition, it can detect both positively and negatively correlated expression patterns.

Data is arranged in a matrix where rows are presenting the genes and columns are the different conditions in gene expression datasets. According to Li et al. [37] two genes will be related under a set of conditions if they have identical integer numbers in a representing matrix along the two corresponding rows of the matrix. They gave the name feasible matrix to the submatrix if each pair of rows of the submatrix is (approximately) either the same or the opposite. Thus, they defined the biclustering problem is to detect all the optimal feasible submatrices in a given matrix according to some specified optimization criteria.

The main idea in this algorithm is as follow: For a data matrix, a representing matrix is created where the expression value of a row under each column is represented as an integer value according to their levels and how it changes or not. Then, for the representing matrix, the weighted graph G will be constructed. In this graph, the rows (genes) is presented as vertices, edges connecting every pair of rows. For each edge, a weight will be given to it, which is presenting the similarity level between the two corresponding rows. When a weight goes higher that means there is more similarity between the two rows. A good bicluster should contain a heavier subgraph of G . However, it is not necessary that every heavy subgraph belongs to the bicluster because it may not have similar expression patterns. Finding all heavy subgraphs requires lots of computation and long time, so Li et al. [37] proposed a solution, which says that instead of solving the problem of finding heavy subgraphs in a graph, the bicluster will be build based on the graph that representing the data. After that, algorithm work to find all of the feasible biclusters (I, J) in the data where $\min\{|I|, |J|\}$ is as large as possible, and I is a subset of rows and J a subset of columns.

The complete algorithm

The QUBIC algorithm has two main phases: the first one is to present the data using a qualitative matrix, and then detecting the biclusters in the data one by one. Every one of these biclusters will start with the heaviest unused edge as a seed to build an initial bicluster.

Finally, in iteratively process rows will be added to the biclusters without violating a pre-specified consistency level, which we will be presented later. The qualitative matrix will contain signed integers and zeros based on:

1. On the way of how the expression values change, which included whether the row's value change or not, and in which direction.
2. The ranking of all the upregulating conditions for each gene based on the expression values of the gene under these conditions.

A gene can be checked whether it unchanged under the condition j as follow: For row i in the data matrix, which has n rows and m columns, expression values will be ordered in increasing order as follow:

$v_{i1}, \dots, v_{i,s-1}, v_{i,s}, \dots, v_{i,c-1}, v_{i,c}, v_{i,c+1}, \dots, v_{i,m-s+1}, v_{i,m-s+2}, \dots, v_{im}$ where $c = m/2$ and $s-1 = m \times q$, where q is a user-defined parameter. Thus, a row (gene) i is considered as unchanged under column (condition) j , if and only if its expression value w_{ij} belongs to the interval $(v_{ic} - d_i, v_{ic} + d_i)$, where $d_i = \min\{v_{ic} - v_{is}, v_{i,m-s+1} - v_{ic}\}$.

To rank the conditions: The column (condition) is considered as a down-regulating for a row (gene) i if its value is $\leq v_{ic} - d_i$, and up-regulating condition in the other state. For the row i , we sort all the upregulating conditions in decreasing order of their corresponding expression values. This order will be used as the rank of each up-regulating column for row i . In the same way, the down-regulating conditions will be ranked. However, the sort will be based on the relevant gene-expression values into the increasing order, and the order will be used as the rank of each down-regulating condition for gene i . For each up-regulating column a '+' will be assigned, and a '-' for every down-regulating columns. In the gene expression data, two genes are considered as oppositely regulated across a subset of conditions, if they have identical nonzero integers column-wise except with opposite signs.

Biclustering through finding a heavy subgraph

As it was mentioned above, for each edge, a weight will be assigned. The weight is the number of columns under each of which the two rows have the same nonzero integer. The problem of biclustering is to detect a submatrix from the data matrix M with I a subset of rows and J a subset of columns so that $\min\{|I|, |J|\}$ is maximal and the consistency level of (I, J) is higher than a pre-specified value c , $0 < c \leq 1.0$.

Algorithm will start work with a set S of seeds (edges). First, S will be set to be the sorted list of edges in G to choose whether the seed $e = g_i g_j$ is seed or not if and only if:

1. One of the genes g_i and g_j or both are not in any biclusters that have been founded before.
2. g_i and g_j are in different biclusters $B_1(I_1, J_1)$ and $B_2(I_2, J_2)$ with I_1 and I_2 do not have common elements and $w(e) \geq \max\{|I_1|, |I_2|\}$. Where $w(e)$ is the weight of the edge e .

The QUBIC algorithm will build the initial bicluster (I, J) based on the chosen seed. After that, the bicluster will be expanded along both the vertical and horizontal directions with respect to the preset consistency level. The stop point is when the bicluster is no more can be expanded. The QUBIC algorithm in detail as follows [37]:

In the first step, we stop if S is empty. If it is not empty, we will check the first element in S if it is a seed or not. That element will be removed from S if it is not a seed and moves to the second element. If it is a seed, we will look to find all columns (conditions) under which the two rows (genes) of the seed have all identical nonzero integer values and set these columns of the two rows as the current bicluster $B(I, J)$, then we start the second step.

In the second step, the current bicluster B will be expanded by adding new rows from the data matrix that does not belong to I and these rows are most consistent with bicluster. By the applying the process of adding row we obtain a new bicluster $B'(I', J')$ where I' is I after the process of adding a new row and J' is J after deleting those columns where the

total consistency is lost. If $\min\{|I'|, |J'|\} \geq \min\{|I|, |J|\}$, we set B to B' , then repeat this step. If $\min\{|I'|, |J'|\} < \min\{|I|, |J|\}$ and if the preset consistency level is 1.0, B will be the output and the used seed will be removed from S else will move to the third step.

The third step work on adding many columns as possible to the bicluster B with saving the consistency level below the user-defined value c as follows: For every column in the data matrix and is not in B , the ratio between the number of identical nonzero integers in the rows of I and $|I|$ is computed and if is bigger or equal to c , we add that column. Let $B'(I', J')$ be the new bicluster and T be the consensus sequence of B' consisting of the dominating elements of the columns of B' , where the dominating element is the element with the highest frequency in the column; add as many rows as possible to B' such that each new row has at least $|I'|c$ identical nonzero integers to those of T . Then we start the fourth step.

In this step, we continue expanding the bicluster by adding oppositely regulated rows: let T be the consensus sequence of bicluster B , we add as many rows as possible to the bicluster where each row has at least $|I'|c$ identical nonzero integers. However, they will be added with the opposite signs to those of T . Finally, we have B bicluster as output and starting again from the first step.

According to Li et al. [37], this algorithm has the following features:

1. If a significant bicluster is being detected in the second step but not completed, it could be built later using other edges of the bicluster as seeds.
2. Algorithm able to find both positively co-regulated rows and negatively co-regulated rows.
3. The QUBIC algorithm can be work with user-provided seeds.
4. The QUBIC algorithm does not, in general, suffer from the issue of being stuck in local optima since it uses multiple starting points (seeds) to find each bicluster.

Algorithm requires the following parameter to start working to detect the bicluster: r the range of possible ranks, q the percentage of the regulating columns for each row, c the

required consistency level for the biclusters, o the desired number of the output biclusters, and f the control parameter for overlaps among to-be-identified biclusters.

3.14. CPB Algorithm

Bozdağ et al. [49] introduced a two-step biclustering algorithm, which is known as CPB (Correlated Pattern Biclusters) algorithm. Algorithm based on using Pearson Correlation Coefficient (PCC) with a given gene reference to detected highly correlated biclusters. Algorithm able to detect overlapping biclusters and can detect a negative correlation by using PCC. Bozdağ et al. [49] also in their work, were able to detect biclusters, they also proposed a method to extract correlation information from identified biclusters in an intuitive way. This method will evaluate the uniqueness of the information that has been captured in each bicluster and computes a correlation score for each gene based on how frequently and in how distinct biclusters it co-occurs with the reference gene. In addition, the correlation scores from all datasets are combined to filter out inconsistent information. This approach is illustrated in the following Figure 3.12:

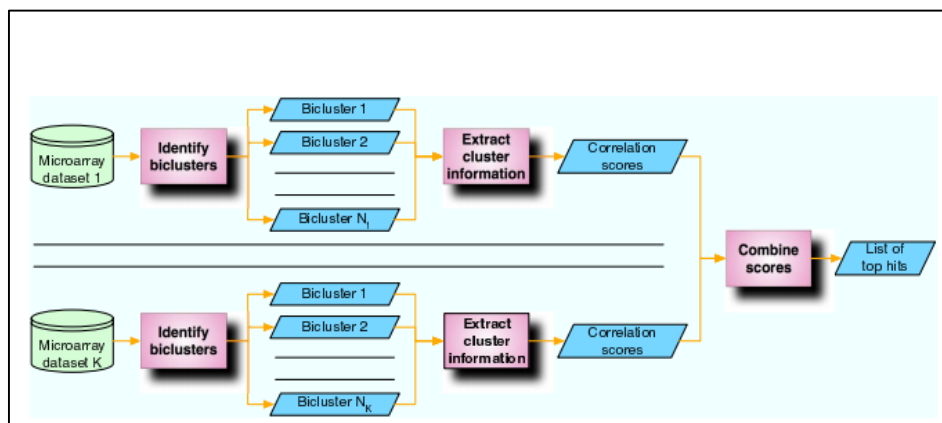


Figure 3.33. Source [49]: Overview of CPB algorithm

Bozdağ et al. [49] addressed two challenges to use the PCC as follow: The first one is the PCC lacks transitivity property. To overcome this problem the proposed instead of measure the closeness to a reference pattern, which requires computing the reference pattern rows (genes) in the same bicluster to measure quality. Therefore, they showed if two rows have a sufficiently high correlation with a reference pattern, there is a lower bound for PCC between each of these two rows. The second challenge is PCC has good results if he used to measure coherence between rows only, but if it is used to measure coherence to columns

simultaneously will not give meaningful results. Therefore, CPB were developed to add columns to the biclusters just if it does not decrease correlation among the rows in the bicluster. Algorithm will compute the effect of adding a column to a bicluster, and the column will be mapped to real numbers and work to capture the change tendency of a gene expression in the bicluster. Then, by computing root mean squared error (RMSE) for each column to evaluate the fit of the column to this tendency pattern.

The complete algorithm

Let A be a data matrix with a subset of rows R and a subset set of columns C . For this matrix, the value a_{rc} represents the relation between row r and column c . CPB algorithm will use PCC to find whether a row belongs to a bicluster $B(X, Y)$ or not, where X is a subset of rows and Y is a subset of columns. $pcc(r, s, Y)$ is the absolute value of PCC between row r and s under the column Y . A row r will be included in a bicluster if it has bigger that user-defined threshold $pcc(r, x_i, Y)$ for all $x_i \in X$. The size of Y also will be controlled to avoid obtain large PCC values merely by chance. The full CPB algorithm is outlined as follows:

Input: A : data matrix, r_r : reference row, ρ : PCC threshold, γ : minimum number of columns.

Output: $B(X, Y)$: biclusters where $r_r \in X, m \geq \gamma, pcc(x_i, x_j, Y) \geq \rho$ for all $x_i, x_j \in X$

Function $CPB(A, r_r, w, \gamma, \rho')$

1- $B(X, Y)$ where $X = \{r_r\}$ and Y is a random subset of columns of A .

2- $\rho'_c \leftarrow \frac{2}{3} \rho'; \rho'_\Delta = \frac{1}{12} \rho'; \gamma_c = m; \gamma_\Delta = \frac{m - \gamma}{4}$

3- repeat

$step \leftarrow 0$

repeat

$step \leftarrow step + 1; B_{save} \leftarrow B$

Compute reference vector T and normalization parameters

if $step \bmod 2 = 1$ then

Update X such that $pocc(x_i, T, Y) > \rho'_c$ for all $x_i \in X$

else

Let r be the row with smallest $pocc(r, T, Y) > \rho'_c$

Update Y $\lim_{x \rightarrow \infty}$ such that $RMSE(y_k) > RMSE(r)$ for all $y_k \in Y$

Until $step > 20$ or $B = B_{save}$

$\rho'_c \leftarrow \rho'_c + \rho'_\Delta; \gamma_c \leftarrow \gamma_c - \gamma_\Delta$

until $\rho'_c > \rho'$

return $B = (X, Y)$

Figure 3.34. CPB Algorithm

The CPB algorithm will start with an initial bicluster $B = (X, Y)$ and work to improve it by adding or taking off rows and columns iteratively. This step is done by comparing PCC between each row and a reference vector T , which is the general tendency of rows in X with respect to the columns in the bicluster while deciding which rows to move. If row r has $pcc(r, T, Y)$ value bigger than a certain threshold, it will be included in the set X and T will be updated. In each iteration of the algorithm, the reference vector T and parameters

elated to normalization of data values will be computed, then either we update X or Y not both in the same time to avoid large fluctuations in the bicluster structure.

The idea of the normalization is to detect a tendency of rows in X in a better way to take into account the different scaling and shifting patterns of rows in the bicluster. The normalized data values in the data matrix are computed as follow:

$$\hat{a}_{x_i y_k} = \frac{a_{x_i y_k} - \alpha_{x_i}}{\beta_{x_i}} \quad \text{for each } x_i \in X \text{ and } y_k \in Y \quad (3.68)$$

where α_{x_i} is the shifting parameter and β_{x_i} is the scaling parameter associated with row x_i . Then each element in T is computed as the arithmetic mean of $\hat{a}_{x_i y_k}$ on all rows $x_i \in X$. The values of T , α_{x_i} and β_{x_i} is computed in an iterative process. The computation process starts with $\alpha_{x_i} = 0$ and $\beta_{x_i} = 1$ as starting values and T will be computed. In order to obtain the best shifting and scaling parameters, which maximize alignment of each row x_i with the reference vector T , the least squares fitting, is being applied on the pairs $\{(t_1, \alpha_{x_i y_1}), \dots, (t_m, \alpha_{x_i y_m})\}$. Then the intercept and slope that obtained in the least squares fitting are assigned to α_{x_i} and β_{x_i} , respectively. Then by using the new values T will be updated and repeat the process until convergence.

As we mentioned before a row r is a member of X if $pcc(r, x_i, Y) > \rho$ for all the rows in X . In order to not testing all the conditions against all of the rows in X , Bozdağ et al. [49] proposed of using ρ' instead of ρ where ρ' is selected such that $pcc(r, T, Y) > \rho'$ must ensure $pcc(r, x_i, Y) > \rho$ for all $x_i \in X$. However, it is very difficult to analytically compute a lower bound for ρ' as a function of ρ because PCC originally is depending on the values and the length of the vectors. To overcome this problem, they generated a reference random vector e with e elements. In addition, they generated more random vectors. Form those random vectors the kept vectors that have an absolute value of PCC with the reference vector greater than ρ' . Finally, they plotted the distribution for the absolute value of PCC between

each pair of these vectors. This experiment verified there is a lower bound for ρ' and increases with ρ as presented in the following Figure:

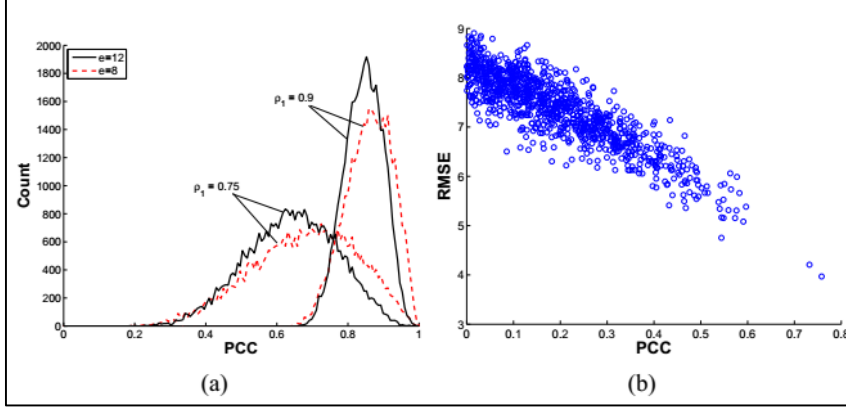


Figure 3.35. Source [49]: (a) Distribution of PCC between pairs of 200 random vectors with e elements that have PCC with reference vector greater than a threshold ρ_1 . (b) Relationship between PCC and RMSE on random vectors.

The value of the root means squared error for a column y_k in the subset Y is computed as follow:

$$RMSE(y_k) = \sqrt{\frac{1}{n} \sum_{i=1}^n (\hat{a}_{x_i y_k} - t_k)^2} \quad (3.69)$$

In addition, if a column c does not belong to Y , RMSE is computed using the same formulation by using t_c instead of t_k where t_c analogous to t_k that quantifies tendency of rows $x_i \in X$ in column c . To include a column to a bicluster it must have an RMSE value below a threshold ε . The threshold ε is controlling the ratio of the number of rows to the number of columns and its value in relation to ρ' . Bozdağ et al. [49] proposed to set the value of RMSE for row r which has the smallest $pcc(r, T, Y)$. RMSE for a row $x_i \in X$ is:

$$RMSE(x_i) = \sqrt{\frac{1}{m} \sum_{k=1}^m (\hat{a}_{x_i y_k} - t_k)^2} \quad (3.70)$$

The biclusters that obtained using CPB algorithm have the ratio n/m , which will be close to N/M . However, in order to have a different ratio k parameter can be used to control this ratio. That will be done by updating the number of columns Y , k times.

The CPB algorithm also uses the parameter ω to make the reference row r_r has a larger impact on decision mechanisms of the algorithm. That is done by assigning a larger weight to the reference row when computing the vector T and RMSE values. Then the total gain from rows except the row r_r will be multiplied by $(1 - \omega)$. In addition, the contribution from r_r is multiplied by ω , where ω is an input parameter. The value of ω must be carefully selected because the high value of it allows discovering patterns that more closely resemble r_r . On other hand, small values will increase the sensitivity.

As we mentioned before Bozdağ et al. [49] introduced the CPB algorithm as a two-step algorithm. The first algorithm that introduced work to detect biclusters from the data. The second step work to evaluate the founded biclusters using correlation information from the biclusters that obtained from different datasets as it presented in Figure 3.12 to obtain the best-corresponded information from multi data sets as follows:

This stage includes three steps: In the first one the uniqueness of the information that captured by each bicluster will be quantified, then correlation score will be computed for each row based on co-occurrence frequency and uniqueness information associated with the row with respect to the reference row. Finally, correlation scores from different datasets will be combined.

When two biclusters $B_v = (X_v, Y_v)$ and $B_w = (X_w, Y_w)$ have no common elements expect of a reference row r_r the biclusters represent two distinct relationships between rows and columns of the data matrix. In addition, if the bicluster B_w , for example, is contained in the B_v bicluster, B_w will be discarded from the results list.

Let $IR(...)$ be the set of biclusters that contain all rows specified in the argument list, and $IC(...)$ be the set of biclusters that contain all columns specified in the argument list. Let a row $r \in X_v$ and a column $c \in Y_v$ in the bicluster $B_v = (X_v, Y_v)$.

3.14. Definition: The bicluster uniqueness for a bicluster B_v with respect to other biclusters in set $IR(r) \cap IC(c)$ on the relationship between r and c is:

$$BU(B_v, r, c) = \frac{\sum_{x_i \in X_v - \{r_r\}} \sum_{y_k \in Y_v} \frac{1}{|IR(r, x_i) \cap IC(c, y_k)|}}{(|X_v| - 1) |Y_v|} \quad (3.71)$$

The value of BU is between $1/|IR(r) \cap IC(c)|$ and 1. Bicluster uniqueness will have 1 as a value for the bicluster B_v if it does not overlap with any biclusters at row r and column c . That means $1/|IR(r, x_i) \cap IC(c, y_k)|$ contains just B_v for all $x_i \in X$ and $y_k \in Y$. In addition, it will take the minimum value when it is completely overlapped.

Uniqueness measure will be used to compute an overlap score $OS(r, c)$. Overlap score will be computed for every row-column pair (r, c) , which used to quantify the amount of different relationships identified between r and c . $OS(r, c)$ is computed by summing the uniqueness measure for all biclusters in $IR(r) \cap IC(c)$:

$$OS(r, c) = \sum_{B_v \in IR(r) \cap IC(c)} BU(B_v, r, c) \quad (3.72)$$

In the next step, for each row, a correlation score $CS(r)$ will be computed, which provides us gathers total evidence on how frequently and in how distinct relationships row r is correlated with the reference row r_r , as follow:

$$CS(r) = \sum_{c \in C} OS(r, c) \quad (3.73)$$

The previous steps help to find more significance and meaningful results by applying this method to different datasets separately and combine correlation scores. The data sets should have the same row labels, which may not be practically possible [49].

4. COMPARISON OF THE BICLUSTERING ALGORITHMS

In this section, a two-stage comparison method will be applied to compare some of the biclustering algorithms. In the first stage, biclustering algorithms will be evaluated using DEA according to some measures. Using DEA helps on choosing the best parameters for the different algorithms and rank them according to some conditions. In the second stage, the results from the first stage will be used to make an order for the different biclustering methods from different sides.

In this section, CC, FLOC, the Plaid Model, xMotif, Bimax, and Qubic algorithms [10, 15, 28, 35-37] (APPENDIX-1 and APPENDIX-2 to see the used packages) will be used. In addition, the benchmark Yeast *Saccharomyces Cerevisiae* cell cycle expression dataset [34] will be used. The data contains 2884 rows (genes) and 17 columns (condition). The original data values have been transformed by scaling and logarithm $x \rightarrow 100\log(10^5 x)$ so the result was a matrix of integers in range between -1 and 595. The matrix contains values with -1 for the missing values and zeros that indicates there is no change under some conditions. However, in this study, we included -1 and the zeros as a normal number in the matrix.

4.1. First Stage: Applying Data Envelopment Analysis

4.1.1. Data envelopment analysis (DEA)

DEA is a mathematical programming method for measuring the relative efficiency of decision-making units (DMUs) with multiple outputs and multiple inputs, which was introduced by Charnes, Cooper, and Rhodes (CCR) [56].

The proposed CCR model assumes constant returns to scale of production technology. In other words, it assumes that all evaluated Decision Making Units (DMUs) are at the optimal production scale stage. However, many production units are not in a state of optimal scale of production, so it is concluded that the technical efficiency of CCR model contains the component of the scale efficiency. Therefore a variable returns to scale based model, which is known BCC s by Banker et al. [24] will be used in evaluating the biclustering result.

The BCC model makes it possible to investigate whether the performance of each DMU was conducted in the region of increasing, constant or decreasing returns to scale in multiple outputs and multiple inputs situations. The output-oriented and the input-oriented BCC model is as follow [23]:

$$\begin{aligned}
 & \max \eta \\
 & \text{subject to :} \\
 & \sum_{j=1}^n x_{ij} \lambda_j \leq x_{i0}, \quad i = 1, 2, \dots, p \\
 & \sum_{j=1}^n y_{rj} \lambda_j \geq \eta y_{r0}, \quad r = 1, 2, \dots, q \\
 & \sum_{k=1}^n \lambda_k = 1 \\
 & \lambda_k \geq 0, \quad k = 1, 2, \dots, n
 \end{aligned} \tag{4.1}$$

where $x_i = (x_{i1}, x_{i2}, \dots, x_{ip})$ is the input vector of DMU_i , $i = 1, 2, \dots, n$ and $x_0 = (x_{01}, x_{02}, \dots, x_{0p})$ is the inputs vector of the target DMU_0 . In the same way, $y_i = (y_{i1}, y_{i2}, \dots, y_{iq})$ is the output vector of DMU_i , $i = 1, 2, \dots, n$ and $y_0 = (y_{01}, y_{02}, \dots, y_{0q})$ is the output vector of the target DMU_0 . The output-oriented BCC model will be used in comparing the biclustering algorithms. We will be looking for maximization of the outputs from the different algorithms by applying each algorithm many times and using the model to look to the best outputs according to some outputs. The output-oriented model will show us which parameters are the best to have maximum outputs but will not show the order of them. So another important DEA technique will be used to make this order. The second technique is known as super-efficiency DEA (see [57]). The difference between the output-oriented BCC and the super-efficiency is that the DMU_0 under evaluation is excluded from the reference set (In our case algorithms which showed the best performance according to their parameters). In other words, algorithms with the best performances that obtained using DEA will be compared, which help us to make order according to their performances.

Each algorithm's parameters were chosen as inputs for the DEA model. These parameters were chosen according to the data type with respect to the previous works with the chosen data. In addition, we worked to cover a good of the possible parameters for each algorithm. The outputs were chosen as follow:

1. The first used output was the size of the biggest 5 biclusters (the sum of number of rows \times number of columns for each bicluster).
2. The number of the detected biclusters from the dataset.
3. The stratification score TS_k .
4. The signVariance.

For the first output we are interested in finding the biggest biclusters in our work, so we have included this variable as an output variable for the biggest 5 biclusters and more number can be included. In the same way, the more is the number of the founded biclusters are bigger that means we have better biclusters because it will contain data are more connected to each other.

The stratification score TS_k (APPENDIX-3 to see the used package) [12] is a measure that can be used to classify the biclusters in 3 types:

The large positive value indicates that the bicluster has strong gene effect only, the large negative score indicates strong condition effect only, and when the score is small and close to zero the means that we have a strong gene as well as condition effects. In this work, we are interested in the third one, which means we are looking for the smallest values in this score without making attention for the sign. Our model is an output-oriented model, which is interested, in the maximum value in the output so the following transformation is being applied:

$$TS1_k = \max(TS_k(\text{values})) - TS_k \quad (4.2)$$

The stratification score is computed as follow:

$$TS_k(b) = \begin{cases} \text{LOG}\left(\frac{T_k(b) + a}{B_k(b) + a}\right) \\ 2 \end{cases} \quad \text{where } k=1 \quad \text{if } SB(b) > 0 \\ \quad \quad \quad \text{if } SB(b) < 0 \end{cases} \quad (4.3)$$

Where b is a number of the bicluster and $0 < a < 1$ is a small fudge factor to offset large ratios based on very small co-expression in both groups of a bicluster. $T_k(b)$, $B_k(b)$ quantify

the T-type (strong gene only effects) and B-type (strong condition only effects) co-expression of genes in a bicluster b , and computed as follows:

$$T_k(b) = \frac{1}{I(b)} \sum_{i=1, i \in b}^{I(b)} \hat{\tau}_{ik}^2(b) - \frac{E_k(b)}{J_k(b)} \quad (4.4)$$

$$B_k(b) = \frac{1}{J_k(b)} \sum_{j=1, j \in b}^{J_k(b)} \hat{\beta}_{jk}^2(b) - \frac{E_k(b)}{I(b)} \quad (4.5)$$

for $k = 1$ and 2 , $I(b)$ is the number of genes in b and $J_k(b)$ is the number of conditions in G_k groups (two groups the first one contain the conditions that included in the b bicluster and the second for ones that were not included) for b . $\hat{\tau}_{ik}$ and $\hat{\beta}_{jk}$ are the estimates values for the effect of genes g_i in G_k and the effect of condition p_{jk} in G_k , respectively. Finally, $SB(b)$ is the differential co-expression score for bicluster b . Strong positive $SB(b)$ indicates strong co-expression in G_1 and weaker or no co-expression in G_2 vice versa, and is computed as follow:

$$SB(b) = LOG \left(\frac{\max(T_1(b) + a, B_1(b) + a)}{\max(T_2(b) + a, B_2(b) + a)} \right) \quad (4.6)$$

The last variable in the outputs is known as *signVariance* (APPENDIX-3 to see the used package)[3], which was introduced by Rodrigo Santamaria and was built based on work of Madeira and Oliveira [9] for classifying biclusters. The variance is computed as follow taken into account a sign transformation will be used whether there is a change in values without looking at the real values and using the following equation:

$$signVariance = \frac{1}{nrow(x) * (nrow(x) - 1)} \sum_{i=1}^{nrow(x)} \sum_{j=1}^{nrow(x)} \left(\frac{1}{ncol(x)} \sum_{k=1}^{ncol(x)} (x[i, k] - x[j, k])^2 \right) \quad (4.7)$$

where are interested making *signVariance* small as we can also so we will find the maximum value and subtract all of the *signVariance* value from the maximum value and

compute the average value for the biggest detected biclusters and work to find the smallest *signVariance* with respect to all of the other outputs.

4.1.2. Results

δ -Biclusters (CC) algorithm

Cheng and Church (CC) algorithm was applied to the dataset 54 times with the following parameters:

1. The residual of the data is equal to 1109.568 in most works is being set to 300. The controlling variable in the CC algorithm is a delta. Before applying CC algorithm is being standardized and delta set to have values equal to $t * residual$ where $t \in [0.1, 0.3, \dots, 0.9]$.
2. The second parameters are the scaling factor α . The all possible values were scanned and the ones that work with the data is chosen. These values were $\{1, 1.3, 1.5, 1.7, 1.9, 2\}$.

Note: For CC algorithm and the other algorithms, most of the parameters can take many values different from the chosen ones. In most of the previous works, they took the same parameters that come as default with the programs, which is used for applying the biclustering algorithms. So other parameters can be chosen and other outputs variables can be taken according to the nature of the data and the field study. The following results tables will have a column named algorithm, which will have the algorithm name and the used parameters. In addition, the bold font will be used to show the algorithms, which showed the best performances using the output-oriented BCC model according to the chosen variables. CC algorithm was applied with different parameters, then both of the output-oriented BCC models, super-efficiency is being applied, and the results are in the following table:

Table 4.1. DEA results for CC algorithm

Algorithm	Score	Rank	Algorithm	Score	Rank
CCD0.008A1	1.0000	1	CCD0.039A1.7	1.0000	5
CCD0.008A1.3	1.0349	26	CCD0.039A1.9	1.1058	45
CCD0.008A1.5	1.0000	4	CCD0.039A2	1.0436	28
CCD0.008A1.7	1.0000	8	CCD0.047A1	1.0000	12
CCD0.008A1.9	1.0000	13	CCD0.047A1.3	1.1143	47
CCD0.008A2	1.0000	17	CCD0.047A1.5	1.0314	25
CCD0.016A1	1.0160	21	CCD0.047A1.7	1.1272	50
CCD0.016A1.3	1.0474	30	CCD0.047A1.9	1.0842	36
CCD0.016A1.5	1.0467	29	CCD0.047A2	1.0093	20
CCD0.016A1.7	1.0000	3	CCD0.054A1	1.1444	52
CCD0.016A1.9	1.0009	19	CCD0.054A1.3	1.1322	51
CCD0.016A2	1.0000	14	CCD0.054A1.5	1.0957	42
CCD0.023A1	1.0000	2	CCD0.054A1.7	1.0000	10
CCD0.023A1.3	1.0000	16	CCD0.054A1.9	1.1634	53
CCD0.023A1.5	1.1063	46	CCD0.054A2	1.0848	37
CCD0.023A1.7	1.0572	33	CCD0.062A1	1.0574	34
CCD0.023A1.9	1.1220	49	CCD0.062A1.3	1.0736	35
CCD0.023A2	1.1188	48	CCD0.062A1.5	1.0917	40
CCD0.031A1	1.0187	22	CCD0.062A1.7	1.0520	31
CCD0.031A1.3	1.0986	43	CCD0.062A1.9	1.0000	18
CCD0.031A1.5	1.0000	7	CCD0.062A2	1.0000	9
CCD0.031A1.7	1.0521	32	CCD0.07A1	1.0288	24
CCD0.031A1.9	1.0000	15	CCD0.07A1.3	1.2080	54
CCD0.031A2	1.0881	38	CCD0.07A1.5	1.0885	39
CCD0.039A1	1.0000	11	CCD0.07A1.7	1.1007	44
CCD0.039A1.3	1.0000	6	CCD0.07A1.9	1.0379	27
CCD0.039A1.5	1.0200	23	CCD0.07A2	1.0946	41

The first column, which includes the names of algorithms, include algorithm name and the used parameters. For example, CCD0.039A1.5 means CC algorithm with delta value equal to 0.039 and α value equal to 1.5.

FLOC algorithm

FLOC algorithm was applied 62 times with the following parameters:

1. The residua threshold r was chosen as the same as the CC algorithm.
2. Gene initial probability p_{Gene} was given three values 0.3, 0.6, 0.9.

3. Sample initial probability was given three values 0.2, 0.5, 0.8.
4. The number of the biclusters founded in the results is not included for the analysis.
5. The number of iterations was set to 225

FLOC algorithm was applied with different parameters then both of the output-oriented BCC models, super-efficiency is being applied, and the results are in the following table (the name of the algorithms contains both FLOC word and the used parameters):

Table 4.2. DEA results for FLOC algorithm

Algorithm	Score	Rank	Algorithm	Score	Rank
FLOCR110.9568pG0.3pS0.2	1.0000	1	FLOCR332.8704pG0.6pS0.8	1.0038	39
FLOCR110.9568pG0.3pS0.5	1.0000	3	FLOCR332.8704pG0.9pS0.2	1.0000	19
FLOCR110.9568pG0.3pS0.8	1.0000	12	FLOCR332.8704pG0.9pS0.5	1.0027	38
FLOCR110.9568pG0.6pS0.5	1.0000	7	FLOCR332.8704pG0.9pS0.8	1.0016	32
FLOCR110.9568pG0.6pS0.8	1.0000	20	FLOCR443.8272pG0.3pS0.2	1.0000	27
FLOCR110.9568pG0.9pS0.2	1.0000	2	FLOCR443.8272pG0.3pS0.5	1.0103	56
FLOCR110.9568pG0.9pS0.5	1.0040	41	FLOCR443.8272pG0.3pS0.8	1.0000	17
FLOCR110.9568pG0.9pS0.8	1.0053	46	FLOCR443.8272pG0.6pS0.2	1.0048	43
FLOCR221.9136pG0.3pS0.5	1.0080	49	FLOCR443.8272pG0.6pS0.5	1.0100	54
FLOCR221.9136pG0.3pS0.8	1.0000	11	FLOCR443.8272pG0.6pS0.8	1.0025	35
FLOCR221.9136pG0.6pS0.2	1.0000	10	FLOCR443.8272pG0.9pS0.2	1.0021	33
FLOCR221.9136pG0.6pS0.5	1.0014	30	FLOCR443.8272pG0.9pS0.5	1.0000	15
FLOCR221.9136pG0.6pS0.8	1.0000	21	FLOCR443.8272pG0.9pS0.8	1.0071	48
FLOCR221.9136pG0.9pS0.2	1.0000	18	FLOCR554.784pG0.3pS0.2	1.0000	6
FLOCR221.9136pG0.9pS0.5	1.0025	34	FLOCR554.784pG0.3pS0.5	1.0130	61
FLOCR221.9136pG0.9pS0.8	1.0113	58	FLOCR554.784pG0.3pS0.8	1.0111	57
FLOCR300pG0.3pS0.2	1.0000	4	FLOCR554.784pG0.6pS0.2	1.0049	44
FLOCR300pG0.3pS0.5	1.0000	24	FLOCR554.784pG0.6pS0.5	1.0000	14
FLOCR300pG0.3pS0.8	1.0000	26	FLOCR554.784pG0.6pS0.8	1.0117	59
FLOCR300pG0.6pS0.2	1.0026	36	FLOCR554.784pG0.9pS0.2	1.0015	31
FLOCR300pG0.6pS0.5	1.0084	51	FLOCR554.784pG0.9pS0.5	1.0000	25
FLOCR300pG0.6pS0.8	1.0000	28	FLOCR554.784pG0.9pS0.8	1.0081	50
FLOCR300pG0.9pS0.2	1.0209	62	FLOCR665.7408pG0.3pS0.2	1.0000	5
FLOCR300pG0.9pS0.5	1.0091	52	FLOCR665.7408pG0.3pS0.5	1.0009	29
FLOCR300pG0.9pS0.8	1.0130	60	FLOCR665.7408pG0.3pS0.8	1.0000	8
FLOCR300pG0.5pS0.5	1.0050	45	FLOCR665.7408pG0.6pS0.2	1.0000	13
FLOCR332.8704pG0.3pS0.2	1.0000	22	FLOCR665.7408pG0.6pS0.5	1.0091	53
FLOCR332.8704pG0.3pS0.5	1.0000	23	FLOCR665.7408pG0.6pS0.8	1.0067	47
FLOCR332.8704pG0.3pS0.8	1.0000	16	FLOCR665.7408pG0.9pS0.2	1.0039	40
FLOCR332.8704pG0.6pS0.2	1.0026	37	FLOCR665.7408pG0.9pS0.5	1.0100	55
FLOCR332.8704pG0.6pS0.5	1.0042	42	FLOCR665.7408pG0.9pS0.8	1.0000	9

The Plaid Model

The plaid model algorithm was applied to the data 64 times. However, the plaid model algorithm gives different results with the same parameters every time so we have repeated it many times until finding 5 biclusters to include in the DEA analysis. The chosen parameters are:

1. The clusters parameters are set to ‘ b ’ which means to apply the clustering on both of rows and columns.
2. Fit model was set to default $fit.model = y \sim m + a + b$
3. Background was set to TRUE, which means that the method will consider that a background layer (constant for all rows and columns) is present in the data matrix.
4. *Row.release* has the values $\{0.5, 0.53, 0.57, 0.6, 0.63, 0.66, 0.68, 0.7\}$, which is used as a threshold to prune rows in the layers depending on row homogeneity.
5. *col.release* the same values in *Row.release* which is used as a threshold to prune columns in layers depending on column homogeneity.
6. The shuffle parameter is set to his default value equal to 3, which work as follow: Before a layer is added, its statistical significance is compared against a number of layers obtained by random defined by this parameter.
7. The *back.fit* parameter was set to zero.

The plaid model algorithm was applied with different parameters then both of the output-oriented BCC model, super-efficiency is being applied, and the results are in the following table (the name of the algorithms contains both of the plaid word and the used parameters):

Table 4.3. DEA results for Plaid Model algorithm

Algorithm	Score	Rank	Algorithm	Score	Rank
PlaidR0.5C0.5	1.0000	1	PlaidR0.63C0.5	1.0087	36
PlaidR0.5C0.53	1.0000	10	PlaidR0.63C0.53	1.0236	53
PlaidR0.5C0.57	1.0000	9	PlaidR0.63C0.57	1.0224	52
PlaidR0.5C0.6	1.0000	7	PlaidR0.63C0.6	1.0131	46
PlaidR0.5C0.63	1.0000	6	PlaidR0.63C0.63	1.0244	54
PlaidR0.5C0.66	1.0050	28	PlaidR0.63C0.66	1.0299	56
PlaidR0.5C0.68	1.0000	8	PlaidR0.63C0.68	1.0372	58
PlaidR0.5C0.7	1.0112	42	PlaidR0.63C0.7	1.0000	17
PlaidR0.53C0.5	1.0000	12	PlaidR0.66C0.5	1.0000	4
PlaidR0.53C0.53	1.0000	2	PlaidR0.66C0.53	1.0000	15
PlaidR0.53C0.57	1.0039	27	PlaidR0.66C0.57	1.0028	25
PlaidR0.53C0.6	1.0000	20	PlaidR0.66C0.6	1.0465	62
PlaidR0.53C0.63	1.0000	11	PlaidR0.66C0.63	1.0088	38
PlaidR0.53C0.66	1.0224	51	PlaidR0.66C0.66	1.0133	47
PlaidR0.53C0.68	1.0116	43	PlaidR0.66C0.68	1.0061	31
PlaidR0.53C0.7	1.0309	57	PlaidR0.66C0.7	1.0055	29
PlaidR0.57C0.5	1.0000	19	PlaidR0.68C0.5	1.0000	18
PlaidR0.57C0.53	1.0000	5	PlaidR0.68C0.53	1.0000	16
PlaidR0.57C0.57	1.0089	39	PlaidR0.68C0.57	1.0528	64
PlaidR0.57C0.6	1.0006	23	PlaidR0.68C0.6	1.0067	33
PlaidR0.57C0.63	1.0182	49	PlaidR0.68C0.63	1.0129	45
PlaidR0.57C0.66	1.0031	26	PlaidR0.68C0.66	1.0123	44
PlaidR0.57C0.68	1.0093	41	PlaidR0.68C0.68	1.0074	34
PlaidR0.57C0.7	1.006	30	PlaidR0.68C0.7	1.0000	22
PlaidR0.6C0.5	1.0000	3	PlaidR0.7C0.5	1.0000	13
PlaidR0.6C0.53	1.0000	14	PlaidR0.7C0.53	1.0167	48
PlaidR0.6C0.57	1.0089	40	PlaidR0.7C0.57	1.0063	32
PlaidR0.6C0.6	1.0082	35	PlaidR0.7C0.6	1.0024	24
PlaidR0.6C0.63	1.0088	37	PlaidR0.7C0.63	1.0393	60
PlaidR0.6C0.66	1.0000	21	PlaidR0.7C0.66	1.0217	50
PlaidR0.6C0.68	1.0385	59	PlaidR0.7C0.68	1.0512	63
PlaidR0.6C0.7	1.0290	55	PlaidR0.7C0.7	1.0430	61

xMotif algorithm

xMotif algorithm was applied to the data 60 times. In addition, the data works with the discrete matrix. The chosen parameters are:

1. n_s the number of chosen columns set to $\{2, 4, 6, 8, 10, 15\}$.
2. n_d the number of repetitions is 1000.

3. s_d the sample size in repetitions are $\{1, 2, 4, 5, 6, 8, 10, 12\}$.
4. The scaling factor α is set to $\{0.01, 0.03, 0.04, 0.05, 0.06, 0.08, 0.09, 0.1, 0.11, 0.12, 0.013, 0.014, 0.15, 0.17, 0.22, 0.23, 0.26, 0.29\}$

The xMotif algorithm was applied with different parameters then both of the output-oriented BCC model, super-efficiency is being applied, and the results are in the following table (the name of the algorithms contains both of the xMotif word and the used parameters):

Table 4.4. DEA results for xMotif algorithm

Algorithm	score	Rank	Algorithm	score	Rank
xMotifNs2Sd1A0.01	1.0000	1	xMotifNs2Sd2A0.15	1.0067	48
xMotifNs2Sd1A0.05	1.0000	9	xMotifNs4Sd1A0.03	1.0000	29
xMotifNs2Sd1A0.08	1.0000	27	xMotifNs4Sd2A0.04	1.0000	16
xMotifNs2Sd1A0.1	1.0000	18	xMotifNs4Sd2A0.09	1.0000	30
xMotifNs2Sd2A0.01	1.0000	2	xMotifNs6Sd2A0.04	1.0000	38
xMotifNs2Sd2A0.06	1.0000	36	xMotifNs6Sd2A0.014	1.0000	39
xMotifNs2Sd2A0.12	1.0000	7	xMotifNs6Sd4A0.05	1.0148	53
xMotifNs2Sd2A0.17	1.0000	24	xMotifNs6Sd4A0.12	1.0033	46
xMotifNs4Sd1A0.01	1.0000	15	xMotifNs6Sd4A0.23	1.0000	21
xMotifNs4Sd1A0.05	1.0000	37	xMotifNs8Sd4A0.05	1.0006	42
xMotifNs4Sd1A0.08	1.0000	10	xMotifNs8Sd4A0.14	1.0150	54
xMotifNs4Sd1A0.11	1.0000	22	xMotifNs8Sd4A0.22	1.0002	41
xMotifNs4Sd2A0.01	1.0227	56	xMotifNs8Sd4A0.26	1.0090	50
xMotifNs4Sd2A0.06	1.0000	32	xMotifNs8Sd6A0.01	1.0000	3
xMotifNs4Sd2A0.12	1.0000	34	xMotifNs8Sd6A0.05	1.0000	33
xMotifNs4Sd2A0.17	1.0121	52	xMotifNs8Sd6A0.09	1.0000	13
xMotifNs6Sd2A0.01	1.0000	20	xMotifNs10Sd5A0.01	1.008	49
xMotifNs6Sd2A0.06	1.0000	25	xMotifNs10Sd5A0.05	1.0000	17
xMotifNs6Sd2A0.12	1.0387	59	xMotifNs10Sd5A0.13	1.0000	11
xMotifNs6Sd2A0.17	1.0001	40	xMotifNs15Sd8A0.01	1.0000	5
xMotifNs6Sd4A0.01	1.0000	6	xMotifNs15Sd8A0.05	1.0092	51
xMotifNs6Sd4A0.08	1.0000	35	xMotifNs15Sd8A0.14	1.0000	28
xMotifNs6Sd4A0.17	1.0000	14	xMotifNs15Sd4A0.01	1.0000	23
xMotifNs6Sd4A0.29	1.0183	55	xMotifNs15Sd6A0.01	1.0000	12
xMotifNs8Sd4A0.01	1.0000	19	xMotifNs15Sd10A0.01	1.0000	4
xMotifNs8Sd4A0.08	1.0029	45	xMotifNs15Sd12A0.01	1.0427	60
xMotifNs8Sd4A0.17	1.0000	26	xMotifNs15Sd4A0.05	1.0044	47
xMotifNs8Sd4A0.29	1.0317	58	xMotifNs15Sd6A0.05	1.0026	44
xMotifNs2Sd2A0.04	1.0026	43	xMotifNs15Sd10A0.05	1.0307	57
xMotifNs2Sd2A0.09	1.0000	31	xMotifNs15Sd12A0.05	1.0000	8

Bimax algorithm

Bimax algorithm was applied to the data 54 times. It searches for submatrices of ones in a logical matrix. The parameters are:

1. *Binarize* threshold has the values $\{69,110,139,179,208,240,271,314,371\}$.
2. The minimum row size of the resulting bicluster sets to 2.
3. The minimum column size of the resulting biclusters sets to $\{2, 4, 7, 9,11,14\}$

The Bimax algorithm was applied with different parameters then both of the output-oriented BCC model, super-efficiency is being applied, and the results are in the following table (the name of the algorithms contains both of the Bimax word and the used parameters):

Table 4.5. DEA results for Bimax algorithm

Algorithm	Score	Rank	Algorithm	Score	Rank
BimaxR2C2B69	1.0000	1	BimaxR2C9B69	1.0000	5
BimaxR2C2B110	1.0004	37	BimaxR2C9B110	1.0017	49
BimaxR2C2B139	1.0000	4	BimaxR2C9B139	1.0000	20
BimaxR2C2B179	1.0000	17	BimaxR2C9B179	1.0000	24
BimaxR2C2B208	1.0000	22	BimaxR2C9B208	1.0003	36
BimaxR2C2B240	1.0000	6	BimaxR2C9B240	1.0008	41
BimaxR2C2B271	1.0011	44	BimaxR2C9B271	1.0026	53
BimaxR2C2B314	1.0000	12	BimaxR2C9B314	1.0001	31
BimaxR2C2B371	1.0000	7	BimaxR2C9B371	1.0000	21
BimaxR2C4B69	1.0000	8	BimaxR2C11B69	1.0000	3
BimaxR2C4B110	1.0002	33	BimaxR2C11B110	1.0009	43
BimaxR2C4B139	1.0008	40	BimaxR2C11B139	1.0012	45
BimaxR2C4B179	1.0000	10	BimaxR2C11B179	1.0042	54
BimaxR2C4B208	1.0006	39	BimaxR2C11B208	1.0000	19
BimaxR2C4B240	1.0000	11	BimaxR2C11B240	1.0000	29
BimaxR2C4B271	1.0004	38	BimaxR2C11B271	1.0023	51
BimaxR2C4B314	1.0000	23	BimaxR2C11B314	1.0001	32
BimaxR2C4B371	1.0000	27	BimaxR2C11B371	1.0000	30
BimaxR2C7B69	1.0017	48	BimaxR2C14B69	1.0000	2
BimaxR2C7B110	1.0009	42	BimaxR2C14B110	1.0000	25
BimaxR2C7B139	1.0003	34	BimaxR2C14B139	1.0000	13
BimaxR2C7B179	1.0000	28	BimaxR2C14B179	1.0000	9
BimaxR2C7B208	1.0003	35	BimaxR2C14B208	1.0024	52
BimaxR2C7B240	1.0022	50	BimaxR2C14B240	1.0000	14
BimaxR2C7B271	1.0014	47	BimaxR2C14B271	1.0013	46
BimaxR2C7B314	1.0000	15	BimaxR2C14B314	1.0000	26
BimaxR2C7B371	1.0000	18	BimaxR2C14B371	1.0000	16

Qubic Algorithm

Qubic algorithm was applied to the data 63 times with the following parameters:

1. Affect the granularity of the biclusters q . The percentage of the regulating conditions for each gene. The choice of q 's value depends on the specific application goals; that is if the goal is to find genes that are responsive to local regulators, we should use a relatively small q -value; otherwise we may want to consider larger q -value. The values are $\{0.01, 0.06, 0.1, 0.2, 0.3, 0.4, 0.49\}$.
2. Affect the granularity of the biclusters r . The range of possible ranks. A user can start with a small value of r (the default value is 1 so the corresponding data matrix consists of values '1', '-1' and '0'), evaluate the results, and then use larger values (should not be larger than half of the number of the columns) to look for fine structures within the identified biclusters. The values are $\{1, 4, 7\}$.
3. The minimum width of columns is set to $\{5, 6, 7\}$.
4. The rest of the parameters are set to their default values.

The Qubic algorithm was applied with different parameters then both of the output-oriented BCC model, super-efficiency is being applied, and the results are in the following table (the name of the algorithms contains both of the Qubic word and the used parameters):

Table 4.6. DEA results for Qubic algorithm

Algorithm	Score	Rank	Algorithm	Score	Rank
QubicQ0.01R1M5	1.0000	1	QubicQ0.2R4M7	1.0162	45
QubicQ0.01R1M6	1.0000	5	QubicQ0.2R7M5	1.0000	18
QubicQ0.01R1M7	1.0000	23	QubicQ0.2R7M6	1.0165	47
QubicQ0.01R4M5	1.0000	3	QubicQ0.2R7M7	1.0177	49
QubicQ0.01R4M6	1.1250	63	QubicQ0.3R1M5	1.0000	22
QubicQ0.01R4M7	1.0000	10	QubicQ0.3R1M6	1.0357	52
QubicQ0.01R7M5	1.0000	25	QubicQ0.3R1M7	1.0498	56
QubicQ0.01R7M6	1.0000	21	QubicQ0.3R4M5	1.0000	26
QubicQ0.01R7M7	1.0000	36	QubicQ0.3R4M6	1.0114	43
QubicQ0.06R1M5	1.0000	11	QubicQ0.3R4M7	1.0133	44
QubicQ0.06R1M6	1.0561	57	QubicQ0.3R7M5	1.0000	16
QubicQ0.06R1M7	1.0000	34	QubicQ0.3R7M6	1.0175	48
QubicQ0.06R4M5	1.0000	28	QubicQ0.3R7M7	1.0384	54
QubicQ0.06R4M6	1.0000	33	QubicQ0.4R1M5	1.0000	9
QubicQ0.06R4M7	1.0049	41	QubicQ0.4R1M6	1.0000	29
QubicQ0.06R7M5	1.0318	51	QubicQ0.4R1M7	1.0041	39
QubicQ0.06R7M6	1.0645	60	QubicQ0.4R4M5	1.0000	31
QubicQ0.06R7M7	1.0224	50	QubicQ0.4R4M6	1.0000	12
QubicQ0.1R1M5	1.0000	4	QubicQ0.4R4M7	1.0009	37
QubicQ0.1R1M6	1.0000	35	QubicQ0.4R7M5	1.0000	14
QubicQ0.1R1M7	1.0607	58	QubicQ0.4R7M6	1.0071	42
QubicQ0.1R4M5	1.0045	40	QubicQ0.4R7M7	1.0365	53
QubicQ0.1R4M6	1.0000	20	QubicQ0.49R1M5	1.0000	7
QubicQ0.1R4M7	1.0163	46	QubicQ0.49R1M6	1.0000	8
QubicQ0.1R7M5	1.0000	15	QubicQ0.49R1M7	1.0000	27
QubicQ0.1R7M6	1.0627	59	QubicQ0.49R4M5	1.0000	32
QubicQ0.1R7M7	1.0735	62	QubicQ0.49R4M6	1.0000	19
QubicQ0.2R1M5	1.0000	13	QubicQ0.49R4M7	1.0000	17
QubicQ0.2R1M6	1.0000	30	QubicQ0.49R7M5	1.0000	24
QubicQ0.2R1M7	1.0000	2	QubicQ0.49R7M6	1.0032	38
QubicQ0.2R4M5	1.0000	6	QubicQ0.49R7M7	1.066	61
QubicQ0.2R4M6	1.0396	55			

4.2. Second Stage: The Quality Comparison

In this stage, the comparison will be done using the results, which were obtained using the DEA method. Algorithms in this stage have been chosen according to their ranks from the best performance. The maximum number of the detected biclusters using each algorithm is being set to 20 at most. In this stage, an algorithm with different parameters according to their ranks will be applied to the dataset until 60 biclusters are obtained.

The original data matrix is presented in the following heatmap figure:

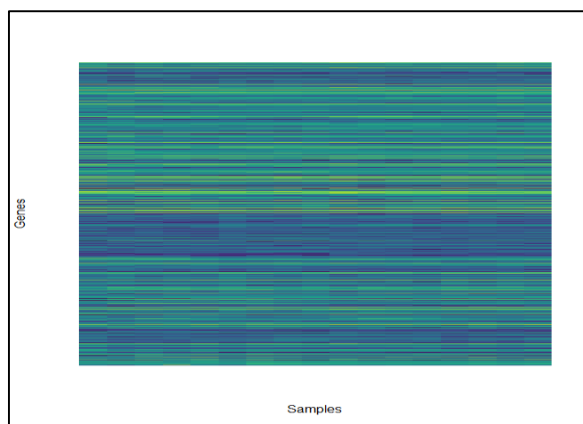


Figure 4.1. Heatmap figure for the original dataset

For CC algorithm, CCD0.008A1, CCD0.023A1 and CCD0.016A1.7 were chosen. The heatmap figures are presented in the following:

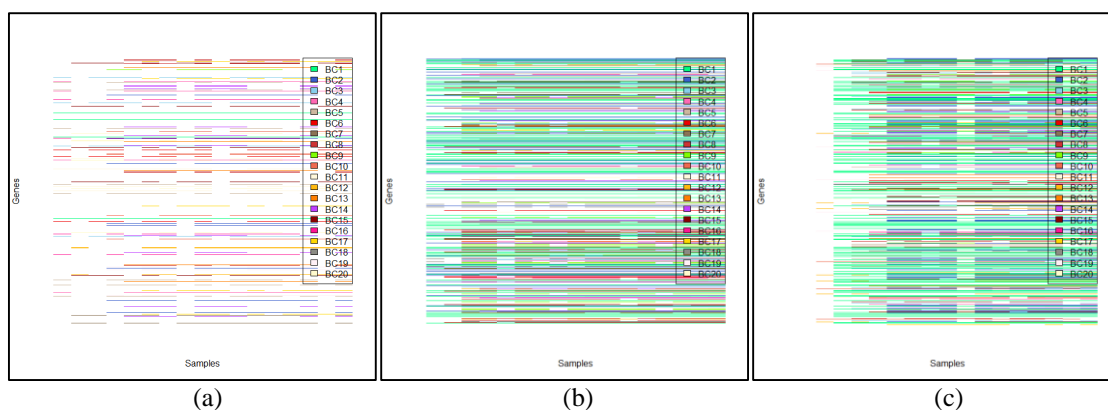


Figure 4.2. Heatmap figures for CC algorithm: (a) CCD0.008A1, (b) CCD0.023A1 and (c) CCD0.016A1.7

For FLOC algorithm, FLOC110.9568pG0.3pS0.2, FLOC110.9568pG0.9pS0.2, FLOC110.9568pG0.3pS0.5 and FLOC300pG0.3pS0.2 were chosen. The heatmap figures for FLOC algorithm with the chosen parameters are:

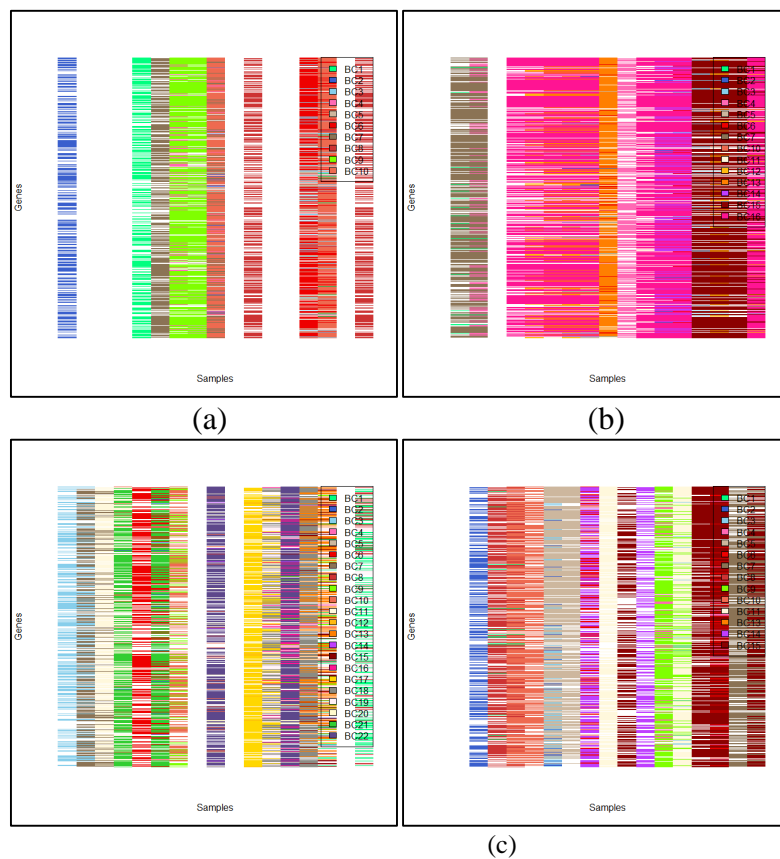


Figure 4.3. Heatmap figures for FLOC algorithm: (a) FLOC R110.9568pG0.3pS0.2, (b) FLOC R110.9568pG0.9pS0.2, (c) FLOC R110.9568pG0.3pS0.5 and (d) FLOC R300pG0.3pS0.2

For Plaid model algorithm, PlaidR0.5C0.5, PlaidR0.53C0.53, PlaidR0.6C0.5, PlaidR0.66C0.5 and PlaidR0.57C0.53 were chosen. The heatmap figures for Plaid Model algorithm with the chosen parameters are:

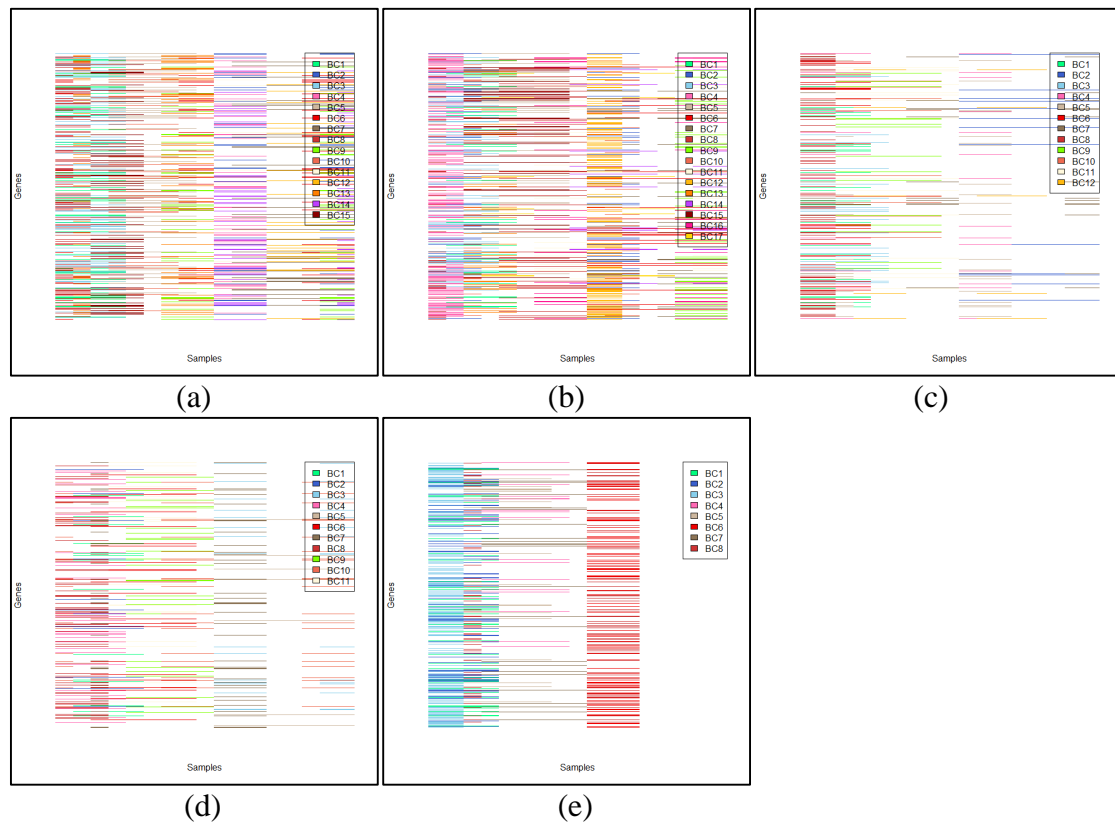


Figure 4.4. Heatmap figures for Plaid Model algorithm: (a) PlaidR0.5C0.5, (b) PlaidR0.53C0.53, (c) PlaidR0.6C0.5, (d) PlaidR0.66C0.5 and (e) PlaidR0.57C0.53

For xMotif algorithm, xMotifNs2Sd1A0.01, xMotifNs2Sd2A0.01, xMotifNs8Sd6A0.01 and xMotifNs15Sd10A0.01 were chosen. The heatmap figures for xMotif algorithm with the chosen parameters are:

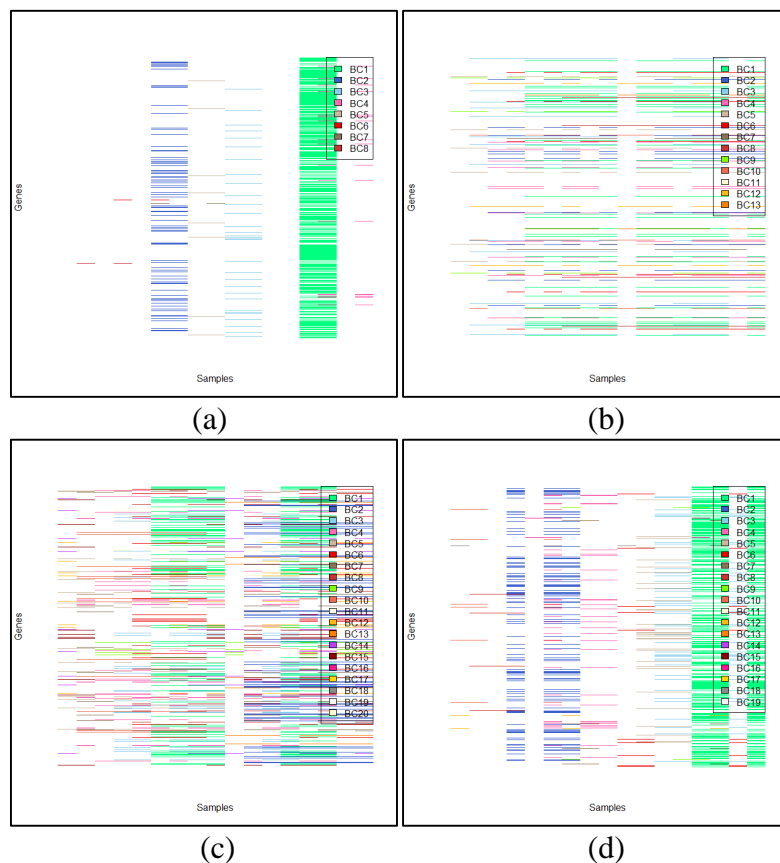


Figure 4.5. Heatmap figures for xMotif algorithm: (a) xMotifNs2Sd1A0.01, (b) xMotifNs2Sd2A0.01, (c) xMotifNs8Sd6A0.01 and (d) xMotifNs15Sd10A0.01

For the Bimax algorithm, BimaxR2C2B69, BimaxR2C14B69 and BimaxR2C11B69 were chosen. The heatmap figures for Bimax algorithm with the chosen parameters are:

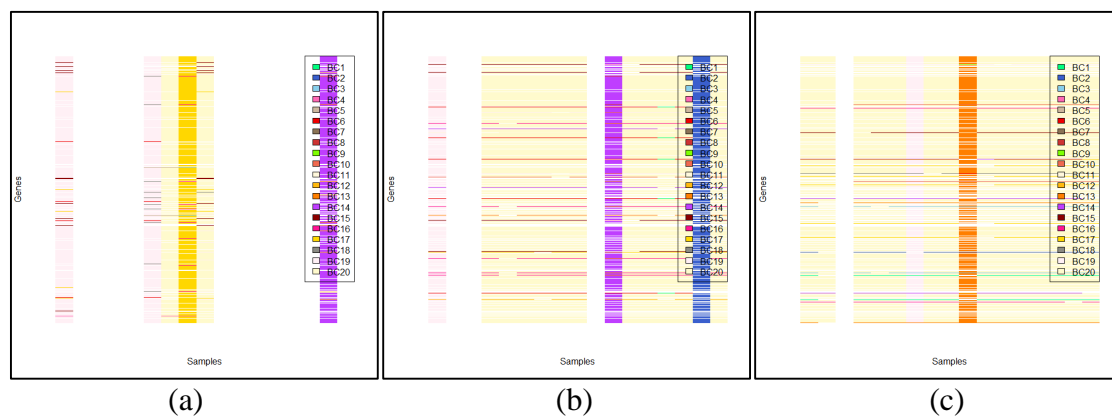


Figure 4.6. Heatmap figures for Bimax algorithm: (a) BimaxR2C2B69, (b) BimaxR2C14B69 and (c) BimaxR2C11B69

For Qubic algorithm, QubicQ0.01R1M5, QubicQ0.2R1M7, QubicQ0.01R4M5, and QubicQ0.1R1M5 were chosen. The heatmap figures for Qubic algorithm with the chosen parameters are:

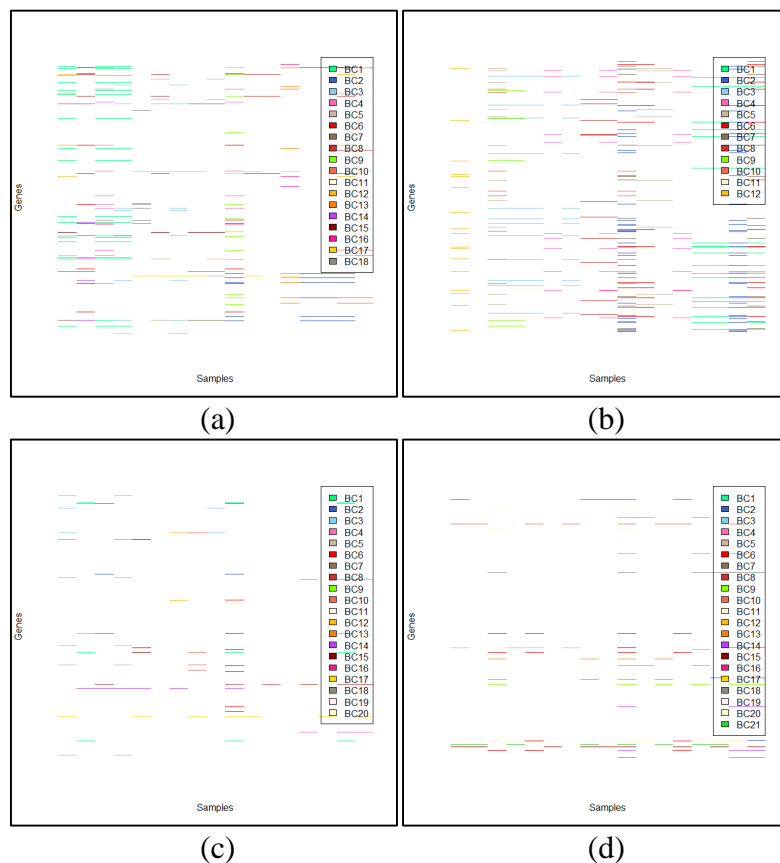


Figure 4.7. Heatmap figures for Qubic algorithm: (a) QubicQ0.01R1M5, (b) QubicQ0.2R1M7, (c) QubicQ0.01R4M5 and (d) QubicQ0.1R1M5

For each of the previous algorithms with the chosen parameters, the comparison will be according to the numbers of the rows, the numbers or the columns, the whole size and the variances of the detect bicluster. In this study, the results for each algorithm with different parameters are considered as one algorithm. For example, CCD0.008A1, CCD0.023A1, and CCD0.016A1.7 will be considered as CC algorithm and so on for the rest.

The numbers of rows, the numbers of the columns and the whole sizes for CC, FLOC, Plaid, xMotif, Bimax and Qubic algorithm are presented in the following boxplot graphics:

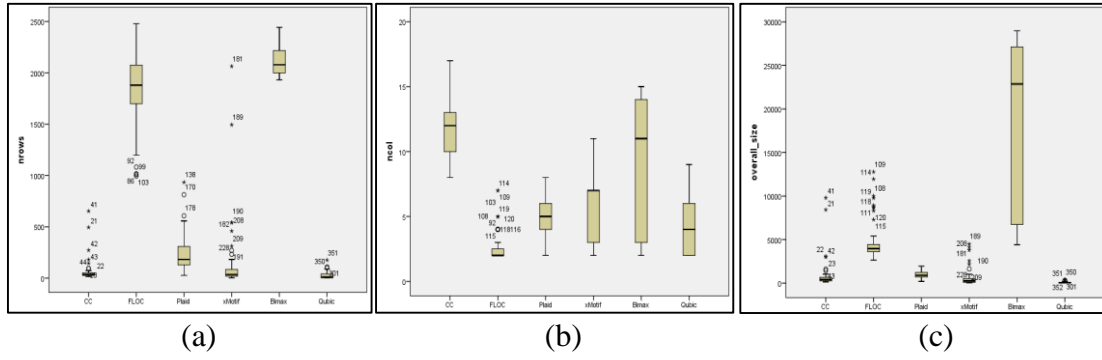


Figure 4.8. The sizes of the detected biclusters: (a) the number of rows, (b) the number of the columns and (c) the whole size for the detected biclusters

Figure 4.8(a) is presenting the numbers of rows for the detected biclusters with the different used algorithms using the boxplot graphic. The graphic shows that the FLOC and the Bimax algorithms have the biggest numbers of the rows for the detect biclusters. Figure 4.8(b) is presenting the numbers of columns for the detected biclusters, which also shows the same similar numbers for the columns in the used dataset. Finally, the figure 4.8(c) is representing the sizes of the detected biclusters (number of rows \times number of columns) and we can see that the Bimax algorithm has the biggest biclusters.

Besides the sizes, other important measures were chosen. The measures are known as constance and coherence measures (APPENDIX-3 to see the used package), which were introduced by Rodrigo Santamaria [3]. These measures are computed for each bicluster separately. There are 4 types of the measures, which have been built based on Madeira and Oliveira [9]. These types are constant-variance, additive-variance, multiplicative-variance, and sign-variance. These measures are included in the ‘biclust’ package [58]. An important parameter for using these measures is the dimension of the variance parameter, which can take row, column or both values. In this study, ‘both’ were chosen.

The constant-variance is a function that returns the corresponding variance of rows as the average of the sum of Euclidean distances between all rows of the bicluster x :

$$\frac{1}{nrow(x) * (nrow(x) - 1)} \sum_{i=1}^{nrow(x)} \sum_{j=1}^{nrow(x)} \left(\frac{1}{ncol(x)} \sum_{k=1}^{ncol(x)} (x[i, k] - x[j, k])^2 \right) \quad (4.8)$$

In addition, the additive-variance, the multiplicative-variance, and the sign-variance can be computed using the equation 4.8. However, the computation is done by applying additive, a multiplicative or a sign transformation to the bicluster.

Using the 4 types of the variance helps in deciding which algorithm is better according to the 4 types. In addition, a value equal to 0 means the bicluster is ideally constant or coherent and value that bigger than 1.5 determine the bicluster is not coherent or constant.

The constant-variance, the additive-variance, the multiplicative-variance and the sign-variance for CC, FLOC, Plaid, xMotif, Bimax and Qubic algorithm are presented in the following boxplot graphics:

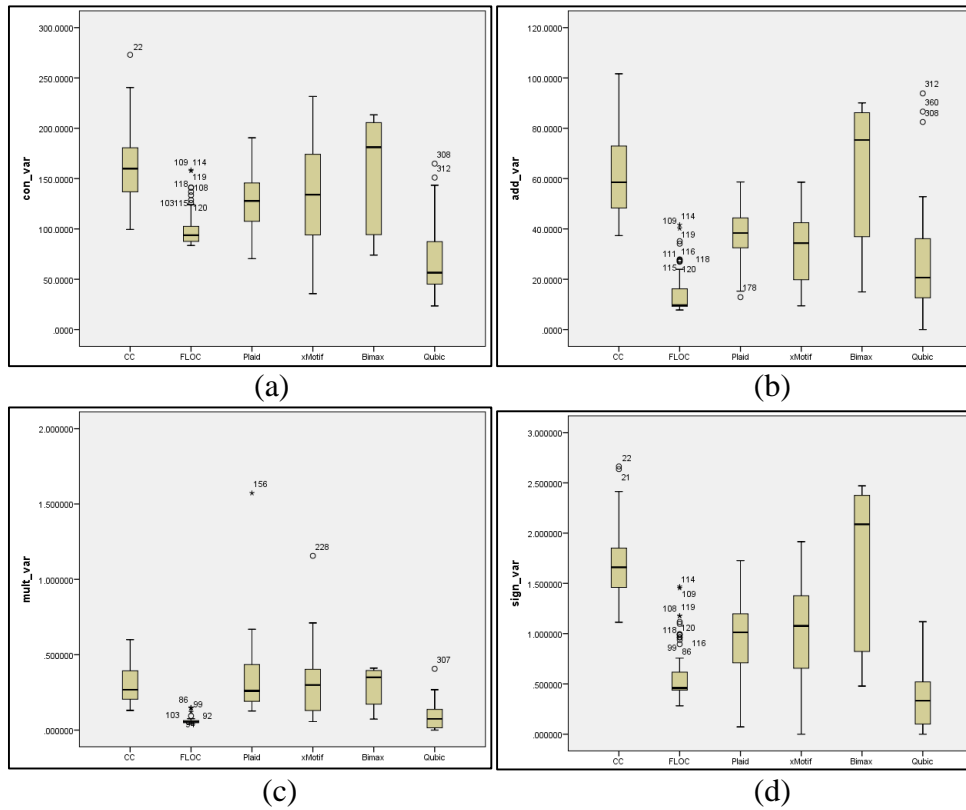


Figure 4.9. The boxplot figures for the different variances types for the detected biclusters: (a) Constant-Variance, (b) Additive-Variance, (c) Multiplicative-Variance and (d) Sign-Variance

Figure 4.9 as presented above summarized the values for each of the constant-variance (figure 4.9(a)), additive-variance (figure 4.9(b)), multiplicative-variance (figure 4.9(c)) and sign-variance (figure 4.10 (d)), which will be compared using the Kruskal-Wallis (KW) test [59] in the following.

The next step is to compare each of the previous measures. The nonparametric method KW test is chosen for these comparisons. This method is an alternative method of the One-Way Analysis of Variance (ANOVA). The propose of choosing the nonparametric method that the data was tested using Kolmogorov–Smirnov test for normality and none of the variables have a normal distribution as presented in Table 4.7. In addition, as we can see in the boxplot figures we have some outliers. The outliers were not deleted because we are interested in all of the values. KW test compares the mean ranks for each group, which give good results with datasets that have outliers. The KW test will tell us whether there is a difference between the groups but will not show where the differences are. So if the KW shows that there are statistically significant differences, the Dunn’s post hoc test [60] will be used in pairs to detect the differences. The Dunn’s test uses the Bonferroni adjustment, which uses adjustment p-value by multiplying the value of the computed p-value by the total number of the tests. This step will reduce the amount of the error for the pairwise comparison tests.

Table 4.7. Test of normality

	Kolmogorov-Smirnov			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
number of rows	.271	223	.000	.754	223	.000
number of columns	.215	223	.000	.854	223	.000
whole size	.325	223	.000	.642	223	.000
constant-variance	.133	223	.000	.948	223	.000
additive-variance	.097	223	.000	.915	223	.000
multiplicative-variance	.113	223	.000	.828	223	.000
sign-variance	.154	223	.000	.917	223	.000

The Table 4.8 is presenting the numbers of the accepted biclusters, which will be used in the second stage of the comparison. Just in the multiplicative-variance row, we can see that we were not able to include 60 biclusters for CC, FLOC, Plaid Model and Qubic algorithm. That because of the outputs in R program were presented as infinity.

Table 4.8. The number of accepted values for the detected values

Number of valid value///Algorithm	CC	FLOC	Plaid	xMotif	Bimax	Qubic
Number of Rows	60	60	60	60	60	60
Number of columns	60	60	60	60	60	60
Size	60	60	60	60	60	60
Constant-variance	60	60	60	60	60	60
Additive-variance	60	60	60	60	60	60
Multiplicative-variance	28	43	11	60	60	21
Sign-variance	60	60	60	60	60	60

Number of rows

KW with $\alpha = 0.05$ is used to test whether there are statistically significant differences between CC, FLOC, Plaid, xMotif, Bimax and Qubic algorithm for the number of rows in the detected biclusters. The results are $\chi^2_5 = 287.35$; $p = 0.000 < 0.05$, which means that there are high statistically significant differences between the groups. Table 4.9 presents the pairwise comparisons of the used algorithms as follow:

Table 4.9. Multiple comparisons table for the number of rows

algorithm 1-algorithm 2	median 1	median 2	Test Statistic	Adj. Sig.
Qubic-CC	11.0	35.5	44.667	0.281
Qubic-xMotif	11.0	35.0	-46.408	0.219
Qubic-Plaid	11.0	182.0	128.142	0.000**
Qubic-FLOC	11.0	1880.0	217.000	0.000**
Qubic-Bimax	11.0	2078.5	250.683	0.000**
CC-xMotif	35.5	35.0	-1.742	1.000
CC-Plaid	35.5	182.0	-83.475	0.000**
CC-FLOC	35.5	1880.0	-172.333	0.000**
CC-Bimax	35.5	2078.5	206.017	0.000**
xMotif-Plaid	35.0	182.0	81.733	0.000**
xMotif-FLOC	35.0	1880.0	170.592	0.000**
xMotif-Bimax	35.0	2078.5	204.275	0.000**
Plaid-FLOC	182.0	1880.0	88.858	0.000**
Plaid-Bimax	182.0	2078.5	122.542	0.000**
FLOC-Bimax	1880.0	2078.5	33.683	1.000

As we can see from the table all of the groups are different from each other except the results obtained using Qubic-CC, Qubic-xMotif, CC-xMotif and FLOC-Bimax algorithms. The mean rank values for each group are presented in the following figure:

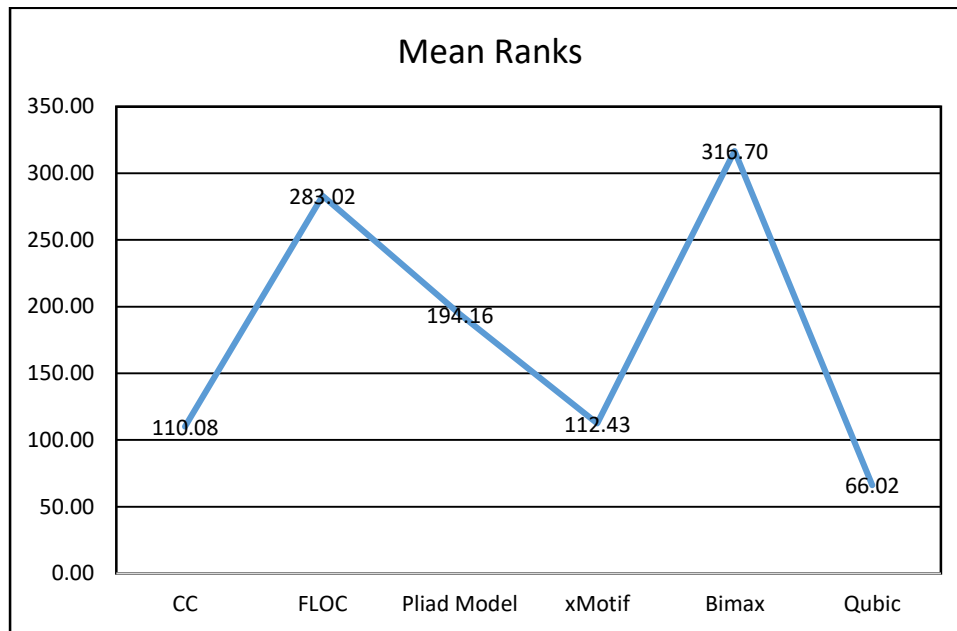


Figure 4.10. Line chart for the mean ranks of the number of rows for each algorithm

According to the previous tests in Table 4.9 and Figure 4.10 with this dataset and the used conditions in the DEA analysis, the best performance was obtained using the Bimax algorithm with no significant difference with the FLOC algorithm, which means that the two algorithms were able to detect biclusters with the biggest numbers of rows. Plaid algorithm comes in the second place for the numbers of rows in the detected biclusters. xMotif, CC and Qubic algorithms with no significant difference come in the last place with the smallest numbers of rows for the used dataset and the variables that were used in the DEA stage.

Note: big numbers of rows may not be a good every time. In this study the comparison is based on the logic of biggest sizes is better and smallest variances are also better.

Number of columns

KW test with $\alpha = 0.05$ is used to test whether there are statistically significant differences between CC, FLOC, Plaid, xMotif, Bimax and Qubic algorithm for the numbers of columns in the detected biclusters. The result is $\chi^2_5 = 176.687$; $p = 0.000 < 0.05$, which means that

there are high statistically significant differences between the groups. Table 4.10 presents the pairwise comparisons of the used algorithms as follow:

Table 4.10. Multiple comparisons table for the number of columns

algorithm 1-algorithm 2	median 1	median 2	Test Statistic	Adj. Sig.
FLOC-Qubic	2	4	-59.033	0.025**
FLOC-Plaid	2	5	-89.933	0.000**
FLOC-xMotif	2	7	-105.742	0.000**
FLOC-Bimax	2	11	159.950	0.000**
FLOC-CC	2	12	226.992	0.000**
Qubic-Plaid	4	5	30.900	1.000
Qubic-xMotif	4	7	-46.708	0.195
Qubic-Bimax	4	11	100.917	0.000**
Qubic-CC	4	12	167.958	0.000**
Plaid-xMotif	5	7	-15.808	1.000
Plaid-Bimax	5	11	70.017	0.000**
Plaid-CC	5	12	137.058	0.000**
xMotif-Bimax	7	11	54.208	0.059
xMotif-CC	7	12	121.250	0.000**
Bimax-CC	11	12	-67.042	0.000**

As we can see from the results in Table 4.10 all of the groups are different from each other except the results obtained using Qubic-Plaid, Qubic-xMotif, Plaid-xMotif and xMotif-Bimax algorithms. The mean rank values for each group are presented in the following figure:

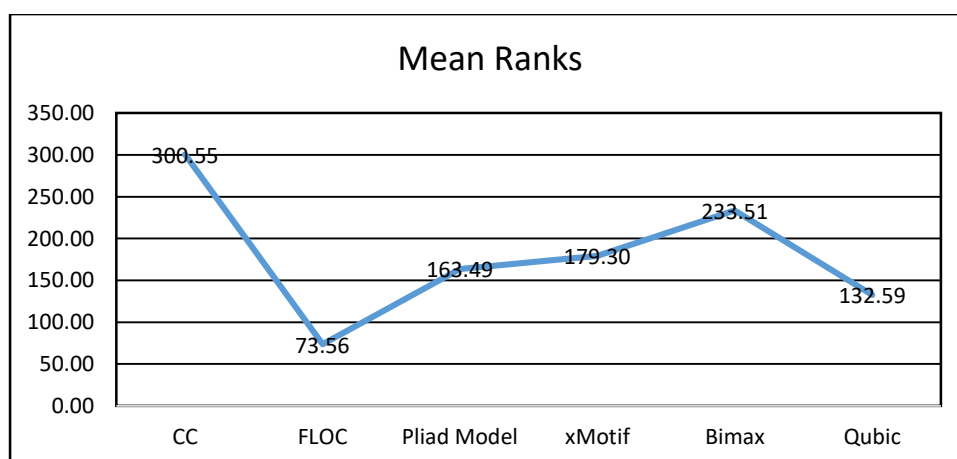


Figure 4.11. Line chart for the mean ranks of the number of columns for each algorithm

According to the previous tests in Table 4.10 and Figure 4.11 with this dataset and the used conditions in the DEA analysis, the best performance for the numbers of columns in the detected biclusters was with the CC algorithm. In the second place, Bimax and xMotif algorithms. Plaid algorithm and Qubic algorithms come in the third place for the numbers of columns. The worst performance for the numbers of columns was with FLOC algorithm for the detected biclusters from the used data and the used variables in the DEA stage.

Size of the biclusters

KW test with $\alpha = 0.05$ is used to test whether there are statistically significant differences between CC, FLOC, Plaid, xMotif, Bimax and Qubic algorithm for the sizes of the detected biclusters. The result is $\chi^2_5 = 298.549$; $p = 0.000 < 0.05$, which means that there are high statistically significant differences between the groups. Table 4.11 presents the pairwise comparisons of the used algorithms as follow:

Table 4.11. Multiple comparisons table for the biclusters sizes

algorithm 1-algorithm 2	median 1	median 2	Test Statistic	Adj. Sig.
Qubic-xMotif	36.0	245.0	-61.983	0.017**
Qubic-CC	36.0	400.0	98.517	0.000**
Qubic-Plaid	36.0	895.5	141.250	0.000**
Qubic-FLOC	36.0	3958.0	224.500	0.000**
Qubic-Bimax	36.0	22863.5	279.750	0.000**
xMotif-CC	245.0	400.0	36.533	0.818
xMotif-Plaid	245.0	895.5	79.267	0.000**
xMotif-FLOC	245.0	3958.0	162.517	0.000**
xMotif-Bimax	245.0	22863.5	217.767	0.000**
CC-Plaid	400.0	895.5	-42.733	0.368
CC-FLOC	400.0	3958.0	-125.983	0.000**
CC-Bimax	400.0	22863.5	181.233	0.000**
Plaid-FLOC	895.5	3958.0	83.250	0.000**
Plaid-Bimax	895.5	22863.5	138.500	0.000**
FLOC-Bimax	3958.0	22863.5	55.250	0.055

As we can see from the results in Table 4.11, all of the groups are different from each other except the results obtained using xMotif-CC, CC-Plaid, and FLOC-Bimax algorithms. The mean ranks values for each group are presented in the following figure:

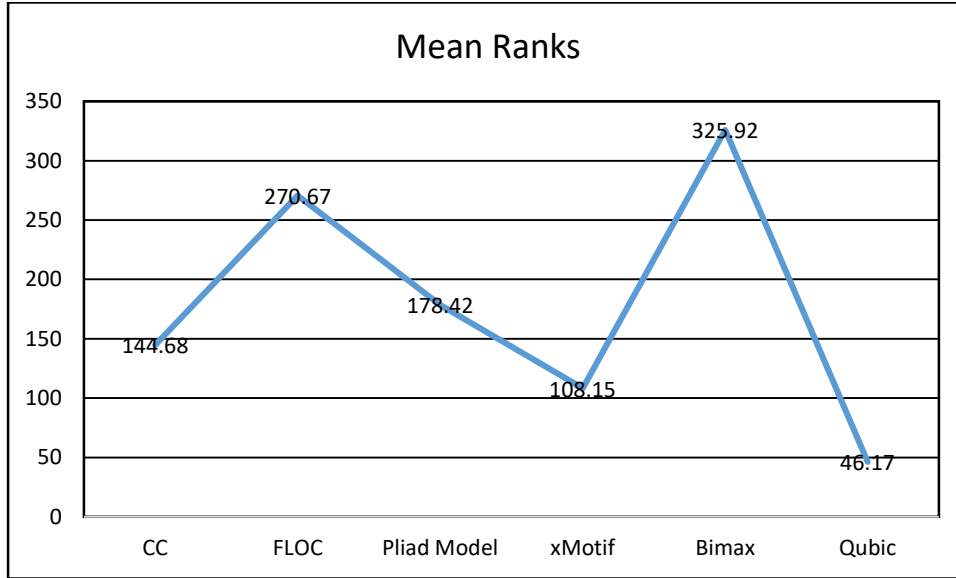


Figure 4.12. Line chart for the mean ranks of the sizes of the detected biclusters for each algorithm

According to the previous tests in Table 4.11 and Figure 4.12 with this dataset and the used conditions in the DEA analysis, the best performance for the sizes of the detected biclusters was with the Bimax and FLOC algorithms. In the second place as presented in the figure, Plaid model, CC algorithms. In the third place, xMotif algorithm comes with no significant difference with the CC algorithm. The smallest bicluster sizes were obtained using the Qubic algorithms for the used dataset and according to the used variables in the DEA stage.

Constant-variance

KW test with $\alpha = 0.05$ is used to test whether there are statistically significant differences between CC, FLOC, Plaid, xMotif, Bimax and Qubic algorithms according to the constant-variance measure. The result is $\chi^2_5 = 147.258$; $p = 0.000 < 0.05$, which means that there are high statistically significant differences between the groups. Table 4.12 presents the pairwise comparisons of the used algorithms as follow:

Table 4.12. Multiple comparisons table for the constant-variance values

algorithm 1-algorithm 2	median 1	median 2	Test Statistic	Adj. Sig.
Qubic-FLOC	56.536	93.770	63.617	0.012**
Qubic-Plaid	56.536	127.836	130.167	0.000**
Qubic-xMotif	56.5356	133.996	-133.733	0.000**
Qubic-Bimax	56.536	181.184	176.100	0.000**
Qubic-CC	56.536	159.941	195.083	0.000**
FLOC-Plaid	93.770	127.836	-66.550	0.007**
FLOC-xMotif	93.770	133.996	-70.117	0.003**
FLOC-Bimax	93.770	181.184	112.483	0.000**
FLOC-CC	93.770	159.941	131.467	0.000**
Plaid-xMotif	127.836	133.996	-3.567	1
Plaid-Bimax	127.836	181.184	45.933	0.234
Plaid-CC	127.836	159.941	64.917	0.01**
xMotif-Bimax	133.996	181.184	42.367	0.386
xMotif-CC	133.996	159.941	61.350	0.019**
Bimax-CC	181.184	159.941	-18.983	1

As we can see from Table 4.12 all of the groups are different from each other except the results obtained using Plaid-xMotif, Plaid-Bimax, xMotif-Bimax and Bimax-CC algorithms. The mean ranks values for each group are presented in the following figure:

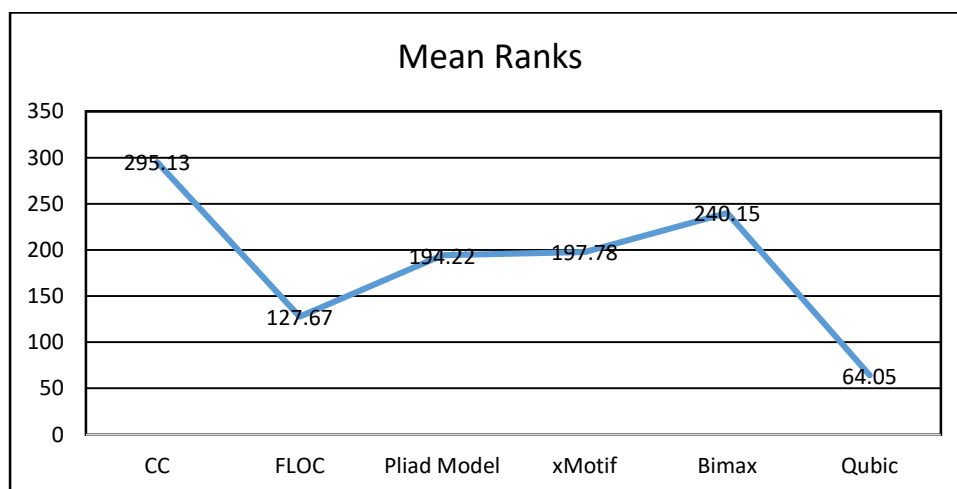


Figure 4.13. Line chart for the mean ranks of the constant-variance values for each algorithm

According to the previous tests in Table 4.12 and Figure 4.13 with this dataset and the used conditions in the DEA analysis, the best performance for the constant-variance measure for the detected biclusters was with the Qubic algorithm. However, we must notice the smallest constant-variance and the smallest sizes at the same time. FLOC algorithm comes in the second place with good sizes. In addition, the Plaid model, xMotif, and Bimax with no statistically significant differences between them, come in the third place. CC algorithm has detected the biggest constant-variance values in the used dataset and according to the used variables in the DEA stage.

Additive-variance

The KW test with $\alpha=0.05$ is used to test whether there are statistically significant differences between CC, FLOC, Plaid, xMotif, Bimax and Qubic algorithms according to the additive-variance measure. The result is $\chi^2_5 = 200.628$; $p = 0.000 < 0.05$, which means that there are high statistically significant differences between the groups. Table 4.13 presents the pairwise comparisons of the used algorithms as follow:

Table 4.13. Multiple comparisons table for the additive-variance values

algorithm 1-algorithm 2	median 1	median 2	Test Statistic	Adj. Sig.
FLOC-Qubic	9.715	20.673	-59.233	0.027**
FLOC-xMotif	9.715	34.348	-100.250	0.000**
FLOC-Plaid	9.715	38.389	-130.783	0.000**
FLOC-Bimax	9.715	75.340	208.733	0.000**
FLOC-CC	9.715	58.525	218.700	0.000**
Qubic-xMotif	20.673	34.348	-41.017	0.463
Qubic-Plaid	20.673	38.389	71.550	0.002**
Qubic-Bimax	20.673	75.340	149.500	0.000**
Qubic-CC	20.673	58.525	159.467	0.000**
xMotif-Plaid	34.348	38.389	30.533	1
xMotif-Bimax	34.348	75.340	108.483	0.000**
xMotif-CC	34.348	58.525	118.450	0.000**
Plaid-Bimax	38.389	75.340	77.950	0.001**
Plaid-CC	38.389	58.525	87.917	0.000**
Bimax-CC	75.340	58.525	-9.967	1

As we can see from the table all of the groups are different from each other except the results obtained using Qubic-xMotif, xMotif-Plaid and Bimax-CC algorithms. The mean ranks value for each group are presented in the following figure:

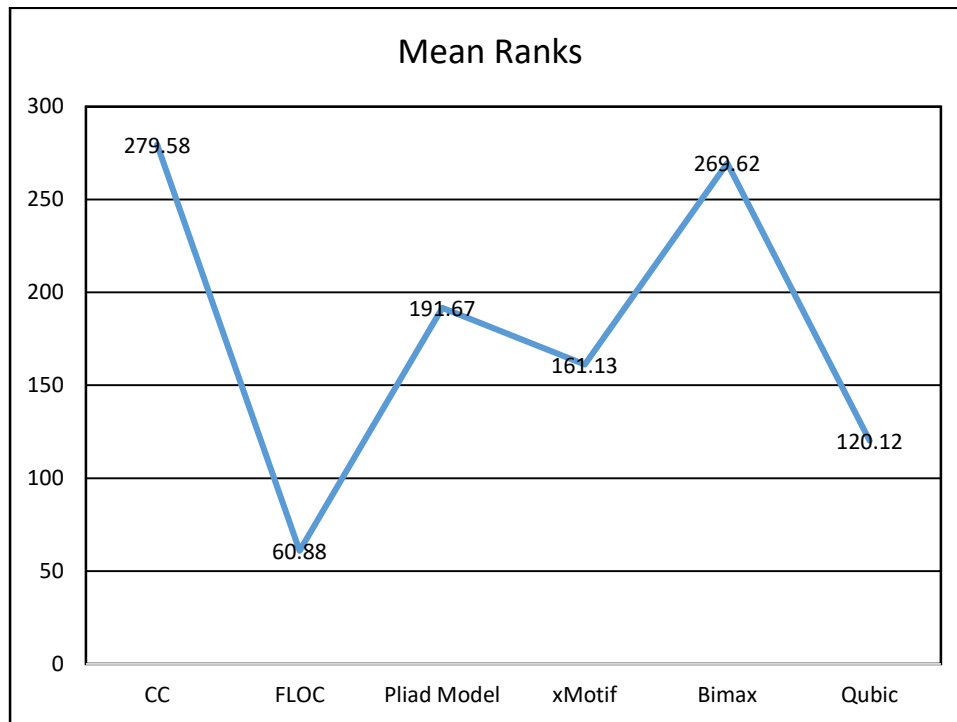


Figure 4.14. Line chart for the mean ranks of the additive-variance values for each algorithm

According to the previous tests in Table 4.13 and Figure 4.14 with this dataset and the used conditions in the DEA analysis, the best performance for the additive-variance measure for the detected biclusters was with the FLOC algorithm. In the second place as presented in the Figure 4.14, Qubic and xmotif algorithms come. In addition, Plaid Model comes in the third place. Finally, both of the Bimax and CC algorithms were able to detect the biggest additive-variance values from the used dataset and according to the used variables in the DEA stage.

Multiplicative-variance

KW test with $\alpha = 0.05$ is used to test whether there are statistically significant differences between CC, FLOC, Plaid, xMotif, Bimax and Qubic algorithms according to the multiplicative-variance measure. The result is $\chi^2_5 = 107.955$; $p = 0.000 < 0.05$, which means that there are high statistically significant differences between the groups. Table 4.14 presents the pairwise comparisons of the used algorithms as follow:

Table 4.14: Multiple comparisons table for the multiplicative-variance values

algorithm 1-algorithm 2	median 1	median 2	Test Statistic	Adj. Sig.
FLOC-Qubic	0.0546	0.0745	-20.955	1.000
FLOC-xMotif	0.0546	0.2994	-99.590	0.000**
FLOC-Plaid	0.0546	0.2601	-107.543	0.000**
FLOC-Bimax	0.0546	0.3501	108.157	0.000**
FLOC-CC	0.0546	0.2676	109.371	0.000**
Qubic-xMotif	0.0745	0.2994	-78.636	0.000**
Qubic-Plaid	0.0745	0.2601	86.589	0.005**
Qubic-Bimax	0.0745	0.3501	87.202	0.000**
Qubic-CC	0.0745	0.2676	88.417	0.000**
xMotif-Plaid	0.2994	0.2601	7.953	1.000
xMotif-Bimax	0.2994	0.3501	8.567	1.000
xMotif-CC	0.2994	0.2676	9.781	1.000
Plaid-Bimax	0.2601	0.3501	0.614	1.000
Plaid-CC	0.2601	0.2676	1.828	1.000
Bimax-CC	0.3501	0.2676	-1.214	1.000

As we can see from the table all of the groups are different from each other expect the results obtained using FLOC-Qubic, xMotif-Plaid, xMotif-Bimax, xMotif-CC, Plaid-Bimax, Plaid-CC and Bimax-CC algorithms. The mean ranks values for each group are presented in the following figure:

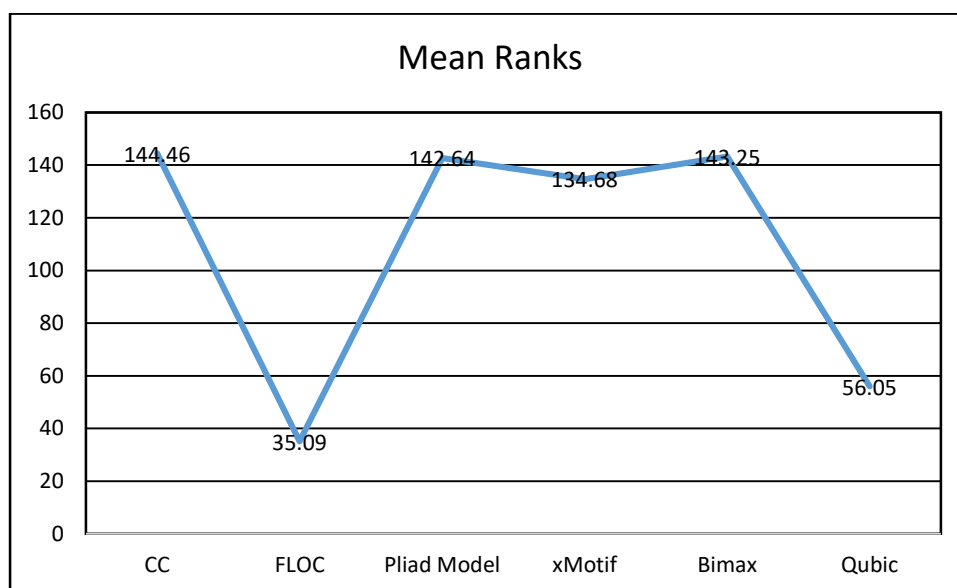


Figure 4.15. Line chart for the mean ranks of the multiplicative-variance values for each algorithm

Not like other measures, multiplicative-variance values may go high. Therefore, some values go to infinity in R program (which considered as missing values). Therefore, the best performance can be considered as which algorithms were able to detect 60 biclusters with small multiplicative-variance. According to this idea, xMotif and Bimax will be the best followed by FLOC algorithm. In addition, Plaid Model algorithm will be the last one.

On other hand using the same evaluation that used for the previous measures: According to the previous tests in Table 4.14 and Figure 4.15 with this dataset and the used conditions in the DEA analysis, the best performance for the multiplicative-variance measure for the detected biclusters was with the FLOC and Qubic algorithms with the smallest means ranks for the medians of the multiplicative-variance values. Bimax, xMotif, Plaid model and CC algorithms come in the second place for the multiplicative-variance measure.

Sign-variance

KW test with $\alpha = 0.05$ is used to test whether there are statistically significant differences between CC, FLOC, Plaid, xMotif, Bimax and Qubic algorithms according to the sign-variance measure. The result is $\chi^2_5 = 205.181$; $p = 0.000 < 0.05$, which means that there are high statistically significant differences between the groups. Table 4.15 presents the pairwise comparisons of the used algorithms as follow:

Table 4.15. Multiple comparisons table for the sign-variance values

algorithm 1-algorithm 2	median 1	median 2	Test Statistic	Adj. Sig.
Qubic-FLOC	0.3335	0.4591	44.408	0.291
Qubic-Plaid	0.3335	1.0125	111.458	0.000**
Qubic-xMotif	0.3335	1.0771	-121.667	0.000**
Qubic-Bimax	0.3335	2.0879	205.275	0.000**
Qubic-CC	0.3335	1.6599	218.142	0.000**
FLOC-Plaid	0.4591	1.0125	-67.050	0.006**
FLOC-xMotif	0.4591	1.0771	-77.258	0.001**
FLOC-Bimax	0.4591	2.0879	160.867	0.000**
FLOC-CC	0.4591	1.6599	173.733	0.000**
Plaid-xMotif	1.0125	1.0771	-10.208	1.000
Plaid-Bimax	1.0125	2.0879	93.817	0.000**
Plaid-CC	1.0125	1.6599	106.683	0.000**
xMotif-Bimax	1.0771	2.0879	83.608	0.000**
xMotif-CC	1.0771	1.6599	96.475	0.000**
Bimax-CC	2.0879	1.6599	-12.867	1.000

As we can see from the table all of the groups are different from each other expect the results obtained using Qubic-FLOC, Plaid-xmotif and Bimax-CC algorithms. The mean ranks value for each group are presented in the following figure:

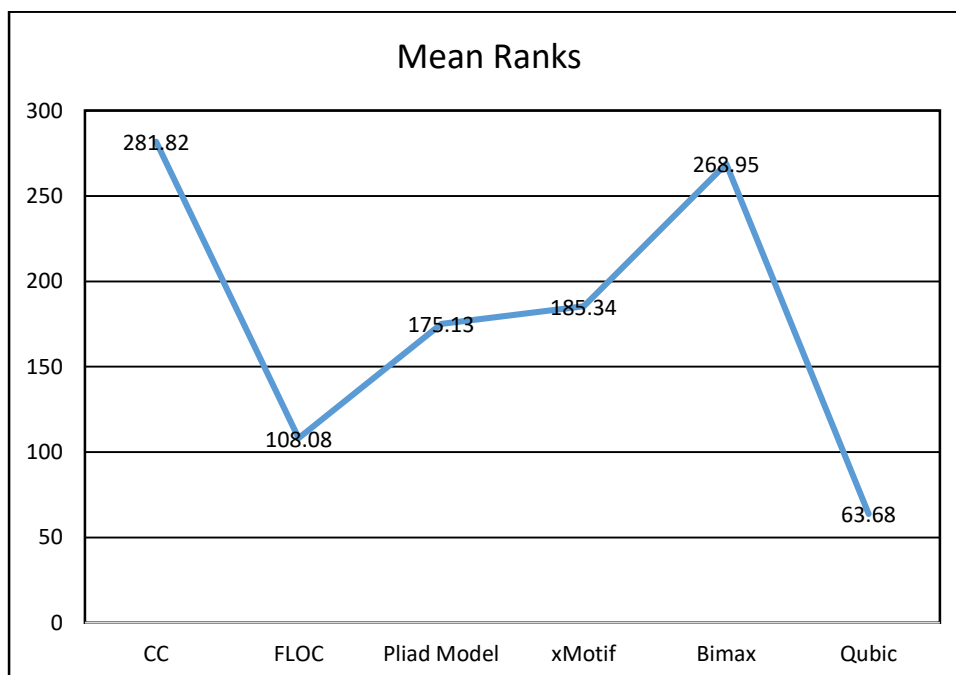


Figure 4.16. Line chart for the mean ranks of the sign-variance values for each algorithm

According to the previous tests in Table 4.15 and Figure 4.16 with this dataset and the used conditions in the DEA analysis, the best performance for the sign-variance measure for the detected biclusters was with the Qubic and FLOC algorithms. Plaid model and xMotif come in the second place. Finally, the worst performance for this measure was with the Bimax and CC algorithms for the used dataset and according to the used variables in the DEA stage.

4.3. Discussion and Recommendations

Biclustering method is one of the most important techniques in the data mining, which can be applied in any field if the dataset can be presented as a data matrix. Many algorithms have been introduced until now [9]. Each algorithm has advantage and disadvantages. Algorithms that introduced in this work can be summarized as follow:

Most of the introduced biclustering algorithms were developed originally to deal with gene expression datasets. Even though Block Clustering algorithm was applied to voting results in the US and UN [2]. This algorithm opens the door to develop many algorithms to deal with the gene expression data and any kind of data, which can be presented as a data matrix shape. The main idea in this algorithm is to split the data matrix into k bicluster using variance to evaluate the detected biclusters. The biggest disadvantages of this algorithm are the early splitting of the data matrix, which may cause losing lots of important patterns in the data and the obtained biclusters are non-overlap with constant values biclusters [40]. Finally, Tibshirani et al. [61] developed a practical way to use DC algorithm by finding a lot number of blocks instead of finding K blocks and combine them using a backward pruning method to have the number of biclusters equal to K .

DC algorithm [2] is the first known biclustering algorithm, but Chang and Church [28] were the first to apply the biclustering concept to gene expression data with their algorithm using the mean square residue to find the biclusters in the data. CC algorithm was tested in their work using yeast data [62] and human B-cells expression data [63] and was able to find more interesting patterns than the patterns can be detected using clustering methods. In addition, the CC algorithm is able to find constant or coherent type biclusters. To use this algorithm, the data should have no missing values. However, if there are missing values it will be replaced by random numbers. In addition, the CC algorithm will detect a single bicluster in

every iteration and replace it with random numbers in the same way that used to deal with missing values. Inserting random values may cause finding meaningless biclusters.

FLOC algorithm [15] was introduced to improve the way that CC algorithm [28] works by giving biclusters with a controlled amount of missing values instead of inserting random values to the data. In addition, FLOC algorithm works faster than CC algorithm. FLOC algorithm will discover the wanted number of the biclusters in the same time. In addition, like CC algorithm it will look for arbitrarily positioned overlapping biclusters with coherent values type. The FLOC algorithm depends on the initial biclusters, which makes it fails to find a high-quality biclusters lot of time [64].

Plaid model [35, 51] is additive biclusters model. Algorithm will work to detect K biclusters that modeled as the sum of a background effect, cluster effect, row effects, column effects and a random noise. The Plaid model algorithm will work to find Arbitrary positioned overlapping biclusters with coherent values and find K bicluster at the same time. Algorithm is located in distribution parameter identification category that works to fit bicluster membership parameters into a model. Finally, Plaid Model is known as the most flexible algorithm because we can control many features so we can obtain users wanted biclusters according to the kind of the researchers.

Getz et al. [43, 52] developed CTWC, which can use any clustering method to find biclusters. They advised using Superparamagnetic clustering algorithm SPC because they thought it is more suitable for gene expression data. Before applying CTWC, the data values are being normalized by dividing each column by its mean then each row by its mean also to detect biclusters with constant columns. CTWC is following Combined Clusters strategy, which means finding clusters using any clustering method and combining them using specific conditions.

The ITWC algorithm was introduced by Tang et al. [44] to find biclusters with possible row overlapping. It follows clusters combine strategy. It looks for one bicluster with coherent values each iteration.

δ -pCluster algorithm was proposed by Wang et al. [45] to improve the quality of the gene expression data analysis. They proposed using pCluster value to evaluate the quality of the

submatrices. This algorithm has important features like each submatrix in the pCluster (bicluster) is also a pCluster, and the pClusters do not contain outliers. Algorithm is able to find coherent values with additive model or by changing the values with its logarithms multiplicative model pClusters.

SAMBA algorithm [46] was introduced to detect coherent evolution or constant types of biclusters. Algorithm can find arbitrarily positioned Overlapping K biclusters at the same time. Algorithm is working on Exhaustive Bicluster Enumeration approach, which means it will search for all the possible biclusters of the data matrix can be made find the best biclusters. In addition, algorithm cannot deal well with the noise in the data [41].

Murali and Kasif [36] introduced the xMotif algorithm, which follows a greedy iterative search and chooses locally optimal sets and hopes that they might be globally optimal. The idea may not give good results, especially with not very good parameters. xMotif Algorithm is designed to detect biclusters with the coherent evolution of rows type, it can also detect biclusters with constant values on rows, and it cannot deal very well with binary data sets. The xMotif algorithm has a problem with dealing with the noise in the data [65]. xMotif algorithm has time in complexity equal to $O(N \times n_d \times n_s)$.

ROBA algorithm was introduced by Tchagang and Tewfix [47] has time in complexity equal to $O(LN_b NM)$ where L is the number of distinct values present in the $N \times M$ input data matrix and N_b is the number of the biclusters. Algorithm follows matrix algebra strategy to detect four biclusters types. It deals with missing data and the noise in the data, which make it a good algorithm to be used to detect the bicluster in the data.

The main goal of proposing Bimax algorithm by Prelic et al. [10] to be a reference method ,which helps to choose other methods parameters. This algorithm will work to detect submatrix (bicluster) that have ones and cannot be in a bigger submatrix.

RMSBE algorithm was introduced by Wang and Liu [48] and works to detect biclusters of constant or additive types. RMSBE algorithm is able to find different bicluster sizes and deal well with deferent noise levels. It is unnecessary to mask previously discovered biclusters

during the process [66]. Algorithm can find nearly squared biclusters and can be extended to find rectangular biclusters.

Qubic algorithm was introduced by Li et al. [37], which employs a graph-theoretic approach to solve the biclustering problem is less sensitive to outliers from many other biclustering methods [67]. It is good to be used with very big data sets. In addition, algorithm able to find many types of the biclusters especially constant columns or rows bicluster types in a reasonable time.

CPB algorithm was introduced by Bozdağ et al. [49] and uses Pearson correlation coefficient between columns and rows to detect highly correlated biclusters in the data. Algorithm starts with a randomly chosen subset of columns and rows and improves the quality by adding or moving rows and columns from the bicluster iteratively. Algorithm well suited for identifying shift-scale patterns. The CPB algorithm is good to analyze multi datasets and merge results from each dataset and can be used at the same time for one data set.

As presented in the previous preview, each one of the biclustering algorithms has some of its important properties. In addition, algorithms can be classified into groups according to the target. The Target can be a specific bicluster type but there are many algorithms to have this type in different ways. Therefore, comparing biclustering algorithms is very important to help in choosing the right algorithms.

As introduced in the literature view section there are many works for evolution and comparing the biclustering works. In our thesis, a two-stage comparison way was introduced and being applied to 6 biclustering algorithms.

In the first stage, the DEA method was used to compare the 6 biclustering algorithms, which were repeated more than 50 times with different parameters. Then using the output-oriented BCC model and the super-efficiency DEA techniques to give a rank or order according to some variables. In this study as presented before, some general conditions are chosen in the DEA stage like variance or size or other variables to give an example. In real life studies, the experts will choose the variable that will be used in the DEA stage according to the field study.

After each one of the chosen algorithms were ranked in the DEA stage. Starting from the best algorithms to generate 20 biclusters at most from each one. Some of the chosen algorithms may not be able to generate 20 biclusters so we continue choosing algorithms until we have 60 biclusters for each algorithm. The 60 biclusters will be from different parameters values in each algorithm. However, we are dealing with them as if they were generated using just the same parameters values. In this study, CC, FLOC, Plaid Model, xMotif, Bimax and Qubic algorithms were chosen. For these 60 biclusters from each algorithm, which give us 360 biclusters, the non-parametric methods for comparing medians (KW test and Dunn's test) used to compare them according to size and the variance.

The results from the two-stage comparison for the yeast dataset and the chosen variables in the DEA stage:

1. The best performance for the numbers of rows in the detected biclusters were obtained using the Bimax and FLOC algorithms, respectively, while Qubic, CC, and xMotif algorithms have the worst performance with detecting biclusters with the smallest number of rows within it.
2. For the numbers of columns, the best performance recorded using CC algorithm. FLOC algorithm detected biclusters with the smallest numbers of columns.
3. The biggest biclusters sizes were detected using the Bimax and FLOC algorithms. In addition, Qubic algorithm was able to detect the smallest biclusters.
4. As presented in the boxplot Figures none of the chosen algorithms were able to detect biclusters with constant values types because all of the values of constant-variance are bigger than 1.5. However, the biclusters that were detected using the Qubic algorithm where have the smallest constant-values with the smallest sizes also. In addition, the CC algorithm has the highest constant-variance values with no statistically significant difference with significance level equal to $\alpha = 0.05$ according to the chosen variables in the DEA stage and the dataset.
5. In addition, the chosen biclusters were not able to detect biclusters with an additive model, with expecting to 5 ideal biclusters with additive-variance values equal to 0. Like in constant-variance if we looked at the smallest values: The detected biclusters using FLOC algorithm were the best in this study. The worst performance was with the Bimax and CC algorithms for the additive-variance values.

6. Just with xMotif and Bimax algorithms we were able to detect the chosen number (60) with the multiplicative model were the multiplicative-variance values smaller the 1.5. However, by looking just to the values FLOC and Qubic algorithms were able to detect biclusters with smaller variance values followed by Qubic algorithms with small biclusters sizes and some multiplicative values equal to zero. The worst performance was with the rest of the used algorithms with no statistically significant differences.
7. Finally, for the sign-variance, which is important, because it is the one, which is used in the DEA, stage. FLOC, Plaid Model and Qubic algorithms where able to detect the full number of biclusters with sign-variance values smaller than 1.5. For the 6 algorithms, FLOC algorithm and Qubic algorithms were the best with no statistically significant differences between them according to the KW test and Dunn's tests. However, by taking in account the sizes, FLOC algorithm will be better. In addition, CC and Bimax also have no statistically significant differences between them, which make them come in the last place.

The previous results are presented in the following table:

Table 4.16. The results of the two-stage comparison study

Rank	Rows	Columns	Size	constant	additive	Multiplicative	Sign
1	Bimax FLOC	CC	Bimax FLOC	Qubic	FLOC	FLOC Qubic	FLOC Qubic
2	Plaid	Bimax xMotif	Plaid CC	FLOC	Qubic xMotif	Bimax Plaid xMotif CC	Plaid xMotif
3	xMotif CC Qubic	Plaid Qubic	xMotif	Plaid xMotif Bimax	Plaid		Bimax CC
4		FLOC	Qubic	CC	Bimax CC		

The result in the previous table is based on using the DEA method to choose the best parameters according to some variable. Of course, choosing other variables in the DEA stage may affect the results. That means the results could not be generalized. The chosen variables have been chosen according to a statistical logic. Each field study has its own characteristics. In addition, all of the studies that were presented in the previous studies section and most of the studies, in general, did not use the ensemble method [3]. The results in the Tables 4.1 to 4.6, which presents the DEA methods results, showed that the starting points for each

algorithm were the best. That comes because one of the used variables in the DEA stage was the sign-variance. In addition, most of the introduced biclustering algorithms work to detect biclusters with similar patterns, which make the variance values in the detected biclusters smaller with make these results logical somehow. So using the ensemble method help to merge the best results from the DEA stage to have more features.

By comparing some of the previous results from the literature studies with our results that presented in Table 4.16. In the study, which was done by Nepomuceno [17], as presented in the previous works section if we looked at the numbers of rows in the detected biclusters, the numbers of rows was bigger CC then xMotif and Bimax algorithms. However, the numbers of rows in our results showed that the Bimax algorithm was able to detect the bigger numbers of rows in the detected biclusters. In addition, xMotif and Bimax algorithm have no statistically significant difference between them. In addition, with the same used data, the CC algorithm was able to detect biclusters with a bigger numbers of columns than Bimax and xMotif algorithm, which match our results. For the bicluster sizes in the previous study, the CC algorithm was able to detect biclusters with bigger sizes than Bimax and xMotif. However, here the results showed that the bigger bicluster sizes were with the Bimax algorithm and xMotif still the last. In addition, as presented in our results the FLOC algorithm is able to detect bigger bicluster the also was obtained in Yin and Liu's work [20] as the results in table 4.16.

In addition, for the results in Eren et al.'s work [22], which used other datasets and compared the Bimax, CC, Plaid Model, Qubic and xMotif algorithms according to algorithms ability to detect different biclusters types. In that study, the xMotif algorithm was the best in detecting biclusters with constant type, while in our study the Qubic algorithm was the best of course with yeast dataset here. For additive types, the result of the previous study showed that the Plaid Model algorithm was the best, but here in Table 4.16 as is presented the FLOC algorithm was the best and Plaid Model was not so good in comparing with the other algorithms. Finally, the multiplicative model in Eren et al.' study, CC algorithm had a good performance in detecting this type, but in our results both of the FLOC and Qubic algorithms have good performances but with a big difference in the sizes of the detected biclusters.

Finally, not all of the results can be compared because each one of the previous works in the biclustering field have been done to compare the biclustering algorithm from different sides.

In addition, in this study, an idea has been proposed to choose the best parameters, and make a classification for some of the biclustering algorithms with a single dataset. The generalization of results cannot be done until more algorithms must be used, more variables in the DEA stage must be included and more datasets is being included in bigger studies.

REFERENCES

1. Han, J., Kamber, M., Pei, J. (2012). *Data Mining Concepts and Techniques*(Third edition). Waltham:Elsevier, 1-38.
2. Hartigan, J.A. (1972). Direct Clustering of a Data Matrix. *Journal of the American Statistical Association*, 67(337), 123-129.
3. Kaiser, S. (2001). *Biclustering: methods, software and application*. Doctoral dissertation, Ludwig Maximilian University, M n h.
4. Hartigan, J.A. (1975). *Clustering algorithms*. New York: Wiley, 1-28.
5. Kasim, A., Shkedy, Z., Talloen, W., Hochreiter, S. (2016). *Applied Biclustering Methods for Big and High Dimensional Data Using R*. Chapman and Hall/CRC
6. Vicente, R.S. (2009). *Visual analysis of gene expression data by means of biclustering*. Doctoral dissertation, University of Salamanca, Salamanca.
7. Giordani, I. (2009). *Relational clustering for knowledge discovery in life sciences*. Doctoral dissertation, University of Milano-Bicocca: Milan.
8. Vittapu, M.S. (2017). Synchronized clustering: a review on systematic comparisons and validation of prominent block-clustering algorithms. *International Journal of Engineering and Information Systems (IJEAIS)*, 1(4), 28-37.
9. Madeira, S.C., Oliveira, A.L. (2004). Biclustering algorithms for biological data analysis: A survey. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 1(1), 24-45.
10. Prelic, A., Bleuler, S., Zimmermann, P. Wil, A., Buhlmann, P., Gruissem, W., Hennig, L., Thiele, L. and Zitzler, E. (2006). A systematic comparison and evaluation of biclustering methods for gene expression data bioinformatics. *Oxford Univ Press*, 22, 1122-1129.
11. Tchagang, A.B., Pan, Y., Famili, F., Tewfik, A.H., & Benos, P.V. (2011). Biclustering of dna microarray data: theory, evaluation, and applications. *Handbook of Research on Computational and Systems Biolog*, 148-186.
12. Chia, B. K. H. and Karuturi, R. K. M. (2010). Differential co-expression framework to quantify goodness of biclusters and compare biclustering algorithms. *Algorithms for Molecular Biology*, 5(1), 5-23.
13. Pontes, B., G rdez, R. and Aguilar-Ruiz, J. S. (2015). Quality measures for gene expression biclusters. *PloS one*, 10(3), e0115497.
14. Padilha, V. A. and Campello, R. J. (2017). A systematic comparative evaluation of biclustering techniques. *BMC bioinformatics*, 18(1), 55.
15. Yang, J., Wang, H., Wang, W., Yu, P.S. (2003). Enhanced biclustering on expression data. *Proc. Third IEEE Conf. Bioinformatics and Bioeng*, 3, 321-327.

16. Ayadi, W., Elloumi, M. and Hao, J. K. (2009). A biclustering algorithm based on a Bicluster Enumeration Tree: application to DNA microarray data. *BioData Mining*, 2(1), 9.
17. Nepomuceno, J.A., Troncoso, A., Aguilar-Ruiz, J. S. (2011). Biclustering of gene expression data by correlation-based scatter search. *BioData Mining*, 4(1), 3.
18. Oghabian, A., Kilpinen, S., Hautaniemi, S., & Czeizler, E. (2014). Biclustering methods: biological relevance and application in gene expression analysis. *PloS one*, 9(3).
19. Beatriz, P., Raúl, G., Aguilar-Ruiz, J. S. (2013). Configurable pattern-based evolutionary biclustering of gene expression data. *Algorithms for Molecular Biology*, 8(1), 4.
20. Yin, L., Liu, Y. (2016). Biclustering analysis and comparison for gene expression data. *Revista de la Facultad de Ingeniería U.C.V*, 31(12), 183-198.
21. Gu, J., Liu, J. S. (2008). Bayesian biclustering of gene expression data. *BMC Genomics*, 9(1), 4.
22. Eren, K., Deveci, M., Küçüktunç, O. and Çatalyürek, Ü. V. (2012). A comparative analysis of biclustering algorithms for gene expression data. *Briefings in Bioinformatics*, 14(3), 279-292.
23. Cooper, W.W., Seiford, L. M., Tone, K. (2006). Introduction to data envelopment analysis and its uses. *US: Springer*, 45-59.
24. Banker, R.D., Cooper, W. W., Seiford, L. M., Zhu, J. (2011). Returns to scale in DEA. *International Series in Operations Research and Management Science*, 41-70.
25. Freitas, A., Ayadi, W., Elloumi, M., Oliveira, L.J., Hao, J.K. (2013). A survey on biclustering of gene expression data. *Biological Knowledge Discovery Handbook: Preprocessing, Mining, and Postprocessing of Biological Data*, 591-608.
26. Musacchia, F. (2013). *Biclustering of gene expression data: hybridization of grasp with other heuristic/metaheuristic Approaches*. Doctoral dissertation, University in Naples.
27. Mirkin, B. (1996). Mathematical classification and clustering: from how to what and why. In classification, data analysis, and data highways. *Springer*, 172-181.
28. Cheng, Y. and Church G. M. (2000). Biclustering of expression data. *International Conference on Intelligent Systems for Molecular Biology*, 8, 93-103.
29. Leite, C.A.M. (2016). *Domain Oriented Biclustering Validation*. Doctoral dissertation, University of Porto, Portugal.
30. hia, B. K. H. and Karuturi, R. K. M. (2010). Differential co-expression framework to quantify goodness of biclusters and compare biclustering algorithms. *Algorithms for Molecular Biology*, 5(1), 23.

31. Pontes, B., Giráldez, R., Aguilar-Ruiz, J.S. (2015). Biclustering on expression data: a review. *Journal of Biomedical Informatics*, 57, 163–180.
32. Lu, C.C. (2014). Evaluation of heuristics using data envelopment analysis. *International Journal of Information Technology & Decision Making*, 13(4), 795–810.
33. Lu, C.C. (2015). Robust data envelopment analysis approaches for evaluating algorithmic performance. *Elsevier*, 81, 78-89.
34. Tavazoie, S., Hughes, J.D., Campbell, M.J., Cho, R.J., Church, G.M. (1999). Systematic determination of genetic network architecture. *Nature Genetics*, 22, 281-285.
35. Lazzeroni, L. and Owen, A. (2002). Plaid models for gene expression data. *Statistica Sinica*, 61-86.
36. Murali, T. M. and Kasif, S. (2003). Extracting conserved gene expression motifs from gene expression data. Pacific Symposium on Biocomputing. 8, 77-88.
37. Li, G., Ma, Q., Tang, H., Paterson, A.H., & Xu, Y. (2009). Qubic: a qualitative biclustering algorithm for analyses of gene expression data. *Nucleic Acids Research*, 37(15).
38. Gasch, A., Spellman, P., Kao C., Carmel-Harel, O., Eisen, M., Storz, G., Botstein, D., Brown, P. (2000). Genomic expression programs in the response of yeast cells. *Mol Biol Cell*, 11(12), 4241-4257.
39. Kilpinen, S., Autio, R., Ojala, K., Iljin, K., Bucher, E., et al. (2008). Systematic bioinformatic analysis of expression levels of 17,330 human genes across 9,783 samples from 175 types of healthy and pathological tissues. *Genome Biology*, 9.
40. Balanza, B.P. (2013). Evolutionary Biclustering of Gene Expression Data: Shifting and Scaling Pattern-based Evaluation. Doctoral dissertation, University of Seville, Spain.
41. Mina, E. (2010). *Applying Biclustering to Understand the Molecular Basis of Phenotypic Diversity*. Doctoral dissertation, Utrecht University: Netherlands.
42. Al-Akwaa, F.M. (2012). Analysis of gene expression data using biclustering algorithms. *Functional Genomics Germana Meroni*. IntechOpen.
43. Getz, G., Levine, E., Domany E. (2000). Coupled two-way clustering of dna microarray data. *Proceedings of the National Academy of Sciences*, 97(22), 12079-12084.
44. Tang, C., Zhang, L., Zhang, A., Ramanathan, M. (2001). Interrelated two-way clustering: an unsupervised approach for gene expression data analysis. In: *Proc. of the 2nd IEEE international Symposium on Bioinformatics and Bioengineering*, 41-48.
45. Wang, H., Wang, W., Yang, J., Yu, P. S. (2002). Clustering by pattern similarity in large data sets. In M. F. B. Moon, & A. Ailamaki (Eds.), *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 394-405.

46. Tanay, A., Sharan, R., Shamir, R. (2002). Discovering statistically significant biclusters in gene expression data. *Bioinformatics*, 18(suppl_1), S136–S144.
47. Tchagang, A.B., Tewfik, A. H. (2005). Robust biclustering algorithm (ROBA) for DNA microarray data analysis. *European Signal Processing Conference (EUSIPCO)*, Antalya, Turkey, 2005.
48. Liu, X., Wang, L. (2007). Computing the maximum similarity bi-clusters of gene expression data. *BIOINFORMATICS*, 23(1), 50–56.
49. Bozdağ, D., Parvin, J.D., Catalyurek, U.V. (2009). A Biclustering Method to Discover Co-regulated Genes Using Diverse Gene Expression Datasets. In: *Rajasekaran S. (eds) Bioinformatics and Computational Biology. Lecture Notes in Computer Science, Springer, Berlin, Heidelberg*, 5462, 151-163.
50. Chan, W.H. (2009). SNAP biclustering. Doctoral dissertation, Virginia Polytechnic Institute and State University.
51. Turner, H., Trevor B. and Wojtek K. (2005). Improved biclustering of microarray data demonstrated through systematic performance tests. *Computational Statistics and Data Analysis* 48(2), 235-254.
52. Getz, G.S., Levine, E., Domany, E. (2008). Coupled two-way clustering analysis of gene microarray data. *Proceedings of the National Academy of Sciences of the United States of America*.
53. Chen, J.R., Chang, Y. I. (2009). A condition-enumeration tree method for mining biclusters from dna microarray data sets. *Biosystems*, 97(1), 44-59.
54. Golumbic, M.C. (1980). Algorithmic graph theory and perfect graphs. New York: Academic Press.
55. Tanay, A., Analysis of Gene Expression Data Lecture 5. 2005.
56. Charnes, A., Cooper, W. W., Rhodes, E. (1999). Measuring the efficiency of decision making units. *European Journal of Operational Research*, 2(6), 429-444.
57. Seiford, L.M. (1999). Infeasibility of super-efficiency data envelopment analysis models. *INFOR Journal*, 37(2), 174-187.
58. Kaiser, S., Santamaria, R., Khamiakova, T., Sil, M., Theron, R., Quintales, L., Leisch, F., Troyer, E. (2015). *Package 'biclust'*. R Package "<https://cran.r-project.org/web/packages/biclust/index.html>", 2015.
59. Kruskal, W.H., Wallis, A. (1952). Use of ranks in one-criterion variance analysis. *Journal of the American Statistical Association*, 47(260), 583–621.
60. Dunn, O.J. (1964). Multiple comparisons using rank sums. *Technometrics*, 6, 241-252.
61. Tibshirani, R., Hastie, T., Eisen, M., Ross, D., Botstein, D., Brown, P. (1999). Clustering Methods for the Analysis of DNA Microarray Data.

62. Cho, R.J., Campbell, M. J., Winzeler, E. A., et al. (1998). A genome-wide transcriptional analysis of the mitotic cell cycle. *Molecular Cell*, 2, 65–73.
63. Alizadeh, A.A., Eisen, M. B., et al. (2000), Distinct types of diffuse large b-cell lymphoma identified by gene expression profiling. *Nature*, 403(3), 503-511.
64. Rengeswaran, B., Natarajan, A.M., & Premalatha, K. (2015), Stellar-mass black hole optimization for biclustering microarray gene expression data. *Applied Artificial Intelligence*, 29, 353-381.
65. Orzechowski, P., Boryczko, K. (2016). Propagation-based biclustering algorithm for extracting inclusion-maximal motifs. *Computing and Informatics*, 35, 391–410.
66. Xiao, J., Wang, L., Liu, X., Jiang, T. (2008). An efficient voting algorithm for finding additive biclusters with random background. *Journal Of Computational Biology*, 15(10), 1275–1293.
67. Zhang, J.D., Badi, L., Ebeling, M. (2013). Qualitative Biclustering with Bioconductor Package Rqubic.

APPENDICES

APPENDIX-1. Installing and loading packages in R software.

RcmdrPlugin.BiclustGUI library contains packages to apply CC, FLOC, Plaid model, xMotif, Bimax and Qubic algorithm. In addition, it contains Chia and Karuturi function and Coherence measures.

```
install.packages("RcmdrPlugin.BiclustGUI")
```

```
library(RcmdrPlugin.BiclustGUI)
```

biclust package is used to apply the CC, Plaid model, xMotif, and Bimax algorithms. In addition, it contains Chia and Karuturi function and Coherence measures.

```
install.packages("biclust")
```

```
library(biclust)
```

BicARE is used to apply the FLOC algorithm.

```
source("http://bioconductor.org/biocLite.R")
```

```
biocLite("BicARE")
```

```
library(BicARE)
```

rqubic is used to apply the Qubic algorithm

```
source("http://bioconductor.org/biocLite.R")
```

```
biocLite("rqubic")
```

```
library(rqubic)
```

APPENDIX-2. Using the biclustering libraries in R software.

Applying the CC algorithm`biclust(x, method=BCCC(), delta = 1.0, alpha=1.5, number=100)`*# Applying the FLOC algorithm*`FLOC(sample.bicData, k=15, pGene=0.3, pSample=0.6, r=0.01, 10, 8, 200)`*# Applying the Plaid model algorithm*`biclust(x, method=BCPlaid(), cluster="b", fit.model = y ~ m + a + b, background = TRUE,
background.layer = NA, background.df = 1, row.release = 0.7, col.release = 0.7, shuffle =
3, back.fit = 0, max.layers = 20, iter.startup = 5, iter.layer = 10, verbose = TRUE)`*# Applying the xMotif algorithm*`biclust(x, method=BCXmotifs(), ns=10, nd=10, sd=5, alpha=0.05, number=100)`*# Applying the Bimax algorithm*`biclust(x, method=BCBimax(), minr=2, minc=2, number=100)`*# Applying the Qubic algorithm*`quantileDiscretize(as.ExprSet(used_matrix), q=0.49,rank=7)``generateSeeds(data.disc,minColWidth=7)``quBicluster(seeds, eset, report.no = 100L, tolerance = 0.95, filter.proportion = 1)`

APPENDIX-3. Chia and Karuturi function and Coherence measures in R software.

Computing the classification scores as described in the paper of Chia and Karuturi.`ChiaKaruturi(x, bicResult, number)`*# Computing the Coherence measures following Madeira and Oliveira classification of biclusters.*`constantVariance(x, resultSet, number, dimension="both")``additiveVariance(x, resultSet, number, dimension="both")``multiplicativeVariance(x, resultSet, number, dimension="both")``signVariance(x, resultSet, number, dimension="both")`

CURRICULUM VITE

Personal Information

Surname / Name : HOMAIDA, Ammar
 Nationality : Syrian
 Date and Place of Birth : 23.07.1989, Aleppo
 Marital Status : Married
 Phone number : 0 (539) 305 41 31
 E-mail : ammarhomaيدا89@gmail.com
 Alternative E-mail : ammar.homaيدا@gazi.edu.tr



Education

Degree	School/Program	Graduation Date
MSc	Gazi University/Statistics	Ongoing
Undergraduate	University of Aleppo/Mathematical Statistics	2013
High School	Ahmad Shanan	2007

Professional Experience

Year	Place of Work	Position
2017-Ongoing	Ankara (Private work)	Translator
2015-Ongoing	Ankara (Private work)	SPSS and Statistics Private Instructor
2015-Ongoing	Ankara (Private work)	Statistical Research Specialist
2014-2015	University of Aleppo	Graduate Assistant
2008-2009	Al-Ber Pharmaceutical Distribution Company	Accountant and Deputy Manager

Native Language

Arabic

Foreign Languages

English, Turkish

Publication:

Homaida, A., Kocatürk, A., Altunkaynak, B. (2018). Comparing biclustering algorithms: A simulation study. *International Journal of Scientific & Engineering Research*, 9(5), 127-135.

Hobbies: Swimming, reading, table tennis.



GAZİ GELECEKTİR..