الجمهوريـــة الجزائريـــة الديمقراطيـــة الشعبيـــة

République Algérienne Démocratique et Populaire

وزارة الـتعلـيـم الـعـالي و الـبـحـث الـعلـمـي

Ministère de l’Enseignement Supérieur et de la Recherche Scientifique

Université MUSTAPHA Stambouli Mascara
Faculté des Sciences Exactes
Département d’Informatique

جامعة مصطفى أسطمبولي معسكر

THESE de DOCTORAT de 3ème cycle

Spécialité : Informatique

Option : Intelligence Artificielle

Intitulée

Algorithmes de Bi-regroupement pour l’Analyse des Données Complexes.

Présentée par : Charfaoui Younes

Le Jeudi 10/10/2024 **à 16h00**

Devant le jury :

Président	*Debakla mohamed*	*MCA*	*Univ Mascara*
Directeur de thèse	*BOUFERA FATMA*	*MCA*	*Univ Mascara*
C0- Directeur	*Houari Amina*	*MCB*	*Univ Mascara*
Examinateur	*MEFTAH BOUDJELAL*	*PR*	*Univ Mascara*
Examinateur	*Rebbah Mohamed*	*PR*	*Univ Mascara*
Examinateur	*Dahmani Youcef*	*PR*	*Univ Tiaret*

Année Universitaire : 2024/2025

I would like to dedicate this thesis to

My beloved parents

Your unwavering love, encouragement, and sacrifices have been the guiding light in my journey. This work is dedicated to you, the pillars of my strength. Your support has fueled my aspirations and shaped my path. I am forever grateful for the love and values you've instilled in me. This achievement is a reflection of the lessons learned from your wisdom. Thank you for being my constant source of inspiration.

My dear siblings, Latif, Lina, and Fouad

In every step of my journey, your closeness has been my source of joy and strength. This dedication is a testament to the bond we share and the shared memories that have enriched my life. Latif, with your cheerful spirit; Lina, with your kindness and laughter; Fouad, with your unwavering support – each of you has left an indelible mark on my heart. This accomplishment is as much yours as it is mine. Thank you for being my companions on this beautiful journey.

Acknowledgement

I extend my sincere appreciation to *Mrs. BOUFERA Fatma* for her unwavering support, guidance, and expertise throughout the course of my academic endeavors. Her commitment to excellence, insightful feedback, and encouragement have been instrumental in the successful completion of this thesis. I am truly grateful for her dedication to my growth and her invaluable contributions to my academic journey.

I would like to express my deep gratitude to *Mrs. HOUARI Amina* for her invaluable support and mentorship during the development of this thesis. Her expertise, encouragement, and constructive feedback have played a pivotal role in shaping the outcome of this research. I appreciate her commitment to fostering a conducive learning environment and for being an inspiring guiding force throughout my academic pursuit.

Abstract

Biclustering, a well-known bioinformatics technique, is essential for analyzing gene expression data because it reveals patterns and identifies groupings of genes that behave similarly under particular conditions. This thesis aims to contribute to the field through the introduction of three distinct approaches: a Differential Evolution-based method, a Multi-Objective Differential Evolution-based approach featuring a novel adaptive mutation operator known as BBDE, and a final method that employs Convolutional Denoising Autoencoders (CDAs) for preprocessing followed by Artificial Bee Colony (ABC) for biclustering. Each strategy displays its usefulness via comprehensive findings, contributing to the progress of biclustering techniques and improving gene expression data analysis in computational genomics.

Keywords: Biclustering, Microarray data analysis, Gene Expression, Differential Evolution, Multi-objective , Artificial Bee Colony, Convolutional Autoencoders.

Résumé

Le biclustering, une téchnique de bioinformatique bien connu, est essentiel pour analyser les données d'expression génique car il révèle des motifs et identifie des regroupements de gènes qui se comportent de manière similaire sous des conditions particulières. Cette thèse vise à contribuer au domaine en introduisant trois approches distinctes : une méthode basée sur l'évolution différentielle, une approche basée sur l'évolution différentielle multi-objectif avec un nouvel opérateur de mutation adaptatif appelé BBDE, et une méthode finale qui utilise des autoencodeurs de débruitage convolutionnels (CDA) pour le prétraitement suivi de l'essaim artificiel de colonies d'abeilles (ABC) pour le biclustering. Chaque stratégie démontre son utilité à travers des résultats complets, contribuant ainsi à l'avancement des techniques de biclustering et améliorant l'analyse des données d'expression génique en génomique computationnelle.

Mots Clée : Biclustering, Analyse de données de microréseaux, Expression génique, Évolution différentielle, Multi-objectif, Colonie d'abeilles artificielles, Autoencodeurs convolutionnels.

Contents

List of Figures **i**

List of Tables **ii**

1 General Introduction **1**

2 State Of The Art **5**

2.1 Introduction . . . 5

2.2 Biclustering . . . 5

2.2.1 Bicluster Definition . . . 6

2.2.2 Types of Biclusters . . . 6

2.2.3 Biclustering as an Optimization Problem . . . 7

2.2.4 Metrics . . . 7

2.3 Related Work . . . 10

2.3.1 Systematic Biclustering Algorithms . . . 10

2.3.2 Stochastic Biclustering Algorithms . . . 11

2.3.3 Optimization Frameworks in Biclustering . . . 12

2.3.4 Comprehensive Overview of Exisiting Biclustering Methods . . . 12

2.3.5 Evolutionary Algorithms and Swarm Intelligence for Biclustering Optimization . . . 17

2.3.6 Analysis . . . 18

2.3.7 Addressing Noise in Gene Expression Data . . . 19

2.4 Conclusion . . . 21

3 DeBic: A Differential Evolution Biclustering Algorithm for Microarray Data Analysis. **22**

3.1 Introduction . . . 22

3.2 Differential Evolution . . . 22

3.2.1 Algorithm of Differential Evolution . . . 24

3.2.2 Mutation Operator 25
3.2.3 Crossover Operator 26
3.2.4 The benefits of the DE Algorithm 27
3.3 DeBic Algorithm Description 27
3.3.1 Algorithm Description 28
3.4 Experimental Results 30
3.5 Conclusion 34

4 AMoDeBic: An Adaptive Multi-Objective Differential Evolution Biclustering Algorithm of Microarray Data Using a Biclustering Binary Mutation Operator 35
4.1 Introduction 35
4.2 Differential Evolution 36
4.2.1 Algorithm of Differential Evolution 36
4.2.2 Multi-objective Differential Evolution 36
4.2.3 Binary Differential Evolution 38
4.3 AMoDeBic Algorithm 41
4.3.1 Algorithm Description 41
4.3.2 Illustrative Example 45
4.3.3 Parameter Setting 47
4.3.4 Computational Complexity 48
4.3.5 Multi-Objective Fitness Functions 49
4.4 Experimental Results 50
4.4.1 Results on Synthetic Data 51
4.4.2 Results on Real Data 54
4.5 Conclusion 63

5 CDABC: A Convolutional Denoising Autoencoder with Artificial Bee Colony Biclustering Algorithm for Gene Expression 64
5.1 Introduction 64
5.2 Autoencoders 64
5.2.1 Denoising Autoencoders 65
5.2.2 Convolutional Denoising Autoencoders 65
5.3 Artificial Bee Colony 66
5.4 Proposed Approach 69
5.4.1 Representation 69
5.4.2 Pre-processing: CDAE 69

5.4.3 Biclustering: ABC 71
5.4.4 Illustrative Example 75
5.5 Experimental Results 77
5.5.1 Synthetic Data Results 77
5.5.2 Real Data Results 80
5.6 Conclusion 88

6 General Conclusion **89**

Bibliography **93**

A Biclustering Metric Calculation Examples **A**

List of Figures

2.1 Types of Biclusters. 7

3.1 Representation of Binary String Encoding for Bicluster Solution. 28
3.2 14 biclusters Discovered By DeBic in The Yeast Cell-Cycle Dataset. 32

4.1 Pareto Optimality Visualization in 2D Objective Space. 37
4.2 Representation of Binary String Encoding for Bicluster Solution. 42
4.3 Noise Effect. A: Constant Biclusters, B: Additive Biclusters. 53
4.4 Overlap Effect. A: Constant Biclusters, B: Additive Biclusters. 54
4.5 Proportions of biclusters significantly enriched by GO annotations (Saccharomyces Cerevisiae dataset) 58
4.6 Proportions of biclusters significantly enriched by GO annotations (Yeast cell-cycle dataset) 59
4.7 Nine Biclusters discovered in the Yeast dataset. 61
4.8 Nine Biclusters discovered in the Saccharomyces Cerevisiae dataset. 61

5.1 Representation of Binary String Encoding for Bicluster Solution. 69
5.2 Architecture of the Proposed CDAE. 70
5.3 Noise Effect. A: Constant Biclusters, B: Additive Biclusters. 79
5.4 Overlap Effect. A: Constant Biclusters, B: Additive Biclusters. 80
5.5 Proportions of Biclusters Significantly Enriched by GO Annotations (Yeast cell-cycle Dataset) 83
5.6 Proportions of Biclusters Significantly Enriched by GO Annotations (Saccharomyces Cerevisiae Dataset) 84
5.7 Six Biclusters Discovered in the Yeast Cell-Cycle Dataset by CDABC. . . . 86
5.8 Six Biclusters Discovered in the Saccharomyces Cerevisiae Dataset by CDABC. 87

List of Tables

3.1 Hyper Parameter Specification . . . 31
3.2 Biological Processes and Corresponding P-values in the Previous Bicluster. 33
3.3 DeBic Performance Comparison To Other Algorithms. . . . 34

4.1 Hyper Parameter Specification. . . . 51
4.2 Human B-Cell Lymphoma coverage for different algorithms. . . . 56
4.3 Yeast Cell-Cycle coverage for different algorithms. . . . 56
4.4 Five biclusters found By AMoDeBic on Yeast dataset. . . . 57
4.5 Five biclusters found By AMoDeBic on the Human B-Cell Lymphoma dataset. . . . 57
4.6 AMoDeBic Performance Comparison To Other Algorithms. . . . 59
4.7 Significant GO terms of two biclusters extracted from Yeast Cell-Cycle dataset using AMoDeBic. . . . 62
4.8 Significant GO terms of two biclusters extracted from Saccharomyces Cerevisiae dataset using AMoDeBic. . . . 62

5.1 Comparison of Yeast Cell-Cycle Coverage by Different Algorithms. . . . 82
5.2 Comparison of Human B-Cell Lymphoma Coverage by Different Algorithms. 82
5.3 Significant GO terms of two biclusters extracted from Yeast Cell-Cycle dataset using CDABC. . . . 85
5.4 Significant GO terms of two biclusters extracted from Saccharomyces Cerevisiae dataset using CDABC. . . . 85

List of Algorithms

1 Differential Evolution with Scheme DE/rand/1/bin 24
2 Binomial Crossover 26
3 Exponential Crossover 27
4 DeBic Algorithm Steps 30
5 AMoDeBic Algorithm Steps 45
6 CDABC Algorithm 74

Chapter 1

General Introduction

Context and Motivation

The rapid expansion of data in many domains is changing the way that information is used and processed. Large-scale datasets are produced by fields including healthcare, finance, social sciences, and environmental research, which poses particular management and interpretation difficulties. This increase in data necessitates advanced methods that can effectively manage large volumes while deriving insightful information. Big data is becoming a vital component of contemporary research and application, thus analytical tools need to be not just rapid and accurate but also scalable, noise-resistant, and capable of identifying intricate patterns in data.

Biclustering has emerged as an effective analytical method for handling big data's complexity in a variety of fields. In contrast to conventional clustering techniques, which examine data along a single dimension at a time, biclustering enables the simultaneous grouping of rows and columns, identifying significant patterns that could only be present in particular dataset subsets. As an approach, it is especially well-suited for high-dimensional data because of its dual-dimension methodology, which allows it to reveal subtle patterns and localized associations that could otherwise go overlooked. Biclustering is an effective and adaptable way to manage and comprehend big data by breaking up massive datasets into logical groupings that make it easier to find pertinent correlations.

Big data difficulties are particularly noticeable in the field of genomics. Large-scale data gathering has been made possible by the quick development of genomic technology; microarray analysis, for example, has produced enormous volumes of gene expression data. Rich insights into genetic mechanisms and their implications across a variety of domains, including evolutionary biology, environmental studies, and biomedical research, are provided by this data, which is arranged into matrices with genes along rows and circumstances along columns. Nevertheless, in order to uncover significant patterns, the size and complexity of genetic data require sophisticated computational techniques.

In genomics, where conventional clustering frequently overlooks the complex interactions between genes and their expressions under various conditions, Biclustering is especially helpful. It provides a deeper understanding of gene regulation and interaction networks by identifying gene groupings that exhibit consistent expression characteristics under particular subsets of conditions. As a result, biclustering becomes the perfect technique for examining huge genomic datasets, emphasizing pertinent biological processes and offering new opportunities for investigation and learning.

The biclustering problem was introduced by [1], signifying a notable milestone in the field. The initial solution was devised by [2], who formulated a greedy search heuristic technique, offering an efficient approach to generate biclusters with approximate optimality. Subsequent to these foundational contributions, the field of biclustering has experienced a surge of innovations. For example, [3] proposed FLOC, a technique explicitly crafted to concurrently reveal multiple, potentially overlapping biclusters. Furthermore, [4] introduced a novel biclustering approach, centering on the identification of order-preserving sub-matrices (OPSM). Additionally, [5] introduced the reference technique Bimax, utilizing a straightforward binary reference model. This progression of methodologies underscores the diversification and expansion of biclustering techniques, each tailored to address specific aspects and nuances within the broader domain of data clustering and pattern discovery.

Furthermore, the increasing adoption of evolutionary algorithms (EAs) in biclustering is attributed to their effectiveness in navigating complex optimization challenges. For instance, [6] proposed an EA framework for biclustering, refining Cheng and Church's method by integrating an EA-based local search. Another notable contribution comes from [7], who introduced a multi-objective EA biclustering algorithm featuring integrated local search heuristics. Additionally, [8] suggested an EA-based biclustering method incorporating four objectives: Mean Squared Residue (MSR), row variance, bicluster size, and overlapping ratio. The prevalence of evolutionary algorithm based biclustering approaches emphasizes the considerable computational complexity associated with biclustering, a challenge further underscored by its classification as an NP-hard problem [9]. The NP-hardness of biclustering indicates the computational intricacy involved in finding an optimal solution or arranging biclusters to meet specific criteria within a reasonable timeframe.

In conjunction with this, the expansive nature of gene expression data introduces a dynamic dimension to the computational intricacies faced by evolutionary algorithm (EA)-based biclustering approaches. Due to the expansion of the datasets, the search space and the potential number of biclusters follow an exponential trajectory. The vastness of this expanding space poses a formidable challenge for exhaustive search and exploration. Consequently, existing biclustering algorithms, exemplified by established approaches such as [10], [11], [10], grapple with the intricate task of navigating this extensive search space efficiently. Moreover, the struggle to effectively balance various objectives becomes more pronounced, hindering the identification of biclusters that strike a harmonious compromise between conflicting criteria.

Research Contributions

In light of these shortcomings, the research presented in this work contributes to the evolving landscape of biclustering and computational genomics.

1. DeBic: The first Differential Evolution (DE) based biclustering algorithm, it harnesses the robust search capabilities of DE, it's strengths in global optimization and effectiveness in handling high-dimensional spaces. DeBic aims to simultaneously and efficiently detect multiple high-quality biclusters through the adaptation of Differential Evolution (DE) to a binary search space, coupled with the implementation of a different mutation operator.

2. AMoDeBic: a novel approach that capitalizes on the inherent adaptability, robustness, and efficacy of differential evolution. Furthermore, it takes advantage of the multi-objective framework, enabling the concurrent optimization of multiple objectives. In addition, We have specifically crafted a new mutation operator, named Biclustering Binary Differential Evolution Mutation (BBDE), designed to adapt DE to biclustering and tackle it's constraints. This addition significantly boosts the efficacy and performance of Differential Evolution (DE), with the aim of delivering efficient solutions for gene expression data analysis. The enhanced DE approach yields coherent outcomes and has the capacity to recognize multiple biclusters simultaneously.

3. CDABC: This approach integrates Convolutional Denoising Autoencoders (CDAEs)

for preprocessing and denoising, along with the Artificial Bee Colony (ABC) algorithm. CDAEs aim to reduce noise and uncover underlying patterns in the gene expression data, enhancing the biclustering process executed by the ABC algorithm, renowned for its effectiveness in optimizing complex problems. Specifically, we incorporated few adaptations in the ABC algorithm to enable its direct application on binary representations within biclusters. By combining the strengths of CDAEs and adapted ABC, our approach enhances the identification of meaningful gene expression patterns and facilitates more accurate bicluster discovery.

Thesis Organization

Following the initial chapter, which serves as the general introduction, the subsequent sections of this thesis are organized as follows:

- Chapter 2: diligently explores the realm of biclustering, providing an all-encompassing overview of current approaches while conducting an in-depth analysis of their respective strengths and limitations.
- Chapter 3: immerses itself in the domain of differential evolution, undertaking a meticulous exploration of the algorithm's principles, steps, and strengths. Furthermore, it introduces and elucidates the DeBic algorithm, revealing its underlying principles, algorithmic details, and showcasing results achieved in the specific context of biclustering gene expression data.
- Chapter 4: extends the exploration of differential evolution (DE) to encompass its multi-objective framework and binary DE. These components are crucial in introducing the AMoDeBic algorithm, accompanied by the innovative mutation operator BBDE, a novel addition to the field of biclustering. The chapter meticulously outlines the algorithm's structure and highlights compelling results obtained through its application.
- Chapter 5: conducts a comprehensive examination of both autoencoders and Artificial Bee Colony. After a thorough review, we introduce the CDABC algorithm, a fusion of these two techniques designed for efficient biclustering. The chapter reveals algorithmic details, providing a nuanced understanding of its inner workings, and presents convincing results that validate its effectiveness.
- Chapter 6: A concluding section where we concisely summarize our innovative contributions and essential findings, and outline potential avenues for future research in the evolving field of biclustering and computational genomics.

Chapter 2

State Of The Art

2.1 Introduction

In the expansive domain of biclustering, a variety of strategies have emerged to solve the challenging task of identifying patterns within two-dimensional data matrices. This chapter provides a structured exploration of biclustering, beginning with a comprehensive review of its fundamental components. We will explain the formal concept of biclustering, distinguish between the many types of biclusters, and highlight the metrics utilized in the quantitative evaluation of biclustering outcomes.

Proceeding to the cutting edge of biclustering research—the state-of-the-art biclustering approaches—our study will thoroughly analyze these advanced techniques, examining their fundamental concepts, strengths, and applications. This investigation aims to offer a comprehensive overview of the landscape of biclustering.

2.2 Biclustering

Biclustering is a technique that aims to identify the best biclusters within a given dataset by arranging data into a matrix and simultaneously assigning the rows and columns of the matrix.

Biclustering is an essential task that reveals hidden patterns within large datasets, allowing insights that standard approaches might overlook. Biclustering in gene expression analysis reveals genes that exhibit similar activity under certain conditions, contributing to the study of biological processes, disease profiling, and medication responses [12], [13]. Beyond genomics, biclustering has applications in text mining, image analysis, and social network analysis, aiding targeted marketing, picture segmentation, and community recognition. Biclustering improves knowledge extraction and decision-making across varied domains by overcoming the obstacles provided by high-dimensional data, making it a vital tool for pattern detection and data-driven discovery.

A DNA microarray dataset is presented in a data matrix in genomic data, where genes are represented by rows, conditions are represented by columns, and gene expression levels are represented by cells. The formal definition of this data matrix is as follows: $G = \{1, 2, ..., n\}$ a set of indices of n genes, $C = \{1, 2, ..., m\}$ a set of indices of m conditions, and $M(G, C)$ the data matrix associated with G and C.

In this section, we'll explore the definition of biclusters, their various types, and the metrics utilized to evaluate them. Additionally, we'll delve into biclustering as an optimization problem.

2.2.1 Bicluster Definition

A bicluster is a group of genes that are associated with a group of conditions where these genes are co-expressed. Thus, it is considered to be a subset of the M matrix; it can be formally defined as follows: A bicluster is a pair $(g; c)$, where $g \in G$ and $c \in C$.

This study aims to find as many consistent biclusters as possible, which indicate a set of genes that exhibit the same behavior under the same set of conditions.

2.2.2 Types of Biclusters

Madeira and Oliveira thoroughly analyze a wide range of bicluster types in [9], providing detailed insights into particular choices among these types. The following is a brief description of several of these types:

2.2.2.1 Bicluster with constant values

Constant biclusters have consistent gene expression values across genes and conditions, indicating homogeneity and consistent gene behavior. They also provide stability and are less affected by experimental noise, making it easier to identify genes with strong expression patterns. Additionaly, they reveal genes that are crucial to essential activities, metabolism, or structural components that remain stable under varying settings. As a result, these biclusters serve an important role in identifying baseline expression or vital housekeeping genes required for cellular functioning.

2.2.2.2 Bicluster with constant values on rows or columns

In contrast to constant biclusters, which have consistent values across both genes and conditions simultaneously, biclusters with constant values on rows or columns represent consistent gene expression within specified conditions. These biclusters uncover genes that stay stable under certain circumstances, assisting in the discovery of essential biological processes or invariant reactions. Their uses range from identifying housekeeping genes to gaining a better understanding of stable gene relationships.

2.2.2.3 Bicluster with coherent values

Biclusters with coherent values demonstrate synchronized expression patterns across genes and conditions and emphasize genes whose expression varies over time, assisting in the decoding of dynamic processes and identifying genes involved in specific pathways. In contrast to biclusters with constant values, they capture dynamic shifts and trends, improving our knowledge of temporal biological events.

3	3	3	3
3	3	3	3
3	3	3	3
3	3	3	3

Constant Bicluster

3	3	3	3
4	4	4	4
1	1	1	1
0.5	0.5	0.5	0.5

Row Constant Bicluster

3	4	1	0.5
3	4	1	0.5
3	4	1	0.5
3	4	1	0.5

Column Constant Bicluster

3	5	6	9
0	1	3	8
2	3	5	8
0.5	3	3	5

Coherent Bicluster

Figure 2.1: Types of Biclusters.

2.2.3 Biclustering as an Optimization Problem

Since efficiency is the foundation of the biclustering process, extracting biclusters from a data matrix may be considered a combinatorial optimization issue. The biclustering problem seeks to extract maximally sized biclusters that meet a coherence requirement. This allows the computer to sift through and sort a massive quantity of data in less time than single clustering methods.

2.2.4 Metrics

The vast majority of algorithm comparisons are performed on synthetic or real-world datasets, with metrics or measurements used for evaluation or validation. In the literature, several bicluster assessment measures have been proposed [14]. We will look at some of the most common bicluster metrics.

2.2.4.1 Mean Squared Residue (MSR)

Cheng and Church proposed the MSR metric [2], which is the most widely used coherence measure in biclustering algorithms [14]; it calculates the average squared difference between gene expression values and the bicluster's mean expression value. It assesses how well the genes inside the bicluster align with the central tendency of the bicluster. MSR for a bicluster B that consists of I rows and J columns is defined as follows:

$$MSR(B) = \frac{1}{|I|.|J|} \sum_{i=1}^{|I|} \sum_{j=1}^{|J|} (b_{ij} - b_{iJ} - b_{Ij} + b_{IJ})^2 \tag{2.1}$$

Where:
b_{ij}: represents the expression value of gene i under condition j within the bicluster.
b_{iJ}: represents the mean expression value of all genes across condition j in the bicluster. It is calculated as follows: $b_{iJ} = \frac{1}{|J|} \sum_{j \in J} b_{ij}$
b_{Ij}: represents the mean expression value of gene i across all conditions in the bicluster. It is calculated as follows: $b_{Ij} = \frac{1}{|I|} \sum_{j \in J} b_{ij}$
b_{IJ}: denotes the overall mean expression value of the bicluster. Calculated as follows: $b_{IJ} = \frac{1}{|I|.|J|} \sum_{i \in I, j \in J} b_{ij}$

2.2.4.2 Variance (Var)

Hartigan proposed the variance measure to locate biclusters with constant values [1]. It is a coherence metric that assesses the variation in gene expression values within a cluster. It measures how much the expression levels of genes within the bicluster depart from the mean expression value of the bicluster. Greater variance suggests more variability in gene expression levels within the bicluster. Formally, it is as follows:

$$Var(B) = \sum_{i=1}^{|I|} \sum_{j=1}^{|I|} (b_{ij} - b_{Ij})^2 \tag{2.2}$$

Where:
B: refers to the bicluster under consideration.
$|I|$ and $|J|$: the numbers of genes and conditions in the bicluster, respectively. b_{ij}: represents the expression value of gene i under condition j within the bicluster.

b_{Ij}: represents the mean expression value of gene i across all conditions in the bicluster. The Variance metric calculates the squared difference between individual expression values b_{ij} and the mean expression value b_{Ij} of gene i across all conditions in the bicluster. These squared differences are accumulated by the summing terms for all elements inside the bicluster.

The row variance (rVar) like the variance measure, assesses the diversity of gene expression within a bicluster. The variance is then normalized by the number of genes and conditions in the bicluster. The formal definition of the rVar measure is:

$$rVar(B) = \frac{1}{|I|.|J|} \sum_{i=1}^{|I|} \sum_{j=1}^{|I|} (b_i j - b_I j)^2 \tag{2.3}$$

2.2.4.3 Relevance Index (RI)

Yip et al. developed a new assessment metric known as the relevance index (RI) [15] . RI evaluates the quality of a bicluster as the sum of the relevance indices of the columns. The relevance index R_{Ij} for column $j \in J$ is defined as follows:

$$R_{Ij} = 1 - \frac{\sigma_{Ij}^2}{\sigma_j^2} \tag{2.4}$$

Where:
σ_{Ij}^2: refers to the variance of the values in column j for the bicluster.
σ_j^2: represents the variance of the values in column j across the entire dataset.

2.2.4.4 Virtual Error (VE)

Pontes et al introduced the VE function to detect shifting or scaling patterns in biclusters [16] . VE measures the difference between a gene's expression values in a bicluster and the predicted profile for that gene across all circumstances within the bicluster. Given a bicluster B containing I genes and J conditions, VE is defined as follows:

$$VE(B) = \frac{1}{I.J} \sum_{i=1}^{i=I} \sum_{j=1}^{j=J} (b'_{ij} - P'_i) \tag{2.5}$$

Where:
b'_{ij}: represents the expression value of gene i under condition j in the bicluster.

P'_i: the expected profile value of gene i across all conditions within the bicluster.

2.3 Related Work

CC, introduced by Cheng and Church [2], stands as a groundbreaking biclustering algorithm tailored for gene expression data analysis. This approach effectively addresses the limitations found in conventional clustering methods, recognizing the diverse biological functions of individual genes and the varying behavior of gene groups under different conditions. Biclustering navigates the intricate landscape of noisy gene expression data by pinpointing specific groups of genes and conditions within a matrix that share similar patterns. Beyond its applications in microarray and gene expression studies, Biclustering has spread across various fields, including pattern discovery [17], text mining [18], marketing [19], web searches [20], and biomedicine [21]. However, given the challenges posed by noise and the vast number of potential combinations, discerning the inclusion of rows and columns in a bicluster becomes a complex task. Biclustering, classified as an NP-hard problem, faces computational intricacies as the scale of input data increases. Consequently, many biclustering algorithms adopt heuristic search methods to identify biclusters and approximate the problem by locating suboptimal solutions. In the literature, biclustering approaches are broadly categorized into two main types: the systematic search approach (Heuristic) and the stochastic search (Metaheuristic).

2.3.1 Systematic Biclustering Algorithms

The systematic search-based biclustering algorithms includes the following approaches:

2.3.1.1 Divide and Conquer Approach

The Divide and Conquer technique begins with the complete data matrix as the initial bicluster. Then, repeatedly partition this bicluster into numerous biclusters that satisfy specific properties until the termination requirements are verified, as shown in [22]. This method is characterized by its speed; however, it may eliminate promising biclusters by dividing them before assessing them. Bimax [5] is an instance of a biclustering approach that uses divide and conquer strategy

2.3.1.2 Greedy Iterative Search Approach

This method constructs biclusters using an iterative procedure that tries to maximize or decrease specific functions. This procedure is based on adding or removing rows or columns until no further additions or deletions are possible. Despite its speed, this approach has the disadvantage of disregarding some good biclusters by adding or deleting rows or columns. OPSM [4] and BicFinder [23] are two algorithms that take this technique.

2.3.1.3 Bicluster Enumeration Approach

This approach identifies the best biclusters, which requires an exhaustive enumeration of all possible biclusters in the data matrix. The advantage of this approach is the certainty of finding the best biclusters, but its major drawback is its long duration of calculation and a significant amount of memory use. Among the algorithms using this approach, there are BiMine [24], BiMine+ [25], and DeBi [26].

2.3.2 Stochastic Biclustering Algorithms

The stochastic search-based biclustering algorithms includes the following approaches:

2.3.2.1 Neighborhood Search

This method begins with a solution: the complete data matrix or just a bicluster. Then, each iteration improves the present solution by adding or removing rows or columns to increase or decrease particular functions. Unlike the Greedy Iterative Search Approach, the previously removed rows or columns can be added back in this strategy. CC Algorithm [2], PDNS [27], and LSM [10] are examples of algorithms that use this method.

2.3.2.2 Evolutionary Algorithm (EA)

This technique begins by initializing the population, where each individual represents a potential solution for this population. It then evaluates each individual using an evaluation function and picks a certain number among them to form the next population using the crossover and mutation operators; this operation is repeated until the stopping condition is met. However, EA can be costly to build and time demanding to run. Furthermore, the results might be challenging to understand. SEBI [8], SMOB [28], CBEB [29], and those presented by Nepomuceno et al. in [11] [30] , MOEA [7], and DeBic [31] are some of the algorithms that use this method.

2.3.2.3 Hybrid Approaches

Hybrid biclustering approaches integrate numerous methodologies to improve the quality, robustness, and interpretability of biclustering. They increase solution quality, navigate solution spaces more robustly, and provide a clearer understanding of patterns by integrating varied algorithms. However, they confront obstacles regarding methods compatibility, parameter adjustment, and computing complexity. In addition, integrating approaches can cause assumptions and objectives clash, and determining optimal parameters can be difficult and time-consuming. BiHEA [32] and SSB [33] are instances of hybrid biclustering approaches.

2.3.3 Optimization Frameworks in Biclustering

Biclustering algorithms often operate within optimization frameworks, classifiable as single-objective or multi-objective [34].

1. *Single-Objective Optimization Framework:* In this framework, algorithms seek a singular, optimal solution by optimizing a predefined criterion or objective function. Examples include SEBI [8] ,BiHEA [11], CBEB [29], SSB [33], BPSO [35].

2. *Multi-Objective Optimization Framework:* Given this framework, deals with problems featuring conflicting objectives, aiming to find a set of solutions that represent trade-offs among these objectives. Examples include MOEA [7], SMOB [28], MOSPO [36], MOACO [37].

2.3.4 Comprehensive Overview of Exisiting Biclustering Methods

1. *Bimax:* Bimax [5] is a biclustering algorithm that uses a straightforward data representation to find optimal biclusters efficiently. It employs a binary data model where genes and conditions are either present (1) or absent (0) in a bicluster. The algorithm scans each gene and expands conditions while maintaining the gene's presence. This approach is simple and fast, making it a good baseline for understanding biclustering concepts and comparing with more complex methods. However, its simplicity might overlook intricate patterns in the data.

2. *OPSM:* The OPSM (Order-Preserving Sub-Matrices) algorithm [4] introduces a unique approach to biclustering by defining biclusters as sub-matrices where gene expression levels adhere to a consistent linear ordering of experiments. The algorithm's main objective is to identify rows that exhibit similar ordering tendencies

within these biclusters. OPSM employs a probabilistic model as its guiding mechanism, which contributes to the efficient identification of order-preserving biclusters.

3. *ISA:* ISA [38] adopts a versatile approach with a generalized Singular Value Decomposition (SVD) technique to unveil transcription modules. Beginning with randomly selected genes or conditions, the algorithm progressively refines the composition of modules based on defined criteria. This process involves iterative data normalization and the use of resolution-controlling thresholds. Each iteration produces a distinct bicluster, and the initial random seed selection allows for potential overlap of genes and conditions in different biclusters.

4. *Xmotif:* This algorithm [39] introduces a unique perspective by utilizing conserved gene expression motifs, called Xmotifs, which represent gene subsets consistently expressed across specific samples. These motifs are defined using gene expression intervals with constraints on size, conservation, and maximality. A probabilistic algorithm exploits the mathematical structure of Xmotifs to compute the largest one. Through iterative iterations, samples corresponding to each Xmotif are gradually removed, unveiling multiple Xmotifs. However, this approach's reliance on iterative sample removal could potentially lead to information loss and biases in the discovered Xmotifs. The algorithm's performance is closely tied to the initial sample selection, impacting the quality and diversity of the resulting Xmotifs.

5. *Samba:* Samba [40] approaches biclustering through a unique bipartite graph representation of data. They assign weights to edges using probabilistic models that highlight potential biclusters. The algorithm introduces two graph models, one of which incorporates the direction of expression changes. By iteratively utilizing a polynomial method, Samba identifies significant biclusters as weighted sub-graphs. It employs a selection process to find the heaviest sub-graphs, adapting to both signed and unsigned models based on the data. This adaptability enhances the algorithm's robustness, allowing for versatile bicluster discovery.

6. *BicFinder:* BicFinder [23] is a specialized greedy algorithm designed for biclustering that uses a novel evaluation function named the Average Correspondence Similarity Index (ACSI). This function serves as a guiding metric to evaluate the quality of biclusters. The approach also involves the implementation of a directed acyclic graph, which is employed as a strategic tool for extracting biclusters from the input data. These combined features enable BicFinder to effectively identify meaningful patterns within DNA microarray data sets. However, Due to its greedy nature, it might not always lead to globally optimal solutions.

7. *BiMine:* BiMine [24] incorporates three unique aspects. Initially, it employs an innovative assessment metric termed Average Spearman's rho (ASR). Secondly, BiMine utilizes a novel tree structure termed Bicluster Enumeration Tree (BET) to catalog the diverse biclusters uncovered during enumeration. Lastly, BiMine introduces a parametric guideline to curtail the enumeration process, mitigating the proliferation of tree branches that might not yield promising biclusters.

8. *BiMine+:* This is a heuristic enumeration biclustering algorithm designed to extract substantial and coherent biclusters [25]. These biclusters encompass sizable sets of genes and conditions, showing strong correlations in their activities across diverse conditions. The algorithm utilizes a Modified Bicluster Enumeration Tree (MBET) to depict the identified biclusters. Each MBET node holds the gene profile shape of a bicluster. The gene profile shape signifies the pattern of gene expression changes across the conditions. However, it may face scalability challenges with large datasets due to its exhaustive enumeration approach.

9. *DeBi:* The DeBi (Differentially Expressed BIclusters) algorithm [26] introduces a rapid biclustering approach based on frequent itemset mining, a widely recognized data mining technique. DeBi's primary objective is to uncover homogeneous biclusters of maximum size, where genes exhibit strong associations with specific subsets of samples. However, DeBi's dependence on frequent itemset mining may present difficulties when handling large or intricate datasets. Furthermore, its emphasis on gene-sample associations could potentially lead to the oversight of subtle patterns in the data.

10. *CC:* The CC algorithm [2] is designed to generate biclusters using an input expression matrix and an MSR threshold. This method employs a series of phases, involving both node deletion and addition, to maintain the structure of biclusters. To address the problem of overlap, a substitution phase is employed, replacing elements within newly identified biclusters with random values. While the CC algorithm effectively avoids overlap, it has certain limitations. These include potential issues such as element masking and the reliance on a dataset-specific threshold, which can affect the rejection of solutions.

11. *PDNS:* The PDNS algorithm [27] is centered around iterative enhancement of an initial bicluster candidate solution. It does so by systematically exploring the neighborhood of the candidate solution and making incremental modifications to improve its quality. PDNS generates its results through a series of runs, each with a distinct initial solution. However, it's important to note that PDNS generates one bicluster

at a time, and overlapping among reported solutions is not addressed. The algorithm's output is not a comprehensive set of biclusters, but rather a sequence of individual biclusters.

12. *SEBI:* The SEBI [8] algorithm utilizes a weighted matrix to regulate the degree of overlap among biclusters. By controlling the sharing of genes and conditions between biclusters, SEBI manages to control overlap effectively. The algorithm employs three different strategies for manipulating genetic information: one-point crossover, two-points crossover, and uniform crossover. However, one limitation of the SEBI algorithm is that it generates only a single bicluster per run. This requires multiple iterations to identify multiple biclusters effectively

13. *SMOB:* SMOB (Sequential Multi-Objective Biclustering) [28] adopts a sequential approach similar to SEBI, but it diverges in terms of its optimization strategy. While SEBI employs a single-objective evolutionary algorithm, SMOB leverages a multi-objective evolutionary algorithm (MOEA) in an iterative manner. In each iteration of the MOEA, SMOB identifies a bicluster and stores it within a list. The size of this list corresponds to the number of solutions generated by the MOEA calls

14. *CBEB:* The CBEB algorithm [29] combines genetic algorithms and hierarchical clustering to identify biclusters. The method partitions rows of the data matrix into subsets of conditions and then employs parallel genetic algorithms on each subset. The results from these parallel genetic algorithms are combined using an expanding-merging approach to produce biclusters. This method efficiently integrates different techniques to address the biclustering problem. However, its strength lies in producing a single bicluster per iteration, which might necessitate multiple iterations for discovering multiple biclusters.

15. *MOEA:* The Multi-Objective Evolutionary Algorithm (MOEA) [7] is an optimization method inspired by natural evolution. It maintains a population of potential solutions, applies selection, crossover, and mutation to create new solutions, and prioritizes those with better objective values. In addition, Local search heuristics are employed to speed up convergence by refining the chromosomes.

16. *DeBic:* DeBic [31] is a biclustering algorithm that adapts the principles of Differential Evolution (DE) to identify coherent patterns in gene expression data. It starts by generating a population of solutions, and iteratively evolves the population through crossover and mutation operations.

17. *BiHEA:* The BiHEA algorithm [32] incorporates a local search approach that draws inspiration from the CC algorithm. It employs a two-point crossover operator to generate new solutions. Additionally, BiHEA integrates an external archive, which serves as a repository to preserve the best biclusters throughout the evolutionary process. However, BiHEA's reliance on a local search strategy could potentially limit its exploration of the solution space. As a result, it might face challenges in discovering diverse and more global optimal solutions.

18. *SSB:* The SSB algorithm [33] employs binary encoding for its biclustering solutions and incorporates a local search method to enhance biclusters containing positively correlated genes. While these approaches are beneficial, they rely on predefined patterns and may not capture more complex relationships within the data. This can limit their effectiveness in identifying intricate underlying structures.

19. *FLOC:* FLOC [3] is an innovative approach that extends the bicluster model to include null values. It introduces a probabilistic algorithm capable of identifying a collection of potentially overlapping biclusters simultaneously. The FLOC process involves iterations of gene and condition moves, aiming to minimize residual elements optimally. In some cases, specific moves might be temporarily "blocked" to prevent undesirable outcomes, like generating trivial biclusters or violating feature constraints.

20. *MBA:* MBA [41] was designed to uncover negatively correlated genes within microarray data. MBA operates on a set of potential biclusters, leveraging them to generate new solutions via variation operators like combinations and local enhancements. The utilization of a behavior matrix representation guides the local enhancement process, using positive and negative pattern-based neighborhoods.

21. *DdPGA:* the Dynamic deme Parallelized Genetic Algorithm (DdPGA) [42] is a global parallelization (master-slave model) influenced by coarse-grained GA with overlapping sub-population model. The primary goal is to mine biclusters with high row variance, low mean square residue, and low overlapping (MSR).

22. *EBA:* EBA [43] is an evolutionary biclustering algorithm based on genetic operators such as selection, crossover, and mutation. They proposed two distinct methods for each of these operators. The first method is based on a parallel approach and employs four complementary functions: the size function, the mean squared residue function, the average correlation function, and the coefficient of variation function, whereas the second method is based on the aggregation of two functions: the

size function and the average correlation function. In addition, they proposed new crossover and mutation methods for the biclustering problem .

2.3.5 Evolutionary Algorithms and Swarm Intelligence for Biclustering Optimization

Within the realm of biclustering, evolutionary algorithms and swarm intelligence stand out as two remarkable approaches inspired by natural phenomena. They provide a systematic and iterative framework that allows the algorithm to adapt and refine solutions over successive generations, enabling them to navigate the complex search space and overcome challenges posed by noise and diverse patterns in gene expression data. The ability of EAs to strike a balance between exploration and exploitation makes them particularly well-suited for addressing the intricate nature of biclustering problems, making them a valuable asset in advancing the field. These algorithms, namely GA, PSO, ACO, ABC, have demonstrated remarkable capabilities and produced impressive results in the field of biclustering.

GA (Genetic Algorithm) is a nature-inspired approach that falls under the category of Evolutionary approaches, with inspiration drawn from the micro-evolutionary processes that underpin animal evolution. A random population of potential solutions encoded as chromosomes serves as the basis for genetic algorithms. New offspring are created through crossover and mutation, with parents chosen based on fitness. The cycle repeats until the termination requirements are met. Swarm-based algorithms use dynamic objective functions and are adaptive, in contrast to Genetic Algorithms (GA), which only use trajectory-based objective functions to evaluate solutions; swarm-based algorithms are able to respond to shifting problem dynamics because of their adaptability. [44] is a case of GA-based biclustering.

PSO (Particle Swarm Optimisation) is a swarm-based algorithm that draws its inspiration from the social behavior of bird flocking. Random solutions are used to initialize the particle population. For every generation, each solution updates its velocity, which, as the name suggests, includes both direction and rate of change of position. This updates the solution's position with respect to the global Pareto front. MOSPO [36] is an algorithm that applies PSO in the biclustering field.

ACO (Ant Colony Optimization) is inspired by real ant foraging behavior. In their search for food, ants engage in an exploratory phase in which they move randomly around their nest. When ants find a food source, they examine its quantity and quality before returning to the nest, leaving a pheromone trail on the ground. The algorithm then uses this trail of pheromones to navigate and effectively solve the task of locating food sources. Each ant in the algorithm moves from one state to another, representing a more complete solution, depositing some amount of pheromone that represents the desirability of the move. Each move is probabilistically selected from a feasible set of allowed moves. MOACO [37] and bicACO [45] are two algorithms that perform biclustering using ACO.

ABC (Artificial Bee Colony): The Artificial Bee Colony (ABC) swarm intelligence optimization algorithm is based on the notion that an objective can be optimized by replicating the intelligent behavior of bees. It imitates how bees look for food sources and share their discoveries to direct colony exploration. Few works that used ABC for clustering purposes [46, 47] showed how it significantly outperformed other approaches like DE (Differential Evolution) and PSO. In [35], the authors proposed work employs a discrete version of the Artificial Bee Colony optimization algorithm for biclustering of web usage data to produce optimal biclusters (i.e., highly correlated biclusters). It was demonstrated on a real dataset, and the results showed that the proposed approach was able to find significant biclusters of high quality and that it outperformed Binary Particle Swarm Optimization (BPSO).

2.3.6 Analysis

In the study conducted by [5], a comprehensive assessment and comparison of five methods was performed including CC [2], OPSM [4], ISA [38], Xmotif [39] and Samba [40].

The evaluation encompassed both synthetic and actual datasets. The synthetic datasets involved biclusters with constant and additive values, undergoing systematic noise increments and increased overlap. Real data evaluation involved incorporating biological insights using GO annotations [48], metabolic pathway maps [49], and protein-protein interaction information [50].

In general, the ISA, Samba, and OPSM methods perform well. While some methods perform better in certain scenarios, they perform poorly in others. Therefore, biclustering methods are difficult to evaluate and compare because the results depend highly on the scenario under consideration. Moreover, despite the existence of numerous biclustering algorithms, there are still several significant challenges to overcome, such as the lack of data available to define the type of specific biclusters to look for, the amount of noise present in the data matrices, the computation time required due to the complex calculations that are frequently required and the problem's multi-objective nature, because both the MSR and the bicluster size must be optimized at the same time.

After examining all of these methods, it was revealed that evolutionary algorithms are the most widely employed approaches for solving the biclustering problem. It is explained by the numerous benefits provided by EA, which include the ability to handle noise and incomplete data, flexibility, parallelism, and non-deterministic nature. These benefits make evolutionary algorithms a powerful approach for the biclustering problem, and they have been shown to produce state-of-the-art results on a variety of benchmark datasets. Among the evolutionary algorithms that have been shown to be effective for biclustering problems are: Genetic Algorithm (GA), Particle Swarm Optimization (PSO), Artificial Bee Colony (ABC). In contrast, Differential Evolution (DE) is another type of evolutionary algorithm that has several advantages over other evolutionary algorithms, including simplicity, robustness, efficiency, versatility, and flexibility. Despite being a powerful and versatile optimization algorithm capable of solving a wide range of optimization problems, particularly those involving discrete variables, DE has never been applied to bicluster microarray data. After researching differential evolution, its process, and its many benefits, we discovered that it could be used for biclustering gene expression data with a few modifications to the original algorithm.

Similarily, the Artificial Bee Colony (ABC) algorithm provides numerous potential advantages over other swarm-based approaches, and it demonstrates remarkable adaptability, robustness, ease of implementation, the potential for parallelization, and the ability to explore and exploit the solution space dynamically. Its inspiration from honey bee foraging behavior offers a unique perspective for solving optimization problems. Nevertheless, the utilization of ABC in biclustering remains largely unexplored due to a limited number of studies that have employed this algorithm for such purposes.

2.3.7 Addressing Noise in Gene Expression Data

Conversely, a shared limitation observed in the mentioned algorithms and approaches is the absence of a data preprocessing stage dedicated to adequately prepare and denoise the gene expression data before its utilization by the chosen biclustering algorithm. Gene expression data can be noisy since there are many different sources of noise due to various factors like experimental errors, technical flaws, and biological variability.

The noise in the gene expression data can introduce unwanted variations and distort the underlying gene expression patterns causing the accurate biclustering to be hampered and, therefore, resulting in the formation of fictitious or unstable biclusters, where the recognized patterns do not accurately reflect true biological relationships.

Numerous denoising methods, including Autoencoders (AE), Principal Component Analysis (PCA), Singular Value Decomposition (SVD), and Wavelet Denoising, can be used on gene expression data. Denoising autoencoders, however, are a superior choice over other denoising techniques for a variety of reasons; Autoencoders are potent neural network models that can recognize complex patterns in gene expression data and learn non-linear relationships between the noisy input and the clean output, in contrast to conventional denoising techniques that rely on linear filters or statistical techniques. Autoencoders are algorithms that learn a compressed data representation and then extract useful features that aid in noise filtering and information preservation which makes them well-suited for denoising gene expression data [51, 52].

Among different types of autoencoders, Convolutional Denoising Autoencoders (CDAEs) are exceptional at identifying regional patterns and spatial relationships in data. Despite the fact that gene expression data is not inherently spatial, it might still contain regional patterns or dependencies that the convolutional layers of a CDAE can recognize. Finding gene expression patterns linked to particular biological processes or regulatory mechanisms can be a result of this. Additionally, CDAEs' ability to denoise data enables them to reconstruct clear gene expression profiles from distorted or noisy data. Utilizing Convolutional Denoising Autoencoders (CDAEs) as a preprocessing and denoising technique for gene expression data can enhance the accuracy and reliability of subsequent analyses, including biclustering and differential gene expression analysis.

Differential Evolution (DE) and Artificial Bee Colony (ABC) hold enormous promise in the field of bilcustering for addressing the identified challenges, and their unexplored potential represents a significant neglected potential, so we have introduced three approaches to effectively address that. Chapters 3 and 4 provide a thorough examination of Differential Evolution (DE) in the context of biclustering. Each chapter focuses on a specific DE approach, giving in-depth discussions and insights on its application and potential contribution to addressing the stated challenges. In addition, Chapter 5 will provide a dedicated approach to address noise issues and the limited exploration of ABC in biclustering.

2.4 Conclusion

Throughout this chapter, we learned about the significance of biclustering as a strategy for identifying complex relationships within biological datasets. Understanding diverse bicluster types and evaluation metrics provides researchers with essential tools for pattern detection and algorithm evaluation. Furthermore, we examined state-of-the-art biclustering approaches, offering a thorough review of their methodology and contributions to the field of genomics.

Looking ahead, the next chapter will introduce a novel contribution – the first-ever Differential Evolution approach for biclustering. This approach aims to leverage evolutionary computation for adaptive and effective analysis of genomic data, providing a practical and innovative addition to the existing biclustering methodologies.

Chapter 3

DeBic: A Differential Evolution Biclustering Algorithm for Microarray Data Analysis.

3.1 Introduction

In the realm of optimization algorithms, Differential Evolution (DE) has garnered widespread acclaim for its simplicity, robustness, and efficiency in addressing a variety of optimization challenges. Its versatility extends to applications in diverse fields, including engineering, finance, machine learning, and computational biology. This chapter provides a thorough exploration of the algorithm's principles, steps, and strengths, encompassing its mutation and crossover operators. Subsequently, it introduces the first-ever approach to the biclustering problem utilizing the differential evolution algorithm. The chapter elaborates on its adaptation to the biclustering problem using a unique mutation operator and presents the results achieved by DeBic or real world datasets.

3.2 Differential Evolution

Recently, Differential evolution has grown in popularity for tackling situations involving continuous optimization; it has been effectively employed in various scientific and technical sectors.

The strength of differential evolution is to develop offspring by utilizing directional information within the population. [53] proposed DE as a simple and efficient heuristic for global optimization over continuous spaces. Their technique demonstrated DE's effectiveness in addressing optimization problems.

DE was examined alongside Genetic Algorithms (GAs) in a comparative study examining optimization strategies for combinatorial issues in [54], where DE was rigorously tested on its convergence speed, solution quality, and computing efficiency. The findings show how DE's performance was more robust and yielded more valuable outcomes when compared to GAs. Similarly, DE was compared to the PSO for constrained optimization in [55], and the results demonstrate that DE exceeds PSO in terms of repeatability (robustness) and the number of iterations necessary to move the solution candidate within the feasible range.

DE is a competitive optimization strategy for real-parameter numerical optimization problems. It is a simple yet powerful method for global optimization over continuous spaces that uses the greedy selection criteria to choose which competitors will be retained in the following generation.

Unlike traditional genetic algorithms, DE relies on distance and directional information via unit vectors for reproduction. Its reproduction operator involves a mutation step to generate a trial vector, which the crossover operator subsequently uses to produce one offspring. The sizes of mutation steps are estimated as weighted differences between randomly chosen individuals. the following is a clear explanation of the main algorithmic steps of DE:

1. **Initialization:** DE starts by generating a population of potential solutions, with the size of the population being determined by the user.

2. **Mutation:** this step involves the creation of new individuals by perturbing existing ones. This is done by computing the difference between two randomly selected individuals and then scaling it by a factor, this scaled difference is then applied to another individual, resulting in a modified solution.

3. **Crossover:** this involves randomly choosing components from the mutated solution and mixing them with the remaining components from the original solution to create what is called a trial solution.

4. **Selection:** DE then compares the trial solution created by crossover to the original solution. If a trial solution proves to be more successful than the corresponding original solution, it replaces it in the population.

5. **Termination:** The algorithm repeats these steps for a certain number of generations or until a particular convergence condition is fulfilled.

3.2.1 Algorithm of Differential Evolution

Let $x \in \mathcal{R}$ designate a candidate solution from the present population, where D is the optimized dimensionality of the problem, and $f : \mathcal{R} \longrightarrow \mathcal{R}$ is the objective function to be minimized.

Georgioudakis and Plevris performed a comparative study on DE variants in constrained structural optimization in [56]. This work contributes to our understanding of DE's effectiveness in tackling complicated optimization tasks, and it complements our exploration of DE's standard variance in this study. The standard DE method is based on the **DE/rand/1/bin** scheme, it's algorithm is provided as follows:

Algorithm 1: Differential Evolution with Scheme DE/rand/1/bin

```
Data   : $NP$: population size, F: mutation factor, CR: crossover probability,
         MAXFES: maximum number of functions evaluations
1  INITIALIZATION:  G = O; Initialize all NP individuals with random
     positions in the search space;
2  while $FES < MAXFES$ do
3      for $i \leftarrow 0$ to $NP$ do
4          GENERATE:  three individuals $x_{r1}$ , $x_{r2}$ , $x_{r3}$ from the current
             population randomly. These must be distinct from each other and also
             from individual $x_i$, i.e. $r_1 \neq r_2 \neq r_3 \neq i$.
5          MUTATION  Form the donor vector using the formula:
             $Vi = x_{r1} + F(x_{r2} - x_{r3})$
6          CROSSOVER  The trial vector $u_i$ is developed either from the elements
             of the target vector $x_i$ or the elements of the donor vector $v_i$ as follows:
```

$$u_{i,j} = \left\{ \begin{array}{c} v_{i,j}, \quad if \quad r_{i,j} \leq CR \quad or \quad j = j_{rand} \\ x_{i,j}, \quad otherwise \end{array} \right\} \tag{3.1}$$

```
7          Where $i = 1, ..., NP, j = 1, ..., D, r_{i,j} \sim U(0,1)$ is a uniformly distributed
             random number which is generated for each $j$ and $j_{rand} \in 1, ..., D$ is an
             arbitrary integer used to ensure that $u_i \neq x_i$ in all cases.
8          EVALUATE if $f(u_i) \leq f(x_i)$ then replace the individual $x_i$ in the
             population with the trial vector $u_i$ FES = FES + NP
9      end
10     $G = G + 1$
11 end
```

DE is a prominent optimization approach due to its ability to tackle non-convex, multimodal, and nonlinear optimization problems while offering speedy convergence, effective local search capabilities, and programming simplicity. Given its versatility and simplicity, and its ability to explore huge solution areas, DE can be a good choice for complicated, high-dimensional problems including both continuous and discrete variables. Furthermore, DE effectively avoids local optima by using solution populations to increase convergence. DE provides for faster convergence with fewer evaluations when control parameters are kept to a minimum, making it a very useful optimization strategy. However, DE has several limitation including sensitivity to parameter settings, difficulties in adapting to dynamic environments, limited performance on noisy or irregular landscapes. Furthermore, DE has difficulties in retaining its scalability in higher dimensions. As the number of dimensions rises, DE's performance and efficiency could decline, making it difficult to properly navigate complicated solution spaces [57].

3.2.2 Mutation Operator

A difference and a target vector are calculated using two components by the mutation operator in differential evolution. The target vector designates the parent whose direction is given priority in creating the unit vector, whereas the difference vector indicates the differences between two or more parents. The outcome of the mutation process is the unit vector, which is subsequently given to the current central parent —the parent of interest excluded from the mutation process— to be crossed over with. The mutation is formally defined as the following: for each target vector $x_i(i = 1, 2, ..., m)$, a mutant vector is produced as follows:

$$h_i(i+1) = x_{r1} + f(x_{r2} - x_{r3}) \tag{3.2}$$

By looking at the mutation operation formula above, we can easily conclude that it is capable of maintaining the closure only in real numbers field. So despite the simplicity and effectiveness of the differential evolution algorithm in many branches of engineering, applying differential evolution to binary optimization issues with binary decision variables is still uncommon; and to fix this issue, We will employ a new mutation operator known as the semi-probability mutation operator, that incorporates the original mutation operator and a fresh probability-based defined operator [58].

3.2.3 Crossover Operator

Following the mutation phase, the mutant vector undergoes a crossover with the target vector $X_i(g)$ to produce the trial vector $U_i(t+1)$, which has received much less attention from researchers in this field. The crossover operator is equally essential as the mutation operator. Two types of crossover operators are commonly used in DE: binomial crossover and exponential crossover.

3.2.3.1 Binomial Crossover

The binomial crossover is a uniform crossover that uses a binomial distribution. The if statement condition in the algorithm below ensures that at least one component is taken from the mutant vector.

Algorithm 2: Binomial Crossover

```
Function BinCrossover($x_i, y_i$) : vector is
    $j \leftarrow rand_i(\{1, ..., n\})$  for $i \geq 1, .., n$ do
        if $rand_i(0,1) \leq CR$ or $i = j$ then
            $u_i \leftarrow h_i$;
        else
            $u_i \leftarrow x_i$;
        end
    end
    return $u_i$
end
```

3.2.3.2 Exponential Crossover

The exponential crossover simulates a two-point crossover in which the first cut point is chosen at random from $\{0, ..., 1\}$, and the second point is chosen in such a way that L consecutive components are taken from the mutant vector.

Algorithm 3: Exponential Crossover

```
1 Function $ExpCrossover(x_i, h_i) : vector$ is
2     $j \leftarrow rand_i(\{1, ..., n\})$  $i \leftarrow j$  $u_i \leftarrow x_i$  $L \leftarrow 0$  repeat
3         $u_i \leftarrow h_i$  $i \leftarrow i + 1$  $L \leftarrow L + 1$
4     until $rand_i(0, 1) \geq CR$ or $L = n$;
5     return $u_i$
6 end
```

Zaharie invistigated the influence of crossover strategies on DE's performance and behavior in optimization tasks in [59], the experimental findings indicate that binomial crossover produces better outcomes on the theory community's common benchmark issues; thus, our approach will utilize the binomial crossover.

The complexity of this algorithm is shared in [60], where they measured the run time and averaging with a varied dimension size and applied it to different bench-marks. The results lead to an asymptotic bound computational complexity.

3.2.4 The benefits of the DE Algorithm

The power of differential evolution is using directional information within the population to create offspring. The algorithm has been demonstrated to outperform the Genetic Algorithm (G.A.) in [54] and [61] alternatively, Particle Swarm Optimization (PSO) in [55] through numerical benchmarks and experiments because of its simplicity, efficiency, and local searching property, and speediness.

3.3 DeBic Algorithm Description

We suggest a Differential Evolution Based Biclustering algorithm called DeBic. This new approach allows us to detect multiple biclusters simultaneously by using the evolutionary algorithm of differential evolution and adapting it to perform Biclustering on a binary search space with a different mutation operator. Differential evolution is best known for its exceptional robustness in non-convex, multimodal, and nonlinear problem optimizations. It is also known for its quick convergence and straightforward programming.

3.3.1 Algorithm Description

The proposed solution uses differential evolution to drive a population of biclusters solutions toward better solutions. First, it initializes the population by randomly selecting biclusters, like the most of current biclustering algorithms, the biclusters are represented by a binary sequence of predetermined length $(n + m)$. The binary sequence is usually made up of two parts: the first part represents the genes, and the second part represents the conditions, as illustrated in Figure 3.1.

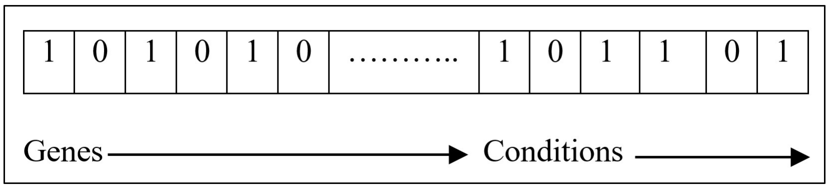

Figure 3.1: Representation of Binary String Encoding for Bicluster Solution.

After initializing the population, the algorithm employs the mutation operator to modify an existing individual; these modifications can be made at random or in accordance with a predetermined strategy. As Discussed in section 3.2.2, we adopted a new mutation operator that allows us to expand DE into the binary space search, proposed by Changshou et al. in [58], called the semi-probability Mutation operator, defined as the following:

$$h_i(t+1) = \begin{cases} x_{r1} + F(x_{r2} - x_{r3}), & \text{if } h_i(t+1) = 0 \text{ or } 1 \\ 0, & \text{if } pr < \text{rand} \\ 1, & \text{otherwise} \end{cases} \tag{3.3}$$

In the next step, the DeBic algorithm uses the crossover strategy to create a new vector by mixing the information of the trial and the target vectors, and it is defined as the following:

$$u_i(t+1) = \begin{cases} h_i(t+1), & \text{if rand} \leq CR \text{ or } j = \text{rand}(i) \\ x_i(t), & \text{otherwise} \end{cases} \tag{3.4}$$

Where $i = 1, 2, \ldots, m; j = 1, 2, \ldots, n; rand(i) \in (1, 2, \ldots, n)$ is the randomly chosen index and $CR \in [0, 1]$ is crossover constant . In other words, certain elements of the mutant individual or at least one parameter chosen at random, as well as some of the target individual's other attributes, are present in the trial individual.

The effect of the mutation with the semi-probability operator alongside the inherited behavior of evolutionary algorithms such as crossover will help maintain diversity in the population since the different evolution will use those new individuals to create other improved individuals, which will get selected or discarded in the selection phase.

Then, we compare the trial individual to the corresponding to decide if it should be a member of the next generation. The selection process relies on the trial participant's fitness survival; if it is considerably better than the current solution, we replace it. After a specific number of iterations, the algorithm reaches stopping criteria and ends the process by returning the resulting population.

The fitness of an individual $X(I, J)$ used in this study is then given by the formula:

$$\text{fitness}(X) = \frac{\text{MSR}(X)}{\delta} + \frac{1}{\text{size}(X)} \tag{3.5}$$

δ is the threshold that represents the most dissimilarity that can exist within a bicluster, and the size function represents the size of the bicluster.

The different steps of DeBic are described in the following:

Algorithm 4: DeBic Algorithm Steps

```
Input  : A Matrix of size $n \times m$, Population Size $p$, Crossover probability $CP$,
         Fitness function $fscore$
Output: $p$ Bicluster solutions
pop ← generating $p$ random solutions for initial population solutions;
popFitness ← calculate the fitness score for all solutions in pop using $fscore$;
while Termination Conditions Isn't met do
    $a, b, c$ ←select 3 random different solutions from pop in different positions
      than $i$ ;
    /* Create mutant using semi probability mutation */
    mutant ← semiProbabilityMutation($a, b, c$);
    trial ←Create Trial Vector using binomial crossover using the mutant and
      the current vectors with $CP$ as crossover probability;
    /* Calculate the score of the new trial vector */
    currentFitness ← fscore(trial);
    if currentFitness > popFitness[$i$] then
        popFitness[$i$] ← currentFitness;
        pop[$i$] ← trial;
    end
end
return $p$ Biclusters pop from last generation;
Function semiProbabilityMutation(a: vector, b: vector, c: vector) : vector is
    $$vector = \begin{cases} a + F(b - c), & \text{if } r = 0 \text{ or } r = 1 \\ 0, & \text{if } pr < \text{rand} \\ 1, & \text{otherwise} \end{cases}$$
    return vector
end
```

3.4 Experimental Results

In order to assess the effectiveness of the DeBic Algorithm for bicluster detection in expression data, we conducted some experiments on the Yeast Saccharomyces Cerevisiae Cell-Cycle expression dataset. The yeast cell cycle is a particularly popular dataset for gene expression data experiments, as the genes functions are well-known. This dataset originated from [56], there are 17 experimental conditions and 2,884 genes in the associated expression matrix.

The parameter settings shown in Table 3.1 were used for all the conducted experiments, and all experiments were carried out on a PC equipped with a 4.30 GHz AMD Ryzen 7 3800XT 8-Core Processor and 64 GB of RAM running with Windows 10.

Hyper Parameter	**Value**
Population Size	75
Generations	100
Initial Crossover Probability	0.75

Table 3.1: Hyper Parameter Specification

14 out of 100 biclusters on the Yeast data set were discovered by the DeBic algorithm, as shown in 3.2:

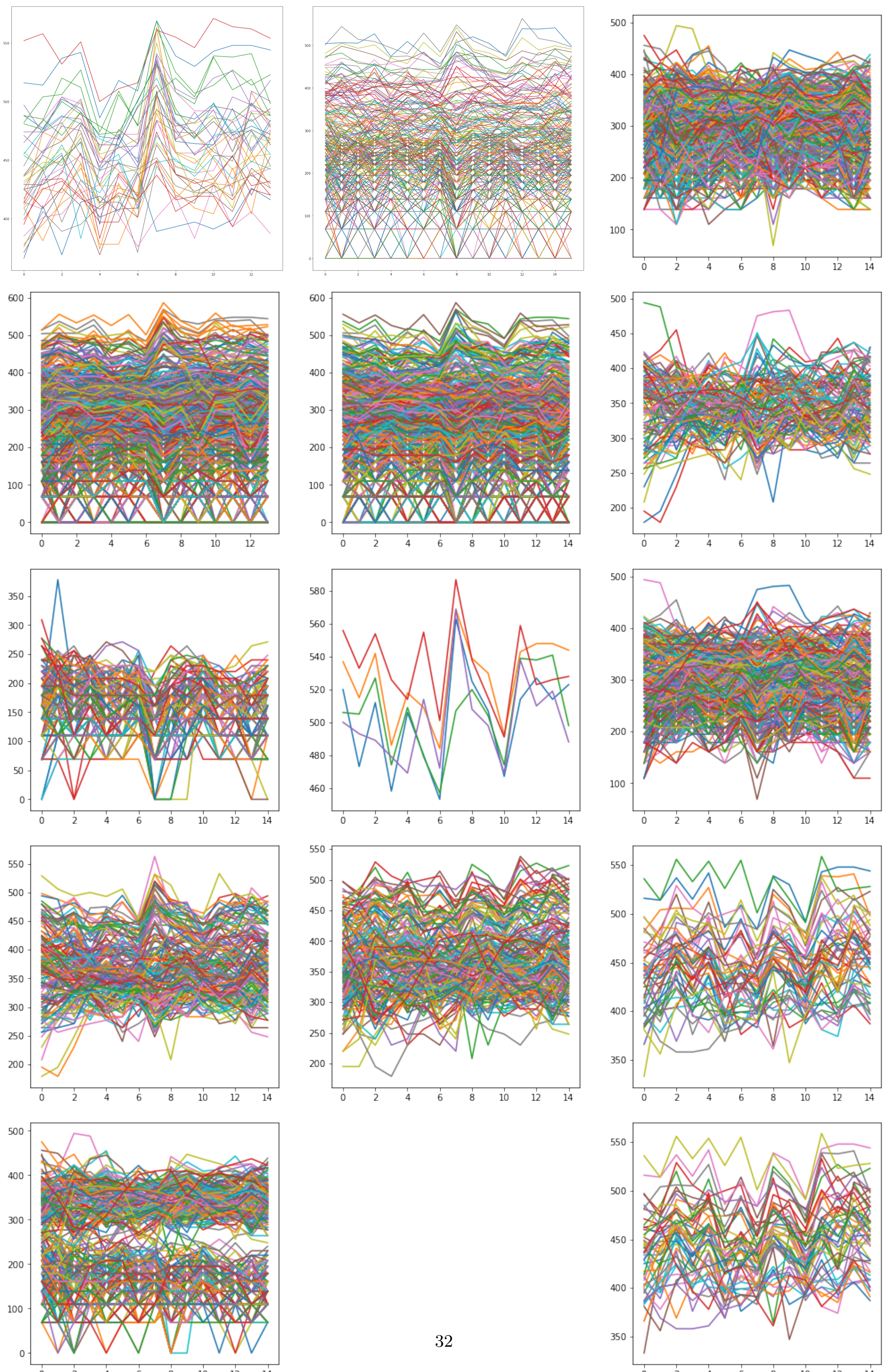

Figure 3.2: 14 biclusters Discovered By DeBic in The Yeast Cell-Cycle Dataset.

The genes exhibit a similar response under a variety of circumstances, as can be shown by a visual examination of the additional biclusters suggested in Fig. 2. The result of a selected set of genes from a bicluster with high biological relevance are shown below: YBR244W YCR102C YDL022W YEL024W YHR037W YHR104W YIL111W YIL124W YJR155W YKL150W YKR009C YLR395C YMR145C YMR152W YMR189W YNL037C YNL134C YOL059W YOR136W YDR185C YDR213W YDR240C YDR294C YDR299W YDR330W YDR466W YDR479C YEL005C YEL024W YEL046C YEL070W YER028C YER054C YER062C YER065C YER068W YER141W YER143W YGL013C YGL130W YGL156W YGL208W YGL219C YGR029W YGR100W YGR146C YHL039W YHL042W YHR037W YHR129C YIL055C YIL111W YIL124W YIL156W YJL003W YJL091C YJL101C YJL191W YJR078W YJR096W YJR155W YKL100C YKL150W YKR004C YKR013W YLL010C YNL336W YNR013C YNR038W YNR077C YOL017W YOL032W YOL059W YOL064C YOL079W YOL096C YOL108C YOL114C YOL154W YOL158C YOR094W YOR136W YOR177C YOR178C YOR181W YOR252W YOR283W YPL007C YPL057C YPL086C YPL094C YPL132W YPL165C YPL186C YPL204W YPL205C YPL208W YPL228W YPL236C YPL258C YPL268W YPL274W YPR017C YPR043W YPR068C YPR113W YPR168W YPR182W.

We also recorded the biological process from the previous bicluster with the corresponding P-values (which is a number describing how likely our data would have occurred by random chance) in Table 3.2.

Biological Process	**P-Value**
polyol metabolic process	5.07e-05
oxidoreductase complex	1.63e-05
oxidoreductase activity, acting on the CH-OH group of donors, NAD or NADP as acceptor	2.15e-07
oxidoreductase activity, acting on the CH-OH group of donors	5.01e-07
oxidoreductase activity	3.73e-10
oxidation-reduction process	3.33e-10
catalytic activity	1.32e-06

Table 3.2: Biological Processes and Corresponding P-values in the Previous Bicluster.

Table 3.3 compares DeBic's performance compared to Cheng and Church's method (hereafter CC) [2], Yang et al's algorithm FLOC [62], and MOEAB [7] for the aver-age residue and average dimension of the discovered biclusters.

Algorithm	**Avg. bicluster size**	**Avg. residue**	**Avg. no. of genes**	**Avg. no. of conditions**
CC	1576.98	204.29	167	11
FLOC	1825.78	**187.54**	195	12.8
MOEA	**10301**	234.87	1095	9.29
DeBic	9889.58	272.13	628.04	**14.5**

Table 3.3: DeBic Performance Comparison To Other Algorithms.

The outcomes of this actual dataset demonstrate the ability of our suggested approach to find biclusters with significant biological relevance.

3.5 Conclusion

This chapter has delved into the versatile realm of Differential Evolution, highlighting its widespread recognition for simplicity, robustness, and efficiency across various optimization challenges. Furthermore, it has effectively provided a comprehensive insight into DE's inner workings. The introduction of DE to the biclustering problem marks a significant milestone in computational genomics. The unique mutation operator incorporated in the adaptation to biclustering, as demonstrated by the DeBic algorithm, adds a novel dimension to the algorithm's application. The chapter concludes with the presentation of results achieved by DeBic on real-world datasets, further validating its efficacy in tackling the biclustering challenges inherent in gene expression data.

In the upcoming chapter, we delve into another implementation of Differential Evolution (DE) within a multi-objective framework tailored for biclustering. This novel approach introduces the Biclustering Binary Mutation Operator (BBDE), representing a unique and adaptive mutation strategy designed explicitly for the biclustering context.

Chapter 4

AMoDeBic: An Adaptive Multi-Objective Differential Evolution Biclustering Algorithm of Microarray Data Using a Biclustering Binary Mutation Operator

4.1 Introduction

This chapter explores a strong optimization approach noted for its ability in solving complex problems across several domains known as Multi-Objective Differential Evolution (MODE). Recognizing the significance of MODE, we embark on a comprehensive exploration of its principles and functionalities. Moreover, the chapter extends its focus to binary Differential Evolution (DE), a critical aspect in AMoDeBic, our novel algorithm that leverages Multi-Objective Differential Evolution for gene group discovery covered in this chapter. Within this innovative approach, we unveil the Biclustering Binary Differential Evolution (BBDE), a new mutation operator designed to enhance the adaptability and robustness of the biclustering process. The chapter systematically evaluates the effectiveness of AMoDeBic by comparing its results to state-of-the-art algorithms, utilizing both synthetic and real datasets.

4.2 Differential Evolution

Differential Evolution (DE) is a population-based, stochastic optimization algorithm for solving nonlinear and numerical optimization tasks using REAL number parameters. The DE algorithm was first introduced by Storn and Price in 1996 [53].

As we covered in Section 3.2, differential Evolution relies upon distance and directional information through unit vectors for reproduction, unlike standard genetic algorithms. Its reproduction operator involves a mutation phase to produce a trial vector, It is subsequently used by the crossover operator to create one offspring. Weighted differences between individuals chosen at random are used to calculate mutation step sizes.

4.2.1 Algorithm of Differential Evolution

A potential solution among the population $x \in \mathcal{R}^D$, D represents the dimensionality of the optimized problem and the objective function that needs to be minimized is $f : \mathcal{R}^D \longrightarrow \mathcal{R}$. The standard DE pseudo code that follows the **DE/rand/1/bin** scheme is as follows [56]:

```
Population Initialization
Repeat
Mutation
Crossover
Selection
Until Termination conditions are met
```

4.2.2 Multi-objective Differential Evolution

As the name implies, a multi-objective method works with more than one objective function. In many practical or real-world situations, numerous objectives or multiple criteria are present in most practical decision-making problems. Multi-objective optimization is sometimes known as vector optimization since a vector of objectives must be simultaneously optimized rather than a single objective. A single optimal solution in such instances no longer exists, but rather a wide variety of solutions of comparable quality.

4.2.2.1 Multi-objective Framework

Unlike single-objective optimization problems, the Multi-objective Differential Evolution (MODE) aims to optimize two or more competing aspects represented by fitness functions. Using single-objective DE to model this situation would include a heuristic evaluation of numerous factors required in establishing such a scalar-combination-type fitness function. On the other hand, MODE presents a set of Pareto-optimal solutions [63] that simultaneously optimize the conflicting criteria of multiple fitness functions.

A solution is Pareto-optimal if it is dominated by no other possible solution, which means that no other solution exists that is greater at least in one objective function value and equal or superior in the different objective function values.

4.2.2.2 Pareto-Based Evaluation

Individuals are evaluated using the Pareto-based rank assignment [64]. Non-dominated individuals are awarded rank 1, reflecting the highest fitness values in the population, and these individuals are withdrawn from the competition. A new group of non-dominated individuals from the rest of the population is ranked as 2 with the following highest fitness values until all individuals in the population are assigned a rank. The fitness sharing strategy [64] is commonly used in MODE in the literature to retain several optimal solutions (cf. Figure 4.1).

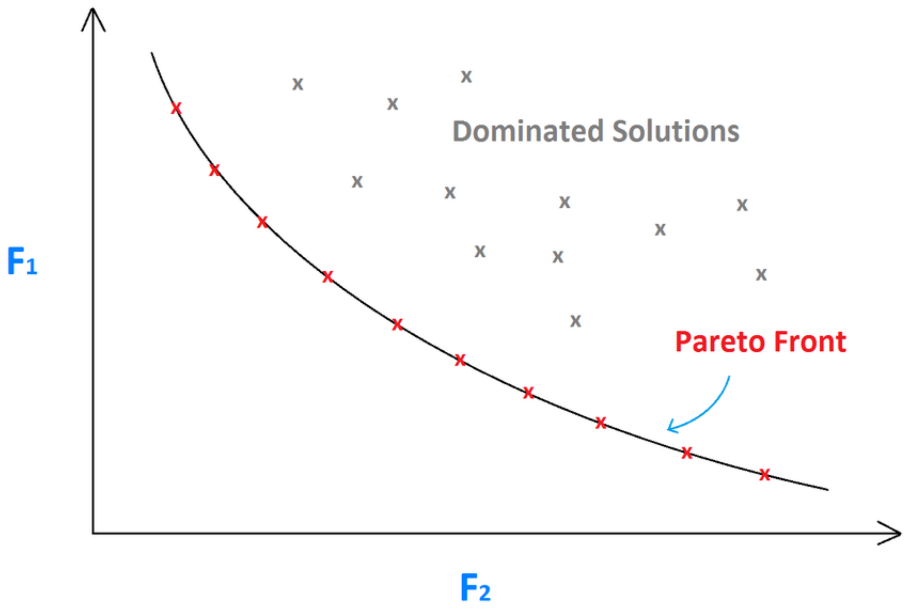

Figure 4.1: Pareto Optimality Visualization in 2D Objective Space.

4.2.2.3 Non-dominated Sorting Genetic Algorithm II

Deb et al. presented a method to implement the fitness-sharing idea in NSGA-II [65] , where individuals within each rank who belong to the least crowded region are penalized less. A crowding distance measure is used to determine the within-rank solution density. It is calculated for a specific individual by adding the total of the normalized distances along each objective dimension between the two individuals within the same rank who define the smallest interval of this individual in that objective dimension.

Among the several multi-objective algorithms, the non-dominated sorting genetic algorithm (NSGA-II) has all the characteristics necessary for a successful MODE [65]. It has been demonstrated that this can converge to the global Pareto front while retaining population diversity. The NSDE [66] algorithm is a straightforward modification of the DE algorithm for addressing multi-objective optimization problems. The operation of NSDE and DE is identical, with the exception of the selection operation, which is adjusted to handle multi-objective optimization problems. Some works have proposed multi-objective biclustering algorithms:

Lashkargir et al. proposed a Hybrid algorithm for discovering biclusters in gene expression data [67], which is based on adaptive multi-objective particle swarm optimization
.

Mitra and Banka developed a novel multi-objective evolutionary biclustering framework that incorporates local search strategies and a new quantitative measure to assess the goodness of the biclusters [7] .

Liu et al. proposed a new multi-objective particle swarm optimization biclustering (MOPSOB) algorithm for detecting coherent patterns in microarray data [68]. In addition, they also proposed the CMOPSOB (Crowding distance-based Multi-objective Particle Swarm Optimization Biclustering), a clustering approach for microarray datasets to cluster genes and highly related conditions in sub-portions of the microarray data [36].

Following our examination of Multi-Objective Differential Evolution, Pareto-based evaluation and the NSGA II algorithm, our exploration naturally extends to another facet of DE: Binary Differential Evolution. This exploration is as crucial since Binary DE plays a vital role in shaping our proposed approach.

4.2.3 Binary Differential Evolution

Binary Differential Evolution (BDE) is a DE variant designed to handle binary variable optimization problems. Each member of the population is represented as a binary string in BDE, and the mutation and crossover operators have been modified to handle binary variables. BDE works by perturbing the individual with the difference between two other individuals in the population to generate a new candidate solution for each individual in the population. A selection operator is also included in BDE to select the best individuals for the next generation.

The mutation operation of DE given by the formula 3.2 shows that it can only maintain closure in the real number field. Individuals are disturbed by the mutant DE vector, and when individuals are subjected to more disturbances, solutions with higher fitness results may result. So creating a suitable mutant vector is the key to DE. The challenge of using DE to solve discrete issues is preserving the mutant vector's capacity for disruption. Due to this, the differential evolution algorithm's application to binary optimization problems with binary decision variables is still uncommon despite its simplicity and success in many engineering fields. The majority of current DE research focuses on issues with continuous optimization. However, given the various benefits of DE and the prevalence of discrete and binary optimization issues, there are some works that studied DE to optimize binary and discrete problems:

Zorarpacı et al. suggested a novel hybrid technique for the feature selection problem of classification tasks that combines the best characteristics of the ABC [69] and DE [53] algorithms and includes a new binary mutation phase for the DE algorithm [70]. After computing the difference vector, the mutant vector is generated for the source individual. To create a mutant vector, the "OR" binary logic operator is used to the components of a randomly selected third vector using a difference vector. The difference vector and the mutant vector are calculated using formulas 4.1 and 4.2:

$$difference\ vector_i^d = \begin{cases} 0 \ \ if x_{r1}^d = x_{r2}^d \\ x_{r1}^d \quad \text{otherwise} \end{cases} \tag{4.1}$$

$$mutant\ vector_i^d = \begin{cases} 1\ if\ differencevector_i^d = 1 \\ x_{r3}^d \quad \text{otherwise} \end{cases} \tag{4.2}$$

The crossover step is then executed using the formula 4.3 as follows:

$$u_i^d = \begin{cases} mutant\ vector_i^d \ \ if\ \sigma \leq CR(t)\ ||\ d = d_{rand} \\ x_i^d \quad \text{otherwise} \end{cases} \tag{4.3}$$

Deng et al. introduced the semi-probability Mutation operator [58], which permits the extension of DE into the binary space search and is defined as follows:

$$h_i(t+1) = \begin{cases} x_{r1} + F(x_{r2} - x_{r3}), & if\ (h_i(t+1) = 0\ or\ 1) \\ 0, & pr < rand \\ 1, & \text{otherwise} \end{cases} \tag{4.4}$$

The probability pr is calculated using the formula 4.5:

$$pr = \frac{1}{1 + e^{-h_i{}^{(t+1)}}} \tag{4.5}$$

Xu and Wang proposed an elite-guiding binary differential evolution (EGBDE) [71] , which includes a novel mutation strategy in which the mutant vector is obtained by comparing every gene of the fittest individual to the corresponding gene of the selected individuals and then randomly setting the bits from the first different bit to the end of the gene.

Gong and Tuson proposed the Binary Differential Evolution (BDE) method [72]. In this method, the continuous difference between two individuals in regular DE is represented by the hamming distance in the binary search space given the distance $D(X(r_2, g), X(r_3, g))$ is d, the scaled distance is:

$$d' = F * d \tag{4.6}$$

The mutant vector is donated by the following formula, with the scaled distance d' as a floating number:

$$D(Mutant, X(r_1, g) = \begin{cases} (\text{int})\ d' + 1, & if\ \ rand < d' \text{ - (int) } d' \\ (\text{int})\ d'\ , & \text{otherwise} \end{cases} \tag{4.7}$$

Engelbrecht and Pampara [73] presented three approaches using DE to optimize binary-valued parameters by converting binary variables to real variables: The angle modulated DE (AMDE) evolves a bitstring generating function using the regular DE like the following:

$$g(x) = sin(2\pi(x-a) \times b \times cos(2\pi(x-a) \times c)) + d \tag{4.8}$$

The second approach is called binDE; it treats the corresponding real variable as having a probability of zero or one.

$$y_{ij}(t) = \begin{cases} 0 \;\; if \; U(0,1) < f(x_{ij}(t)) \\ 1 \quad \text{otherwise} \end{cases} \tag{4.9}$$

where the sigmoid function f is given as follows:

$$f(x) = \frac{1}{1+e^{-x}} \tag{4.10}$$

The normDE is the third approach, and it normalizes the real variables by placing them in the range $[0, 1]$, and then any value less than 0.5 is absorbed to zero, and any value more than that is absorbed to one.

$$y_{ij}(t) = \begin{cases} 0 \;\; if \; x_{ij}(t) < 0.5 \\ 1 \quad \text{otherwise} \end{cases} \tag{4.11}$$

However, these approaches are not intended for biclustering issues. To address this issue, we proposed the BBDE mutation strategy, which is heavily inspired by Cheng and Church's heuristic of node deletion and addition, who were the first to propose the biclustering technique.

4.3 AMoDeBic Algorithm

This study introduces AMoDeBic, a Multi-objective Differential Evolution approach for Biclustering, which integrates the innovative Binary Biclustering Differential Evolution (BBDE) mutation operator. This novel method detects multiple and diverse biclusters simultaneously by adapting a non-dominated sorting genetic algorithm (NSGA-II) that contains all of the characteristics essential for a suitable MODE to execute biclustering on a binary search space.

4.3.1 Algorithm Description

The suggested solution employs differential evolution to steer a population of bicluster solutions toward improved solutions. First, it initializes the parent population P by selecting biclusters at random, and it represents the biclusters with a predetermined binary string size (n+m), as do most of the current biclustering methods. This binary string comprises two-bit strings, one for the genes and the other for the conditions. If the appropriate gene and/or condition is present in the bicluster, a bit is set to one; otherwise, it is reset to zero (cf. Figure 4.2).

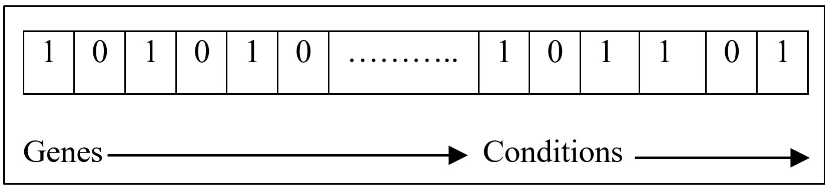

Figure 4.2: Representation of Binary String Encoding for Bicluster Solution.

After initializing the population, the DE algorithm generates the offspring population by employing the mutation operator to alter an existing individual. These modifications can be random or based on a predetermined strategy.
As explained in section 4.2.3, the mutation operator given by the formula 3.2 in the original DE algorithm can only retain the closure in the field of real numbers. As a result, we cannot directly apply the original DE in discrete optimization problems. Therefore, we will employ the new mutation technique proposed in this work called BBDE.

The BBDE operator functions as follows: The first step is to order the randomly selected vectors x_{r1}, x_{r2}, and x_{r3} for the mutation phase based on their fitness scores, with x_{r1} taking the vector with the best fitness score, x_{r2} taking the vector with the second-best fitness scores, and x_{r3} taking the remaining vector, to ensure the elimination of the bad characteristics of the vector x_{r3} that scored less when performing the node deletion operation part in the formula 3.2 $(x_{r2} - x_{r3})$

$$h_i = nodeAddition(x_{r1}, nodeDeletion(x_{r2}, x_{r3}, F), F) \tag{4.12}$$

Suppose only the node deletion and node addition are used after a few iterations. In that case, the resulting individuals will most likely be all 0s (0....0) or all 1s (1...1), significantly impacting the biclustering results. As a solution to this problem, we will generate a random number $rand_i \in \{0, ..., 1\}$ (where i indicates the position of the current bit in the present individual) and use the factor F to indicate the accepted probability of the component difference or addition as the following: if the generated number $rand_i$ is less than F, the algorithm will apply the node deletion/ addition; if $rand_i$ is greater than F, the node operation will be abolished, knowing that the initial value of $F = 0.65$.

After the mutation step, the resulting mutant vector will be used in the crossover operation with a probability CR initialized at 0.75. The new individual will then be tested in the selection phase using the fitness functions; if the score has improved, it is interpreted as positive feedback; thus, the factor F and the crossover probability CR will remain the same for that individual. However, if no improvement is detected in the individuals, F and CR will undergo an adaptation by a slight change using the following formulas:

$$F_i = normrnd(mean(F), 0.1) \tag{4.13}$$

$$CR_i = normrnd(mean(CR), 0.1) \tag{4.14}$$

The new Biclustering Binary Differential Evolution mutation operation with the adapted node deletion and addition operation is summarized as follows:

$$nodeDeletion(x, y, F) = \begin{cases} 0, & x_i = 0 \;\; or \;\; (y_i = 1 \;\; and \;\; rand_i < F) \\ & 1, \quad \text{otherwise} \end{cases} \tag{4.15}$$

$$nodeAddition(x, y, F) = \begin{cases} 1, & x_i = 1 \;\; or \;\; (y_i = 1 \;\; and \;\; rand_i < F) \\ & 0, \quad \text{otherwise} \end{cases} \tag{4.16}$$

The algorithm then employs the crossover approach to generate a new vector by combining the information from the trial and target vectors, which is defined as follows:

$$u_i(t+1) = \left\{ \begin{array}{cc} h_i(t+1), & rand \leq CR \;\; or \;\; j = rand(i) \\ x_i(t), & \text{otherwise} \end{array} \right\} \tag{4.17}$$

Where $i = 1, 2, ..., m, j = 1, 2, ..., n, CR \in [0, 1]$ is the crossover probability, and $rand(i) \in (1, 2, ..., n)$ is the randomly selected index. In other words, the trial individual has some components of the mutant individual or at least one of the characteristics chosen at random, as well as some of the target individual's other parameters. After multiple iterations, the offspring generation Q will be produced. Next, the algorithm performs the selection process as follows:

1. It merges both P and Q generations to prepare for the selection process.
2. It uses the multi-objective fitness functions to calculate the score of all solutions.
3. using the dominance criteria, it calculates the non-dominant front.

4. Finally, it performs the selection process by calculating the crowding distance to select solutions of low crowded regions.

A higher crowding distance is taken into consideration while choosing individuals. The selected number is the amount needed to equalize the size of the old and new parent populations ($size(P + Q)/2$). The selected solutions from the combined population will replace the parent population. The DE will generate a new offspring generation following the above mentioned process. After several iterations, the algorithm achieves ending criteria and terminates by returning the resulting population. The following is a description of AMoDeBic's steps:

Algorithm 5: AMoDeBic Algorithm Steps

```
Input  : A Matrix of size $n \times m$, Population Size $p$,Initial Crossover probability
         $CP$, Initial Factor $F$, objective Fitness function $f1, f2$
Output: $p$ Bicluster solutions
1  $popP \leftarrow$ generating $p$ random solutions for initial population solutions;
2  $popPFitness \leftarrow$ calculate the fitness scores using $f1, f2$ for all solutions in $popP$;
3  $CP \leftarrow$ initialize the array of crossover probabilities CP for all individuals in $p$
     score using initial $CP$;
4  $F \leftarrow$ initialize the array of F factor for all individuals in $p$ score using initial $F$;
5  while Termination Conditions Isn't met do
6      $popQ \leftarrow$Initialize $popQ$ with empty list;
7      for $i \leftarrow 0$ to $p$ do
8          $a, b, c \leftarrow$select 3 random different solutions from $popP$ in different positions
             than $i$ and sort them from best to worst according to their fitness ;
9          /* Create mutant using BBDE mutation operator */
10         $mutant \leftarrow BBDE(a, b, c, F_i)$;
11         $trial \leftarrow$Create Trial Vector using binomial crossover using the $mutant$ and
             the $current$ vectors with $CP_i$ as crossover probability;
12         /* Calculate the score of the new trial vector */
13         $currentFitness \leftarrow calculatefitnessfortrialvector$;
14         if $currentFitness$ is not better than $popFitness[i]$ then
15             /* Update the $CP_i$ and $F_i$ */
16             $CP_i \leftarrow normrnd(mean(F), 0.1)$;
17             $F_i \leftarrow normrnd(mean(CP), 0.1)$;
18         end
19     end
20     $fullPop \leftarrow$merge $popP$ and $popQ$ ;
21     $fullPop \leftarrow$Rank the $fullPop$ using the Dominance criteria;
22     $cd \leftarrow$Calculate crowding distance within $fullPop$;
23     $popP \leftarrow$Select Best first $p$ using $cd$ and ranked solutions from $fullPop$ ;
24 end
25 return $p$ Biclusters $popP$ from last generation;
26 Function BBDE(a: vector, b: vector, c: vector, F: real) : vector is
       $newVector = nodeAddition(a, nodeDeletion(b, c, F), F)$
       $nodeAddition(x, y, F) = \begin{cases} 1, \; if \;\; x_i = 1 \;\; or \;\; (y_i = 1 \;\; and \;\; rand_i < F) \\ 0, \quad \text{otherwise} \end{cases}$
       $nodeDeletion(x, y, F) = \begin{cases} 0, \; if \;\; x_i = 0 \;\; or \;\; (y_i = 1 \;\; and \;\; rand_i < F) \\ 1, \quad \text{otherwise} \end{cases}$
       return $newVector$
27 end
```

4.3.2 Illustrative Example

The following is an example of how the node addition and deletion given by equations 4.16 and 4.15 work:

considering the matrix M(5,5), where the rows represent genes and the columns represent conditions:

$$M = \begin{pmatrix} 5 & 64 & 44 & 2 & 39 \\ 20 & 6 & 3 & 23 & 68 \\ 38 & 19 & 12 & 82 & 4 \\ 1 & 67 & 41 & 27 & 98 \\ 37 & 12 & 20 & 95 & 53 \end{pmatrix}$$

The first population P generated at random contains four solutions (biclusters); as shown in Figure 4.2 in section 4.3.1, the biclusters are represented in binary, and the size of a bicluster depends on the number of genes and conditions in the matrix M; thus, a bicluster is represented by a 10-bit string.

$$p = \begin{array}{c} a \\ b \\ c \\ d \end{array} \begin{pmatrix} 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \end{pmatrix}$$

The genes and conditions included in the first bicluster (a) of the P population are as follows: $a = \begin{pmatrix} 6 & 3 & 68 \\ 67 & 41 & 98 \end{pmatrix}$

We use our new BBDE operator, as described in section 5.1, to perform the mutation operation; it randomly chooses three solutions from the population P, assuming the selected solutions are b,c, and d, and the value of the Factor $F = 0.65$, the node deletion and addition operations are executed as the following:

$$nodeDeletion(c, d, F) = \frac{\begin{array}{cccccccccc} 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \end{array}}{\begin{array}{cccccccccc} 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \end{array}}$$

When performing the node deletion operation presented above, we will generate a random number $rand_i \in \{0, ..., 1\}$ (where i represents the position of the current bit in the current individual), for instance: Because $rand_{10}$ was less than F, the algorithm performed node deletion, yielding a result of 0; however, $rand_9$ was greater than F; therefore the algorithm did not perform the node deletion, obtaining a result of 1, and the same process is repeated for the remaining bits as well as the node addition operation.

$$nodeAddition(b, nodeDeletion(c, d, F), F) = \frac{\begin{array}{cccccccccc} 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \end{array}}{\begin{array}{cccccccccc} 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 \end{array}}$$

The interpretation of the result of BBDE(b,c,d) is the solution consists of one row and four columns; therefore it represents the following bicluster:

$$BBDE(b, c, d) = \begin{pmatrix} 38 & 19 & 12 & 4 \end{pmatrix}$$

4.3.3 Parameter Setting

The selection of parameter settings is frequently influenced by practical experimentation to determine values that offer higher algorithm performance, while also considering the specific importance and impact of these parameters on the algorithm's performance. This is further elaborated in the following discussion, providing insight into the variables that inspired these parameter choices.

Population Size (p): The population size, represented by p, determines the number of evolving solutions in each generation. While a greater population size allows for more extensive investigation of the solution space, it also increases computational time. Choosing p necessitates careful assessment of available resources as well as the problem's complexity.

Initial Crossover Probability (CP): The initial crossover probability (CP) is an important factor in determining the possibility of using the crossover operator to construct a trial vector. Starting with a higher CP number may encourage exploration, whereas starting with a lower value may prioritize exploitation. The value of CP chosen shapes the compromise between these two factors. CP evolves adaptively in our algorithm depending on the incremental improvements detected in the solutions of each iteration.

Initial Factor (F): The initial factor, abbreviated as F, is a component of the BBDA mutation operator that impacts mutation strength. Higher F values encourage exploration over exploitation, while lower levels encourage the latter. The precise adjustment of F ensures that these aspects are balanced. F adapts in our method based on the benefits observed after each repetition.

Termination conditions: Algorithms are terminated based on particular conditions such as maximum iteration count or convergence level. These criteria prevent infinite execution while giving enough time for a satisfactory solution to emerge.

BBDE Mutation Parameters: There are two aspects to the BBDE mutation: node addition and deletion. The algorithm's diversity and exploration abilities are shaped by parameters inside these components, such as random values and comparisons (e.g., $rand_i$, F).

Crowding Distance and Selection: During the selection step, the calculation of crowding distance ensures diversity within selected solutions. Careful tuning of crowding distance parameters provides a harmonic balance between selection pressure and diversity preservation.

Fitness Functions (f1, f2): The choice of fitness functions is determined by the nature of the problem. In our scenario, bicluster size and MSR were used to direct the algorithm toward significant biclusters.

4.3.4 Computational Complexity

The complexity of our approach can be broken down as follows:

- Initializing the population size as p, the vector length as m, Arrays of fitness, crossover probability CP, and the factor F all lead to an initial complexity of $O(p)$.
- The main loop iterates until a termination condition is met, typically a fixed number of iterations, resulting in $O(iteration)$ complexity.
- Within each iteration of the main loop:
  - Initializing *popQ* takes $O(1)$.
  - Selecting three random solutions takes $O(1)$.
  - Applying the $BBDE$ (our proposed mutation operator) takes $O(m)$.
  - Calculating fitness for solutions takes $O(1)$.
  - Updating control parameters CP and F factor takes $O(1)$.

  - Therefore, the overall complexity within each iteration is $O(iteration \times p \times m)$.

- Merging $popP$ and $popQ$ takes $O(1)$.
- The procedure of NSGA-II has a complexity of $O(mN^2)$, where m is the objective number (2 objectives in our approach) and N is the solutions number (which is p), therefore, the result is $O(p^2)$.

Putting it all together, the overall complexity of our approach can be approximated as $O(\text{iteration} \times p \times m + p^2))$. This can further be simplified to: $O(p \times (\text{iteration} \times m + p))$.

This complexity analysis highlights that the computational effort primarily depends on the population size p, the number of iterations, and the vector length m. Our approach efficiently balances these factors, making it capable of handling high-dimensional data with a reasonable computational cost.

4.3.5 Multi-Objective Fitness Functions

We have taken into account a number of strategies to efficiently incorporate multiple objectives in our AMoDeBic algorithm while designing multi-objective fitness functions. Normalizing and scaling the goals to a consistent range is a common strategy, assuming that the domain knowledge is sufficient to calculate the right scaling factors. However, if precise ranges are unknown or if the right individual weighting is still unclear, this approach may be difficult.
Genuine multi-objective optimization forms the foundation of an alternative approach that is frequently more effective. This approach includes pursuing a set of Pareto-optimal solutions rather than attempting to combine objectives into a single function. The optimization process produces a variety of solutions in this paradigm, each of which excels in a different aspect of the objectives. For instance, while some solutions excel primarily in achieving a single objective (e.g., maximizing p), others (e.g., maximizing v), and a variety of solutions achieve various balancing acts between the two, there are many solutions that excel in both.

To achieve this, multi-objective evolutionary algorithms offer a well-established toolkit. Numerous methods have been extensively studied in the literature, including NSGA-II (Non-dominated Sorting Genetic Algorithm II) [65] and SPEA (Strength Pareto Evolutionary Algorithm) [74]. These methods make sure the algorithm keeps the Pareto-optimal solutions diverse, allowing for the discovery of a wide range of trade-off solutions.

In our particular situation, we have used the Pareto optimization method in the AMoDeBic algorithm with two straightforward but effective objectives: larger bicluster size and smaller MSR. By doing this, we want to reach a set of biclustering solutions that represent the optimum trade-offs between various goals, improving the algorithm's ability to recognize intricate patterns in microarray data while adhering to numerous biologically pertinent criteria.

4.4 Experimental Results

The AMoDeBic algorithm is evaluated using both synthetic and real DNA microarray datasets on multiple levels. The validation criteria is as follows:

Using Synthetic Data:

- **Overlap Assessment**
  - This test assesses AMoDeBic's capacity to detect biclusters with shared genes or conditions.
- **Noise Resistance Assessment**
  - This test evaluates AMoDeBic's resilience in detecting significant biclusters in the presence of noise.

Using Real Data:

- **Test on Well-Known Datasets:**
  - Yeast Cell-Cycle expression Dataset [75].
  - Saccharomyces Cerevisiae Dataset [76] .
  - Human B-Cell Lymphoma Dataset [77].
- **Statistical Analysis:**
  - Coverage tests measure the extent of true pattern capture.
  - P-value tests assess bicluster statistical significance.
- **Biological Relevance**

– In-depth assessment of biclusters' biological relevance.

By conducting comprehensive evaluations on synthetic and real datasets, encompassing aspects such as overlap resistance, noise handling, biological relevance, and statistical analysis, AMoDeBic's performance is thoroughly assessed.
The algorithms employed for comparison include the following:

- For synthetic data, we compare our results with the results of some prominent biclustering algorithms used by the community; namely, CC [2], OSPM [4], ISA [38], Bimax [5], and Xmotif [39].
- For real datasets, we compare the performance of our algorithm with the results of CC [2], BicFinder [23], BiMine [24], BiMine+ [25], SEBI [8], MOEA [7], MOP-SOB [68], ISA [38], Bimax [5], and MBA [41].

All the experiments for both synthetic and real datasets were carried out with the parameter settings shown in Table 4.1 on a PC running with Windows 10 and equipped with a 4.30 GHz AMD Ryzen 7 3800XT 8-Core Processor and 64 GB of RAM.

Hyper Parameter	**Value**
Population Size	100
Generations	100
Initial Crossover Probability	0.75
Initial F (Factor)	0.65

Table 4.1: Hyper Parameter Specification.

4.4.1 Results on Synthetic Data

The purpose of experimenting with synthetic data is to see whether an algorithm can extract all of the embedded biclusters. For this test, we use the synthetic data generated by [5] to investigate the effect of noise and overlap in expression matrices with non-overlapping biclusters on the performance of our algorithm. The datasets contain ten implanted biclusters and have been used to study the effects of noise on the performance of the biclustering algorithms, where different types of biclusters, such as constant and additive biclusters, are embedded, spanning ten genes and five conditions. We will use the following gene match score proposed by [5] to evaluate the performance of our algorithm on synthetic data. This measure is defined as follows:

$$S^*_G(M, M_{opt}) = \frac{1}{|M|} \sum_{(G_1,C_1)\in M} \max_{(G_2,C_2)\in M_{opt}} \frac{|G_1 \cap G_2|}{|G_1 \cup G_2|} \tag{4.18}$$

This score is the average of the maximum match scores for all biclusters in M in relation to the biclusters in M_{opt}, where M is the set of extracted biclusters from the biclustering algorithms, and M_{opt} is the set of implanted biclusters.

Figures 4.3 A (noise with constant biclusters) and 4.3 B (noise with additive biclusters) demonstrate the performance of several biclustering approaches for extracting non-overlapping biclusters from scenarios with varying noise levels, while Figures Figure 4.4 A (overlap with constant biclusters) and Figure 4.4 B (overlap with additive biclusters) present the results of these biclustering algorithms with increasing degrees of overlap in the absence of noise.

We find that for non-overlapping biclusters and in the absence of noise (Figure 4.3), the ISA algorithm can identify 100% of the implanted biclusters and that for additive biclusters, both ISA and Bimax can identify almost all of the implanted biclusters. AMoDeBic, on the other hand, can recognize 90% of constant implanted biclusters and 85% of additive biclusters. Only ISA maintained its high percentage at high noise levels for both types of biclusters under noise influence. AMoDeBic clearly outperforms other biclustering methods, AMoDeBic maintains its performance for the extraction of both constant and additive biclusters, even at high noise levels.

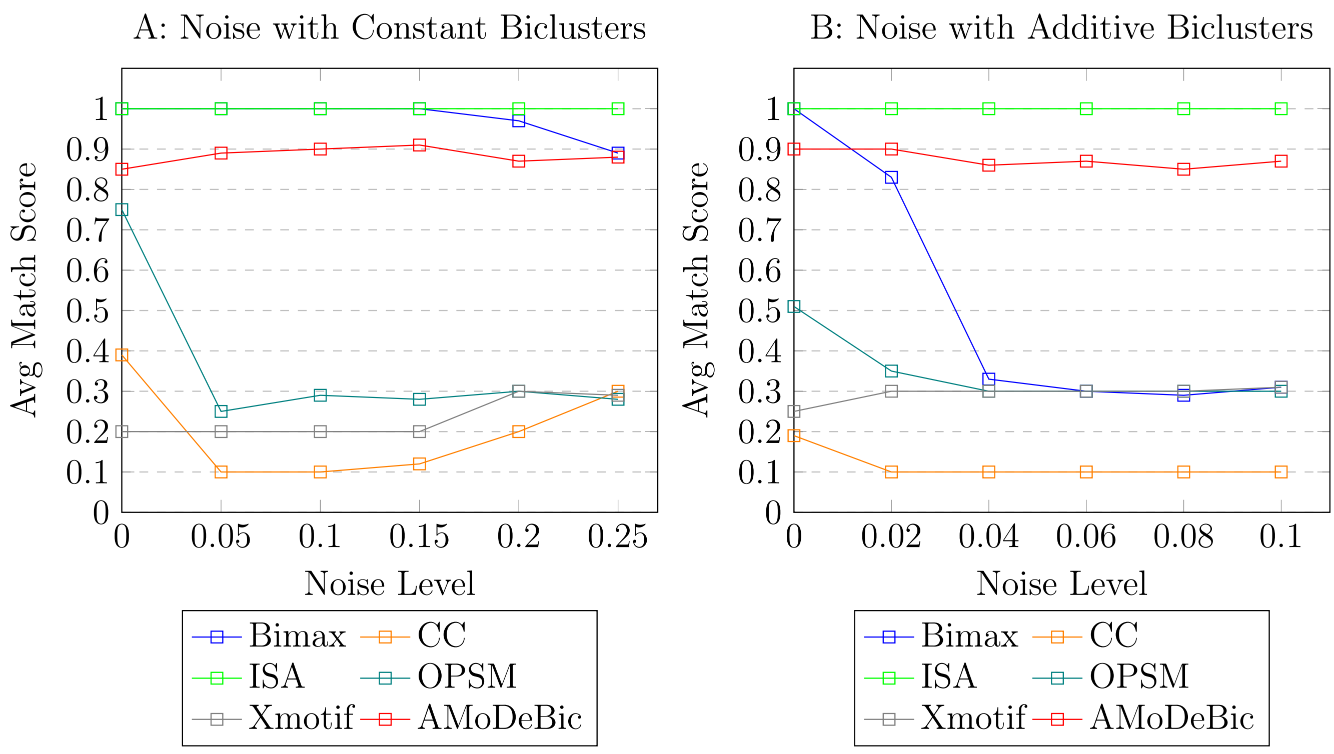

Figure 4.3: Noise Effect. A: Constant Biclusters, B: Additive Biclusters.

Regarding the impact of overlap (Figure 4.4), it is clear that after a certain amount of overlap, the performance of most biclustering methods degrades. Bimax's performance was maintained for both constant and additive biclusters. ISA was at its best at the start. However, as the degree of overlap increased, its performance decreased significantly. AMoDeBic, on the other hand, was able to recognize 90% of constant implanted biclusters and 80% of additive biclusters. Its performance decreased as the overlap increased, but it still outperformed the other algorithms.

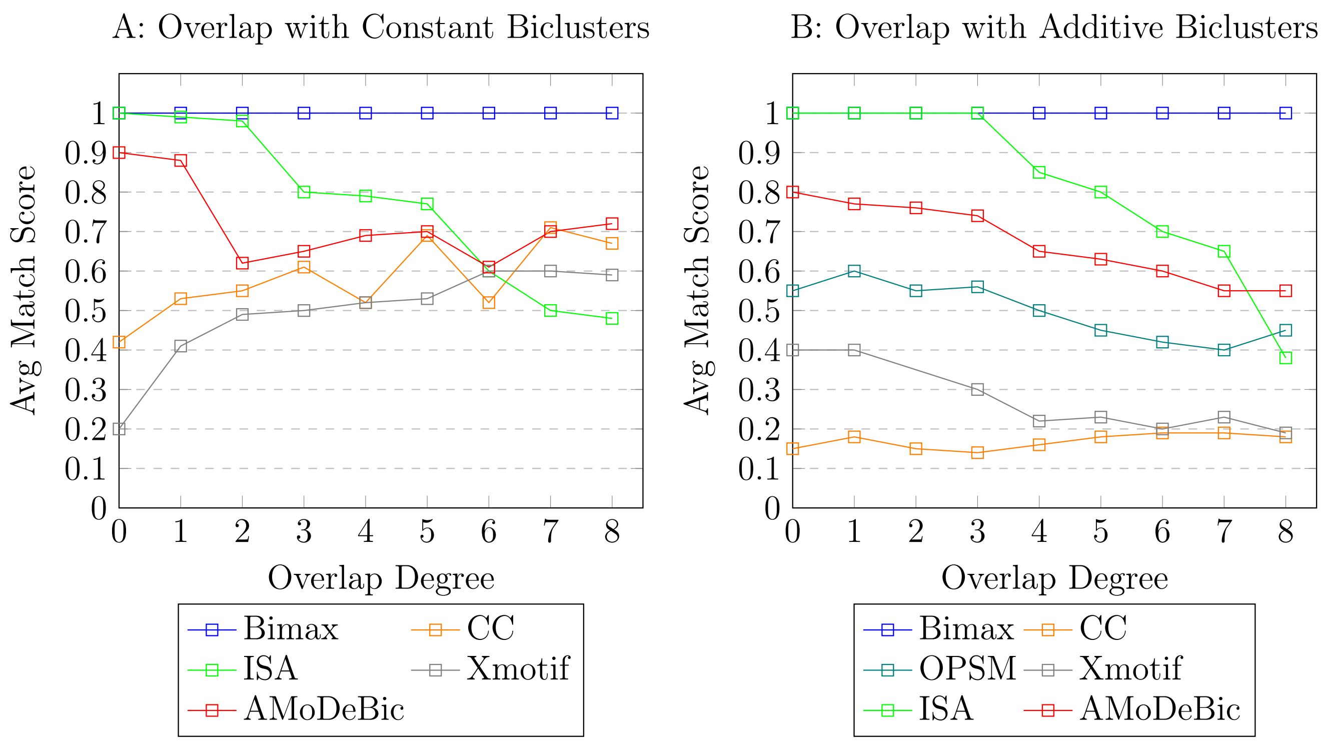

Figure 4.4: Overlap Effect. A: Constant Biclusters, B: Additive Biclusters.

4.4.2 Results on Real Data

After applying and validating the algorithm on synthetic data, we need to validate its applicability on a real-world dataset; this section provides information and gathered results for real datasets.

4.4.2.1 Description of Considered Datasets

1. Yeast Cell-Cycle dataset: The Yeast Cell-Cycle [75] is a particularly prominent dataset in gene expression data. It reveals gene expression profiles across several stages of the yeast cell cycle, providing information about cell regulation, DNA replication, and division mechanisms. It's important for cell and systems biology, drug development, and understanding basic biological processes. This dataset includes the expression profiles of 2,884 genes and 17 experimental conditions.

2. Saccharomyces Cerevisiae dataset: The Saccharomyces Cerevisiae dataset [76] is used for studying basic cellular processes in eukaryotic creatures. Biclustering can reveal light on metabolic processes, cell cycle regulation, and stress responses by detecting coherent gene expression patterns. This dataset contains the expression

levels of 2993 genes under 173 experimental conditions.

3. Human B-Cell Lymphoma dataset: The Human B-Cell Lymphoma dataset [77] has the potential to reveal gene expression patterns linked with B-cell lymphomas, a heterogeneous group of malignancies that influence the immune system. Biclustering can assist in identifying clusters of co-expressed genes that play critical roles in lymphoma development, progression, and treatment response by uncovering underlying biological processes, potentially leading to improved diagnostic and treatment options. This dataset contains 4026 genes and 96 conditions.

4.4.2.2 Statistical Relevance

A crucial stage in biological validation and interpretation of obtained biclusters is statistical validation. This involves identifying the statistical significance of the discovered biclusters to determine whether or not the co-expression patterns seen inside biclusters are statistically significant when compared to random chance. In our approach, we strongly rely on the coverage criterion as well as the p-value criterion to assess the statistical relevance of our algorithms.

Coverage:

The coverage refers to the total number of cells comprised by the retrieved biclusters. This evaluation is used to validate the effectiveness of biclustering methods. Higher coverage often demonstrates the algorithm's capacity to detect important patterns in the data; this can also lead to less overlap between biclusters, ensuring that each bicluster captures a different set of genes and samples.

During the evaluation phase, we rigorously tested our algorithm and conducted a comparative analysis against several established algorithms: CC [2], BicFinder [23], BiMine [24], BiMine+ [24], SEBI [8], MOEA [7], MOPSOB [68]. This test was performed on the datasets Yeast Cell-Cycle and Human B-Cell Lymphoma.

The coverage of the obtained biclusters is shown in Tables 4.2 and 4.3. We notice that most algorithms produce reasonably close results. The biclusters extracted by our algorithm AMoDeBic cover 75.31% (respectively 79.94%) of the genes, 100% of the conditions, and 52.73% (respectively 67.28%) of the cells in the Human B-Cell Lymphoma (respectively Yeast Cell-Cycle) dataset.

The AMoDeBic algorithm exhibits stronger performance compared to BiMine, BiMine+, BicFinder, MOPSOB, MOEA, and SEBI, for both total and gene coverage measures. Additionally, it surpasses the CC algorithm in terms of total coverage, while the CC algorithm achieves better gene coverage.

Furthermore, it is important to note that the CC algorithm uses a masking strategy on groups extracted using random values. This procedure ensures that previously identified genes/conditions are not chosen in subsequent search iterations. This intentional masking strategy makes a significant contribution to achieving broad coverage. As a result, the CC approach outperforms other algorithms on the Yeast Cell-Cycle dataset.

Human B-Cell Lymphoma			
Algorithms	**Total coverage**	**Gene coverage**	**Condition coverage**
BiMine	8.93%	26.15%	100%
BiMine+	21.19%	46.26%	100%
BicFinder	44.24%	55.89%	100%
MOPSOB	36.90%	-	-
MOEA	20.96%	-	-
SEBI	34.07%	38.23%	100%
CC	36.81%	91.58%	100%
AMoDeBic	**52.73%**	**75.31%**	**100%**

Table 4.2: Human B-Cell Lymphoma coverage for different algorithms.

Yeast Cell-Cycle			
Algorithms	**Total coverage**	**Gene coverage**	**Condition coverage**
BiMine	13.36%	32.84%	100%
BiMine+	51.76%	68.65%	100%
BicFinder	55.43%	76.93%	100%
MOPSOB	52.40%	-	-
MOEA	51.34%	-	-
SEBI	38.14%	43.55%	100%
CC	81.47%	97.12%	100%
AMoDeBic	**67.28%**	**79.94%**	**100%**

Table 4.3: Yeast Cell-Cycle coverage for different algorithms.

Tables 4.4 and 4.5 display details regarding five biclusters identified by AMoDeBic within the Yeast Cell-Cycle dataset and the Human B-Cell Lymphoma dataset. The tables include the Mean Squared Residue (MSR) values for each bicluster. Notably, the largest bicluster in these tables comprises 941 genes for the Yeast Cell-Cycle dataset and 1023 genes for the Human B-Cell Lymphoma dataset.

Genes	Conditions	Size	MSR
730	15	10950	299.14
209	13	2717	298.94
507	17	8619	298.047
61	10	610	297.77
941	17	15997	299.172

Table 4.4: Five biclusters found By AMoDeBic on Yeast dataset.

Genes	Conditions	Size	MSR
616	42	25,872	1195.627
824	21	17,304	1196.526
772	28	21,616	1196.988
957	20	19,140	1192.104
1023	31	31,713	1199.632

Table 4.5: Five biclusters found By AMoDeBic on the Human B-Cell Lymphoma dataset.

P-value:

This statistical evaluation method assess if the observed patterns inside the biclusters are statistically significant or could have occurred by chance to evaluate the significance of each bicluster's characteristics in the dataset. This method provides a solid statistical framework for assessing the biclusters' dependability and meaningfulness, allowing for more informed decisions concerning their quality and significance, with the most favorable biclusters displaying an adjusted p-value of less than 0.001%.

We conducted the p-value test using the web tool **FuncAssociate** [1] [78]. The assessment includes the Yeast Cell-Cycle and Saccharomyces Cerevisiae datasets, as well as a comparison of our algorithm's performance to the results of CC [2], ISA [38], Bimax [5], and MBA [41] algorithms.

Figures 4.5 and 4.6 present the outcomes achieved from the Saccharomyces Cerevisiae and Yeast Cell-Cycle datasets, considering diverse adjusted p-values (p = 5%; 1%; 0.5%; 0.1%; 0.001%) for each algorithm across the spectrum of total biclusters.

[1]http://llama.mshri.on.ca/funcassociate/

Analyzing the Saccharomyces Cerevisiae dataset (Figure 4.5), both the AMoDeBic and MBA results showcase that approximately 98% of the identified biclusters hold statistical significance, boasting an adjusted p-value of 0.001%. Remarkably, AMoDeBic and MBA yield identical results for other adjusted p-values. Conversely, CC, ISA, and CC exhibit consistently average performance across all p-values, signaling less distinguished outcomes.

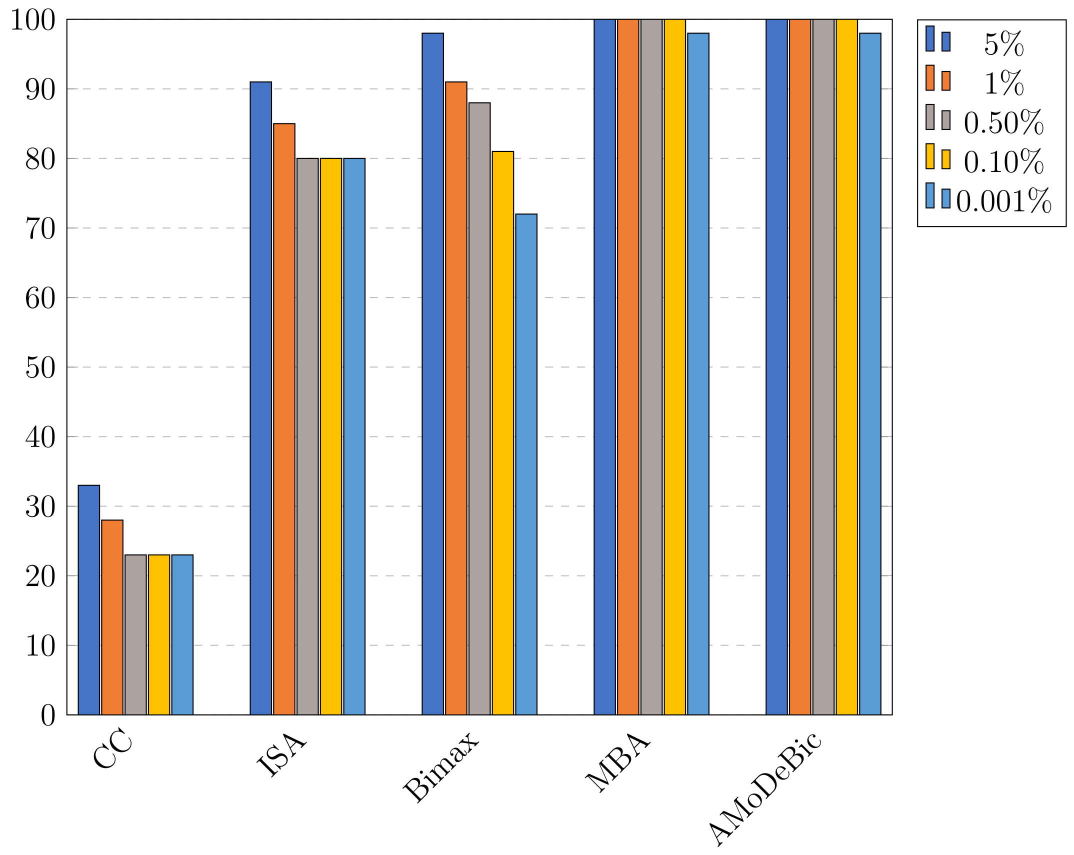

Figure 4.5: Proportions of biclusters significantly enriched by GO annotations (Saccharomyces Cerevisiae dataset)

Examining the Yeast Cell-Cycle dataset (Figure 4.6), we observe that all the biclusters identified by the AMoDeBic algorithm attain statistical significance for adjusted p-values of 0.5%, 1%, and 5%. In contrast, Bimax achieve a complete coverage of extracted biclusters at a p-value of 1%. Notably, ISA and CC demonstrate their best performance when the p-value is set to 5%. Consequently, our results surpass those of Bimax, ISA, and CC. However, it's pertinent to acknowledge that Bimax exhibits enhanced performance when p is less than 0.1%.

Table 4.6 presents a comparative analysis of AMoDeBic's performance on the Yeast Cell-Cycle dataset in contrast to algorithm like CC [2], FLOC [3], MOEA [7], and DeBic [31]. The comparison is based on both the average residue and average dimension of the

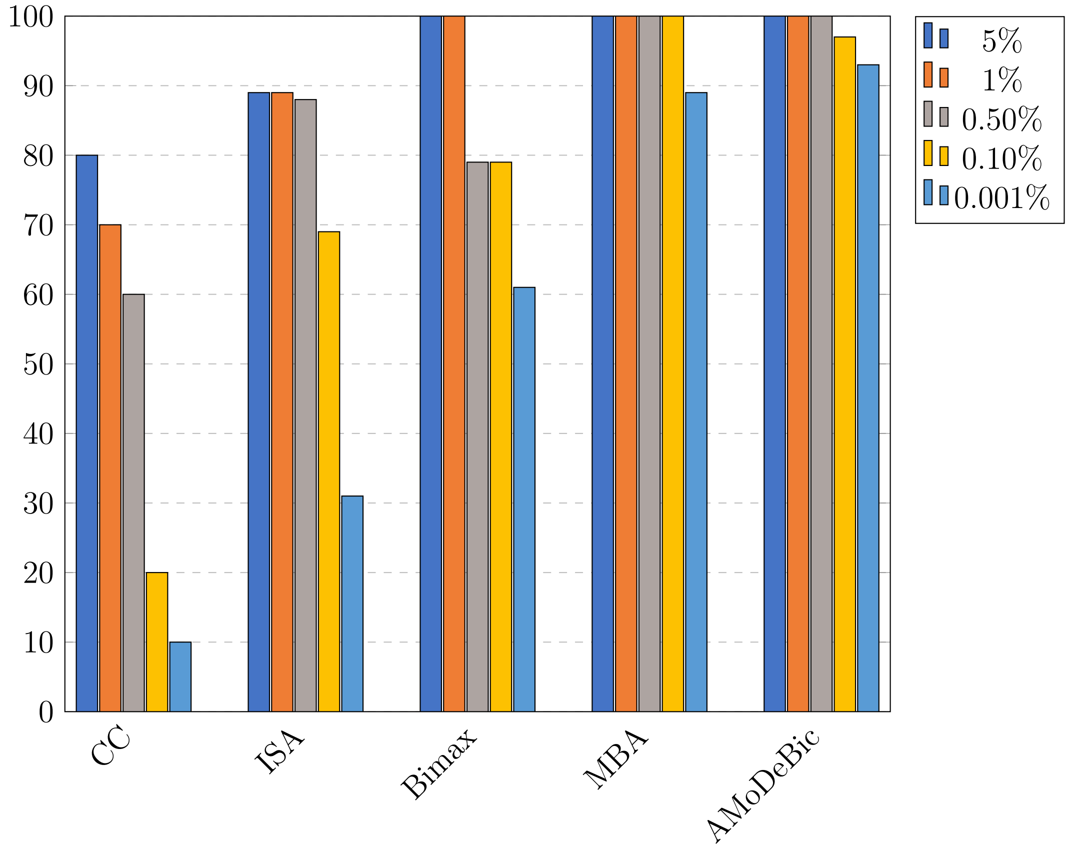

Figure 4.6: Proportions of biclusters significantly enriched by GO annotations (Yeast cell-cycle dataset)

identified biclusters.

The provided table demonstrates the capability of our algorithm to identify large biclusters while maintaining an acceptable Mean Squared Residue (MSR). Noteworthy is our algorithm's superior performance in terms of the average number of genes compared to some of the other algorithms. Additionally, it surpasses all algorithms in terms of the average number of conditions discovered within the Yeast dataset.

Algorithm	**Avg. bicluster size**	**Avg. residue**	**Avg. no. of genes**	**Avg. no. of conditions**
CC	1576.98	204.29	167	11
FLOC	1825.78	187.54	195	12.8
MOEA	10301	234.87	1095	9.29
DeBic	9889.58	272.13	628.04	14.5
AMoDeBic	10756.012	297.02	721.88	14.9

Table 4.6: AMoDeBic Performance Comparison To Other Algorithms.

the statistical analysis, supported by the examination of adjusted p-values and coverage metrics, contributes greatly to the decisive conclusions. Using modified p-values, we can evaluate the statistical significance of the extracted biclusters, distinguishing between random occurrences and meaningful patterns. Whereas the coverage parameter assures that the detected biclusters cover a significant amount of the data, strengthening their statistical and biological significance.

Furthermore, comparing our algorithm's performance with established approaches, particularly in terms of bicluster size, and the number of conditions addressed, validates our findings. By objectively measuring statistical significance and taking into account the breadth of bicluster coverage, this combination technique strengthens the robustness of our study's conclusions.

4.4.2.3 Biological Relevance

Biological relevance is an essential phase for validating a biclustering approach as it guarantees that the identified patterns are consistent with existing biological knowledge. Relevant biclusters represent meaningful functional relationships within biological systems, improving the results' credibility and interpretability.

This assessment employs the widely-used web tool **GOTermFinder** [2], which identifies significant shared Gene Ontology (GO) terms among selected gene groups. The GO is categorized into biological process, molecular function, and cellular component. This evaluation is performed on both the Yeast Cell-Cycle and Saccharomyces Cerevisiae datasets.

Figures 4.7 and 4.8 showcase nine biclusters identified by the AMoDeBic algorithm within the Yeast Cell-Cycle and Saccharomyces Cerevisiae datasets, respectively. In these figures, genes display comparable behavior under specific conditions, as discerned through visual inspection.

[2]https://www.yeastgenome.org/goTermFinder

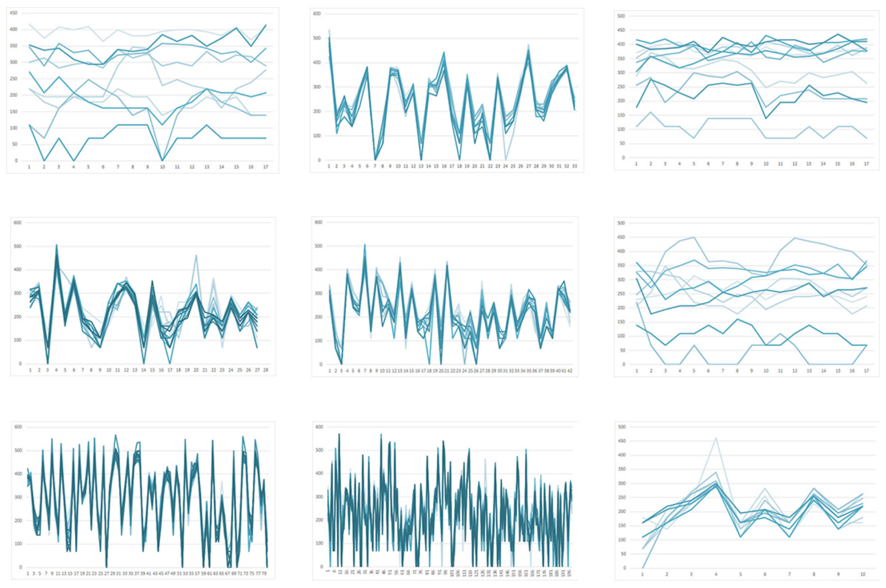
Figure 4.7: Nine Biclusters discovered in the Yeast dataset.

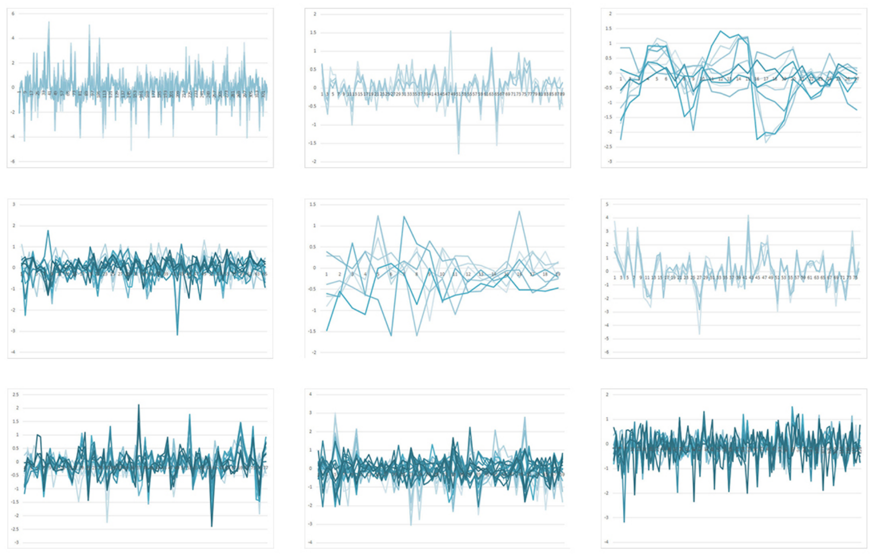
Figure 4.8: Nine Biclusters discovered in the Saccharomyces Cerevisiae dataset.

The following list contains the most significant Gene Ontology (GO) terms shared within the first bicluster extracted from the yeast cell-cycle dataset, particularly related to nuclear nucleosome activity:
YGL211W YGR243W YGR151C YNL031C YBR010W YBR009C YGR142W YGL248W YGL250W YGL262W YGR149W YGR206W YGR289C YGR190C YGR065C YGR088W

YGL209W YGL230C YGR039W YGR053C YGR090W YGL236C YGR023W YGR143W YPL127C YGR010W YGR152C YGR035C YNL030W YGL204C YGR259C YGR041W YBL002W YGR109C YGR265W YGR214W YGR207C YGR164W YGR099W YGR238C YHL007C YGR150C YGR153W YGL255W YGR129W YGR146C YGR131W YGR122W YGR212W YGL246C YGR019W YGR112W YBL003C YGR140W YGR108W YGR245C YGR159C YGL219C YGL229C YGL232W YGR216C YGR187C YGR104C YGL218W YGR189C YGL214W YGR239C YGR032W YGR089W YGR138C YGR154C YGR200C YGR058W YGR043C YGR114C YDR225W YGR042W YGR197C YDR224C YGR113W YGR118W YGR263C YGR014W YGR288W

Tables 4.7 and 4.8 display the biological annotations for two randomly chosen biclusters within the yeast cell-cycle and Saccharomyces Cerevisiae datasets, respectively. These annotations are presented across the previously mentioned Gene Ontology (GO) categories, showcasing the most noteworthy GO terms associated with each bicluster.

GO Terms	**Bicluster 1**	**Bicluster 2**
Biological Process	biological regulation (29.1%, 9.78e-29) regulation of cellular process (22.3%, 9.98e-36) biological regulation (29.1%, 9.78e-29)	heterocycle metabolic process (29.4%, 0.00069) organelle organization (18.5%, 0.00030) nucleic acid metabolic process (25.0%, 0.00013)
Cellular component	nucleus (34.7%, 3.80e-13) membrane-bounded organelle (63.9%, 0.00026) intracellular organelle (68.1%, 0.00093)	protein-containing complex (31.4%, 9.39e-14) organelle lumen (14.6%, 0.00348) organelle (68.2%, 4.60e-05)
Molecular function	binding (48.9%, 4.18e-05) ion binding (23.4%, 0.00101) organic cyclic compound binding (33.1%, 0.00832)	heterocyclic compound binding (32.9%,0.00065) nucleic acid binding (23.1%, 1.90e-05) DNA binding (8.5%,8.12e-10)

Table 4.7: Significant GO terms of two biclusters extracted from Yeast Cell-Cycle dataset using AMoDeBic.

GO Terms	**Bicluster 1**	**Bicluster 2**
Biological Process	Mitotic Nuclear Division (2.3%, 1.43e-18) Cell Cycle Process (9.1%, 3.10e-11) organelle fission (3.7%, 5.74e-16)	regulation of cell cycle (4.1%, 3.65e-19) cell cycle phase transition (3.1%, 2.49e-18) regulation of biological process (24.2%, 1.55e-07)
Cellular component	polymeric cytoskeletal fiber (1.1%, 3.96e-10) microtubule cytoskeleton (2.0%, 2.72e-12) supramolecular complex (3.7%, 2.30e-06)	polymeric cytoskeletal fiber (1.1%, 2.79e-08) spindle microtubule (0.3%, 1.74e-05) supramolecular complex (3.7%, 8.66e-08)
Molecular function	cytoskeletal protein binding (1.6%, 4.18e-07) tubulin binding (0.7%, 3.05e-09) microtubule binding (0.6%, 8.18e-10)	non-membrane-bounded organelle (21.6%, 0.00354) astral microtubule (0.1%, 0.00020) cytoskeleton (3.6%, 1.16e-05)

Table 4.8: Significant GO terms of two biclusters extracted from Saccharomyces Cerevisiae dataset using AMoDeBic.

Experiments show that the AMoDeBic algorithm, along with the new mutation technique, is effective as a biclustering approach. The algorithm demonstrates its ability to discover numerous significant and large biclusters at the same time. The results show that AMoDeBic is robust in the presence of noise and overlapping patterns. Furthermore, the analysis emphasizes the biological significance of the observed biclusters. Nevertheless, it is important to point out that the a multi-objective differential evolution based biclustering algorithm's performance is sensitive to parameter tuning, requiring careful optimization to achieve the most effective outcomes. Furthermore, the algorithm's processing complexity may become a challenge when dealing with large datasets, emphasizing the importance of addressing scalability concerns for efficient analysis.

4.5 Conclusion

In this chapter, we comprehensively examined essential aspects of Differential Evolution (DE), focusing specifically on Multi-Objective Differential Evolution (MODE) and Binary DE. The interplay between these components is crucial in the proposed MODE-based biclustering approach, AMoDeBic. A significant contribution is the introduction and analysis of the Biclustering Binary Differential Evolution (BBDE) mutation operator, facilitating DE's integration into biclustering. We addressed key considerations, encompassing algorithmic complexity, parameter settings, and the implications of BBDE. Additionally, we carried out experimental assessments on both synthetic and real-world datasets, rigorously comparing the proposed AMODEBIC approach with state-of-the-art methodologies.

The following chapter will focus on a novel approach that harnesses the synergies of two potent approaches: Convolutional Denoising Autoencoders (CDAs) and Artificial Bee Colony (ABC).

Chapter 5

CDABC: A Convolutional Denoising Autoencoder with Artificial Bee Colony Biclustering Algorithm for Gene Expression

5.1 Introduction

In computational genomics, two key methodologies play crucial roles in our proposed solution: Convolutional Denoising Autoencoders (CDAs) and the Artificial Bee Colony (ABC) approach. CDAs serve as adept preprocessing tools, refining gene expression data by mitigating noise and highlighting intrinsic patterns. Simultaneously, the ABC algorithm, inspired by the foraging behavior of bees, emerges as a robust optimization technique for biclustering. This section offers a succinct introduction to CDAs and ABC, setting the stage for a detailed exploration and subsequent unveiling of CDABC—a novel biclustering algorithm seamlessly integrating these two methodologies. The upcoming sections will delve into the algorithmic intricacies of CDABC and present its performance in comparison to established biclustering approaches.

5.2 Autoencoders

An autoencoder is a feed-forward neural network that uses a lower dimensional hidden layer to reconstruct the given input. It is composed of an encoder and a decoder. The encoder maps the input x to a hidden representation z using an affine transformation followed by a non-linear function. This transformation can be defined as follows [79]:

$$z = f(W_x + b) \tag{5.1}$$

The decoder, then, maps back the hidden representation z to reconstruct the original input data using another nonlinear transformation given as the following [79]:

$$x\prime = g(W\prime_z + b\prime) \tag{5.2}$$

where W and b represent the encoder's weight and bias matrices, respectively, and $W\prime$ and $b\prime$ represent the decoder's weight and bias matrices, respectively, while f and g represent nonlinear activation functions. Several choices exist for the functions f and g, including a sigmoid function, hyperbolic tangent, and rectified linear function.

5.2.1 Denoising Autoencoders

Denoising autoencoders (DAEs) are a simple modification of autoencoder neural networks proposed by Vincent et al. [52]. They are trained to denoise an artificially corrupted version of their input rather than reconstruct it. While a regular autoencoder with an overcomplete representation can learn an uninformative identity mapping quickly, a denoising autoencoder (DAE) is forced to extract more meaningful features in order to undertake the much more difficult task of denoising.

The basic idea behind denoising autoencoders is to introduce noise or corruption into the input data $\check{x}$ and then train the network to map it to the corresponding hidden representation z and ultimately to its reconstruction $x\prime$.

5.2.2 Convolutional Denoising Autoencoders

Convolutional Denoising Autoencoders (CDAEs) use the standard DAE architecture with convolutional encoding and decoding layers; they are primarily used for image preprocessing, but their applications go beyond that. While previous research has shown that CDAEs outperform in image-related tasks [80, 81], they show great promise in gene expression analysis since they exploit the patterns and structures inherent in gene expression data by using the full power of convolutional neural networks. CDAEs differ from standard DAEs as they share weights (W) and biases (b) across all gene and condition combinations to capture spatial and temporal dependencies within the dataset while preserving important contextual information.

5.3 Artificial Bee Colony

The Artificial Bee Colony (ABC) algorithm put forth by [82] is a swarm intelligence optimization algorithm that takes its cues from honey bees' foraging habits. In ABC, an artificial bee population represents possible answers to an optimization issue. There are three types of bees: employed bees, onlooker bees, and scout bees, representing the main components of honey bee behavior that are useful for resolving search issues. Scouts and observers are also referred to as unemployed bees.

- *Employed bees*: They are linked to particular food sources (solutions), take advantage of the knowledge they have learned about those sources, and look for new food sources by conducting local searches close to their current locations.
- *Onlooker bees*: They observe the employed bees and choose food sources based on their quality and nectar value (the corresponding fitness values), effectively utilizing promising search space regions.
- *Scout bees*: They bring diversity by randomly seeking out food sources. After a certain number of iterations, they stop using food sources that haven't improved and start looking for new ones.

Initially, all food source positions are discovered by scout bees; then, each employed bee is associated with a single food source. The employed bees can be thought of as trying to maximize some function of the quality and quantity of the nectar value of the food source. A food source's location is communicated to the onlooker bees with a probability inversely correlated with the nectar value of the food source once it has been discovered. This mechanism ensures that more bees are attracted to food sources with higher nectar values, allowing them to exploit these rich sources. When a nectar source is completely consumed, the bee abandons it and begins searching the immediate area for new nectar sources.

The ABC algorithm is iterative or cycle-based, with each cycle consisting of the employed bee phase, onlooker bee phase, and scout bee phase. The algorithm seeks to improve the solutions iteratively by exploring the search space and utilizing the promising solutions discovered thus far. It continues until a termination condition, such as reaching a maximum number of iterations or reaching a satisfactory solution, is met.

The algorithmic structure of Artificial Bee Colony is as follows:

```
Initialization Phase
Repeat
Employed Bees Phase
Onlooker Bees Phase
Scout Bees Phase
Until Cycle = Maximum Cycle Number or a Maximum CPU time
```

The Artificial Bee Colony delivers promising results when solving optimization problems; however, it is not originally designed to solve binary optimization problems where the solutions have a binary representation; this is due to the Eq. 5.3 used in the employed bees phase to generate new solutions:

$$v_{ij} = x_{ij} + \theta_{ij}(x_{ij} - x_{kj}) \tag{5.3}$$

Eq. 5.3 was particularly intended for real-valued optimization and can't be directly applied to binary solutions; therefore, some modifications to this formula are required in order to use ABC for binary optimization problems. Among the studies and adaptations that have been offered in the literature to address this issue in this context are:

In [83] Kiran and Gündüz presented an XOR-based modification for the ABC algorithm's solution-updating equation in order to tackle binary optimization problems called binary ABC (binABC), in which the positions of the solutions are updated in the employed and onlooker bee phases using the following equation instead of the formula 5.3:

$$V_i^j = X_i^j \oplus [\varphi(X_i^j \oplus X_k^j)] \tag{5.4}$$

Where $i, k \in \{1, 2, ..., N\}, j \in \{1, 2, ..., D\}$and $i \neq k$, V_i^j is the jth dimension of the ith candidate solution, X_i^j is the jth dimension of the ith employed bee, X_k^j is the jth dimension of the kth employed bee, $\oplus$ is The XOR logic operator, and φ is the logic NOT gate with 50% probability. If φ is less than 0.5, the result obtained by $(X_i^j \oplus X_k^j)$ is inverted; otherwise, the result is not inverted.

As another contribution to the binary optimization with ABC, Durgut [84] presents an updated version of binABC as well as an iterative process for dynamically running the rule described by [83] for more dimensions than a single one. The estimation of φ is carried out pragmatically by taking into account the neighboring solution. In contrast to the strategy proposed by Kiran et al. [83] in Eq. 5.3, ibinABC employs a dynamic normalization factor,φ, that varies with each iteration. Eq. 5.3 replaces the initial fixed threshold of $\varphi = 0.5$ with a dynamically adaptable threshold. Due to increasing randomness, the predetermined threshold reduces the exploitation potential, particularly in later phases. Eq. 5.5 shows an innovative approach for calculating. This rule states that φ is set to 0 if the new solution is worse, and is assigned a computed value between [0,1] if the solution is better. The value of φ is updated based on the current iteration, t:

$$\varphi = \begin{cases} \varphi_{max} - (\frac{\varphi_{max}-\varphi_{min}}{t_{max}}) \times t; & F(x_k) < F(x_i) \\ 0; & \text{otherwise} \end{cases} \tag{5.5}$$

where φ_{max} and φ_{min} represent the upper and lower limits of the defined range.

In [85], The suggested method provides a variable for each dimension of the optimization problem to track changes from 0 to 1 or 1 to 0. In the jth dimension, where C_{01}^j counts change from 0 to 1, and C_{10}^j counts change from 1 to 0. Following the generation of a candidate solution using Eq. 5.4, a greedy choice is made between the candidate and the present solution. If the candidate solution is superior, the counters are updated by counting the bits that change from 0 to 1 or 1 to 0 and increasing by 1. The search equation for onlooker bees is revised to incorporate this knowledge into the generation of candidate solutions as the following:

$$P_{01}^j = C_{01}^j/(C_{01}^j + C_{10}^j) \tag{5.6}$$

$$P_{10}^j = C_{10}^j/(C_{10}^j + C_{01}^j) \tag{5.7}$$

$$V_{i,j} = \begin{cases} 0; & if \ \ (rand < P_{10}^j) \\ 1; & \text{otherwise} \end{cases} \tag{5.8}$$

With P_{01}^j being the probability of transitioning from 0 to 1 and P_{10}^j being the probability of transitioning from 1 to 0 for the associated jth decision variable.

In our approach CDABC, we attempt to bridge the gap between the original ABC algorithm and the unique requirements of binary optimization for biclustering by incorporating a new technique for the generation of new solutions, which will be tackled further in Section 5.4.

5.4 Proposed Approach

This research presents CDABC, a novel Biclustering approach that combines two powerful methodologies: Convolutional Denoising Autoencoder and Artificial Bee Colony. In CDABC, the data undergoes a preprocessing phase where the CDAE effectively denoises the data while simultaneously preserving and identifying significant patterns. Subsequently, the ABC Algorithm is slightly adapted and applied to the denoised data to perform biclustering. This innovative approach excels in detecting multiple, diverse, and high-quality biclusters, even in the presence of noise in the data.

5.4.1 Representation

In biclustering, solutions (biclusters) are generally represented by a binary string of size $(n + m)$, where n represents the number of genes and m represents the number of conditions. This binary string is composed of two strings; the first n bits represent the genes, and the remaining m bits represent the conditions. The value of each bit represents the presence of a given gene and/or condition in a bicluster, with one signifying presence and zero indicating absence (Fig. 5.1).

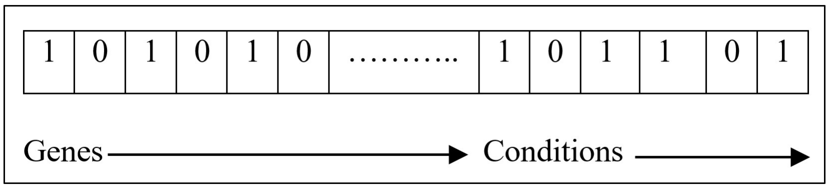

Figure 5.1: Representation of Binary String Encoding for Bicluster Solution.

5.4.2 Pre-processing: CDAE

In this study, the Convolutional Denoising Autoencoder (CDAE) was trained using datasets generated by specialized software designed for generating biclustering datasets Called *G-Bic* [1] [86]. Two datasets were generated: a noise-free dataset and a noisy dataset. The noisy dataset was obtained by introducing a noise parameter to the generation process, thereby incorporating varying levels of noise into the data. This approach aimed to train the CDAE to effectively identify and eliminate noise when used on other datasets in the experimental phase; notably, the Yeast Cell-Cycle and the Saccharomyces Cerevisiae real-life datasets were among the datasets used for evaluation. By evaluating the performance

[1]https://github.com/jplobo1313/G-Bic

of the CDAE on these datasets, we were able to assess its ability to denoise gene expression data and capture meaningful patterns across different noise conditions.

As shown in Fig. 5.2, the proposed CDAE, like all Autoencoders, is made up of two major components: an encoder and a decoder with several different layer types each.

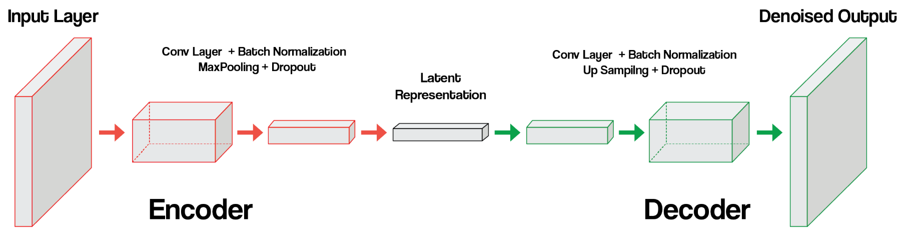

Figure 5.2: Architecture of the Proposed CDAE.

In the encoding section, the gene expression matrix is fed into the first convolutional layer (input layer). This layer is made up of 32 filters of size 3×3 that slide over the input, producing convolutions to capture local patterns while employing ReLU (Rectified Linear Unit) as the activation function. Then, a batch normalization layer is used to improve training speed, reduce overfitting, and assist the model in learning generalizable representations. Following that, the max-pooling layer with a 2×2 size filter is in charge of reducing the spatial dimensions of the input feature maps, resulting in the extraction of the most important patterns while eliminating noise. Finally, a dropout technique with a 0.25 rate is incorporated, which basically drops out 25% of the neurons during training to force the model to rely only on the remaining neurons to learn relevant patterns, reducing overfitting and making the training more efficient. The following convolutional layer is fairly identical to the previous one, with the exception that it has 64 filters that are 3×3 in size.

A feature map that depicts the compressed data is obtained through the encoding procedure. **The decoder part**, which is inversely symmetric to the encoder, the first deconvolutional layer consists of 64 3×3-sized filters, and the second one has 32 3×3-sized filters, both using the ReLU activation function. Following this, the deconvolutional layers work to recover spatial information, upsample the feature maps, and enhance the quality of the denoised gene expression reconstruction. The output layer consists of a deconvolutional layer with a single 3×3 filter. This layer uses a sigmoid activation function to produce the final output.

After the gene expression data has been successfully denoised by the Convolutional Denoising Autoencoder (CDAE) and the pertinent patterns have been captured, the Biclustering algorithm is next implemented using the Artificial Bee Colony (ABC) optimization method.

5.4.3 Biclustering: ABC

The ABC algorithm, as previously explained in Section 5.3, makes use of three categories of bees: employed, onlooker, and scout bees. Scout bees begin by randomly generating the initial population of PS food sources (solutions) in the search space. PS represents the population size, and each solution x_i (where $i = 1, 2, ..., PS$) is represented as a D-dimensional vector. The dimensionality, denoted by D, represents the number of optimization parameters. Each food source gets assigned to an employed bee.

After the initialization phase, the algorithm enters an iterative procedure, where each employed bee generates a new solution by selecting a solution x_i from the population P and then applying modifications to it. As previously discussed, the formula used for generating the new solution is applicable only to real numbers and produces solutions that are also real numbers. To address this issue, we modified the ABC algorithm by altering this formula so that it may be implemented directly on biclusters with binary representations. This modification is inspired by Cheng and Church's heuristic of node deletion and addition to generate the new solutions; in addition to changing the formula, we made a few adjustments to promote more diversity in the solutions; as a result, the CDABC generates three solutions rather than just one as in the original ABC. This procedure goes as follows:

First, for every randomly selected solution x_i, we select three different solutions a, b, and c; then we derive three neighboring solutions from x_i denoted as X_1, X_2, and X_3 by altering one bit in the gene sequence, the condition sequence, and both sequences, respectively, and utilize them to produce three new solutions for each x_i. this can be achieved using the following formula:

$$h_{i1} = nodeAddition(a, nodeDeletion(X_1, x_i, F), F) \tag{5.9}$$

$$h_{i2} = nodeAddition(b, nodeDeletion(X_2, x_i, F), F) \tag{5.10}$$

$$h_{i3} = nodeAddition(c, nodeDeletion(X_3, x_i, F), F) \tag{5.11}$$

The node deletion and addition operations are defined as the following:

$$nodeDeletion(x, y, F) = \begin{cases} 0, & x_j = 0 \;\; or \;\; (y_j = 1 \;\; and \;\; rand_j < F) \\ 1, & \text{otherwise} \end{cases} \tag{5.12}$$

$$nodeAddition(x, y, F) = \begin{cases} 1, & x_j = 1 \;\; or \;\; (y_j = 1 \;\; and \;\; rand_j < F) \\ 0, & \text{otherwise} \end{cases} \tag{5.13}$$

Where the factor $F = 0.65$ and $rand_j \in \{0, ..., 1\}$ (where j indicates the position of the current bit in the current solution) is a generated random number used together with the factor F to act as a form of node operation acceptance probability in the following manner: the node deletion/ addition will be performed only if the generated number $rand_j$ is less than F; otherwise, the node operation will be discarded.

After generating three new solutions from the same selected solution x_i, the bees then assess the fitness value of the freshly generated solutions using an objective function; if the fitness value of one of the new solutions is greater than the fitness value of the prior one, the bee memorizes the new solution and replaces the old one in the population. If not, the same solution is retained.

In the following phase, the employed bees share the nectar information (fitness value) of the food sources (solutions) with the onlooker bees, where they assess the nectar information obtained from the employed bees and select two different food sources x_i and a based on a probability $Prob_i$ that is proportional to its nectar amount. The probability is calculated using the following formulas:

$$Prob_i = \frac{f(x_i)}{\sum_{n=1}^{PS}(f(x_n))} \tag{5.14}$$

With PS being the size of the population and $f(x_i)$ being the fitness value of the solution x_i (the nectar information of the food source in position i). Then, a modification process takes place on the selected food source x_i (solution) where only one new solution h_i will be generated using Eq. 5.9, then compare $f(x_i)$ and $f(h_i)$; the solution that has the greater fitness value will be retained in the new population. Following the completion of these steps by all onlooker bees, the algorithm then conducts the scout bee phase, where it determines whether a particular solution hasn't been improved for a predetermined number of iterations called the *abandonment threshold*. If a solution exceeds this limit, it gets abandoned, and the employed bee that was assigned to this solution becomes a scout bee; then, scout bees explore the search space and discover completely new solutions to replace the abandoned and less promising ones with.

The ABC then proceeds through the iterations, starting from the employed bee phase once more to the scout bees phase, generating and exploring new solutions while retaining the best ones until a termination condition is met, such as reaching a maximum number of iterations or attaining a desired level of solution quality. The steps of CDABC are described in Algorithm 6:

Algorithm 6: CDABC Algorithm

```
Input  : A Matrix of size $n \times m$, Population Size $PS$, Factor $F$, objective
         Fitness function $f$
Output: $p$ Bicluster solutions
pop ← Initialize the initial population;
calculate the fitness scores using $f$ for all solutions in pop;
$besTSol$ ← set best solution ;
$iteNum$ ← Set maximum number of iterations;
$PS$ ← Set population size ; $Threshold$ ← set threshold;
/* $PS = onlookerBee = employedBee$ */
iteration ← 0;
while iteration < iteNum do
    /* Employed Bee Phase */
    for $i \leftarrow 1$ to employedBee do
        /* Select a random solution $x_i$ */
        a,b,c ← select 3 random solutions from pop in different positions than $i$;
        $X_1 \leftarrow x_i$ with one bit altered in the gene sequence
        $X_2 \leftarrow x_i$ with one bit altered in the condition sequence
        $X_3 \leftarrow x_i$ with one bit altered in both gene and condition sequences
        /* Generate three new solutions */
            $h_{i1} = nodeAddition(a, nodeDeletion(X_1, x_i, F), F)$
            $h_{i2} = nodeAddition(b, nodeDeletion(X_2, x_i, F), F)$
            $h_{i3} = nodeAddition(c, nodeDeletion(X_3, x_i, F), F)$
        /* Calculate the fitness of the new solutions */
        newSolFitness ← calculate the fitness of the three new solutions and
         keep the best one;
        if newSolFitness is better than $f(x_i)$ then
            /* The bee memorizes the new solution with the best fitness
               value */
            Replace $x_i$ with the new solution in pop;
        end
    end
    /* Onlooker Bee Phase */
    for $i \leftarrow 1$ to onlookerBee do
        /* Select two new solutions $x_i$ and $a$ based on probability
           $Prob_i$ */
            $Prob_i = \frac{f(x_i)}{\sum_{n=1}^{PS}(f(x_n))}$
        $X \leftarrow x_i$ with one bit altered ;
        /* Generate a new solution */
            $h_i = nodeAddition(a, nodeDeletion(X, x_i, F), F)$
        Calculate the fitness of the new solution $f(h_i)$;
        if $f(h_i)$ is better than $f(x_i)$ then
            /* The bee memorizes the new solution with the best fitness
               value */
            Replace $x_i$ with $h_i$ in pop;
        end
    end
    /* Scout Bee Phase */
    for $i \leftarrow 1$ to $PS$ do
        if Improvement iterations of $x_i$ exceeds $Threshold$ then
            abandon solution $x_i$ ;
            employed bee of $x_i$ becomes scout bee;
            generate a new solution to replace $x_i$ in pop;
        end
        iteration++;
    end
end
return $p$ Biclusters pop from last generation;
```

5.4.4 Illustrative Example

The following is an example of how the employed bees phase work with consideration of the implemented changes in CDABC.
Considering the matrix M(5,5), where the rows represent genes and the columns represent conditions:

$$M = \begin{pmatrix} 68 & 13 & 5 & 88 & 32 \\ 10 & 16 & 52 & 38 & 11 \\ 14 & 83 & 69 & 67 & 77 \\ 8 & 87 & 14 & 4 & 9 \\ 81 & 49 & 2 & 71 & 94 \end{pmatrix}$$

The algorithm randomly generates the first population P, and then selects 4 random solutions denoted as x_i , a, b, and c, these solutions are biclusters that are represented in binary as shown in Figure 5.1 within the section 3.1. The size of a bicluster depends on the number of genes and conditions in the matrix M; thus, each bicluster is represented by a 10-bit string.

Consider the following selected solutions:

$$\begin{aligned} x_i &= \begin{pmatrix} 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix} \\ a &= \begin{pmatrix} 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \end{pmatrix} \\ b &= \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{pmatrix} \\ c &= \begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \end{pmatrix} \end{aligned}$$

The genes and conditions included in the selected bicluster x_i of the P population are as follows:

$$x_i = \begin{pmatrix} 16 & 38 & 11 \\ 83 & 67 & 77 \\ 87 & 4 & 9 \end{pmatrix}$$

The next step entails creating three neighboring solutions to x_i, denoted as X_1, X_2, and X_3. This is achieved by taking the original solution x_i and making slight adjustments. Specifically, one bit is altered in the gene part, another in the condition part, and a third in both the gene and condition parts, for each of the three solutions. The resulting solutions are as follows:

$$x_i = \begin{pmatrix} 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix}$$
$$X_1 = \begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 \end{pmatrix}$$
$$X_2 = \begin{pmatrix} 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \end{pmatrix}$$
$$X_3 = \begin{pmatrix} 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \end{pmatrix}$$

Afterward, we create completely three new solutions by employing node addition and deletion operations on all the previous solutions using equations 5.9, 5.10 and 5.10:

$$h_{i1} = nodeAddition(a, nodeDeletion(X_1, x_i, F), F)$$

$$h_{i2} = nodeAddition(b, nodeDeletion(X_2, x_i, F), F)$$

$$h_{i3} = nodeAddition(c, nodeDeletion(X_3, x_i, F), F)$$

Initially, we execute the node deletion operation in the following manner:

$$nodeDeletion(X_1, x_i, F) = \begin{array}{cccccccccc} 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ \hline 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \end{array}$$

In the process of performing the mentioned node deletion operation, a random factor is generated Specifically, we generate a random number $rand_i \in 0, ..., 1$, where i represents the position of the current bit in the individual. For example, if $rand_9$ is less than F, the algorithm carries out a node deletion, resulting in 0. On the other hand, if $rand_1 0$ is greater than F, the algorithm skips the node deletion, resulting in 1. This process is repeated for the remaining bits and for the node addition operation as well.

$$nodeAddition(a, nodeDeletion(X_1, x_i, F), F) = \begin{array}{cccccccccc} 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ \hline 1 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \end{array}$$

The result of the solution h_{i1} is interpreted as representing a bicluster with four row and two columns, it can be denoted as follows:

$$h_{i1} = \begin{pmatrix} 5 & 32 \\ 52 & 11 \\ 69 & 77 \\ 2 & 94 \end{pmatrix}$$

The same process is replicated for the other two solutions, h_{i2} and h_{i3}.

5.5 Experimental Results

We conducted the experiments using both synthetic data and real data in order to examine and evaluate the performance of our novel algorithm CDABC.

The experiments covered in this section for both synthetic and real data were conducted using the parameter settings presented in Table 1. The experiments were performed on a Windows 10 PC with an AMD Ryzen 7 3800XT 8-Core Processor running at 4.30 GHz and equipped with 64 GB of RAM.

5.5.1 Synthetic Data Results

The synthetic data allow us to explore the CDABC's capacity to identify embedded biclusters while also rigorously investigating other factors, such as noise and overlap. The synthetic data used for this test was proposed by Prelic [5], and has been previously used to study the noise effect on biclustering. The datasets include ten implants of various bicluster types, including constant and additive biclusters, as well as ten genes and five conditions.

To assess the efficacy of our algorithm on synthetic data, we will employ Prelic's [5] gene match score, and compare the results with those obtained using state-of-the-art biclustering algorithms, including CC [2], OPSM [4], ISA [38], BiMax [5], Xmotif [39], and AMoDeBic [87]. The score represents the average of the highest match scores between the biclusters extracted by the biclustering algorithm in set M and the implanted biclusters in set M_opt. This metric is defined as follows:

$$S^*_G(M, M_{opt}) = \frac{1}{|M|} \sum_{(G_1,C_1)\in M} \max_{(G_2,C_2)\in M_{opt}} \frac{|G_1 \cap G_2|}{|G_1 \cup G_2|} \tag{5.15}$$

Fig. 5.3.A depicts the performance of several biclustering algorithms in extracting non-overlapping biclusters under differing degrees of noise, with a focus on scenarios involving constant biclusters. Similarly, Fig. 5.3.B shows how different biclustering methods perform in cases with additive biclusters and varying noise levels.

In Fig. 5.4.A, the emphasis changes to instances with bicluster overlap with consistent patterns. The results of the biclustering methods with increasing degrees of overlap and no noise are shown. Fig. 5.4.B also shows the results of these algorithms in scenarios with additive biclusters and increasing levels of overlap in the absence of noise.

Our analysis shows that the ISA algorithm detects 100% of the implanted biclusters in the circumstances with no noise and non-overlapping biclusters (Fig. 5.3). Furthermore, in settings with additive biclusters, both ISA and BiMax can recognize virtually all of the implanted biclusters. CDABC, on the other hand, achieves recognition rates of more than 90% for both constant and additive implanted biclusters.
Remarkably, when significant noise levels are present, ISA maintains its high detection rate for both types of biclusters. In contrast, when the noise level increases, BiMax's ability to identify implanted additive biclusters decreases. CDABC, on the other hand, has consistent performance in extracting more than 90% of both constant and additive biclusters, even in the presence of high noise levels. Notably, CDABC clearly surpasses other biclustering algorithms in terms of performance, indicating its great performance.

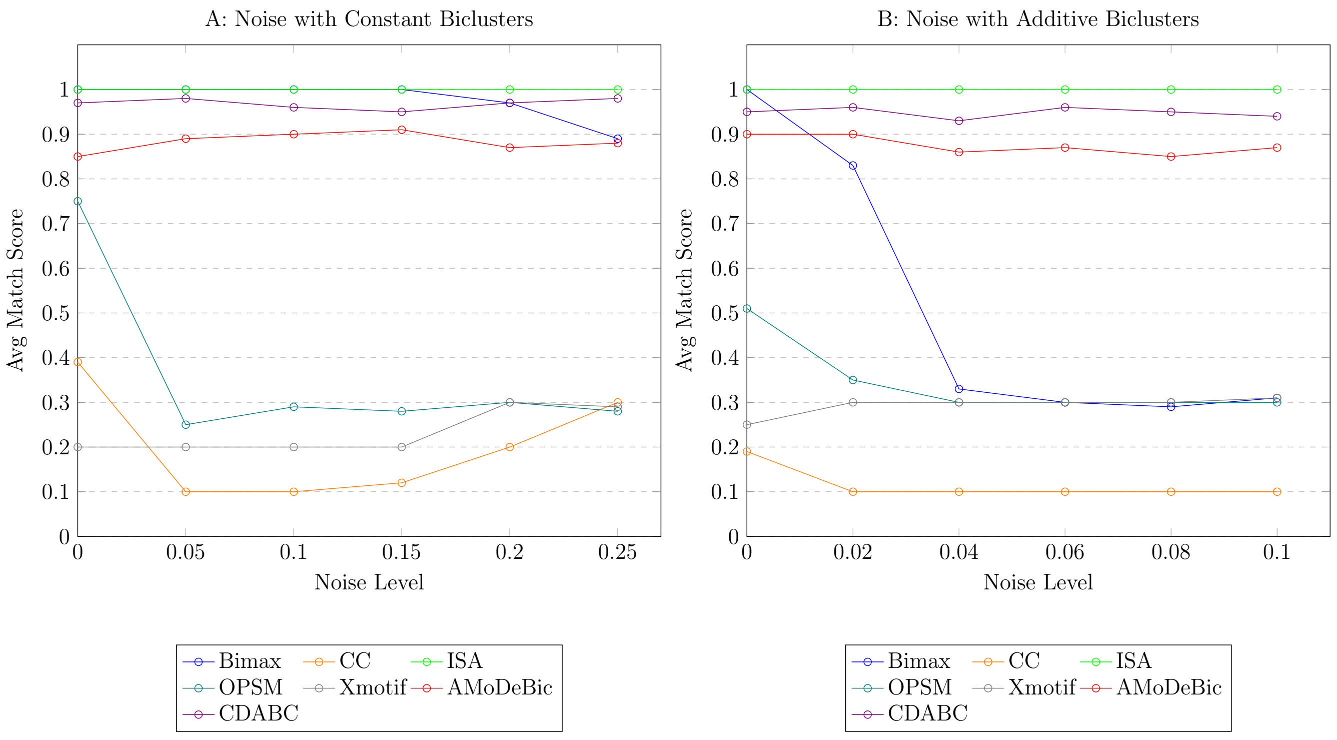

Figure 5.3: Noise Effect. A: Constant Biclusters, B: Additive Biclusters.

In terms of overlap impact (Fig. 5.4), it is clear that the performance of most biclustering approaches degrades as the level of overlap grows. Notably, BiMax exhibits consistent results for both constant and additive biclusters. ISA functions well at first, but as the degree of overlap increases, its performance degrades substantially. AMoDeBic achieves a recognition rate of 90% for constant implanted biclusters and 80% for additive biclusters, but its performance decreases substantially.
CDABC, on the other hand, distinguishes itself by detecting almost 90% of constant biclusters and 80% of additive biclusters. While its performance drops when overlap increases, the decrease is not as significant when compared to other techniques. CDABC surpasses the other approaches significantly, demonstrating its effectiveness and robustness in dealing with overlapping biclusters.

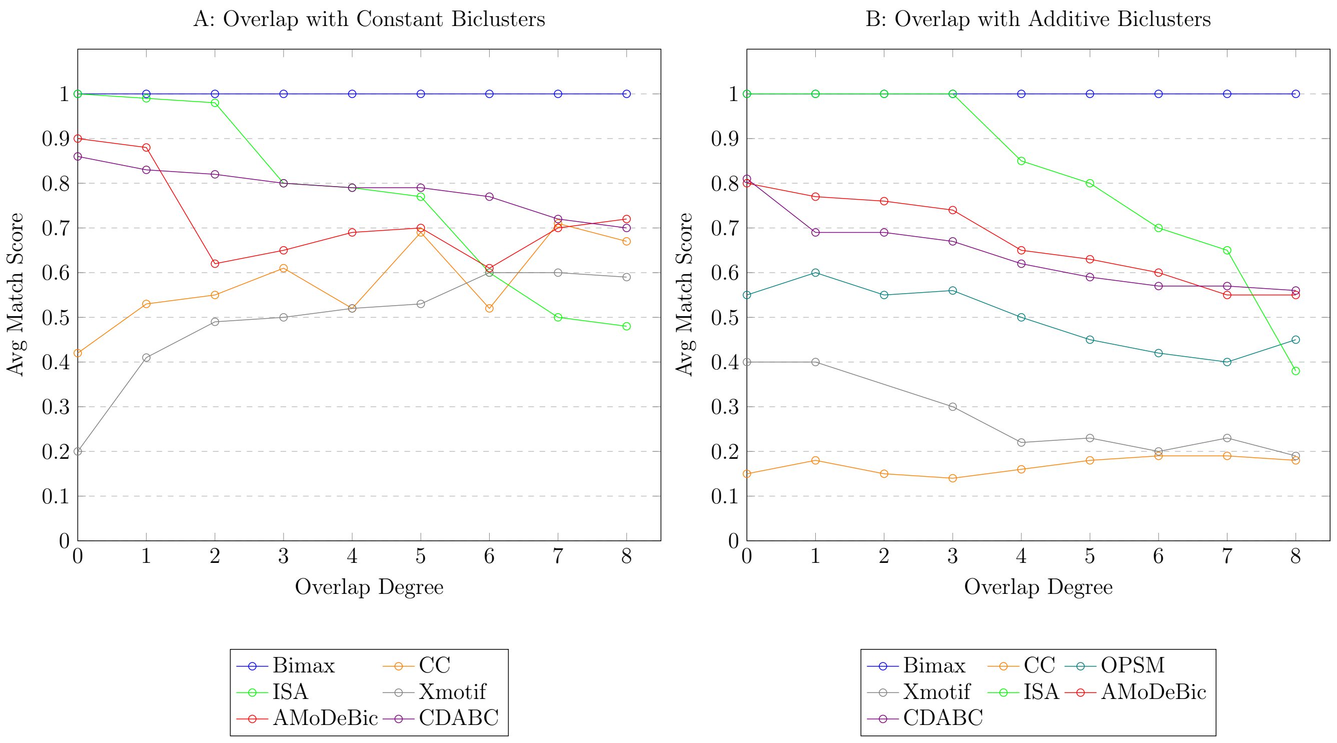

Figure 5.4: Overlap Effect. A: Constant Biclusters, B: Additive Biclusters.

5.5.2 Real Data Results

In addition to synthetic data, we conducted the experiments on three well-known real-world datasets: the Yeast Cell-Cycle Expression Data Set, the Human B-Cell Lymphoma Dataset, and the Saccharomyces Cerevisiae Dataset, where the results were compared to those of CC [2], BicFinder [23], BiMine+ [25], SEBI [8], MOEA [7], MOPSOB [68], ISA [38], BiMax [5], MBA [41], and AMoDeBic [87]. The following is a description of the considered datasets:

1. *The Yeast Cell-Cycle:* This is a particularly prominent dataset in gene expression data [2]. This dataset includes 2,884 gene expression profiles and 17 experimental conditions where each gene's function is well understood.

2. *Saccharomyces Cerevisiae:* This dataset covers the expression levels of 2993 genes under 173 different experimental settings.

3. *Human B-Cell Lymphoma:* This is another well-known dataset for gene expression data. It consists of 4026 genes and 96 conditions [77].

5.5.2.1 Statistical Relevance

The statistical significance of the biclusters discovered by our algorithm CDABC is critical for its validation. This evaluation is based on the coverage and p-value criteria.

1. Coverage

The coverage parameter specifies how many cells in the gene expression array are covered by the biclusters. It is critical for validating biclustering algorithms since it demonstrates the algorithms' capacity to capture and detect relevant patterns in data. This evaluation was performed on the datasets Yeast Cell-Cycle and Human B-Cell Lymphoma, and the outcomes are compared to those of the following algorithms: CC [2], BicFinder [23], BiMine [24], BiMine+ [25], SEBI [8], MOEA [7], MOPSOB [68], Trimax [88] and AMoDeBic [87].

The coverage results of the extracted biclusters are shown in Tables 5.1 and 5.2; the algorithms' results are quite comparable. CDABC covers 100% of the conditions in both datasets. Regarding gene coverage, it covers84.78% in the Yeast Cell-Cycle dataset and 80.96% in the Human B-Cell Lymphoma dataset. In terms of cell coverage, it covers 70.29% in the Yeast Cell-Cycle dataset and 44.94% in the Human B-Cell Lymphoma dataset.

On the Yeast Cell-Cycle dataset, the CC algorithm exhibits the most advantageous results. It is important to note that the CC method uses a masking technique that prevents the selection of previously identified genes or circumstances during subsequent search procedures by excluding groups retrieved with random values. This masking strategy greatly aids in obtaining thorough coverage. With the exception of the CC algorithm, CDABC has the highest gene coverage, condition coverage, and total coverage of any algorithm.

Yeast Cell-Cycle			
Algorithms	**Total coverage**	**Gene coverage**	**Condition coverage**
BiMine	13.36%	32.84%	100%
BiMine+	51.76%	68.65%	100%
BicFinder	55.43%	76.93%	100%
MOPSOB	52.40%	-	-
MOEA	51.34%	-	-
SEBI	38.14%	43.55%	100%
CC	**81.47%**	**97.12%**	100%
Trimax	15.32%	22.09%	70.59%
AMoDeBic	67.28%	79.94%	100%
CDABC	**70.29%**	**84.78%**	**100%**

Table 5.1: Comparison of Yeast Cell-Cycle Coverage by Different Algorithms.

On the Human B-Cell Lymphoma dataset, CDABC surpasses BiMine, Bimine+, BicFinder, MOPSOB, MOEA, and SEBI in terms of coverage outcomes in both total coverage and gene coverage. However, the CC algorithm surpasses CDABC for gene coverage in particular. On the other hand, CDABC performs better than the CC algorithm in terms of total coverage. AModebic also has higher outcomes in terms of total coverage, whereas CDABC outperforms in terms of gene coverage.

Human B-Cell Lymphoma			
Algorithms	**Total coverage**	**Gene coverage**	**Condition coverage**
BiMine	8.93%	26.15%	100%
BiMine+	21.19%	46.26%	100%
BicFinder	44.24%	55.89%	100%
MOPSOB	36.90%	-	-
MOEA	20.96%	-	-
SEBI	34.07%	38.23%	100%
CC	36.81%	**91.58%**	100%
Trimax	8.50%	46.32%	11.46%
AMoDeBic	**52.73%**	75.31%	100%
CDABC	**44.94%**	**80.96%**	**100%**

Table 5.2: Comparison of Human B-Cell Lymphoma Coverage by Different Algorithms.

2. P-value

In this test, we determine the adjusted significance scores for each bicluster using the online program FuncAssociate [2] [78]. The biclusters with an adjusted p-value of less than

[2]http://llama.mshri.on.ca/funcassociate/

0.001% are considered to be of the highest quality. This assessment was performed on the datasets Yeast Cell-Cycle and the Saccharomyces Cerevisiae datasets, and the results are compared with CC [2], ISA [38], BiMax [5], MBA [41], Trimax [88] and AMoDeBic [87].

The findings of the Yeast Cell-Cycle and Saccharomyces Cerevisiae datasets with various adjusted p-values (p = 5%; 1%; 0.5%; 0.1%; 0.001%) for each algorithm over the percentage of total biclusters are shown in Fig. 5.5 and Fig. 5.6.

For the Yeast Cell-Cycle dataset, the algorithms BiMax, Trimax, MBA, AMoDeBic, and CDABC all were able to achieve 100% of the extracted biclusters when the p-value was 5% and 1%. In addition, AMoDeBic obtained 100% when p-value = 0.5%, and MBA achieved 100% with a p-value greater than 0.001%. CDABC, on the other hand, was able to achieve 99% when p-value = 0.5% and 96% with p-value = 0.001%, which proves that CDABC performed better than MBA and AMoDeBic, both of which achieved under 95% of the biclusters.

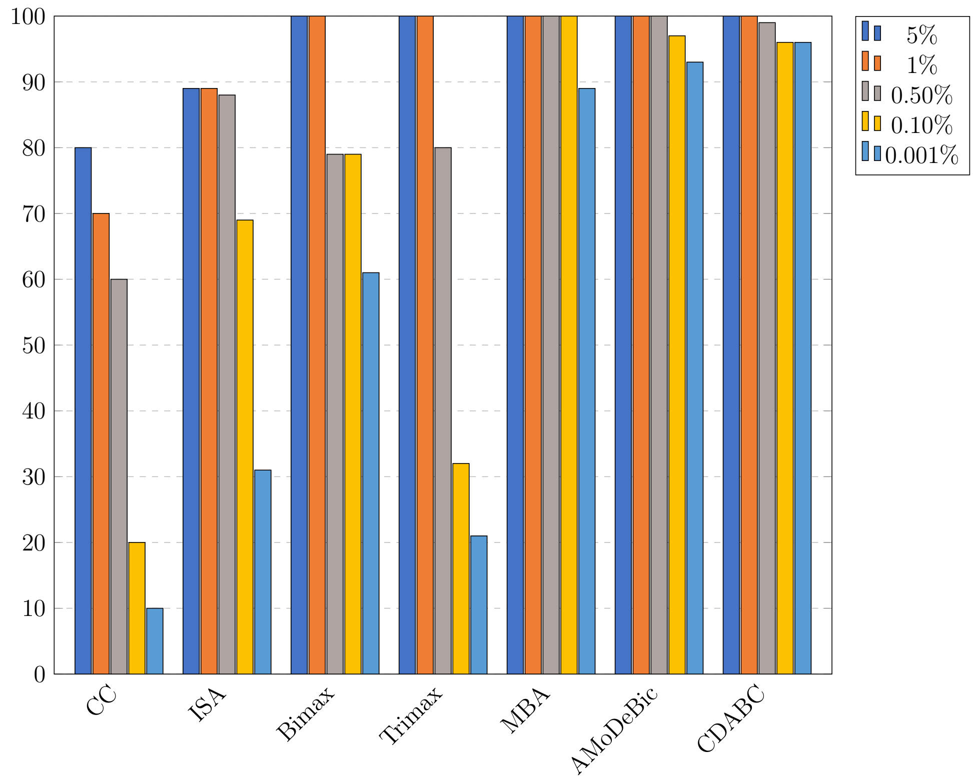

Figure 5.5: Proportions of Biclusters Significantly Enriched by GO Annotations (Yeast cell-cycle Dataset)

For the Saccharomyces Cerevisiae Dataset, it is remarkable that Trimax has the best outcomes for all adjusted P-values. When the p-value is 5%, 1%, 0.5%, and 0.1%, MBA and AMoDeBic were able to extract 100% of the biclusters, and when the p-value is 0.001%, they were able to extract approximately 98%. CDABC, however, displays notable performance in extracting 100% when the p-value is 5%, 1%, and around 99% for 0.5%, 0.1%, and 0.001%.

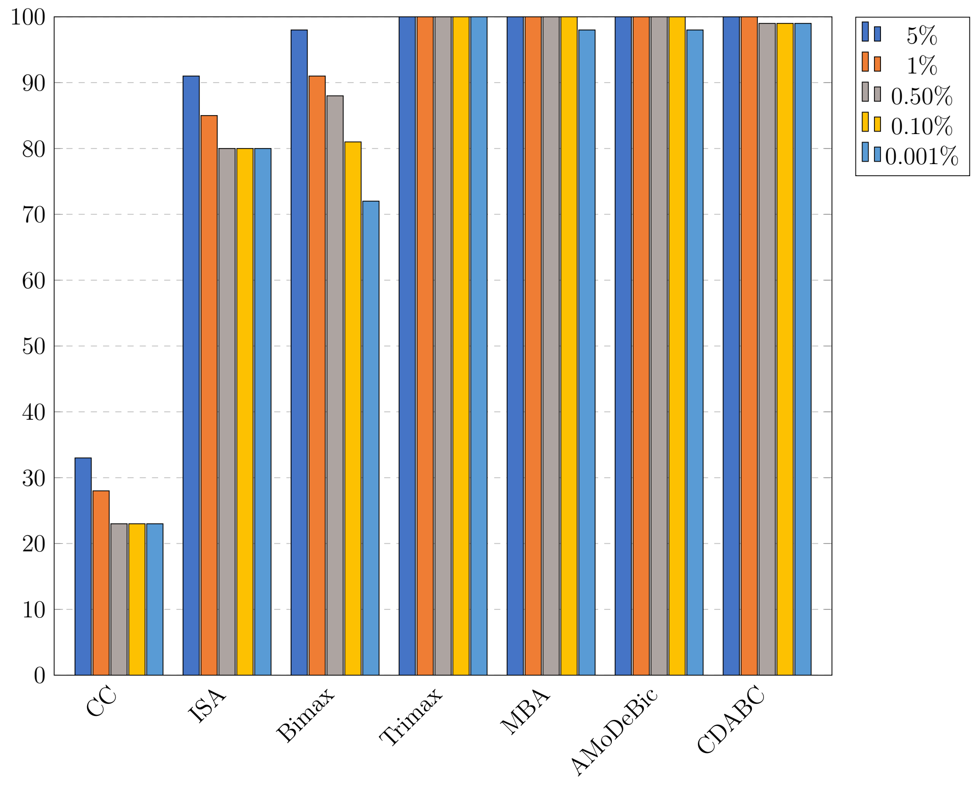

Figure 5.6: Proportions of Biclusters Significantly Enriched by GO Annotations (Saccharomyces Cerevisiae Dataset)

5.5.2.2 Biological Relevance

Another crucial factor in evaluating a biclustering algorithm is the quality of the retrieved biclusters, and this could be assessed using a biological criterion that checks whether the genes of a bicluster share biological traits. This evaluation is carried out using *GOTermFinder* [3], a well-known web application that discovers important shared Gene Ontology (GO) terms among specified sets of genes. The GO is divided into three categories: biological process, molecular function, and cellular component. In this test, we used The Yeast Cell-Cycle and Saccharomyces Cerevisiae datasets.

[3]https://www.yeastgenome.org/goTermFinder

The biological annotations of two randomly selected biclusters by CDABC in the Yeast Cell-Cycle and Saccharomyces Cerevisiae datasets, respectively, with regard to the previously described GO categories are presented in tables 5.3 and 5.4. The tables highlight the most important GO terms related to these biclusters.

GO Terms	**Bicluster 1**	**Bicluster 2**
Biological Process	regulation of meiotic cell cycle(49.2%, 7.10e-55 cellular response to stimulus(57.4%,1.06e-10)	positive regulation of nitrogen compound metabolic process(39.3%, 1.49e-09) negative regulation of sexual sporulation resulting in formation of a cellular spore (4.9%, 0.00035)
Cellular component	nuclear lumen(41.7%,1.75e-05) membrane part(100.0%, 0.00023) organelle envelope(87.5%, 8.49e-06)	non-membrane-bounded organelle(90.0%, 0.00024) MCM core complex(50.0%,1.27e-08) condensed chromosome (16.4%,7.10e-06)
Molecular function	histone chaperone activity (8.3%, 3.34e-05) DNA binding (41.7%, 5.33e-06)	ATP-dependent activity (38.9%, 1.41e-08) helicase activity(27.8%, 4.33e-09)

Table 5.3: Significant GO terms of two biclusters extracted from Yeast Cell-Cycle dataset using CDABC.

GO Terms	**Bicluster 1**	**Bicluster 2**
Biological Process	positive regulation of cell cycle (66.7%, 3.71e-14) mitotic cell cycle phase transition(93.3%, 1.72e-19)	mitotic spindle assembly (66.7%, 3.82e-08) organelle fission (83.3%, 4.57e-05)
Cellular component	supramolecular complex (58.3%, 2.30e-06) condensed chromosome, centromeric region (33.3%, 0.00024)	catalytic complex(38.4%, 1.93e-08) microtubule organizing center (33.3%, 0.00021)
Molecular function	protein binding (58.3%, 0.00317) microtubule motor activity(16.7%, 0.00321)	RNA pseudouridylation guide activity (38.4%, 6.66e-58) cytoskeletal protein binding(50.0%, 4.18e-07)

Table 5.4: Significant GO terms of two biclusters extracted from Saccharomyces Cerevisiae dataset using CDABC.

Fig. 5.7 and Fig. 5.8 illustrate six biclusters found by the CDABC algorithm in the Yeast Cell-Cycle and Saccharomyces Cerevisiae datasets. These biclusters are distinguished by genes that exhibit comparable patterns of behavior under specified situations, as established by visual inspection.

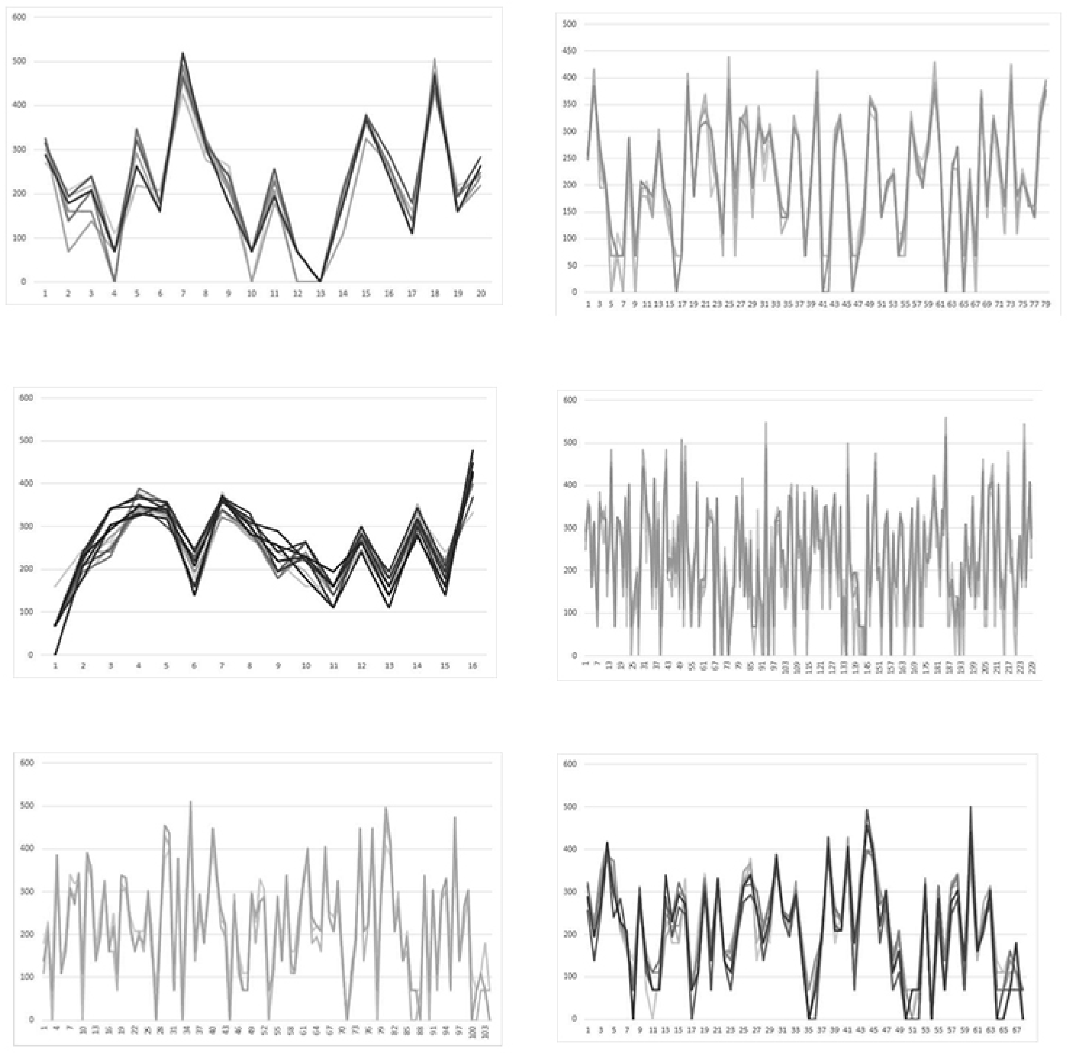

Figure 5.7: Six Biclusters Discovered in the Yeast Cell-Cycle Dataset by CDABC.

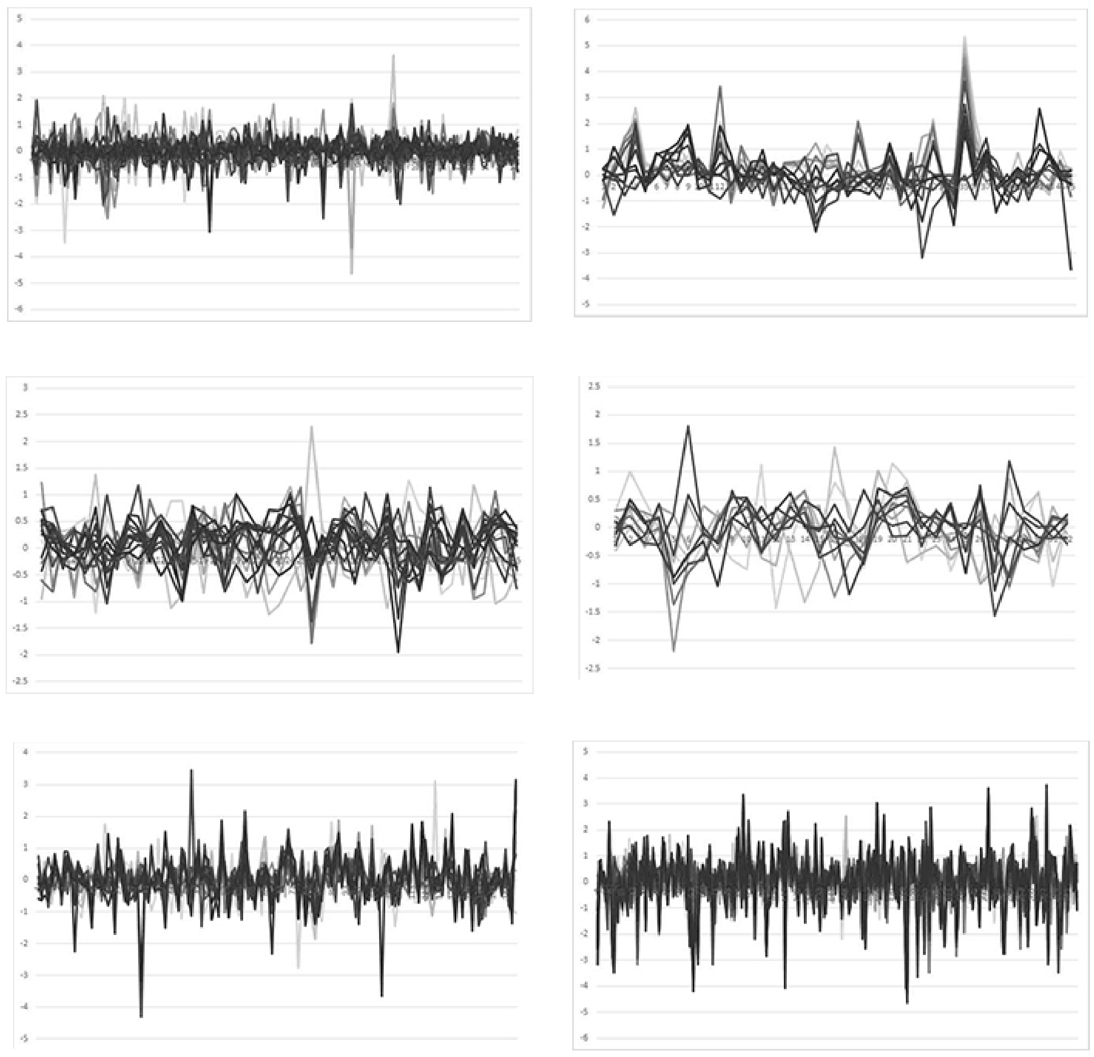

Figure 5.8: Six Biclusters Discovered in the Saccharomyces Cerevisiae Dataset by CDABC.

Considering all the findings in this section, our new method, CDABC, demonstrated impressive and noteworthy results on both synthetic and real data, showed robustness to noise and overlap, demonstrated significant coverage, and demonstrated the shared biological traits among biclusters.

5.6 Conclusion

This chapter actively explores a new approach that leverages the strengths inherent in two distinct methods: Convolutional Denoising Autoencoders (CDAs) and Artificial Bee Colony (ABC). Through an in-depth examination, we review and delve into the specifics of both CDAs and ABC, gaining a thorough understanding of their individual uses and strengths. Subsequently, we present the CDABC algorithm, providing a comprehensive explanation of the active modifications and adaptations implemented to the ABC algorithm. These changes are tailored to enhance its functionality, making it well-suited to effectively address the unique challenges posed by biclustering tasks. Lastly, the effectiveness of CDABC was validated through experimental results obtained from both synthetic and real data. The outcomes were systematically compared to established approaches, further affirming the algorithm's performance in the field of biclustering challenges.

Chapter 6

General Conclusion

Thesis Summary

The findings of this thesis provide various avenues for further research. This section outlines various promising routes of future study, including the following:

In the growing field of bioinformatics, where maintaining massive volumes of biological data is difficult, advanced analytical techniques become essential. Biclustering, an essential methodology in this vast landscape, is crucial for analyzing subtle patterns in gene expression data. This thesis, which falls under the broad category of biclustering techniques, seeks to present practical solutions targeted at directly tackling significant difficulties. The primary goal is to improve the analytical capabilities necessary for overcoming the difficulties inherent in large biological datasets. The goal is to enable a more nuanced and efficient analysis of gene expression patterns, therefore significantly contributing to the progress and evolution of the biclustering field.

The thesis is organized into six chapters. It begins with an overview of bioinformatics and the function of biclustering in the analysis of gene expression data. The second chapter includes a comprehensive evaluation of known biclustering algorithms, establishing the foundation for the presentation of three unique techniques in the following chapters—DeBiC, AMoDeBic, and CDABC. Each chapter is geared to address different challenges in biclustering approaches, adding to the field's overall comprehension. The last chapter summarizes the findings and emphasizes the possible influence of the suggested approaches on future research in biclustering.

Our contributions are distinguished by the introduction of innovative biclustering

methodologies, each carefully tailored to meet specific issues in the field. Our initial contribution to DeBiC strategically uses Differential Evolution's extensive search capabilities, global optimization skills, and efficacy in handling high-dimensional landscapes. Recognizing that DE's initial mutation operator was not specifically designed to handle binary values, which is critical for encoding biclusters in biclustering. DeBiC uses a separate mutation operator known as the semi-probability mutation. The primary objective of DeBiC is to efficiently discover several high-quality biclusters at the same time, setting the groundwork for future methodological advances in biclustering.
The experimental results demonstrate the efficacy of using a differential evolutionary algorithm inside a binary search space for global optimization. This approach proves helpful in collecting significant biclusters of interest with minimum residuals within a reasonable timeframe. This work represents an initial exploration into the field of differential evolution algorithms for biclustering, setting the foundation for further examination and development in this promising area of research.

The second important contribution in this thesis is AMoDeBic, which is a multi-objective biclustering approach based on DeBiC's fundamental concepts. Comparably, it incorporates the resilient elements of Differential Evolution into a multi-objective optimization framework. Notably, AMoDeBic is the first Multi-Objective Differential Evolution (MODE) technique for biclustering, representing a significant advancement in the field of research. This contribution additionally introduces an entirely novel mutation operator: the adaptive Biclustering Binary Differential Evolution (BBDE) mutation operator, which incorporates both node addition and deletion. The purpose of this mutation operator, which is controlled by a factor F, is to promote increased diversity in the solutions, therefore, it significantly enhances the algorithm's ability to simultaneously and effectively identify large and diverse biclusters and permits the extension of differential evolution's continuous search feature to binary search spaces, demonstrating an effective method for tackling the complexity issues associated with biclustering.

In validation experiments making use of the Yeast Cell-Cycle, Human Cell B, and Saccharomyces datasets, AMoDeBic showcased notable advantages over existing approaches. The discovered biclusters were exceptionally diverse, resistant to noise and overlap, and had significant biological significance. These findings were obtained within a reasonable timeframe, demonstrating the algorithm's efficiency and effectiveness. Our multi-objective differential evolution biclustering approach stands out as the first of its kind as it addresses conflicting objectives and enhances bicluster quality. The strategic combination of the BBDE mutation operator and multi-objective optimization principles establishes

AMoDeBic as a powerful method for improving our understanding of gene expression patterns, accelerating biological discoveries, and making significant contributions to the evolving field of biclustering approaches.

The third contribution of this thesis presents an original approach to biclustering by combining two powerful methods: the Convolutional Denoising Autoencoder (CDAE) as a preprocessing step for denoising gene expression data and identifying meaningful patterns, and the Artificial Bee Intelligence (ABC) algorithm, which has been altered to accommodate biclustering tasks. To evaluate the proposed approach, experiments were performed on the frequently used Yeast Cell-Cycle, Human Cell B, and Saccharomyces datasets. The results reveal various benefits over other biclustering approaches, including noise robustness and the capacity to successfully discover varying and high-quality biclusters.

Future Research

Future research will focus on the use of the suggested biclustering algorithms in various constantly evolving fields such as cybersecurity, healthcare, text mining, and others. DeBiC and AMoDeBic, as well as CDABC, while they were initially designed for gene expression research, have outstanding flexibility and pattern recognition skills. When we contemplate their incorporation into cybersecurity, the potential resides in their capacity to identify small trends across heterogeneous datasets, providing an original approach of virus identification. Beyond standard approaches, biclustering algorithms have the ability to handle emerging threats, identify polymorphic malware, and decrease false positives. While computing needs and real-time analysis must be addressed, the idea of strengthening cybersecurity defenses with biclustering algorithms is a compelling avenue for exploration, research, and innovation.

Another intriguing perspective is to eliminate the standard practice of needing upfront parameters for population and bicluster numbers in typical biclustering approaches. Instead, we strive to create an innovative model that emphasizes flexibility, removing the necessity for predefined numbers. The dynamic and changing characteristics of biological datasets make it difficult to predict these numbers ahead of time. The complexity and variety of biological datasets make it challenging to precisely predict the ideal number of populations and clusters at the start of the research. Employing a dynamic framework that seamlessly adapts to the intricacies of datasets not only offers an interesting area for continued research but also ensures a more intuitive and user-friendly experience and increased applicability to datasets of different complexity.

An additional noteworthy consideration for future work involves concentrating on enhancing the speed and accuracy of biclustering approaches. The goal is to address computational challenges, optimize algorithms, increase accuracy, and introduce innovative strategies to ensure efficient and precise analysis of biological data.

Publication List

International journals

Charfaoui, Y., Houari, A., & Boufera, F. (2024). AMoDeBic: An adaptive Multi-objective Differential Evolution biclustering algorithm of microarray data using a biclustering binary mutation operator. Expert Systems with Applications, 238, 121863.

International conferences

Charfaoui, Y., Houari, A., & Boufera, F. (2022, November). DeBic: A Differential Evolution Biclustering Algorithm for Microarray Data Analysis. In International Conference on Artificial Intelligence: Theories and Applications (pp. 288-302). Cham: Springer Nature Switzerland.

Bibliography

[1] John A Hartigan. Direct clustering of a data matrix. *Journal of the american statistical association*, 67(337):123–129, 1972.

[2] Yizong Cheng and George M Church. Biclustering of expression data. In *Ismb*, volume 8, pages 93–103, 2000.

[3] Jiong Yang, Haixun Wang, Wei Wang, and Philip S Yu. An improved biclustering method for analyzing gene expression profiles. *International Journal on Artificial Intelligence Tools*, 14(05):771–789, 2005.

[4] Amir Ben-Dor, Benny Chor, Richard Karp, and Zohar Yakhini. Discovering local structure in gene expression data: the order-preserving submatrix problem. *Journal of Computational Biology*, pages 49–57, 2002.

[5] Amela Prelić, Stefan Bleuler, Philip Zimmermann, Anja Wille, Peter Bühlmann, Wilhelm Gruissem, Lars Hennig, Lothar Thiele, and Eckart Zitzler. A systematic comparison and evaluation of biclustering methods for gene expression data. *Bioinformatics*, 22(9):1122–1129, 2006.

[6] Stefan Bleuler, Amela Prelic, and Eckart Zitzler. An ea framework for biclustering of gene expression data. In *Proceedings of the 2004 congress on evolutionary computation (IEEE Cat. No. 04TH8753)*, volume 1, pages 166–173. IEEE, 2004.

[7] Sushmita Mitra and Haider Banka. Multi-objective evolutionary biclustering of gene expression data. *Pattern Recognition*, 39(12):2464–2477, 2006.

[8] Federico Divina and Jesus S Aguilar-Ruiz. Biclustering of expression data with evolutionary computation. *IEEE transactions on knowledge and data engineering*, 18(5):590–602, 2006.

[9] Sara C Madeira and Arlindo L Oliveira. Biclustering algorithms for biological data analysis: a survey. *IEEE/ACM transactions on computational biology and bioinformatics*, 1(1):24–45, 2004.

[10] Ons Maâtouk, Wassim Ayadi, Hend Bouziri, and Béatrice Duval. Evolutionary local search algorithm for the biclustering of gene expression data based on biological knowledge. *Applied Soft Computing*, 104:107177, 2021.

[11] Juan A Nepomuceno, Alicia Troncoso, Isabel A Nepomuceno-Chamorro, and Jesús S Aguilar-Ruiz. Integrating biological knowledge based on functional annotations for biclustering of gene expression data. *Computer methods and programs in biomedicine*, 119(3):163–180, 2015.

[12] Amina Houari and Sadok Ben Yahia. Top-k formal concepts for identifying positively and negatively correlated biclusters. In *Model and Data Engineering: 10th International Conference, MEDI 2021, Tallinn, Estonia, June 21–23, 2021, Proceedings 10*, pages 156–172. Springer, 2021.

[13] Amina Houari, Wassim Ayadi, and Sadok Ben Yahia. A new fca-based method for identifying biclusters in gene expression data. *International Journal of Machine Learning and Cybernetics*, 9:1879–1893, 2018.

[14] Marta DM Noronha, Rui Henriques, Sara C Madeira, and Luis E Zárate. Impact of metrics on biclustering solution and quality: a review. *Pattern Recognition*, 127:108612, 2022.

[15] Kevin Y Yip, David W Cheung, and Michael K Ng. Harp: A practical projected clustering algorithm. *IEEE Transactions on knowledge and data engineering*, 16(11):1387–1397, 2004.

[16] Beatriz Pontes, Federico Divina, Raúl Giráldez, and Jesús S Aguilar-Ruiz. Virtual error: A new measure for evolutionary biclustering. In *Evolutionary Computation, Machine Learning and Data Mining in Bioinformatics: 5th European Conference, EvoBIO 2007, Valencia, Spain, April 11-13, 2007. Proceedings 5*, pages 217–226. Springer, 2007.

[17] Patryk Orzechowski, Moshe Sipper, Xiuzhen Huang, and Jason H Moore. Ebic: an artificial intelligence-based parallel biclustering algorithm for pattern discovery. *arXiv preprint arXiv:1801.03039*, 2018.

[18] Patryk Orzechowski and Krzysztof Boryczko. Text mining with hybrid biclustering algorithms. In *International Conference on Artificial Intelligence and Soft Computing*, pages 102–113. Springer, 2016.

[19] Anis R Amna and Agus Hermanto. Implementation of bcbimax algorithm to determine customer segmentation based on customer market and behavior. In *2017 4th International Conference on Computer Applications and Information Processing Technology (CAIPT)*, pages 1–5. IEEE, 2017.

[20] Ricardo Cachucho, Kaihua Liu, Siegfried Nijssen, and Arno Knobbe. Bipeline: a web-based visualization tool for biclustering of multivariate time series. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2016, Riva del Garda, Italy, September 19-23, 2016, Proceedings, Part III 16*, pages 12–16. Springer, 2016.

[21] Saziye Deniz Oguz Arikan and Cem Iyigun. A supervised biclustering optimization model for feature selection in biomedical dataset classification. In *Data Mining and Big Data: First International Conference, DMBD 2016, Bali, Indonesia, June 25-30, 2016. Proceedings 1*, pages 196–204. Springer, 2016.

[22] Amela Prelić, Stefan Bleuler, Philip Zimmermann, Anja Wille, Peter Bühlmann, Wilhelm Gruissem, Lars Hennig, Lothar Thiele, and Eckart Zitzler. A systematic comparison and evaluation of biclustering methods for gene expression data. *Bioinformatics*, 22(9):1122–1129, 2006.

[23] Wassim Ayadi, Mourad Elloumi, and Jin-Kao Hao. Bicfinder: a biclustering algorithm for microarray data analysis. *Knowledge and information systems*, 30:341–358, 2012.

[24] Wassim Ayadi, Mourad Elloumi, and Jin-Kao Hao. A biclustering algorithm based on a bicluster enumeration tree: application to dna microarray data. *BioData mining*, 2(1):1–16, 2009.

[25] Wassim Ayadi, Mourad Elloumi, and Jin Kao Hao. Bimine+: an efficient algorithm for discovering relevant biclusters of dna microarray data. *Knowledge-Based Systems*, 35:224–234, 2012.

[26] Akdes Serin and Martin Vingron. Debi: Discovering differentially expressed biclusters using a frequent itemset approach. *Algorithms for Molecular Biology*, 6(1):1–12, 2011.

[27] Wassim Ayadi, Mourad Elloumi, and Jin-Kao Hao. Pattern-driven neighborhood search for biclustering of microarray data. *BMC bioinformatics*, 13:1–11, 2012.

[28] Federico Divina and Jesus S Aguilar-Ruiz. A multi-objective approach to discover biclusters in microarray data. *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, pages 385–392, 2007.

[29] Qinghua Huang, Dacheng Tao, Xuelong Li, and Alan Liew. Parallelized evolutionary learning for detection of biclusters in gene expression data. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 9(2):560–570, 2011.

[30] Juan A Nepomuceno, Alicia Troncoso, Isabel A Nepomuceno-Chamorro, and Jesús S Aguilar-Ruiz. Pairwise gene go-based measures for biclustering of high-dimensional expression data. *BioData mining*, 11(1):1–19, 2018.

[31] Younes Charfaoui, Amina Houari, and Fatma Boufera. Debic: A differential evolution biclustering algorithm for microarray data analysis. In *International Conference on Artificial Intelligence: Theories and Applications*, pages 288–302. Springer, 2022.

[32] Cristian Andrés Gallo, Jessica Andrea Carballido, and Ignacio Ponzoni. Bihea: a hybrid evolutionary approach for microarray biclustering. In *Advances in Bioinformatics and Computational Biology: 4th Brazilian Symposium on Bioinformatics, BSB 2009, Porto Alegre, Brazil, July 29-31, 2009. Proceedings 4*, pages 36–47. Springer, 2009.

[33] Juan A Nepomuceno, Alicia Troncoso, and Jesús S Aguilar-Ruiz. Biclustering of gene expression data by correlation-based scatter search. *BioData mining*, 4:1–17, 2011.

[34] Adán José-García, Julie Jacques, Vincent Sobanski, and Clarisse Dhaenens. Metaheuristic biclustering algorithms: From state-of-the-art to future opportunities. *ACM Computing Surveys*, 56(3):1–38, 2023.

[35] R Rathipriya and K Thangavel. A discrete artificial bees colony inspired biclustering algorithm. *International Journal of Swarm Intelligence Research (IJSIR)*, 3(1):30–42, 2012.

[36] Junwan Liu, Zhoujun Li, Xiaohua Hu, and Yiming Chen. Biclustering of microarray data with mospo based on crowding distance. In *BMC bioinformatics*, volume 10, pages 1–10. BioMed Central, 2009.

[37] Junwan Liu, Zhoujun Li, Xiaohua Hu, and Yiming Chen. Moaco biclustering of gene expression data. *International Journal of Functional Informatics and Personalised Medicine*, 3(1):58–72, 2010.

[38] Jan Ihmels, Sven Bergmann, and Naama Barkai. Defining transcription modules using large-scale gene expression data. *Bioinformatics*, 20(13):1993–2003, 2004.

[39] TM Murali and Simon Kasif. Extracting conserved gene expression motifs from gene expression data. In *Biocomputing 2003*, pages 77–88. World Scientific, 2002.

[40] Amos Tanay, Roded Sharan, and Ron Shamir. Discovering statistically significant biclusters in gene expression data. *Bioinformatics*, 18(suppl_1):S136–S144, 2002.

[41] Wassim Ayadi and Jin-Kao Hao. A memetic algorithm for discovering negative correlation biclusters of dna microarray data. *Neurocomputing*, 145:14–22, 2014.

[42] Shreya Mishra and Swati Vipsita. Biclustering of gene expression microarray data using dynamic deme parallelized genetic algorithm (ddpga). In *2017 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB)*, pages 1–8. IEEE, 2017.

[43] Ons Maâtouk, Wassim Ayadi, Hend Bouziri, and Béatrice Duval. Evolutionary biclustering algorithms: an experimental study on microarray data. *Soft Computing*, 23:7671–7697, 2019.

[44] Anupam Chakraborty and Hitashyam Maka. Biclustering of gene expression data using genetic algorithm. In *2005 IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology*, pages 1–8. IEEE, 2005.

[45] Fabrício O de França, Guilherme P Coelho, and Fernando J Von Zuben. bicaco: An ant colony inspired biclustering algorithm. In *Ant Colony Optimization and Swarm Intelligence: 6th International Conference, ANTS 2008, Brussels, Belgium, September 22-24, 2008. Proceedings 6*, pages 401–402. Springer, 2008.

[46] S Sudhakar Ilango, S Vimal, Madasamy Kaliappan, and P Subbulakshmi. Optimization using artificial bee colony based clustering approach for big data. *Cluster Computing*, 22:12169–12177, 2019.

[47] Fuding Xie, Fangfei Li, Cunkuan Lei, Jun Yang, and Yong Zhang. Unsupervised band selection based on artificial bee colony algorithm for hyperspectral image classification. *Applied Soft Computing*, 75:428–440, 2019.

[48] Michael Ashburner, Catherine A Ball, Judith A Blake, David Botstein, Heather Butler, J Michael Cherry, Allan P Davis, Kara Dolinski, Selina S Dwight, Janan T Eppig, et al. Gene ontology: tool for the unification of biology. *Nature genetics*, 25(1):25–29, 2000.

[49] Audrey P Gasch, Paul T Spellman, Camilla M Kao, Orna Carmel-Harel, Michael B Eisen, Gisela Storz, David Botstein, and Patrick O Brown. Genomic expression programs in the response of yeast cells to environmental changes. *Molecular biology of the cell*, 11(12):4241–4257, 2000.

[50] Anja Wille, Philip Zimmermann, Eva Vranová, Andreas Fürholz, Oliver Laule, Stefan Bleuler, Lars Hennig, Amela Prelić, Peter von Rohr, Lothar Thiele, et al. Sparse graphical gaussian modeling of the isoprenoid gene network in arabidopsis thaliana. *Genome biology*, 5(11):1–13, 2004.

[51] Laura Macías-García, José María Luna-Romera, Jorge García-Gutiérrez, Maria Martinez-Ballesteros, José C Riquelme-Santos, and Ricardo González-Cámpora. A study of the suitability of autoencoders for preprocessing data in breast cancer experimentation. *Journal of biomedical informatics*, 72:33–44, 2017.

[52] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103, 2008.

[53] Rainer Storn and Kenneth Price. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, 11:341–359, 1997.

[54] Brian Hegerty, Chih-Cheng Hung, and Kristen Kasprak. A comparative study on differential evolution and genetic algorithms for some combinatorial problems. In *Proceedings of 8th Mexican international conference on artificial intelligence*, volume 9, page 13, 2009.

[55] Mahmud Iwan, Rini Akmeliawati, Tarig Faisal, and Hayder MAA Al-Assadi. Performance comparison of differential evolution and particle swarm optimization in constrained optimization. *Procedia Engineering*, 41:1323–1328, 2012.

[56] Manolis Georgioudakis and Vagelis Plevris. A comparative study of differential evolution variants in constrained structural optimization. *Frontiers in Built Environment*, 6:102, 2020.

[57] Zhenyu Yang, Ke Tang, and Xin Yao. Scalability of generalized adaptive differential evolution for large-scale continuous optimization. *Soft Computing*, 15:2141–2155, 2011.

[58] Changshou Deng, Bingyan Zhao, Yanlin Yang, and Hai Zhang. Binary encoding differential evolution for combinatorial optimization problems. *International Journal of Education and Management Engineering*, 1(3):59–66, 2011.

[59] Daniela Zaharie. Influence of crossover on the behavior of differential evolution algorithms. *Applied soft computing*, 9(3):1126–1138, 2009.

[60] M Fatih Tasgetiren, Yun-Chia Liang, Gunes Gencyilmaz, and Ipek Eker. A differential evolution algorithm for continuous function optimization. In *IEEE Congress on Evolutionary Computation*, volume 1, 2005.

[61] Aristotelis E Charalampakis and George C Tsiatas. Critical evaluation of metaheuristic algorithms for weight minimization of truss structures. *Frontiers in Built Environment*, 5:113, 2019.

[62] Jiong Yang, Haixun Wang, Wei Wang, and Philip Yu. Enhanced biclustering on expression data. In *Third IEEE Symposium on Bioinformatics and Bioengineering, 2003. Proceedings.*, pages 321–327. IEEE, 2003.

[63] Kalyanmoy Deb. *Multi-objective optimisation using evolutionary algorithms: an introduction*. Springer, 2011.

[64] David E Golberg. Genetic algorithms in search, optimization, and machine learning. *Addion wesley*, 1989(102):36, 1989.

[65] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation*, 6(2):182–197, 2002.

[66] Rakesh Angira and BV Babu. Non-dominated sorting differential evolution (nsde): An extension of differential evolution for multi-objective optimization. In *IICAI*, pages 1428–1443, 2005.

[67] Mohsen Lashkargir, S Amirhassan Monadjemi, and Ahmad Baraani Dastjerdi. A new biclustering method for gene expersion data based on adaptive multi objective particle swarm optimization. In *2009 Second International Conference on Computer and Electrical Engineering*, volume 1, pages 559–563. IEEE, 2009.

[68] Junwan Liu, Zhoujun Li, Feifei Liu, and Yiming Chen. Multi-objective particle swarm optimization biclustering of microarray data. In *2008 IEEE International Conference on Bioinformatics and Biomedicine*, pages 363–366. IEEE, 2008.

[69] Dervis Karaboga and Bahriye Basturk. Artificial bee colony (abc) optimization algorithm for solving constrained optimization problems. In *Foundations of Fuzzy Logic and Soft Computing: 12th International Fuzzy Systems Association World Congress, IFSA 2007, Cancun, Mexico, June 18-21, 2007. Proceedings 12*, pages 789–798. Springer, 2007.

[70] Ezgi Zorarpacı and Selma Ayşe Özel. A hybrid approach of differential evolution and artificial bee colony for feature selection. *Expert Systems with Applications*, 62:91–103, 2016.

[71] Gening Xu and Xingfeng Wang. Elite-guiding binary differential evolution. In *2015 6th International Conference on Manufacturing Science and Engineering*, pages 860–865. Atlantis Press, 2015.

[72] Tao Gong and Andrew L Tuson. Differential evolution for binary encoding. In *Soft Computing in Industrial Applications: Recent Trends*, pages 251–262. Springer, 2007.

[73] Andries P Engelbrecht and Gary Pampara. Binary differential evolution strategies. In *2007 IEEE congress on evolutionary computation*, pages 1942–1947. IEEE, 2007.

[74] Eckart Zitzler and Lothar Thiele. An evolutionary algorithm for multiobjective optimization: The strength pareto approach. *TIK report*, 43, 1998.

[75] Audrey P Gasch, Paul T Spellman, Camilla M Kao, Orna Carmel-Harel, Michael B Eisen, Gisela Storz, David Botstein, and Patrick O Brown. Genomic expression programs in the response of yeast cells to environmental changes. *Molecular biology of the cell*, 11(12):4241–4257, 2000.

[76] Audrey P Gasch, Paul T Spellman, Camilla M Kao, Orna Carmel-Harel, Michael B Eisen, Gisela Storz, David Botstein, and Patrick O Brown. Genomic expression programs in the response of yeast cells to environmental changes. *Molecular biology of the cell*, 11(12):4241–4257, 2000.

[77] Ash A Alizadeh, Michael B Eisen, R Eric Davis, Chi Ma, Izidore S Lossos, Andreas Rosenwald, Jennifer C Boldrick, Hajeer Sabet, Truc Tran, Xin Yu, et al. Distinct types of diffuse large b-cell lymphoma identified by gene expression profiling. *Nature*, 403(6769):503–511, 2000.

[78] Gabriel F Berriz, Oliver D King, Barbara Bryant, Chris Sander, and Frederick P Roth. Characterizing gene sets with funcassociate. *Bioinformatics*, 19(18):2502–2504, 2003.

[79] Hsin-Tien Chiang, Yi-Yen Hsieh, Szu-Wei Fu, Kuo-Hsuan Hung, Yu Tsao, and Shao-Yi Chien. Noise reduction in ecg signals using fully convolutional denoising autoencoders. *IEEE Access*, 7:60806–60813, 2019.

[80] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103, 2008.

[81] Jonathan Masci, Ueli Meier, Dan Cireşan, and Jürgen Schmidhuber. Stacked convolutional auto-encoders for hierarchical feature extraction. In *Artificial Neural Networks and Machine Learning–ICANN 2011: 21st International Conference on Artificial Neural Networks, Espoo, Finland, June 14-17, 2011, Proceedings, Part I 21*, pages 52–59. Springer, 2011.

[82] Dervis Karaboga and Bahriye Basturk. On the performance of artificial bee colony (abc) algorithm. *Applied soft computing*, 8(1):687–697, 2008.

[83] Mustafa Servet Kiran and MESUT Gündüz. Xor-based artificial bee colony algorithm for binary optimization. *Turkish Journal of Electrical Engineering and Computer Sciences*, 21(8):2307–2328, 2013.

[84] Rafet Durgut. Improved binary artificial bee colony algorithm. *Frontiers of Information Technology & Electronic Engineering*, 22(8):1080–1091, 2021.

[85] Mustafa Servet Kiran. A binary artificial bee colony algorithm and its performance assessment. *Expert Systems with Applications*, 175:114817, 2021.

[86] jplobo1313. G-Bic. https://github.com/jplobo1313/G-Bic, 2022.

[87] Younes Charfaoui, Amina Houari, and Fatma Boufera. Amodebic: An adaptive multi-objective differential evolution biclustering algorithm of microarray data using a biclustering binary mutation operator. *Expert Systems with Applications*, 238:121863, 2024.

[88] Mehdi Kaytoue, Sergei O Kuznetsov, Juraj Macko, and Amedeo Napoli. Biclustering meets triadic concept analysis. *Annals of Mathematics and Artificial Intelligence*, 70:55–79, 2014.

Appendix A

Biclustering Metric Calculation Examples

A.1 MSR: Calculation Example

MSR for a bicluster B that consists of I rows and J columns is defined as follows:

$$(2.1) MSR(B) = \frac{1}{|I|.|J|} \sum_{i=1}^{|I|} \sum_{j=1}^{|J|} (b_{ij} - b_{iJ} - b_{Ij} + b_{IJ})^2 \tag{A.1}$$

Where:

b_{ij}: represents the expression value of gene i under condition j within the bicluster.

b_{iJ}: represents the mean expression value of all genes across condition j in the bicluster. It is calculated as follows: $b_{iJ} = \frac{1}{|J|} \sum_{j \in J} b_{ij}$

b_{Ij}: represents the mean expression value of gene i across all conditions in the bicluster. It is calculated as follows: $b_{Ij} = \frac{1}{|I|} \sum_{j \in J} b_{ij}$

b_{IJ}: denotes the overall mean expression value of the bicluster. Calculated as follows: $b_{IJ} = \frac{1}{|I|.|J|} \sum_{i \in I, j \in J} b_{ij}$

Example: Consider the genes and conditions included in the first bicluster as follows:

$$B = \begin{pmatrix} 6 & 3 & 68 \\ 67 & 41 & 98 \end{pmatrix}$$

When $i = 0, j = 0$

$$b_{00} = 6$$

$$b_{IJ} = \frac{6+3+68+67+41+98}{6} = \frac{283}{6} \approx 47.167$$

$$b_{0J} = \frac{6+3+68}{3} = \frac{77}{3} \approx 25.667$$

$$b_{I0} = \frac{6+67}{2} = \frac{73}{2} \approx 36.500$$

Substitute these values into the MSR equation and keep iterating:

$$MSR(B) = \frac{1}{2 \times 3} \sum_{i=1}^{2} \sum_{j=1}^{3} (b_{ij} - b_{iJ} - b_{Ij} + b_{IJ})^2$$

After plugging in the values and performing the calculations:

$$MSR(B) \approx 43.167$$

A.2 Var: Calculation Example

The variance metric is formally defined as follows:

$$(2.2) Var(B) = \sum_{i=1}^{|I|} \sum_{j=1}^{|I|} (b_{ij} - b_{Ij})^2 \tag{A.2}$$

Where:
B: refers to the bicluster under consideration.
$|I|$ and $|J|$: the numbers of genes and conditions in the bicluster, respectively. b_{ij}: represents the expression value of gene i under condition j within the bicluster.
b_{Ij}: represents the mean expression value of gene i across all conditions in the bicluster.
The Variance metric calculates the squared difference between individual expression values b_{ij} and the mean expression value b_{Ij} of gene i across all conditions in the bicluster. These squared differences are accumulated by the summing terms for all elements inside the bicluster.

Example: Consider the genes and conditions included in the first bicluster as follows:

$$B = \begin{pmatrix} 6 & 3 & 68 \\ 67 & 41 & 98 \end{pmatrix}$$

When $i = 0, j = 0$

$$b_{00} = 6$$

$$b_{I0} = \frac{6+67}{2} = \frac{73}{2} \approx 36.500$$

Substitute these values into the Variance equation and keep iterating::

$$Var(B) = \sum_{i=1}^{2} \sum_{j=1}^{3} (b_{ij} - b_{Ij})^2$$

After plugging in the values and performing the calculations:

$$Var(B) \approx 4321.33$$