UNIVERSIDADE ESTADUAL DE CAMPINAS
Faculdade de Engenharia Elétrica e de Computação

Rosana Veroneze

Enumerating all maximal biclusters in numerical datasets

Enumerando todos os biclusters maximais em conjuntos de dados numéricos

Campinas

2016

UNIVERSIDADE ESTADUAL DE CAMPINAS
Faculdade de Engenharia Elétrica e de Computação

Rosana Veroneze

Enumerating all maximal biclusters in numerical datasets

Enumerando todos os biclusters maximais em conjuntos de dados numéricos

Thesis presented to the School of Electrical and Computer Engineering of the University of Campinas in partial fulfillment of the requirements for the degree of Doctor in Electrical Engineering, in the area of Computer Engineering.

Tese apresentada à Faculdade de Engenharia Elétrica e de Computação da Universidade Estadual de Campinas como parte dos requisitos exigidos para a obtenção do título de Doutora em Engenharia Elétrica, na Área de Engenharia de Computação.

Supervisor: Prof. Dr. Fernando J. Von Zuben

Este exemplar corresponde à versão final da tese defendida pela aluna Rosana Veroneze, e orientada pelo Prof. Dr. Fernando J. Von Zuben

Campinas

2016

Agência(s) de fomento e nº(s) de processo(s): CAPES

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca da Área de Engenharia e Arquitetura
Luciana Pietrosanto Milla - CRB 8/8129

V599e

Veroneze, Rosana, 1982-
Enumerating all maximal biclusters in numerical datasets / Rosana Veroneze. – Campinas, SP : [s.n.], 2016.

Orientador: Fernando José Von Zuben.
Tese (doutorado) – Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Conjunto de dados. 2. Estatística e dados numéricos. 3. Algoritmos. 4. Heurística. I. Von Zuben, Fernando José,1968-. II. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. III. Título.

Informações para Biblioteca Digital

Título em outro idioma: Enumerando todos os biclusters maximais em conjuntos de dados numéricos
Palavras-chave em inglês:
Data set
Statistics and numerical data
Algorithms
Heuristic
Área de concentração: Engenharia de Computação
Titulação: Doutora em Engenharia Elétrica
Banca examinadora:
Fernando José Von Zuben [Orientador]
Guilherme de Alencar Barreto
Fabricio Olivetti de França
Tiago Fernandes Tavares
Levy Boccato
Data de defesa: 20-06-2016
Programa de Pós-Graduação: Engenharia Elétrica

COMISSÃO JULGADORA - TESE DE DOUTORADO

Candidata: Rosana Veroneze **RA:** 075570

Data da Defesa: 20 de junho de 2016

Título da Tese: "Enumerating all maximal biclusters in numerical datasets (Enumerando todos os biclusters maximais em conjuntos de dados numéricos)"

Prof. Dr. Fernando José Von Zuben (Presidente, FEEC/UNICAMP)
Prof. Dr. Guilherme de Alencar Barreto (DETI/UFC)
Prof. Dr. Fabricio Olivetti de França (CMCC /UFABC)
Prof. Dr. Tiago Fernandes Tavares (FEEC/UNICAMP)
Prof. Dr. Levy Boccato (FEEC/UNICAMP)

A ata de defesa, com as respectivas assinaturas dos membros da Comissão Julgadora, encontra-se no processo de vida acadêmica do aluno.

Dedico esta tese a todos os seres.

Acknowledgements

"...Nossos talentos e habilidades, tudo provém da bondade dos outros; tivemos que ser ensinados a comer, andar, falar e escrever. Mesmo a língua que falamos não foi inventada por nós mesmos, mas é o produto de muitas gerações. Sem ela, não poderíamos nos comunicar com os outros nem compartilhar de suas ideias... Todos os recursos e instalações que damos como certos, garantidos - como casas, carros, estradas, lojas, escolas, hospitais e cinemas - são produzidos somente através da bondade dos outros...

... precisamos dos outros para o nosso bem-estar físico, emocional e espiritual. Sem os outros, não somos nada. Nossa impressão de que somos uma ilha, um indivíduo autossuficiente e independente, não possui relação alguma com a realidade. É mais próximo da realidade nos imaginarmos como uma célula no vasto corpo da vida - distintos, porém intimamente ligados a todos os seres vivos. Não podemos existir sem os outros, e eles, por sua vez, são afetados por tudo o que fazemos. A ideia de que é possível garantir nosso próprio bem-estar enquanto negligenciamos o bem-estar dos outros, ou até mesmo à custa deles, é completamente irrealista". Geshe Kelsang Gyatso

Estudar e aprender sempre foram uma paixão para mim. Concluir esse doutorado é a realização de um sonho de criança. Eu não sabia que existia graduação, mestrado, e muito menos doutorado. Sabia apenas que estudaria até a quarta série no sítio, que depois começaria a estudar na cidade, e que não queria parar de estudar. Queria estudar até o fim, mas nem sabia (e nem sei) o que é esse fim. Esse desejo de aprender e de melhorar me move, mas nada seria possível sem os outros. Sou imensamente grata a todos, tanto àqueles que participam da minha vida mais diretamente, como àqueles que não. Quero poder retribuir todo o bem que me fizeram e me fazem. Não porque os outros esperam de mim um pagamento, mas porque meu coração é cheio de gratidão. Assim sendo, deixo aqui registrado o meu agradecimento a todos os seres, sem exceção. Possa eu me tornar capaz de retribuir tanta bondade.

Se somos células no vasto corpo da vida, eu influencio a todos e todos me influenciam. Porém, o conceito de vizinhança se aplica às células desse corpo, certamente! E sendo específica em relação à elaboração desta tese, são muitos os que participaram ativamente!

Sou muito grata a todos os professores inspiradores que tive em minha vida. Desde a professora Márcia até o professor Fernando, todos foram muito importantes. Cito aqui alguns nomes, correndo o risco de cometer alguma injustiça: Márcia, Regina, Luciana, Maristela, Rogério, Antonieta, Elizete, Edvar, Elaine, Silmara, Almerinda, Robson, Edu-

ardo, Romis, Renato e Fernando. Também agradeço aos meus colegas professores que tanto me inspiraram: Dilermando, Graça, Aldo, Renata e Luciana.

Graça, que bom que você pôde participar da minha defesa. Foi muito bom contar com sua presença nesse dia. Obrigada por toda a ajuda que você me deu para eu poder tocar minha pesquisa enquanto lecionava na FATEC.

Ao meu orientador, Fernando, agradeço por toda a sua dedicação. Foi muito bom ter sido orientada por alguém tão competente e, ao mesmo tempo, humilde, simples e sensível. São muitos os ensinamentos que levarei para a vida toda! Obrigada por ter me ajudado em alguns momentos chatinhos da vida, por ter sempre me incentivado e acreditado em meu potencial!

Alan, meu voto é seu! Obrigada pela ajuda com a análise de complexidade dos algoritmos. Mas, principalmente, obrigada por ser esse amigo tão lindo, sempre muito solidário com todos!

André e Salomão (Salomão, onde quer que você esteja, sinto sua falta), obrigada por todos os dias que estudamos juntos! Que grande oportunidade eu tive de trabalhar com vocês.

Marcos, muito obrigada por estruturar e organizar a infraestrutura do LBiC!

Conrado, obrigada por estar sempre me dando ótimos palpites sobre minha pesquisa!

Obrigada a todos do LBiC que reiniciaram as máquinas para mim! Estou devendo várias para vocês!

A todos do LBiC e do GRUDE, obrigada por serem tão fofinhos! Quão afortunada sou eu em ter a oportunidade de conviver com pessoas tão inteligentes, humildes e compassivas!

A todos os membros da banca, meu muito obrigada pelas contribuições e comentários junto à minha pesquisa.

Eu agradeço à Faculdade de Engenharia Elétrica e de Computação e ao Departamento de Engenharia de Computação e Automação Industrial, pelo fornecimento de instalações e condições de trabalho apropriadas.

Eu agradeço a todos os funcionários e professores da Unicamp que trabalham para nossa universidade ser um local de grande amadurecimento humano.

Eu agradeço à CAPES por ter financiado minha pesquisa.

Eu agradeço a todos os pesquisadores que foram fundamentais para o desenvolvimento da minha pesquisa, como Andrews, Madeira, Ganter, Wille, Kuznetsov, Kaytoue, Zaki, Pei e Napoli.

Também, agradeço muito às pessoas que não fazem parte desse ambiente acadêmico,

mas que me deram todo suporte para eu trabalhar durante esses anos em minha pesquisa.

Eu agradeço aos meus pais pelo dom da vida. Agradeço por terem cuidado de mim e criado todas as condições para que eu tenha uma vida humana plena.

Eu agradeço aos meus avós pelo mimo. Meu coração transborda o amor que me deram.

Eu agradeço aos meus irmãos pela relação intensa, cheia de brigas e brincadeiras na infância, e cheia de companheirismo para a vida toda.

Eu agradeço a todos da minha família - pais, avós, irmãos, tios e primos - por serem a minha família querida.

Eu agradeço ao Gustavinho - meu primo, meu irmão, meu sobrinho, meu filho, meu amigo, meu amor - por fazer de mim uma pessoa mais altruísta.

Eu agradeço a todos os meus amigos e amigas por todos os momentos em que rimos e choramos juntos.

Eu agradeço à Sangha por me ajudar a reconhecer o potencial da vida humana. Um agradecimento especial à minha professora, Andréia, que sempre me aconselha e me direciona por bons caminhos.

Eu agradeço ao Nilton por todo seu zelo por mim. Os comes e bebes que ele preparou para a defesa, que estavam uma delícia, é uma amostra de todo o seu enorme carinho.

Eu agradeço a todos os meus amigos do NBT por me incentivarem e se empolgarem com minhas conquistas acadêmicas.

Peço desculpa a todas as pessoas que porventura eu me esqueci de mencionar por causa de minhas limitações de memória. Mas saibam que as marcas deixadas por vocês são sentidas com o coração. O tempo não as podem apagar.

Chego ao fim dessa seção e agradeço por poder agradecer! Que bom que é agradecer!

"Haveria couro suficiente
Para cobrir a superfície da Terra?
Porém, usar couro apenas nas solas dos pés
Equivale a cobrir a Terra inteira."
Shantideva

Abstract

Biclustering has proved to be a powerful data analysis technique due to its wide success in various application domains. However, the existing literature presents efficient solutions only for enumerating maximal biclusters with constant values, or heuristic-based approaches which cannot find all biclusters or even ensure the maximality of the obtained biclusters. Here, we present a general family of biclustering algorithms for enumerating all maximal biclusters with (*i*) constant values on rows, (*ii*) constant values on columns, or (*iii*) coherent values. Versions for perfect and for perturbed biclusters are provided. Our algorithms have four key properties (only the algorithm for perturbed biclusters with coherent values fails to exhibit the first property): they are (1) efficient (take polynomial time per pattern), (2) complete (find all maximal biclusters), (3) correct (all biclusters attend the user-defined measure of similarity), and (4) non-redundant (all the obtained biclusters are maximal and the same bicluster is not enumerated twice). They are based on a generalization of an efficient formal concept analysis algorithm called In-Close2. Experimental results point to the necessity of having efficient enumerative biclustering algorithms and provide a valuable insight into the scalability of our family of algorithms and its sensitivity to user-defined parameters. Our algorithms were successfully applied in two real-world problems: (*i*) the gene ontology enrichment analysis, and (*ii*) the analysis and identification of biomarkers. In the first application, we show how pseudo enumerative biclustering algorithms (that miss at least one of the three last key properties) fail in finding all enriched biclusters and all genes that belong to these biclusters. In the second application, we show the usefulness of our algorithms in testing the discriminatory capability of proposed biomarkers, and in proposing biomarkers from scratch. We also present a way of extracting associative classification rules from our biclusters, guiding to the possibility of building classifiers based on our solutions.

Keywords: Efficient enumeration; Maximal biclusters; Numerical datasets; Multiple types of biclusters; Perfect and perturbed biclusters.

Resumo

A biclusterização provou ser uma poderosa técnica de análise de dados devido ao seu amplo sucesso em vários domínios de aplicação. No entanto, a literatura existente apresenta soluções eficientes apenas para enumerar biclusters maximais com valores constantes, ou abordagens baseadas em heurísticas que não podem encontrar todos os biclusters ou nem mesmo garantir a maximalidade dos biclusters obtidos. Nesta tese, nós apresentamos uma família genérica de algoritmos de biclusterização para enumerar todos os biclusters maximais com (i) valores constantes nas linhas, (ii) valores constantes nas colunas, ou (iii) valores coerentes. Fornecemos versões para biclusters perfeitos e perturbados. Nossos algoritmos têm quatro propriedades-chave (apenas o algoritmo para biclusters perturbados com valores coerentes não possui a primeira propriedade): eles são (1) eficientes (têm tempo polinomial por bicluster), (2) completos (encontram todos os biclusters maximais), (3) corretos (todos os biclusters atendem a medida de similaridade definida pelo usuário), e (4) não-redundantes (todos os biclusters obtidos são maximais e o mesmo bicluster não é enumerado mais de uma vez). Os algoritmos propostos são baseados em uma generalização de um eficiente algoritmo de análise de conceitos formais, chamado In-Close2. Os resultados experimentais apontam para a necessidade de termos algoritmos enumerativos de biclusterização eficientes, e nos fornecem informações valiosas sobre a escalabilidade de nossa família de algoritmos, como também de sua sensibilidade aos parâmetros definidos pelo usuário. Nossos algoritmos foram aplicados com sucesso em dois problemas do mundo real: (i) a análise de enriquecimento de ontologias gênicas, e (ii) a análise e identificação de biomarcadores. Na primeira aplicação, mostramos como algoritmos pseudo enumerativos de biclusterização (que não possuem pelo menos uma das três últimas propriedades-chave) falham em encontrar todos os biclusters enriquecidos, bem como todos os genes que pertencem a esses biclusters. Na segunda aplicação, mostramos a utilidade de nossos algoritmos para testar a capacidade discriminatória de biomarcadores propostos na literatura, bem como para propor biomarcadores a partir do zero. Apresentamos também uma maneira de extrair regras de classificação associativas dos nossos biclusters, apontando para a possibilidade de construção de classificadores com base em nossas soluções.

Palavras-chaves: Enumeração eficiente; Biclusters maximais; Conjuntos de dados numéricos; Múltiplos tipos de biclusters; Biclusters perfeitos e perturbados.

List of Figures

List of Tables

Contents

1 Introduction

The fields of machine learning and data mining have been attracting a lot of attention due to the increasing necessity of extracting useful information from a huge amount of available data.

Clustering is one of the most popular data mining techniques to knowledge discovery in datasets [53]. It consists in grouping the data into clusters, such that the data points inside a cluster are more similar to each other, or at least to a subset of them, than they are with the data points outside the cluster. Usual clustering techniques assign a data point to a cluster based on global similarities, i.e., similarity measures computed across all attributes. These similarity measures are usually based on distance functions, such as Euclidean distance or Manhattan distance. Besides, most of the clustering methods are only capable of assigning a data point to a single cluster. These characteristics are undesirable in many scenarios, such as analysis of gene expression, text mining, fraud detection, and market basket analysis. For instance, in the analysis of gene expression, a gene could be co-expressed considering only a subset of the conditions. The distance functions are also not suitable for measuring the coherence between genes, because there may be strong coherence between a set of genes even when their expression levels are too far apart from each other when we measure them by a distance function [106]. Furthermore, a gene can belong to none, one, or more than one group. Therefore, the traditional clustering is very inflexible in scenarios like this one.

Biclustering is a local approach for clustering that overcomes these clustering limitations. It operates simultaneously over the set of objects and attributes of a data matrix, looking for submatrices constituted of subsets of objects that have a highly consistent pattern across a subset of attributes. Biclustering methods are able to consider coherence measures which are more general than distance functions, such as Euclidean and Manhattan distances, and hence are going to find biclusters supporting more general affinities than conventional numerical proximity of elements [106].

Given that the concept behind the biclustering approach is appealing in biosciences, biclustering has great value in finding interesting patterns in microarray expression data [86, 111]. Indeed, the application of biclustering is fully disseminated and not limited to biological data. For instance, we can mention: dimensionality reduction [2], text mining [13, 33], collaborative filtering [7, 30, 98, 99], and treatment of missing data [13, 28, 31, 104]. Moreover, the importance of biclustering continues to increase, as researchers are (*i*) finding new applications in scientific and commercial domains, including bioinformatics, social network analysis, and text mining; and (*ii*) unveiling the connection

between biclustering and several other important problems, including subspace clustering [63], frequent pattern mining (FPM) [43], and formal concept analysis (FCA) [39].

Biclustering may be interpreted as a hard combinatorial optimization problem. The more flexible the bicluster structure, the more complex the problem, and we are considering the most flexible structure in this thesis: arbitrarily positioned overlapping biclusters [79]. Thereby, an object/attribute can belong to none, one, or more than one bicluster. In this scenario, finding all maximal biclusters in a data matrix is an NP-hard problem [80]. Due to this, most of the proposed algorithms are heuristic-based [79], and many of them consider an inflexible bicluster structure and mine a number of biclusters defined a priori. The heuristic-based algorithms potentially produce sub-optimal solutions, missing important biclusters and not guaranteeing the maximality of the identified ones. Some examples of well-known heuristic-based biclustering algorithms are: CC [23], FLOC [107], ROCC [32], ISA [52], Plaid [69], and OPSM [14]. For surveys, refer to Madeira and Oliveira [79], Busygin *et al.* [18] and Pontes *et al.* [96].

In FCA and related areas, such as FPM and graph theory, we have plenty of algorithms for enumerating all maximal biclusters with constant values (CTV) in a binary dataset. These maximal CTV biclusters are called formal concepts in FCA, closed frequent itemsets (or patterns) in FPM [1] , and maximal bicliques in graph theory (for more details about the connection of these areas, see Chapter 3, Section 3.2). Some examples of algorithms are: Makino and Uno [81], Eppstein *et al.* [35], Close-by-One (CbO) [65], In-Close [8], In-Close2 [9], FCbO [62], CHARM [109], and LCM [101]. Their enumeration process is characterized by being:

1. Efficient: it takes polynomial time per pattern, i.e., it takes polynomial time to enumerate the first bicluster and takes polynomial time between enumerating two consecutive biclusters. It is the best one can computationally do in such scenario. If done properly, such algorithm will have time complexity linear in the number of biclusters and polynomial in the input size. Moreover, if the number of maximal biclusters is polynomial in the input size, the overall algorithm will be a polynomial time algorithm.

2. Complete: it finds all maximal biclusters. A complete enumeration guarantees to include the results produced by any other biclustering solution (given the same restrictions of similarity and size). So, such biclustering solution is at least of equal quality, but probably of better quality, when compared with the solution provided by any other contender.

3. Correct: all found biclusters attend the user-defined measure of similarity. For instance, in the case of the aforementioned algorithms, all biclusters are submatrices

[1] Being more specific, a closed frequent itemset corresponds to the column-set of a bicluster.

of ones. Complete and correct enumerators are crucial for some applications such as the identification of biological indicators [76] and classification based on associations [74].

4. Non-redundant: all biclusters are maximal and it does not enumerate the same maximal bicluster twice. It is very important because the number of biclusters produced from a dataset can be very large. So, it is useful to identify the smallest representative set of biclusters from which all other biclusters can be derived [92]. The set of all maximal biclusters is necessary and sufficient to capture all the information about the biclusters, and has a much smaller cardinality than the set of all attainable biclusters [108]. It is important to note that the algorithm must have a smart solution to avoid redundancy, otherwise it will not be efficient. For instance, a procedure to be avoided is to check if a new bicluster is not redundant by comparing with all previously mined biclusters.

Once the researchers found the link between these related areas and biclustering, many algorithms have been proposed to deal with numerical (not only binary, but also integer or real-valued) datasets and other types of biclusters. In fact, nowadays the state-of-the-art biclustering algorithms are based on FPM [49]. Many proposals, such as [49, 80, 82, 87, 88, 97], binarize the data and apply the aforementioned algorithms. However, binarizing the dataset leads to loss of information, and guides to the necessity of tedious Boolean property encoding phases [16]. Therefore, there are also proposals to deal directly with numerical datasets, such as [16, 57, 91, 95, 111]. Without binarizing the numerical dataset, we are going to show in Chapter 4 that there is no proposal in the literature able to enumerate biclusters with (*i*) constant value on columns (CVC), (*ii*) constant values on rows (CVR), or (*iii*) coherent values (CHV) (see definitions in Chapter 2), so that the aforementioned four properties are preserved in this extended scenario (not only binary values in the dataset). Although some authors claim that their proposals do preserve these four properties, a more careful analysis to be presented in Chapter 4 shows that they all fail to exhibit one or more of these four properties. So, the main aim of this thesis is to fill these gaps. In fact, we are proposing algorithms capable of preserving these four properties when enumerating perfect CVC biclusters, perturbed CVC biclusters, and perfect CHV biclusters. The problem of enumerating CVR biclusters is equivalent to enumerating CVC biclusters. We are also proposing an algorithm with the last three of these properties to enumerate perturbed CHV biclusters. Note that CVC, CVR and CHV biclusters are a generalization of CTV biclusters [79] (for more details see Chapter 2, Section 2.1.1). We call our family of algorithms *RIn-Close* because they are generalizations of the FCA algorithm In-Close2 [9]. Tables 7 and 8 (see Chapter 4) show a technical comparison between our proposals and the competitors, attesting that we are proposing a number of improvements when enumerating biclusters from numerical datasets.

1.1 Goals

Due to the complexity of the biclustering problems, most of the proposed biclustering algorithms are heuristic-based, leading to sub-optimal solutions. The majority of the enumerative biclustering approaches requires the binarization of the data matrix, which leads to loss of information. The existent enumerative algorithms that deal directly with numerical datasets do not have all the four key properties of an enumerative biclustering algorithm (i.e., (1) efficiency, (2) completeness, (3) correctness, and (4) non-redundancy). The main goal of this thesis is to fill these gaps. For this, we are proposing a family of enumerative biclustering algorithms, called RIn-Close, that are the most complete from the literature. Our algorithms to enumerate perfect or perturbed CVC (or CVR) biclusters, and our algorithm to mine perfect CHV biclusters, are the first ones with these four key properties. Our algorithm to enumerate perturbed CHV biclusters is the first one with these last 3 key properties. An algorithm to enumerate perturbed CHV biclusters with all four key properties remains an open problem.

Still regarding to enumerative biclustering algorithms, we show how (*i*) it is possible to use our algorithms in datasets with categorical attributes, (*ii*) the procedure to enumerate CHV biclusters can be easily adapted to enumerate other types of biclusters, such as OPSM biclusters, and (*iii*) the missing data can be easily handled when using RIn-Close.

We also made several experiments with our family of enumerative biclustering algorithms in order to:

- Test RIn-Close's scalability when varying some characteristics of the datasets. These experimental results indicate to the user when it is feasible to use our family of algorithms in data analysis.
- Test RIn-Close's sensitivity to its main parameters, which indicates to the user how to set RIn-Close's parameters.
- Test RIn-Close's sensitivity to missing values in the dataset.
- Indicate the distinct aspects of enumerative biclustering algorithms when compared to heuristic-based biclustering algorithms.
- Indicate the distinct aspects of an actual enumerative biclustering algorithm when compared to a pseudo enumerative biclustering algorithm.
- Indicate the usefulness of RIn-Close algorithms to solve real-world problems.
- Indicate the main advantages of applying enumerative biclustering algorithms to the gene ontology enrichment analysis.

- Indicate the potential of using enumerative biclustering algorithms to test the discriminatory capability of proposed biomarkers.
- Indicate the potential of using enumerative biclustering algorithms to propose biomarkers.
- Show how we can extract associative classification rules from biclusters that are not binary.

1.2 Thesis Organization

The remaining content of the thesis is organized as follows:

Chapter 2 introduces definitions and mathematical formulations for biclustering. It reviews the main types of bicluters and the links between them. It also presents the main bicluster structures and the methods used to identify the biclusters. The concept of maximal biclusters and properties of our bicluster definitions are presented. Some bicluster metrics and indices are also outlined.

Chapter 3 reviews main concepts of FCA. The algorithm In-Close2, which was generalized to propose the RIn-Close family of algorithms, is presented. It also outlines the links between FCA, biclustering, association mining (more specifically, frequent itemset mining), and graph theory.

Chapter 4 places the RIn-Close family of algorithms before the existing biclustering algorithms in the literature. It also highlights common strategies used by enumerative biclustering algorithms to handle missing data and categorical attributes.

Chapter 5 presents the main contributions of this thesis, more specifically the RIn-Close family of enumerative algorithms for mining maximal CVC, CVR, or CHV biclusters. It also presents our proposal of handling missing data when using RIn-Close, and presents how to mine other types of biclusters (in addition to CVC, CVR, and CHV biclusters) by means of adapting the procedure to enumerate CHV biclusters.

Chapter 6 presents and discusses experimental results. These experiments include the analysis of RIn-Close's scalability, the analysis of RIn-Close's sensitivity to its main parameters and to missing data, the comparison between an enumerative algorithm and heuristics, the comparison of a true enumerative algorithm against a pseudo enumerative algorithm when applied to gene ontology enrichment analysis, and analysis and identification of biomarkers.

Chapter 7 presents conclusions, limitations, and suggestions of future research avenues supported by our family of biclustering algorithms.

2 Biclustering

The term *biclustering* was introduced by Mirkin [83] to describe the simultaneous clustering of the sets of rows and columns of a data matrix. More recently, the term was used in the analysis of gene expression data [23]. Cheng and Church [23] significantly contributed to the popularization of biclustering techniques with their heuristic-based algorithm called CC. However, Hartigan [48] was the first one to propose an algorithm for biclustering, using the term *direct clustering*. Other terms that are found in literature are: co-clustering, two-way clustering and bidimensional clustering, among others [79].

2.1 Definitions and Taxonomy of Biclusters

Let $\mathbf{A}_{n\times m}$ be a data matrix with the row index set $X = \{1, 2, ..., n\}$ and the column index set $Y = \{1, 2, ..., m\}$. Each element $a_{ij} \in \mathbf{A}$ represents the relationship between row i and column j. We use (X, Y) to denote the entire matrix $\mathbf{A}$. Considering that $I \subseteq X$ and $J \subseteq Y$, $\mathbf{A}_{IJ} = (I, J)$ denotes the submatrix of $\mathbf{A}$ with the row index subset I and column index subset J.

Definition 2.1. *A* bicluster *is a submatrix (I, J) of the data matrix $\mathbf{A}_{n\times m}$ such that the rows in the index subset $I = \{i_1, ..., i_k\}$ ($I \subseteq X$ and $k \leq n$) exhibits similar behavior across the columns in the index subset $J = \{j_1, ..., j_s\}$ ($J \subseteq Y$ and $s \leq m$), and vice-versa.*

Thus, a bicluster (I, J) is a $k \times s$ submatrix of the matrix $\mathbf{A}$, with not necessarily contiguous rows and columns, such that it meets a certain homogeneity criterion. A biclustering algorithm looks for a set of biclusters $\mathfrak{B} = (I_l, J_l)_{l=1}^{q}$, such that each bicluster (I_l, J_l), $l = 1, ..., q$, satisfies some specific characteristics of homogeneity [79]. Considering these characteristics, there are four major types of biclusters [79]: (i) biclusters with constant values (CTV), (ii) biclusters with constant values on columns (CVC) or rows (CVR), (iii) biclusters with coherent values (CHV), and (iv) biclusters with coherent evolutions (CHE). The total number of biclusters, q, will depend on the features of the selected biclustering algorithm, on the constraints imposed, and on the structure of the dataset being analyzed.

Table 1 shows an example of a data matrix $\mathbf{A}_{9\times 8}$ with two perfect CTV biclusters highlighted: (i) one bicluster is formed by the elements in red; and (ii) the second bicluster is formed by the elements in blue. In turn, Table 2 shows an example of a data matrix $\mathbf{A}_{10\times 10}$ with three perfect CHV biclusters highlighted: (i) one bicluster is formed by the elements in red; (ii) a second bicluster is formed by the elements in blue; and (iii) the third bicluster is formed by the elements with the gray background. The second and the third

Table 1 – Example of a data matrix $\mathbf{A}_{9\times8}$ with two perfect CTV biclusters highlighted.

33	39	32	16	39	20	5	39
37	7	37	37	13	18	37	36
6	39	27	7	39	26	39	39
37	39	37	37	39	29	37	39
26	20	34	2	18	31	24	6
4	33	38	12	16	12	9	6
12	6	28	2	31	28	31	11
22	39	31	4	39	27	11	39
39	37	30	33	8	7	21	11

Table 2 – Example of a data matrix $\mathbf{A}_{10\times10}$ with three perfect CHV biclusters highlighted.

31	28	11	43	30	22	42	23	30	25
23	2	30	9	7	18	31	14	19	11
16	39	28	24	27	4	29	1	25	6
12	10	15	26	15	31	25	21	13	8
1	4	4	20	9	20	19	10	11	11
3	33	6	31	31	22	30	12	18	13
15	29	7	12	27	19	18	4	34	31
3	7	6	23	12	22	22	12	10	5
2	13	5	29	18	21	28	11	33	21
31	13	27	14	16	40	26	19	11	6

biclusters overlaps. Figure 1 and 2 show examples of the types of biclusters. In Figure 1, the biclusters do not have any residue, i.e., they are perfect biclusters. In Figure 2, the biclusters have some residue, i.e., they are perturbed. We will define all these types of biclusters in Subsection 2.1.1.

In addition to the type of the biclusters, other two important aspects to consider when choosing a biclustering algorithm are the admissible bicluster structure and the method used to identify the biclusters.

Madeira and Oliveira [79] pointed out nine types of bicluster structures. Among them, non-overlapping biclusters with checkerboard structure, and arbitrarily positioned overlapping biclusters are the most common structures. Basically, the algorithms that look for the first structure just reorder the rows and columns of the data matrix aiming at identifying $e \times f$ biclusters arranged in a grid structure. The second one is the most flexible structure among all nine possibilities. Note that if we consider the checkboard structure, all objects will necessarily belong to f biclusters, and all attributes will necessarily belong to e biclusters. On the other hand, with arbitrarily positioned overlapping biclusters, an object/attribute may belong to none, one, or more than one bicluster. Typically, some stopping criterion, such as the maximum number of biclusters to be identified, is one of the parameters of the heuristics that look for this type of structure. In turn, enumerative algorithms do not receive this kind of parameter and look for all the biclusters that meet

2.0	2.0	2.0	2.0	2.0
2.0	2.0	2.0	2.0	2.0
2.0	2.0	2.0	2.0	2.0
2.0	2.0	2.0	2.0	2.0

CTV bicluster

2.0	2.0	2.0	2.0	2.0
9.1	9.1	9.1	9.1	9.1
5.5	5.5	5.5	5.5	5.5
3.7	3.7	3.7	3.7	3.7

CVR bicluster

1.5	9.0	3.2	8.6	5.4
1.5	9.0	3.2	8.6	5.4
1.5	9.0	3.2	8.6	5.4
1.5	9.0	3.2	8.6	5.4

CVC bicluster

2.7	11.7	1.7	9.7	4.2
4.0	13.0	3.0	11.0	5.5
9.5	18.5	8.5	16.5	11.0
8.0	17.0	7.0	15.0	9.5

CHV bicluster (additive)

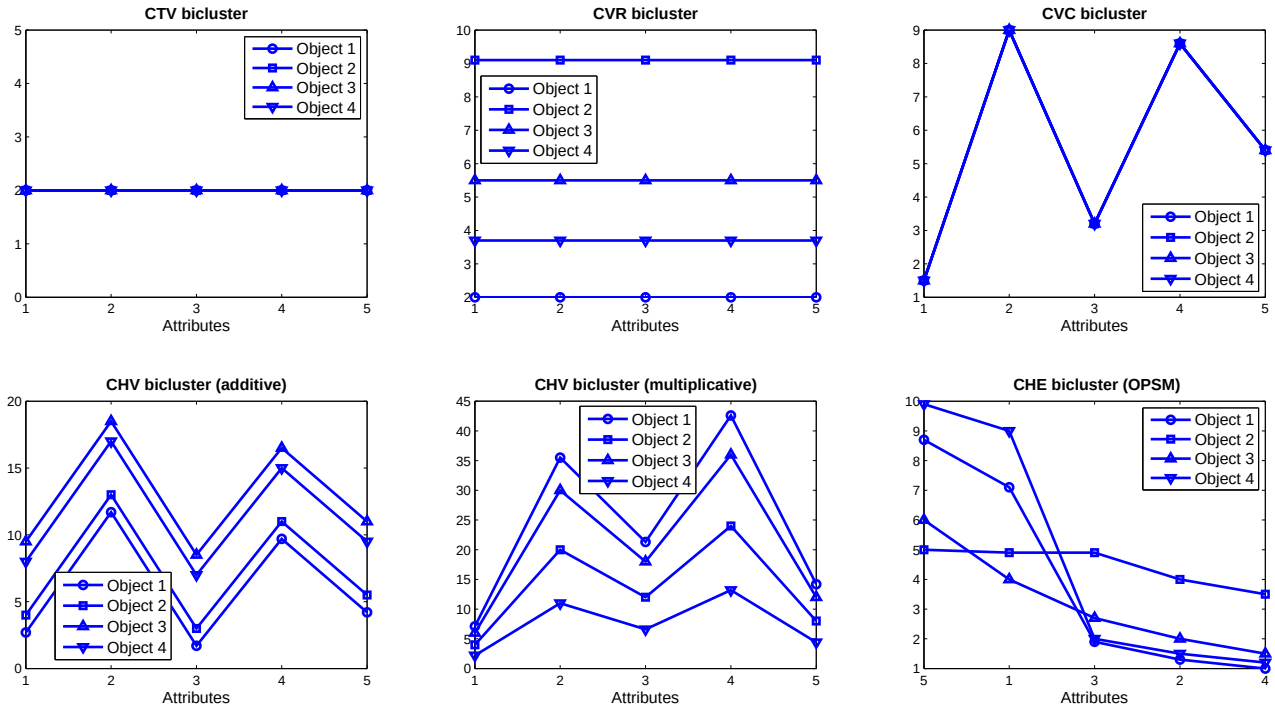
7.1	35.5	21.3	42.6	14.2
4.0	20.0	12.0	24.0	8.0
6.0	30.0	18.0	36.0	12.0
2.2	11.0	6.6	13.2	4.4

CHV bicluster (multiplicative)

7.1	1.3	1.9	1.0	8.7
4.9	4.0	4.9	3.5	5.0
4.0	2.0	2.7	1.5	6.0
9.0	1.5	2.0	1.2	9.9

CHE bicluster (OPSM)

(a) Numerical example.



(b) Graphical example.

Figure 1 – Examples of different types of biclusters. These biclusters are perfect, i.e., they do not have any residue.

2.0	2.0	2.0	2.7	2.5
2.0	3.0	2.0	2.0	2.0
2.0	2.1	2.9	2.0	2.7
2.1	2.4	2.0	2.0	2.0

CTV bicluster

2.0	2.4	2.0	3.0	2.0
9.1	9.1	9.1	9.1	9.1
5.5	5.7	5.5	5.0	5.5
3.5	3.7	3.7	3.7	3.6

CVR bicluster

1.5	9.0	3.2	8.6	5.4
1.5	8.0	3.2	9.0	5.4
2.0	8.4	3.2	8.5	5.4
1.5	8.7	3.2	8.6	5.4

CVC bicluster

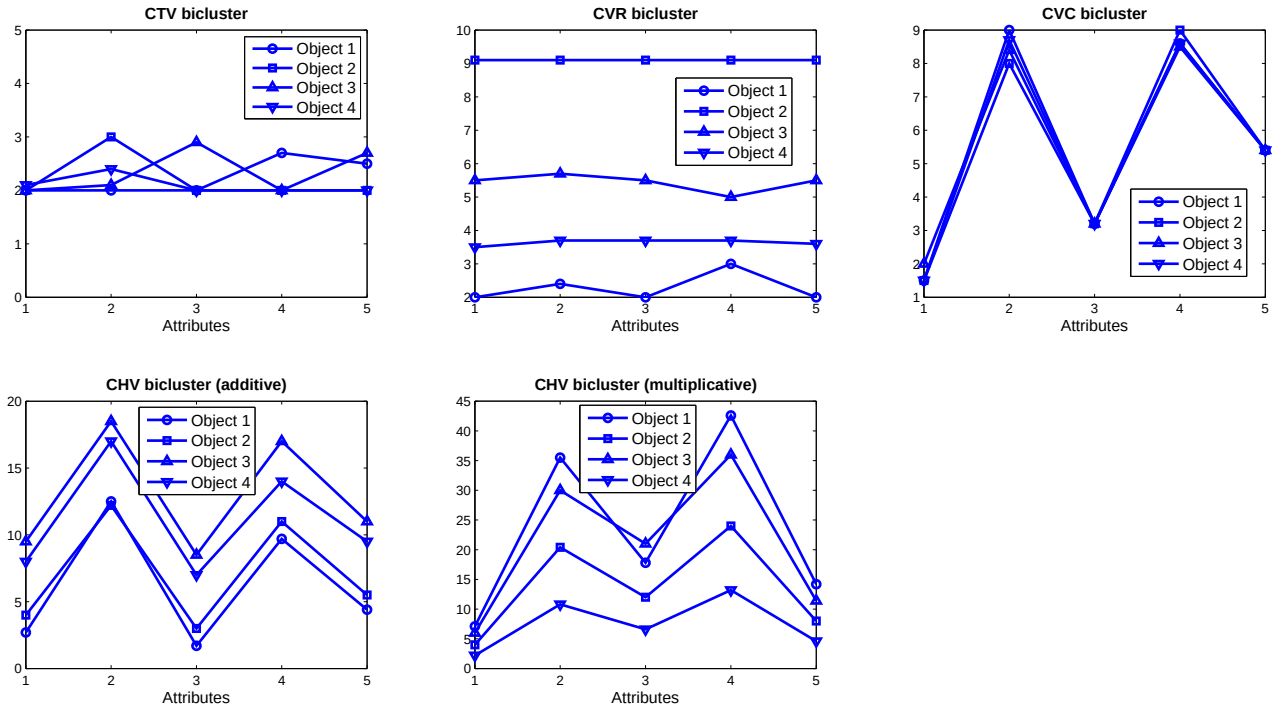
2.7	12.5	1.7	9.7	4.4
4.0	12.2	3.0	11.0	5.5
9.5	18.5	8.5	17.0	11.0
8.0	17.0	7.0	14.0	9.5

CHV bicluster (additive)

7.1	35.5	17.8	42.6	14.2
4.0	20.4	12.0	24.0	8.0
6.0	30.0	21.0	36.0	11.4
2.2	10.8	6.6	13.2	4.6

CHV bicluster (multiplicative)

(a) Numerical example.



(b) Graphical example.

Figure 2 – Examples of different types of biclusters. These biclusters are perturbed, i.e., they have some residue.

the restrictions of similarity and size. Table 2 shows an example of arbitrarily positioned overlapping biclusters.

Among the methods to identify the biclusters, we have [79]:

- Iterative row and column clustering combination: standard clustering algorithms are applied separately on the column and row dimensions of the data matrix, and then the results are combined, using some kind of iterative procedure, to obtain the biclusters.
- Divide and conquer: the problem is divided into several sub-problems that are similar to the original, but smaller. These sub-problems are solved recursively, and then the solutions are combined to create a unique solution to the original problem.
- Greedy iterative search: these algorithms create biclusters by adding or removing rows / columns of the biclusters, using a criterion that maximizes the local gain, in the hope that this choice will lead to a good global solution.
- Distribution parameter identification: these algorithms consider that the biclusters are generated using a statistical model, and try to identify the distribution parameters that fit the available data by iteratively minimizing certain optimization criteria.
- Exhaustive bicluster enumeration: these algorithms mine all biclusters of the data matrix that meet some restrictions, usually of similarity and size.

2.1.1 Types of Biclusters

Although perfect biclusters can be found in some data matrices, they are usually masked by noise in real data. Therefore, we will define the perfect and the perturbed cases for all types of biclusters. The perturbed case is always a generalization of the perfect case. A user-defined parameter $\epsilon \geq 0$ determines the maximum residue (perturbation) allowed in a bicluster.

Let $|\zeta|$ be (i) the absolute value (or modulus) of ζ if ζ is a scalar; or (ii) the number of elements in ζ if ζ is a set.

Definition 2.2 (CTV biclusters)**.** *A* perfect CTV (constant value) bicluster *is a submatrix* (I, J) *of a data matrix* $\mathbf{A}_{n\times m}$ *such that* $a_{ij} = a_{kl}$, $\forall i, k \in I$ *and* $\forall j, l \in J$. *A* perturbed CTV bicluster *is a submatrix* (I, J) *of a data matrix* $\mathbf{A}_{n\times m}$ *such that* $|a_{ij} - a_{kl}| \leq \epsilon$, $\forall i, k \in I$ *and* $\forall j, l \in J$, *i.e.,*

$$\max_{i\in I, j\in J}(a_{ij}) - \min_{i\in I, j\in J}(a_{ij}) \leq \epsilon. \tag{2.1}$$

Figure 1 shows a perfect CTV bicluster, and Figure 2 shows a perturbed CTV bicluster that can be obtained using $\epsilon \geq 1$.

Definition 2.3 (CVR biclusters)**.** *A* perfect CVR (constant value on rows) bicluster *is a submatrix* (I, J) *of a data matrix* $\mathbf{A}_{n \times m}$ *such that* $a_{ij} = a_{il}$, $\forall i \in I$ *and* $\forall j, l \in J$. *A* perturbed CVR bicluster *is a submatrix* (I, J) *of a data matrix* $\mathbf{A}_{n \times m}$ *such that* $|a_{ij} - a_{il}| \leq \epsilon$, $\forall i \in I$ *and* $\forall j, l \in J$, *i.e.,*

$$\max_{j \in J}(a_{ij}) - \min_{j \in J}(a_{ij}) \leq \epsilon, \forall i \in I. \tag{2.2}$$

Figure 1 shows a perfect CVR bicluster, and Figure 2 shows a perturbed CVR bicluster that can be obtained using $\epsilon \geq 1$.

Definition 2.4 (CVC biclusters)**.** *A* perfect CVC (constant value on columns) bicluster *is a submatrix* (I, J) *of a data matrix* $\mathbf{A}_{n \times m}$ *such that* $a_{ij} = a_{kj}$, $\forall i, k \in I$ *and* $\forall j \in J$. *A* perturbed CVC bicluster *is a submatrix* (I, J) *such that* $|a_{ij} - a_{kj}| \leq \epsilon$, $\forall i, k \in I$ *and* $\forall j \in J$, *i.e.,*

$$\max_{i \in I}(a_{ij}) - \min_{i \in I}(a_{ij}) \leq \epsilon, \forall j \in J. \tag{2.3}$$

Figure 1 shows a perfect CVC bicluster, and Figure 2 shows a perturbed CVC bicluster that can be obtained using $\epsilon \geq 1$.

Note that the definition of a CVR bicluster is the equivalent transpose of the definition of a CVC bicluster. So, we can mine CVR biclusters by transposing the original data matrix and using an algorithm to mine CVC biclusters [91].

There are two perspectives for CHV (coherent value) biclusters: (*i*) additive model, and (*ii*) multiplicative model. Biclusters based on the additive model are called *shifting biclusters*. Biclusters based on the multiplicative model are called *scaling biclusters*. Any row (column) of a perfect shifting bicluster can be obtained by adding a constant value to any other row (column) of the bicluster. For instance, in the shifting (additive CHV) bicluster of Figure 1: the second row is equal to the first row plus 1.3, the third column is equal to the second column plus -10, and so on. Similarly, any row (column) of a perfect scaling bicluster can be obtained by multiplying a constant value to any other row (column) of the bicluster. For instance, in the scaling (multiplicative CHV) bicluster of Figure 1: the third row is equal to the fourth row times 2.73, the second column is equal to the first column times 5, and so on.

Definition 2.5 (CHV biclusters - additive model)**.** *Let* $Z^{jl} = \{a_{ij} - a_{il}\}_{i \in I}$, $j, l \in J$. *For instance, consider the shifting bicluster of Figure 2, and let* $j = 2$ *and* $l = 3$, *then*

$Z^{23} = \{10.5, 9.5, 10, 10\}$. *A* perfect shifting bicluster *is a submatrix* (I, J) *of a data matrix* $\mathbf{A}_{n \times m}$ *such that all elements of the set* Z^{jl}, $\forall j, l \in J$, *are equal, i.e.,* $z = w$, $\forall z, w \in Z^{jl}$, $\forall j, l \in J$. *A* perturbed shifting bicluster *is a submatrix* (I, J) *such that* $|z - w| \leq \epsilon$, $\forall z, w \in Z^{jl}$, $\forall j, l \in J$, *i.e.,*

$$\max(Z^{jl}) - \min(Z^{jl}) \leq \epsilon, \forall j, l \in J. \tag{2.4}$$

Figure 1 shows a perfect shifting bicluster, and Figure 2 shows a perturbed shifting bicluster that can be obtained using $\epsilon \geq 1.8$.

Definition 2.6 (CHV biclusters - multiplicative model)**.** *Let* $Z^{jl} = \{a_{ij}/a_{il}\}_{i \in I}$, $j, l \in J$. *A* perfect scaling bicluster *is a submatrix* (I, J) *of a data matrix* $\mathbf{A}_{n \times m}$ *such that all elements of the set* Z^{jl}, $\forall j, l \in J$, *are equal, i.e.,* $z = w$, $\forall z, w \in Z^{jl}$, $\forall j, l \in J$. *A* perturbed scaling bicluster *is a submatrix* (I, J) *such that*

$$\max(Z^{jl}) / \min(Z^{jl}) - 1 \leq \epsilon, \forall j, l \in J. \tag{2.5}$$

Figure 1 shows a perfect scaling bicluster, and Figure 2 shows a perturbed scaling bicluster that can be obtained using $\epsilon \geq 0.47$.

The problems of mining shifting and scaling biclusters are equivalent. Using an algorithm to mine shifting (scaling) biclusters, we can mine scaling (shifting) biclusters by previously taking the logarithm (exponential) of all entries of the data matrix (see Lemmas 2.1 and 2.2). Therefore, we are going to focus only on the additive model in the remainder of this thesis.

Lemma 2.1. *If* (I, J) *is a scaling bicluster in the data matrix* $\mathbf{A}$ *with residue less or equal to* ϵ, *then* (I, J) *is a shifting bicluster in* $\log(\mathbf{A})$ *with residue less or equal to* $\log(\epsilon + 1)$ *(where* $\log(\mathbf{A})$ *is a data matrix obtained by applying the logarithm function to each value* $a_{ij} \in \mathbf{A}$*).*

Proof. Without loss of generality, let a_{ij}, a_{il}, a_{kj} and a_{kl} be any four elements of the scaling bicluster (I, J). So, we have:

$$\frac{a_{ij}/a_{il}}{a_{kj}/a_{kl}} - 1 \leq \epsilon$$

$$\frac{a_{ij}/a_{il}}{a_{kj}/a_{kl}} \leq \epsilon + 1$$

Applying the logarithm function, we have:

$$\log(\frac{a_{ij}/a_{il}}{a_{kj}/a_{kl}}) \leq \log(\epsilon + 1)$$

$$\log(a_{ij}/a_{il}) - \log(a_{kj}/a_{kl}) \leq \log(\epsilon + 1)$$

$$(\log(a_{ij}) - \log(a_{il})) - (\log(a_{kj}) - \log(a_{kl})) \leq \log(\epsilon + 1)$$

□

Lemma 2.2. *If (I, J) is a shifting bicluster in the data matrix $\mathbf{A}$ with residue less or equal to ϵ, then (I, J) is a scaling bicluster in $e^{\mathbf{A}}$ with residue less or equal to $e^{\epsilon} - 1$ (where $e^{\mathbf{A}}$ is a data matrix obtained by applying the exponential function to each value $a_{ij} \in \mathbf{A}$).*

Proof. Without loss of generality, let a_{ij}, a_{il}, a_{kj} and a_{kl} be any four elements of the shifting bicluster (I, J). So, we have:

$$(a_{ij} - a_{il}) - (a_{kj} - a_{kl}) \leq \epsilon$$

Applying the exponential function, we have:

$$e^{(a_{ij}-a_{il})-(a_{kj}-a_{kl})} \leq e^{\epsilon}$$

$$\frac{e^{(a_{ij}-a_{il})}}{e^{(a_{kj}-a_{kl})}} \leq e^{\epsilon}$$

$$\frac{e^{a_{ij}}/e^{a_{il}}}{e^{a_{kj}}/e^{a_{kl}}} - 1 \leq e^{\epsilon} - 1$$

□

Another interesting property of mining CHV biclusters is that it is a symmetric problem. The CHV biclusters are fully preserved when rows become columns and columns become rows of the matrix (see Lemma 2.3).

Lemma 2.3. *If (I, J) is a CHV bicluster in the data matrix $\mathbf{A}$ with residue less or equal to ϵ, then (J, I) is a CHV bicluster in $\mathbf{A}^{\intercal}$ with residue less or equal to ϵ.*

Proof. Let

$$\begin{bmatrix} x & y \\ w & z \end{bmatrix}$$

be any arbitrary 2×2 submatrix of the bicluster (I, J). So, $|(x - y) - (w - z)| \leq \epsilon$, and

$$\begin{bmatrix} x & w \\ y & z \end{bmatrix}$$

is an arbitrary 2×2 submatrix of the bicluster (J, I). So, we must prove that $|(x - w) - (y - z)| \leq \epsilon$. In fact, reorganizing the elements of the summation produces

$$|(x - y) - (w - z)| = |(x - w) - (y - z)|$$

and the demonstration is concluded.

□

Biclustering algorithms for finding CHE (coherent evolution) biclusters address the problem of finding coherent evolutions across the rows and/or columns of the data matrix regardless of their exact values [79]. There are many subtypes of CHE biclusters, and the order-preserving submatrix (OPSM) biclusters are the most famous among them. Figure 1 shows an example of an OPSM bicluster.

Definition 2.7 (OPSM biclusters)**.** *An* OPSM bicluster *is a submatrix* (I, J) *of a data matrix* $\mathbf{A}_{n \times m}$ *such that there is a permutation* $P = \{p_1, p_2, ..., p_s\}$ *of the set of columns* J*, where* $a_{ip_1} \leq a_{ip_2} \leq ... \leq a_{ip_s}$*,* $\forall i \in I$*.*

Figure 1 shows an OPSM bicluster, where $P = \{4, 2, 3, 1, 5\}$.

We can easily notice that, in the perfect case, CVR and CVC biclusters are generalizations of CTV biclusters. Similarly, CHV biclusters are generalizations of CVR and CVC biclusters. Conformably, OPSM biclusters are generalizations of CHV biclusters. Those inferences are also true for the perturbed cases, as shown by Lemmas 2.4 to 2.6.

Lemma 2.4. *A CTV bicluster with residue* ϵ *is a CVC (CVR) bicluster with residue* ϵ' *such that* $\epsilon' \leq \epsilon$*.*

Proof. If a CTV bicluster has residue ϵ, it means that the variation inside it is ϵ. Consequently, the maximum variation in a bicluster column is ϵ. Therefore, $\epsilon' \leq \epsilon$.

The proof for a CVR bicluster is equivalent. □

Lemma 2.5. *A CVC (CVR) bicluster with residue* ϵ *is a CHV bicluster with residue* ϵ' *such that* $\epsilon' \leq 2\epsilon$*.*

Proof. Without loss of generality, let pick any two columns j and l of the CVC bicluster (I, J). So, we have that

$$\begin{aligned}\max_i(a_{ij} - a_{il}) - \min_i(a_{ij} - a_{il}) &\leq (\max_i(a_{ij}) - \min_i(a_{il})) - (\min_i(a_{ij}) - \max_i(a_{il})) \\ &= \max_i(a_{ij}) - \min_i(a_{il}) + \max_i(a_{il}) - \min_i(a_{ij}) \\ &\leq 2\epsilon.\end{aligned}$$

The proof for a CVR bicluster is equivalent. □

Lemma 2.6. *A bicluster is an OPSM when the differences between all its pairs of columns have the same signal.*

Proof. If the difference between two columns of a bicluster has the same signal for all elements, then we know the order between these two columns. If we have this for all pairs of columns, then we can assemble a permutation so that all columns are rearranged from the column with the lowest values to the column with the highest values. So, we have an OPSM by definition. □

In Appendix A, we will exemplify the usage of biclustering in a kind of dataset that is very popular nowadays: a dataset containg ratings from users on movies. We will demonstrated some limitations of the traditional clustering techniques when applied to this kind of dataset, and we will provide examples of mining different types of biclusters from this dataset.

2.2 Metrics and indices

Here, we will outline some bicluster metrics and indices to ease the reading of this work.

The *volume* of a bicluster (I, J) is given by $|I| \times |J|$.

The *overlap* between two biclusters (I, J) and (I', J') is given by:

$$ove((I, J), (I', J')) = \frac{|I \cap I'| \times |J \cap J'|}{\min(|I \times J|, |I' \times J'|)} \tag{2.6}$$

Let $\mathfrak{B} = (I_l, J_l)$, $l = 1, ..., q$, be a biclustering solution, composed of q biclusters. The *span* of the solution $\mathfrak{B}$ is given by:

$$span(\mathfrak{B}) = \bigcup_{(I_l, J_l)} I_l \times J_l, \tag{2.7}$$

The *coverage* of the solution $\mathfrak{B}$ is given by:

$$cov(\mathfrak{B}) = |span(\mathfrak{B})|, \tag{2.8}$$

i.e, the number of cells of the data matrix covered by at least one bicluster. It is more usual to present the coverage in terms of percentage, i.e, $100 \times cov(\mathfrak{B})/(n \times m)$.

The *global overlap* of the solution $\mathfrak{B}$ is given by:

$$oveg(\mathfrak{B}) = \frac{\sum_{(I_l \times J_l)} |I_l \times J_l| - cov(\mathfrak{B})}{cov(\mathfrak{B})}. \tag{2.9}$$

If we have a reference bicluster solution $\dot{\mathfrak{B}}$, we can measure how good is a found bicluster solution $\mathfrak{B}$ by means of an external evaluation [50]. We will use *precision* and *recall* to this end:

$$prec(\mathfrak{B}, \dot{\mathfrak{B}}) = \frac{|span(\mathfrak{B}) \cap span(\dot{\mathfrak{B}})|}{cov(\mathfrak{B})} \tag{2.10}$$

$$rec(\mathfrak{B}, \dot{\mathfrak{B}}) = prec(\dot{\mathfrak{B}}, \mathfrak{B}) \tag{2.11}$$

2.3 Maximality and Properties

Definition 2.8 (Maximal bicluster)**.** *Given the desired characteristics of homogeneity, a bicluster (I, J) is called a* maximal bicluster *if and only if:*

- $\forall x \in X \setminus I$, $(I \cup \{x\}, J)$ *is not a (valid) bicluster, and*
- $\forall y \in Y \setminus J$, $(I, J \cup \{y\})$ *is not a (valid) bicluster.*

It means that a bicluster is maximal if we cannot add any object/attribute to it without violating the desired characteristics of homogeneity. For instance, a CTV bicluster (I, J) is called a maximal CTV bicluster iff:

- $\forall x \in X \setminus I$, $\max_{i \in I \cup \{x\}, j \in J}(a_{ij}) - \min_{i \in I \cup \{x\}, j \in J}(a_{ij}) > \epsilon$, and
- $\forall y \in Y \setminus J$, $\max_{i \in I, j \in J \cup \{y\}}(a_{ij}) - \min_{i \in I, j \in J \cup \{y\}}(a_{ij}) > \epsilon$.

For all bicluster definitions given in Subsection 2.1.1, we have the following properties.

Property 2.1 (Anti-Monotonicity)**.** *Let (I, J) be a bicluster. Any submatrix (I', J'), where $I' \subseteq I$ and $J' \subseteq J$, is also a bicluster.*

Property 2.2 (Monotonicity)**.** *Let (I, J) be a maximal bicluster. Any supermatrix of (I, J) is not a valid bicluster.*

Usually, the efficient algorithms for enumerating CTV biclusters of ones from binary datasets are based on the monotonicity and anti-monotonicity properties [16]. In fact, we do not know any efficient algorithm for this task that is not based on these properties. RIn-Close (see Chapter 5), in turn, also considers these properties, but now in the context of numerical datasets. Any definition of a bicluster type that meets these properties can be used in our framework. For instance, our definition of a CVC bicluster considers the same maximum residue ϵ for all attributes. However, it is possible to use a different maximum residue for each attribute. Suppose that we are analyzing a dataset where the attributes have different range values. In this case, we could use a different value of ϵ for each column, or we could normalize / scale the domain of the attributes and use the same value of ϵ for all of them. Another example of bicluster's definition that meets the monotonicity and anti-monotonicity properties is the OPSM [14]. On the other hand, among the definitions of bicluster that do not meet these two properties are the ones based on correlation measures, such as Pearson or Spearman (see the proposal of Flores *et al.* [36] for an example). In this case, a bicluster (I, J) that meets a correlation threshold can have submatrices (I', J') that do not meet the correlation threshold, thus violating these two crucial properties used in the enumerative algorithms of FCA and FPM areas.

One of the first shifting biclusters definitions, which is based on the mean squared residue (MSR) [23], is also an example of definition that does not meet the monotonicity and anti-monotonicity properties. The MSR is given by

$$H(I, J) = \frac{1}{|I||J|} \sum_{i \in I, j \in J} (a_{ij} - a_{iJ} - a_{Ij} + a_{IJ})^2, \tag{2.12}$$

where

$$a_{iJ} = \frac{1}{|J|} \sum_{j \in J} a_{ij}, a_{Ij} = \frac{1}{|I|} \sum_{i \in I} a_{ij}, \text{ and } a_{IJ} = \frac{1}{|I||J|} \sum_{i \in I, j \in J} a_{ij}.$$

Biclustering algorithms that are based on this definition look for biclusters (I, J) where $H(I, J) \leq \delta$, where δ is a user-defined parameter. The biclusters are called δ-biclusters. The problem with this definition is that a bicluster (I, J) that is a δ-bicluster (for some value of δ specified by the user) can have submatrices (I', J'), where $I' \subseteq I$ and $J' \subseteq J$, that are not δ-biclusters. Only when we use $\delta = 0$, we will have that all submatrices (I', J') of a δ-bicluster (I, J) are also δ-biclusters. For this reason, we can only establish a one-to-one correspondence between our definition and this one when we are talking about perfect biclusters.

Property 2.3. *If there is a bicluster (I, J) with maximum residue ϵ in the data matrix* $\mathbf{A}$*, then there is a bicluster (I', J'), where $I' \supseteq I$ and $J' \supseteq J$, with maximum residue*

$\epsilon' \geq \epsilon$ *in the data matrix* $\mathbf{A}$.

Property 2.4. *Let* $\mathfrak{B}$ *be an enumerative bicluster solution without restrictions on the minimum number of rows and columns of the biclusters. In this case, we have* $cov(\mathfrak{B}) = 100\%$.

Property 2.4 is supported by the fact that every single element of matrix $\mathbf{A}_{n \times m}$ is a valid bicluster by itself.

Property 2.5. *Let* $\mathfrak{B}_\epsilon$ *be an enumerative bicluster solution with maximum perturbation* ϵ, *and* $\mathfrak{B}_{\epsilon'}$ *be an enumerative bicluster solution with maximum perturbation* ϵ', *where* $\epsilon > \epsilon'$. *Both,* $\mathfrak{B}_\epsilon$ *and* $\mathfrak{B}_{\epsilon'}$, *share the same restrictions on the minimum number of rows and columns of the biclusters. In this case,* $span(\mathfrak{B}_\epsilon) \supseteq span(\mathfrak{B}_{\epsilon'})$, *and therefore* $cov(\mathfrak{B}_\epsilon) \geq cov(\mathfrak{B}_{\epsilon'})$.

Property 2.5 states that the coverage is a monotonic function with respect to the parameter ϵ. However, it does not indicate that the number of biclusters will always increase with ϵ. With an increase in ϵ, new biclusters tend to be found, but it is not always valid. For example, two biclusters found with $\epsilon = x$ can be merged into one with some $\epsilon > x$. In fact, if ϵ is high enough, the entire dataset will be considered a single valid bicluster.

2.4 Chapter Overview

This chapter presented the main concepts of biclustering, which is a local approach for clustering and overcomes many limitations of the traditional clustering techniques . Briefly, a bicluster is a submatrix of the data matrix that meets a certain homogeneity criterion. In respect of this homogeneity criterion, we define the five main types of biclusters, and show the links between them. We also outlined some important properties with respect to our bicluster definitions, and define the concept of maximal bicluster. Bicluster metrics and indices was also presented to facilitate the reading of this thesis.

In the next chapter, we will provide an overview about Formal Concept Analysis (FCA), and its correlated areas. We also will present the In-Close2 algorithm, which we generalized to develop our family of enumerative biclustering algorithms.

3 Formal Concept Analysis

Formal Concept Analysis (FCA) is a field of applied mathematics based on mathematical order theory, in particular on the theory of complete lattices. Here, we will explain the basic principles of FCA. For more details refer to Ganter and Wille [39].

Definition 3.1 (Formal Context)**.** *A* formal context *is a triple* (G, M, I) *of two sets* G *and* M*, and a relation* $I \subseteq G \times M$*. Each* $g \in G$ *is interpreted as an object, and each* $m \in M$ *is interpreted as an attribute. In order to express that an object* g *is in a relation* I *with an attribute* m*, we write* $(g, m) \in I$ *or* gIm*. We read it as "the object* g *has the attribute* m*".*

Notice that a formal context can be easily represented by a binary matrix $\mathbf{D}$, where rows represent objects, and columns represent attributes. We will have $d_{gm} = 1$ if the object g has the attribute m, and we will have $d_{gm} = 0$ otherwise. Table 3 shows an example of a formal context represented by a binary matrix.

For a subset $A \subseteq G$ of objects, we define

$$A' = \{m \in M | \forall g \in A : gIm\} \tag{3.1}$$

as the set of attributes common to the objects in A. Similarly, for a subset $B \subseteq M$, we define:

$$B' = \{g \in G | \forall m \in B : gIm\} \tag{3.2}$$

as the set of objects common to the attributes in B.

Definition 3.2 (Formal Concept)**.** *A* formal concept *of the formal context* (G, M, I) *is a pair* (A, B) *with* $A \subseteq G$*,* $B \subseteq M$*,* $A' = B$*, and* $B' = A$*. The subset* A *of a formal concept* (A, B) *is called* extent*, and the subset* B *is called* intent*.*

Table 3 – Example of a formal context with a formal concept highlighted.

	m_1	m_2	m_3	m_4	m_5	m_6	m_7
g_1	0	0	1	1	1	0	1
g_2	0	0	1	0	0	0	0
g_3	1	1	1	1	1	1	1
g_4	1	0	0	0	1	1	1
g_5	1	0	1	0	0	1	0
g_6	0	0	1	1	0	0	1

By the definition, we see that though many subsets A can generate the same subset B, only the largest (closed) subset A is part of a formal concept (and vice versa). See the example of the formal concept highlighted in Table 3. The common attributes to the objects g_1, g_3, and g_6 are: m_3, m_4, and m_7. In turn, the common objects to the attributes m_3, m_4, and m_7 are: g_1, g_3, and g_6.

The set of all formal concepts of the context (G, M, I) is denoted by $\mathfrak{B}(G, M, I)$.

Proposition 3.1. *If (G, M, I) is a formal context, $A, A_1, A_2 \subseteq G$ are sets of objects and $B, B_1, B_2 \subseteq M$ are sets of attributes, then*

1. $A_1 \subseteq A_2 \Rightarrow A_2' \subseteq A_1'$,
2. $A \subseteq A''$,
3. $A' = A'''$,
4. $B_1 \subseteq B_2 \Rightarrow B_2' \subseteq B_1'$,
5. $B \subseteq B''$,
6. $B' = B'''$, *and*
7. $A \subseteq B' \Leftrightarrow B \subseteq A' \Leftrightarrow A \times B \subseteq I$.

This proposition shows that the two derivation operators $\{(.)', (.)'\}$ form a *Galois connection* between the power sets of G and M [39]. For the proof of the proposition, refer to Ganter and Wille [39].

Definition 3.3 (Galois Connection)**.** *Let $\varphi : P \to Q$ and $\psi : Q \to P$ be maps between two ordered sets $(P, \leq)$ and $(Q, \leq)$. Such a pair of maps is called a* Galois Connection *between the ordered sets if:*

1. $p_1 \leq p_2 \Rightarrow \varphi p_1 \geq \varphi p_2$
2. $q_1 \leq q_2 \Rightarrow \psi q_1 \geq \psi q_2$
3. $p \leq \psi\varphi p$ *and* $q \leq \varphi\psi q$

The two maps then are called dually adjoint to each other.

For every set $A \subseteq G$, A' is an intent of some formal concept, since (A'', A') is always a formal concept. A'' is the smallest extent containing A. Consequently, a set $A \subseteq G$ is an extent if and only if $A = A''$. The union of extents generally does not result in an extent, but the intersection of any number of extents is always an extent. The same applies to intents [39].

Formal concepts are partially ordered by $(A_1, B_1) \leq (A_2, B_2) \Leftrightarrow A_1 \subseteq A_2 (\Leftrightarrow B_2 \subseteq B_1)$. With respect to this partial order, the set of all formal concepts forms a complete lattice called the *concept lattice* of the formal context (G, M, I).

There are several algorithms in the literature that are able to extract the concept lattice of a formal context. Some examples are: Close-by-One (CbO) [65], In-Close [8], In-Close2 [9], and FCbO [62]. As the algorithms that we will propose are based on In-Close2, Subsection 3.1 is devoted to its formalization.

3.1 The main aspects of In-Close2 algorithm

In-Close2 [9] and its precursor In-Close [8] are based conceptually on Close-By-One [65]. These algorithms use the lexicographic approach for mining formal concepts, thus avoiding the discovery of the same formal concept multiple times. In-Close2 [9] is a computationally faster version of In-Close [8]. To achieve a better performance than the one produced by In-Close, In-Close2 (*i*) allows all elements of an intent to be inherited, and (*ii*) implements some optimization and data preprocessing techniques for efficient use of cache memory. This second improvement is only applicable to binary matrices, therefore we will not use it here.

Ganter [37] showed how the lexicographical order of concepts can be used to avoid the search of repeated results. In the mathematical order theory, combinations have a lexicographical order, for instance, {1, 2, 3} comes before {1, 2, 4}, and also before {1, 3} [8]. In-Close and In-Close2 maintain a *current attribute*. The concept next generated is new (*canonical*) if its intent contains no attribute preceding the current attribute. Therefore, to verify canonicity, In-Close/In-Close2 does the following: supposing that B is the current intent, j is the current attribute, and RW is the resulting extent, RW is not canonical if

$$\exists k \in M \setminus B | [k < j] \ \wedge \ [\forall g \in RW : gIk]. \tag{3.3}$$

i.e., if there is an attribute $k < j$ where $k \notin B$ and $RW \subseteq \{k\}'$. See Eq. 3.2 for a definition of $\{k\}'$. The concept of canonicity was introduced in Kuznetsov [66].

Algorithm 3.1 shows In-Close2 pseudocode. It is an adaptation of the pseudocode proposed by Andrews [9] in order to facilitate the reading and the understanding of the algorithm In-Close2. When we use A_r and B_r, it means the extent and the intent of the r-th formal concept, respectively. When we write J_k it means the element of the set J at position k, for instance, if $J = \{2, 5, 7, 8, 13\}$ and $k = 2$, $J_2 = 5$. The same for R_k.

In a main function, we set $(A_1, B_1) \leftarrow (\{1, ..., n\}, \{\})$ (which is called *supremum*), and $r_{new} \leftarrow 1$. Then, we call the function In-Close2 to incrementally close the formal

concept (A_1, B_1), starting at attribute index 1. Thereafter, all formal concepts will be found recursively. During the closure of a formal concept, In-Close2 iterates across the attributes (line 3). If the current attribute j is not an inherited attribute (line 4), In-Close2 computes the candidate to a new extent RW (line 5). The size of RW must be greater than or equal to the value of the parameter $minRow$ (line 6). If the extent RW is the same as the current extent A_r (line 7), then attribute j is added to the current intent B_r (line 8). If the extent RW is not the same as the current extent A_r, In-Close2 tests if RW is canonical (line 9). If yes, the current formal concept (A_r, B_r) will give rise to a child formal concept (lines 10 to 13). After the closure of the current formal concept (A_r, B_r), In-Close2 starts to close its children (lines 19 to 22).

Algorithm 3.1 In-Close2 (adapted from [9])

Input: Binary data matrix $\mathbf{D}_{n\times m}$, minimum number of rows $minRow$, index of the formal concept to be closed r, index of the initial attribute y

```
 1: J ← {}
 2: R ← {}
 3: for j ← y to m do
 4:    if j ∉ B_r then
 5:       RW ← A_r ∩ {j}'
 6:       if |RW| ≥ minRow then
 7:          if |RW| = |A_r| then
 8:             B_r ← B_r ∪ {j}
 9:          else if RW is canonical then
10:             r_new ← r_new + 1 // global variable
11:             J ← J ∪ {j}
12:             R ← R ∪ {r_new}
13:             A_{r_new} ← RW
14:          end if
15:       end if
16:    end if
17: end for
18: Store (A_r, B_r) in the solution 𝔅
19: for k ← 1 to |J| do
20:    B_{R_k} ← B_r ∪ {J_k} // R_k and J_k are the k-th elements of R and J, respectively
21:    In-Close2(D, minRow, R_k, J_k + 1)
22: end for
```

The worst-case processing time of In-Close2 is $O(knm^2)$, where k is the number of formal concepts. The difference between In-Close and In-Close2 pseudocodes is just that in In-Close the recursive call happens after the test of canonicity. Therefore the descendant formal concepts inherits only the current attributes of the parent.

If $minRow = 1$, In-Close2 mines the concept lattice of the formal context represented by the binary matrix $\mathbf{D}$. Otherwise, if $minRow > 1$, In-Close2 mines the set of all *frequent concepts* for the threshold $minRow$, called the *iceberg concept lattice*. The parameter $minRow$ helps to prune the search space, together with the verification of canonicity.

Table 4 – Closure of the supremum formal concept (A_1, B_1) for the data in Table 3.

	RW	Situation
$j = 1$	$\{3, 4, 5\}$	is canonical
$j = 2$	$\{3\}$	$\|RW\| < minRow$
$j = 3$	$\{1, 2, 3, 5, 6\}$	is canonical
$j = 4$	$\{1, 3, 6\}$	is not canonical
$j = 5$	$\{1, 3, 4\}$	is canonical
$j = 6$	$\{3, 4, 5\}$	is not canonical
$j = 7$	$\{1, 3, 4, 6\}$	is canonical

A candidate to a new extent RW does not even start to be an actual extent of a formal concept when it fails to meet both conditions.

In addition to the minimum number of rows $minRow$, we can easily add a minimum number of columns $minCol$ to In-Close2. While In-Close2 loops through the attributes, a formal concept (A_r, B_r) can be discarded if, even adding all remaining attributes to its intent, it will not meet the minimum number of columns $minCol$ (therefore, its next descendants will not meet the minimum number of columns $minCol$ as well). Although this restriction can be checked only during the closure of a formal concept, it will also prune the search space and save computational resources because (i) it stops the construction of a formal concept that will be discarded later, given that it does not meet the restriction $minCol$, and (ii) it avoids generating descendants that will not meet the restriction $minCol$ as well.

3.1.1 Example of In-Close2 Operation

Now, we will provide an example to illustrate the In-Close2 operation. Let $\mathbf{D}$ be the binary matrix in Table 3. The supremum is initialized as $(A_1 = \{1, 2, 3, 4, 5, 6\}, B_1 = \{\})$. Supposing that $minRow = 2$, Table 4 shows all the extents RW computed during the closure of the supremum formal concept (A_1, B_1) (lines 3 to 17 of Algorithm 3.1). Thus, the descendants of (A_1, B_1) are:

- $(A_2 = \{3, 4, 5\}, B_2 = \{1\})$,
- $(A_3 = \{1, 2, 3, 5, 6\}, B_3 = \{3\})$,
- $(A_4 = \{1, 3, 4\}, B_4 = \{5\})$, and
- $(A_5 = \{1, 3, 4, 6\}, B_5 = \{7\})$.

The next step of In-Close2 is to close the descendants of (A_1, B_1) recursively (lines 19 to 22 of Algorithm 3.1).

Table 5 – Closure of the formal concept (A_2, B_2) for the data in Table 3.

	RW	Situation
$j = 2$	$\{3\}$	$\|RW\| < minRow$
$j = 3$	$\{3, 5\}$	is canonical
$j = 4$	$\{3\}$	$\|RW\| < minRow$
$j = 5$	$\{3, 4\}$	is canonical
$j = 6$	$\{3, 4, 5\}$	$RW = A_2$
$j = 7$	$\{3, 4\}$	is not canonical

Table 6 – Formal concepts of the binary data matrix of Table 3 (using $minRow = 2$ and $minCol = 1$).

#	Extent	Intent
1	$\{3, 4, 5\}$	$\{1, 6\}$
2	$\{1, 2, 3, 5, 6\}$	$\{3\}$
3	$\{1, 3, 4\}$	$\{5, 7\}$
4	$\{1, 3, 4, 6\}$	$\{7\}$
5	$\{3, 5\}$	$\{1, 3, 6\}$
6	$\{3, 4\}$	$\{1, 5, 6, 7\}$
7	$\{1, 3, 6\}$	$\{3, 4, 7\}$
8	$\{1, 3\}$	$\{3, 4, 5, 7\}$

To illustrate, let's compute the closure of the formal concept (A_2, B_2). Table 5 shows all the extents RW computed during its closure. After its closure, the formal concept (A_2, B_2) is equal to $(\{3, 4, 5\}, \{1, 6\})$. Remember that the descendants inherit the columns in the intent of their parent. Thus, the descendants of (A_2, B_2) are:

- $(A_6 = \{3, 5\}, B_6 = \{1, 6, 3\})$, and
- $(A_7 = \{3, 4\}, B_7 = \{1, 6, 5\})$.

The next step of In-Close2 is to closure the descendants of (A_2, B_2).

This process continues until In-Close2 computes the closure of all formal concepts satisfying $minRow$. The final result is presented in Table 6.

3.2 Related areas of research in the literature

The problem of extracting the concept lattice from a formal context is the same as extracting all maximal CTV biclusters of ones from a binary data matrix [59]. Table 3 shows an example of a formal context with a formal concept highlighted. As we can see, a formal concept is a maximal CTV bicluster of ones. Extent A and intent B are the set of rows (objects) and columns (attributes) that compose a bicluster, respectively.

The association mining problem is also closely related to FCA. This problem is divided in two sub-problems: (*i*) the frequent itemset (pattern) mining problem, and (*ii*) the problem of mining the association rules (see Subsection 3.2.1 for the basic concepts of association rules) from these itemsets. As the first sub-problem of association mining is the most computationally expensive, almost all researches have been focused on the frequent itemset generation phase.

In terms of FCA, the problem of mining all *frequent itemsets* (patterns) can be described as follows. Given a formal context (G, M, I), determine all subsets $B \subseteq M$ such that the support of B (see Subsection 3.2.1 for the definition of support) is above a user-defined parameter [68]. Examples of algorithms that perform this task are Apriori [3] and Eclat [110].

To reduce the computational cost of the frequent pattern mining problem, some algorithms, such as GenMax [55], mine only the *maximal frequent itemsets*, i.e., those frequent itemsets from which all supersets are infrequent and all subsets are frequent. The problem of this approach is that it leads to a loss of information since the supports of the subsets are not available.

An option to reduce the computational cost without loss of information is to mine only the *closed frequent itemsets*. A frequent itemset B is called closed if there exists no superset $D \supset B$ with $B' = D'$, i.e., a closed frequent itemset is the intent of a formal concept. The closed frequent itemsets are also called *frequent concept intents*. For any itemset B, its concept intent is given by B''. Note that this approach is the most closely related to FCA. Remarkably, a concept lattice contains all necessary information to derive the support of all (frequent) itemsets [68]. Indeed, the set of closed frequent itemsets uniquely determines the exact frequency of all itemsets, and it can be orders of magnitude smaller than the set of all frequent itemsets [109]. Moreover, this approach drastically reduces the number of rules that have to be presented to the user, without any information loss [68]. CHARM [109] is a well-known algorithm to mine all closed frequent itemsets. It exploits the fact that the extents of the formal concepts are irrelevant in the frequent pattern mining problem (just the intents and the cardinality of the extents are relevant). Thus, it drastically cuts down the size of memory required [109].

To exemplify the difference between frequent itemset, maximal frequent itemset and closed frequent itemset, let us use the data matrix of Table 3 and assume that the minimum support is equal to 2. The intent of the formal concept highlighted, $\{m_3, m_4, m_7\}$, is a closed frequent itemset. $\{m_3, m_4, m_5, m_7\}$ is a maximal frequent itemset and all its subsets are frequent itemsets. Figure 3 exemplifies this relation.

The problem of enumerating all maximal bicliques from a bipartite graph is also closely related to FCA. Madeira and Oliveira [79] stated that in the simplest case of biclustering, where we are looking for CTV biclusters of ones in a binary data matrix

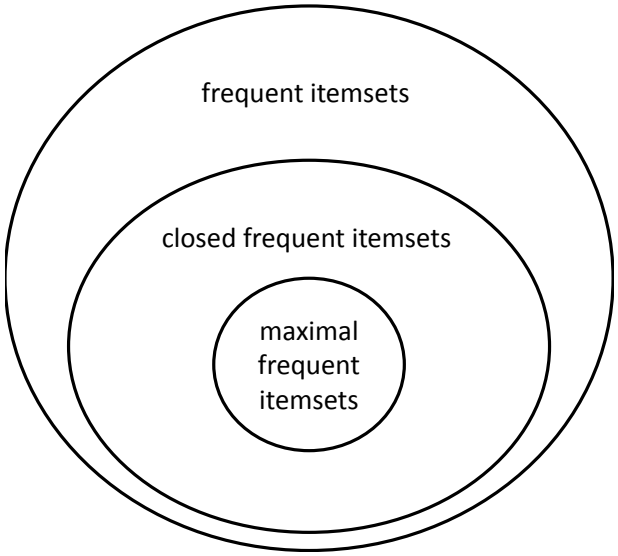


Figure 3 – The relation between frequent itemsets, closed frequent itemsets, and maximal frequent itemsets.

D, a bicluster corresponds to a biclique in the corresponding bipartite graph. Rows and columns of the matrix **D** correspond, respectively, to the first and second sets of vertices of a bipartite graph. For example, in Table 3, the first set of vertices is given by $\{g_1, g_2, g_3, g_4, g_5, g_6\}$, and the second one is given by $\{m_1, m_2, m_3, m_4, m_5, m_6, m_7\}$. Each element d_{ij} is equal to 1 if vertex i is connected to vertex j, and 0 otherwise. In this scenario, a formal concept from the binary matrix **D** is equivalent to a maximal biclique (bicluster). So, finding a concept lattice is also equivalent to finding all maximal bicliques of a bipartite graph. The connection between FCA and the problem of enumerating all maximal bicliques from a bipartite graph is explored in several papers [1, 40, 41]. Moreover, Gély *et al.* [41] pointed out several algorithms to find all maximal bicliques from a bipartite graph, most of them are from the area of FCA.

For an undirected graph without self-loops, each maximal biclique is generated twice if we apply FCA algorithms, i.e., each maximal biclique corresponds to a pair of formal concepts [72]. In fact, Li *et al.* [72] proved the correspondence between maximal bicliques and closed itemsets. However, the authors did not realize that a closed itemset is the intent of a formal concept. Li *et al.* [72] also modified the LCM [101] (an algorithm for finding closed frequent itemsets) to develop the LCM-MBC algorithm, tailored to produce a non-redundant enumeration of all maximal bicliques. For an undirected graph with self-loops, the problem of using FCA is that it may happen that the extent A and the intent B of a formal concept (A, B) will not be disjoint , i.e., $A \cap B \neq \emptyset$. But in the graph literature, we can find specialized algorithms that exploit the specific nature of particular classes of graphs, leading to highly efficient solutions [6]. For example, Eppstein's algorithm [34] is a linear time algorithm for the class of graphs with bounded arboricity.

We also want to emphasize that it may be possible to adapt FCA algorithms to find all maximal cliques of an undirected graph. A clique is a subset of vertices of an undirected graph, such that every two distinct vertices in the clique are adjacent. As FCA algorithms looks for blocks of ones, the adjacency matrix must necessarily have all its diagonal elements equal to 1 (even if the graph does not have self-loops). With this, every formal concept whose extent is equal to the intent represents a clique of the undirected graph. However, there are more efficient methods to enumerate all maximal cliques in the graph theory literature [20].

3.2.1 Basic concepts of association rules

Let (G, M, I) be a formal context, where G is a set of objects, M is a set of items (or attributes), and $I \subseteq G \times M$ is a binary relation between the objects G and the attributes M.

Definition 3.4. *A set $B = \{m_1, m_2, ..., m_k\} \subseteq M$ is called an* itemset *(or a k-*itemset *since it contains k items).*

The *support* of an itemset B is given by

$$sup(B) = |B'|, \tag{3.4}$$

where B' is defined in Eq. 3.2 and $|\zeta|$ is the number of elements in the set ζ. The *relative support* or *frequency* of an itemset B is given by

$$rsup(B) = \frac{sup(B)}{|G|}. \tag{3.5}$$

Definition 3.5. *An itemset B is a* frequent itemset *if $sup(B) \geq minSup$, where $minSup$ is a user-defined threshold.*

Definition 3.6. *An* association rule *is an expression of the form $X \Rightarrow Y$, where X and Y are itemsets and they are disjoint, i.e., $X, Y \subseteq M$ and $X \cap Y = \{\}$. X is called the* body *or* antecedent*, and Y is called the* head *or* consequent *of the rule.*

The *support* of a rule $X \Rightarrow Y$ is given by the number of objects common to the itemset $X \cup Y$:

$$sup(X \Rightarrow Y) = sup(X \cup Y) = |(X \cup Y)'|. \tag{3.6}$$

Thus, the *relative support* of a rule $X \Rightarrow Y$ provides an estimate of the joint probability of the occurrence of X and Y:

$$rsup(X \Rightarrow Y) = \frac{sup(X \cup Y)}{|G|} = P(X \wedge Y). \tag{3.7}$$

The *confidence* of a rule $X \Rightarrow Y$ is the conditional probability that an object contains the items Y given that it contains the items X:

$$conf(X \Rightarrow Y) = P(Y|X) = \frac{P(X \wedge Y)}{P(X)} = \frac{sup(X \cup Y)}{sup(X)}. \tag{3.8}$$

Definition 3.7. *A rule $X \Rightarrow Y$ is a* frequent rule *if $sup(X \Rightarrow Y) \geq minSup$, where $minSup$ is a user-defined threshold.*

Definition 3.8. *A rule $X \Rightarrow Y$ is a strong rule if $conf(X \Rightarrow Y) \geq minConf$, where $minConf$ is a user-defined threshold.*

Definition 3.9. *A rule $X \Rightarrow Y$ is a* class association rule (CAR) *if its consequent Y is a 1-itemset $Y = \{c\}$, where c is a class label. We use $X \Rightarrow c$ instead of $X \Rightarrow \{c\}$ for simplicity.*

The completeness of a CAR $X \Rightarrow c$ is given by

$$comp(X \Rightarrow c) = \frac{sup(X \cup c)}{sup(c)}. \tag{3.9}$$

Thus completeness is the proportion of instances that are predicted by the CAR $X \Rightarrow c$, while confidence is the fraction of correct predictions made by the rule.

3.3 Chapter Overview

This chapter presented the main concepts of FCA, and also described its connections with biclustering, association mining, and graph theory. It also presented the algorithm In-Close2 and gave an example of its operation. Our family of enumerative biclustering algorithms, RIn-Close, are conceptually based on In-Close2.

In the next chapter, we will provide a literature review and face RIn-Close algorithms with their competitors.

4 Related Works

Due to the inherent computational complexity of the problem of finding all maximal biclusters, most of the proposed algorithms are heuristic-based [79]. Some relevant heuristics in the literature are: CC [23], FLOC [107], ROCC [32], ISA [52], Plaid [69], and OPSM [14]. CC looks for biclusters with mean squared residue (MSR) below a user-defined threshold δ. It mines one bicluster at each iteration, and performs random perturbations to the data to mask the already discovered biclusters. FLOC is also based on the MSR, but performs simultaneous bicluster identification. Briefly, the goal of CC and FLOC is to mine a set of biclusters with high average volume given the residue limit δ. ROCC is scalable and very versatile because it can be parametrized to mine several types of biclusters. It works in two steps: (i) find $k \times l$ biclusters arranged in a grid structure, keeping only the sr rows and sc columns with the lowest residue associated with them, and (ii) filter out the biclusters with the largest residue values, and merge similar biclusters. ISA looks for biclusters where their rows have an average value above a threshold t_g, and their columns have an average value above a threshold t_c. ISA starts with $nseed$ biclusters, and iteratively updates the columns and rows of the biclusters until convergence. Plaid fits parameters of a generative model of the data iteratively by minimizing the mean squared error between the modeled data and the true data. OPSM is a deterministic greedy algorithm dedicated to find large OPSM biclusters.

Clearly, even the best heuristics potentially lead to sub-optimal solutions, thus motivating the existence of many proposals of exhaustive bicluster enumeration, guiding to the optimal solution. Most of the work in this area is designed to mine all maximal CTV biclusters of ones from a binary dataset. In FCA, FPM and graph theory, there are several efficient algorithms able to perform this task. Proposals such as [49, 80, 82, 87, 88, 97] binarize the dataset and use FCA, FPM or graph theory algorithms to enumerate the biclusters. To avoid the loss of information of tedious Boolean property encoding phases [16], many proposals deal directly with numerical datasets, as the ones to be described in what follows.

The proposals in [11, 16, 58, 59] are dedicated to enumerate CTV biclusters from numerical datasets.

The RCB algorithm [11] is based on an FPM algorithm called Apriori [4], which has a worst-case time exponential on the number of attributes. Thus, Apriori and the algorithms based on it are not efficient. It is also noteworthy that Apriori mines frequent itemsets, not closed frequent itemsets. Thus, it produces many redundant biclusters. RCB adopts a two step process. First, all the square submatrices that qualify to be a CTV

bicluster are enumerated. Second, these square CTV biclusters are merged to form rectangular CTV biclusters of arbitrary sizes.

The NBS-Miner algorithm [16] mines all maximal CTV biclusters of a numerical dataset. The algorithm starts with the lattice $((\{\}, \{\}), (G, M))$ (whose bottom is $(\{\}, \{\})$ and top is (G, M)), i.e, the lattice containing all possible biclusters. Then, NBS-Miner explores its sublattices using three functions: enumeration, pruning, and propagation. The enumeration function splits recursively the current sublattice into two new sublattices. The prune function is responsible for pruning the sublattices that do not attend to the restriction of similarity (imposed by ϵ, see Eq. 2.1) or maximality. The propagation function implements the reduction of the size of the search space of a sublattice, not considering the entire current sublattice. The algorithm finds a bicluster when it finds a sublattice whose top is equal to the bottom.

Kaytoue *et al.* [59] proposed two FCA-based methods to enumerate CTV biclusters. The former is based on the discretization of the numerical data matrix using *conceptual scaling* [39]. Let W be the set of values that an object $g \in G$ can take for an attribute $m \in M$. First of all, they compute all *tolerance classes* [56] from W. Then, they create one formal context for each class of tolerance and use FCA standard algorithms to enumerate the formal concepts from them. Each formal concept corresponds to a maximal CTV bicluster. The formal contexts are created in a way that avoids finding redundant CTV biclusters, but at a cost of not finding some biclusters. Since the resulting binary tables may be aplenty depending on the number of elements of W and the parameter ϵ, this method is not feasible in many real-world scenarios. The second method is divided in two phases. In the first one, it enumerates all the CVC biclusters using *interval pattern structures* (IPS) [38]. It is noteworthy that this method returns redundant CVC biclusters. It happens because the pattern concepts are maximal by definition, but we lose this property when we convert a pattern concept in a CVC bicluster. In the second phase, CTV biclusters are extracted from the CVC biclusters, but this process is not so clear, because a CVC bicluster can give rise to many CTV biclusters.

In [58], the authors also use tolerance classes over the set of numbers W, and create one formal context for each class of tolerance. They proposed a new algorithm called TriMax to mine the CTV biclusters from these formal contexts. TriMax is able to perform a complete, correct and non-redundat enumeration of all maximal CTV biclusters in a numerical dataset. But due to the scaling process, TriMax may be not feasible in many real-world scenarios too.

In the next two subsections, we highlight the competitors of the algorithms that we are proposing, and compare them with RIn-Close algorithms. This comparison was based on a theoretical study of the competitors, and we also tested the implementations that were made available by the authors.

Table 7 – Comparison of RIn-Close_CVC_P, RIn-Close_CVC and their competitors.

	Complete	Correct	Non-Redundant	Efficient
RIn-Close_CVC_P	✓	✓	✓	✓
RIn-Close_CVC	✓	✓	✓	✓
RAP	✓	✓		
PPS	∘	✓	✓	
IPS	✓	✓		
TCA	✓	✓		

The symbol ✓ indicates that the algorithm has the property. The symbol ∘ indicates that the authors claim their algorithm has the property, but it fails to exhibit the property.

4.1 Enumerating CVC (or CVR) biclusters

RAP [91] is an algorithm created to mine CVC biclusters that is also based on Apriori [4]. The authors did not describe their strategy to avoid redundancy, but we conjecture that the best that can be done is a pairwise comparison of biclusters with k and $k+1$ columns.

In [26, 27], the authors proposed a method based on *partition pattern structure* (PPS) [12]. Their proposal is not able to perform a complete enumeration because the components of the partition of the set G must be disjunct. They proposed a strategy based on a lattice to remove the redundancy, which is much faster than to compare one bicluster against all the others. But it is necessary to mine the redundant biclusters to make this verification, so it is not an efficient method.

In [57], the authors revisited the proposals of mining CVC biclusters using IPS [59] and PPS [26, 27]. They also proposed an approach based on Triadic Concept Analysis (TCA) [71]. From a numerical dataset, they derivate a triadic context given by (M, G, G, Y) such that $(m, g_1, g_2) \in Y$ iif $|d_{g_1m} - d_{g_2m}| \leq \epsilon$. Then, they use standard TCA algorithms to enumerate the triadic concepts, but not all triadic concepts are maximal CVC biclusters.

Table 7 shows a conceptual comparison of these proposals and the algorithms that we are proposing in Chapter 5 to enumerate CVC biclusters. Notice that our proposals are the only algorithms that are able to perform an efficient, complete, correct and non-redundant enumeration of all maximal CVC biclusters.

4.2 Enumerating CHV biclusters

pCluster [106] was the first deterministic algorithm with an enumerative approach to mine CHV biclusters. pCluster computes all row-maximal biclusters with two columns

and all column-maximal biclusters with two rows, prunes the unpromising biclusters, and stores the remaining column-maximal biclusters in a prefix tree. Then, pCluster makes a depth-first search in this prefix tree in order to mine larger biclusters. pCluster has the following shortcomings: it does not find all biclusters, can find biclusters that do not attend the user-defined measure of similarity, and returns redundant biclusters. Furthermore, pCluster's computational complexity is exponential w.r.t. the number of attributes.

Maple [95] is an improved version of pCluster and it is closer to an actual enumerative algorithm. It returns only non-redundant biclusters, but it does not have an efficient approach to do this: for each possible new bicluster, Maple must look at all previously generated biclusters to avoid redundancy. Besides, there are two scenarios where Maple fails to perform a complete and correct enumeration of all maximal biclusters. If two biclusters have the same set of objects and share some attributes, Maple would return only one bicluster containing both of them (thus violating the user-defined measure of similarity). Maple also may miss biclusters due to its routine of pruning unpromising biclusters: Maple keeps an attribute-list ordered by some criterion. If a bicluster has a subset of objects and a superset of attributes of another bicluster, and its extra attributes are subsequently considering Maple's attribute-list, Maple would prune it incorrectly. The worst-case time of Maple's search is also exponential w.r.t. the number of attributes.

MicroCluster [111] constructs a multigraph that represents all row-maximal biclusters with two columns, where nodes represent attributes and edges represent sets of objects. It uses a depth-first search on the multigraph to mine the biclusters. We tested the authors' MicroCluster implementation[1], and we observed that MicroCluster can fail to enumerate all maximal biclusters and can return biclusters that violate the user-defined measure of similarity. Its worst-case time is also exponential w.r.t. the number of attributes. As Maple, MicroCluster must look at all previously generated biclusters to avoid redundancy, which is always a strategy to be avoided due to the high computational cost. After mining a CHV bicluster, MicroCluster verifies if its columns and rows attends user-defined measures of similarity too, in an attempt to mine biclusters that are at the same time CHV, CVC, and CVR biclusters. If this verification fails, the bicluster is discarded. Note that this verification can be done in any biclustering solution.

To reduce computational cost, algorithms such as SeqClus [105] and CPT [44] relaxed the definition of CHV biclusters. Instead of looking for every pair of attributes (or objects), they use a pivot attribute to compute the CHV biclusters. Thus, given a user-defined parameter ϵ, their biclusters will have residue below or equal to 2ϵ (see Lemma 4.1). The biclustering solution of these algorithms depends on the choices of the pivot attributes. So, we can say that this strategy yields an approximate result for the actual enumeration. A genuine complete and correct enumeration of CHV biclusters would

[1] `http://www.cs.rpi.edu/~zaki/www-new/pmwiki.php/Software/Software`

provide the same solution when dealing with the original matrix or with its transpose version. The CHV biclusters are fully preserved when rows become columns and columns become rows of the matrix, see Lemma 2.3. But unfortunately, when resorting to the pivot attribute, those techniques are prone to lose this fundamental property. Accordingly, these algorithms are not able to perform a complete and correct enumeration. As SeqClus and CPT are based on algorithms to mine frequent itemset, not closed frequent itemsets, they return redundant biclusters and are not efficient to mine maximal biclusters. So, this idea of mining CHV biclusters using a pivot attribute could have been better explored. In fact, we could have used this idea in RIn-Close_CHV (as we used in RIn-Close_CHVP because of Lemma 4.1: in this case, if a new candidate attribute is coherent with any attribute of the current bicluster, it will be coherent with all other attributes of the current bicluster), which would lead to an efficient algorithm. But our main goal is to provide an algorithm able to perform a complete and correct enumeration of all maximal CHV biclusters. Moreover, despite our algorithm RIn-Close_CHV does not have polynomial delay, it has good computational performance as we can see with the experimental results in Chapter 6.

Lemma 4.1. *Let (I, J) be a CHV bicluster, $j, l \in J$, and $dist_{jl} = \max(Z^{jl}) - \min(Z^{jl})$. If we have a pivot column $p \in J$ such that $dist_{jp} \leq \epsilon$ and $dist_{lp} \leq \epsilon$, then $dist_{jl} \leq 2\epsilon$.*

Proof.

$$\begin{aligned} dist_{jl} &\leq dist_{jp} + dist_{lp} \\ &\leq 2\epsilon \end{aligned}$$

□

Table 8 shows a conceptual comparison of these proposals and the algorithms that we are proposing in Chapter 5 to enumerate CHV biclusters. Notice that we are proposing algorithms with a number of additional features that are able to support a wider range of application scenarios, including the identification of biological indicators [76] and classification based on associations [74]. Moreover, a bicluster solution provided by these competitors is contained in the RIn-Close's solution, given an adequate parameter ϵ. For instance, if a user parameterizes CPT with $\epsilon = 1$, it would obtain a biclustering solution composed of biclusters with residue up to 2. Parameterizing RIn-Close with $\epsilon = 2$, a user would obtain a biclustering solution containing all the maximal versions of the biclusters of the CPT's solution, and possibly containing additional biclusters.

4.3 Other Considerations

There are some strategies that the RIn-Close family of algorithms and its direct competitors can apply to handle missing values. The simplest one is to remove the rows

Table 8 – Comparison of RIn-Close_CHV_P, RIn-Close_CHV and their competitors.

	Complete	Correct	Non-Redundant	Efficient
RIn-Close_CHV_P	✓	✓	✓	✓
RIn-Close_CHV	✓	✓	✓	
pCluster	○	○		
Maple	○	○	✓	
MicroCluster	○	○	✓	
SeqClus	○	○		
CPT	○	○		

The symbol ✓ indicates that the algorithm has the property. The symbol ○ indicates that the authors claim their algorithm has the property, but it fails to exhibit the property.

or columns (usually the ones with smaller dimension) containing missing values, at the cost of information loss. Another simple strategy is the previous estimation of the missing values using some imputation technique of the literature. The problem with this approach is that it generally will introduce additional noise to the dataset, which may significantly reduce the biclusters' homogeneity, thus promoting unnecessary bicluster partitioning. Essentially, a single large original bicluster with some missing elements may be recovered as dozens of smaller biclusters, possibly with a high overlap among them. Henriques and Madeira [49] proposed the use of additional attributes handled according to a level of relaxation defined by the user. We are going to propose a similar strategy in Chapter 5, adopting its more restrictive version where the missing items are removed (not the entire rows or columns). The advantages of this approach are that it has a low computational cost, avoids information loss, and does not introduce additional noise to the dataset.

The algorithms described here to deal with numerical attributes can promptly be adapted to handle categorical attributes too. For instance, grouping together identical values of a numerical attribute in a CVC bicluster is the same as grouping together identical values of a non-ordinal categorical attribute. Moreover, the case of ordinal categorical attributes can be essentially treated as a numerical attribute, but with discrete values for the parameter ϵ. In both cases, the monotonicity and anti-monotonicity properties are preserved.

4.4 Chapter Overview

This chapter reviewed some well-known heuristic-based biclustering algorithms, together with enumerative biclustering algorithms to mine CTV, CVC (or CVR), and CHV biclusters. It also placed RIn-Close algorithms against its competitors, showing its distinctive aspects and advantages.

We also highlighted how the enumerative biclustering algorithms can handle miss-

ing values and categorical attributes in the data matrix.

In the next chapter, we will present the main contribution of this work: our family of enumerative biclustering algorithms, RIn-Close.

5 Generalizations of In-Close2 to numerical datasets: RIn-Close

In-Close2 has been specifically designed to extract all maximal CTV biclusters of ones from a binary data matrix. Now, we will propose generalizations of In-Close2 to enumerate other types of biclusters from numerical (not only binary, but also integer or real-valued) matrices. We call our family of algorithms RIn-Close.

We chose to adapt In-Close2 because: (*i*) it is easy to understand; (*ii*) it is one of the fastest algorithms of FCA; (*iii*) it has support to the desired minimum number of rows in a bicluster (the parameter $minRow$); (*iv*) it is easy to incorporate a support to the desired minimum number of columns in a bicluster (the parameter $minCol$); and (*v*) In-Close2 starts with all objects in the extent of a formal concept. This latter aspect of In-Close2 is very important when working with integer or real-valued data matrices. For instance, when finding CVC biclusters, given the current attribute, we can look for the subsets of rows of the extent that accomplish the similarity restriction ϵ.

The name of a algorithm of our family is given by the word RIn-Close plus the type of bicluster that it enumerates. For instance, RIn-Close_CVC enumerates all maximal CVC biclusters. If the algorithm is specialized in enumerating perfect biclusters, its name ends with the letter 'P'. For instance, RIn-Close_CVC_P enumerates all maximal perfect CVC biclusters

5.1 Finding biclusters with constant values on columns (CVC biclusters)

The algorithms of this section compute an efficient, complete, correct and non-redundant enumeration of all maximal CVC biclusters. First, we will show how to extract perfect biclusters, because it is the easiest case. After that, given a user-defined parameter $\epsilon > 0$, we will show how to extract non-perfect CVC biclusters with maximum residue ϵ, as defined in Eq. 2.3.

5.1.1 Perfect Biclusters

The adaptation of In-Close2 to enumerate all maximal perfect CVC biclusters, called RIn-Close_CVC_P, is straightforward. We have only one major difference. In In-Close2, each bicluster (A_r, B_r) can generate just one descendant per attribute, whereas in RIn-Close_CVC_P, each bicluster (A_r, B_r) can generate multiple descendants per at-

tribute. It happens because In-Close2 looks for blocks of 1s, whereas RIn-Close_CVC_P looks for any blocks of constant values on columns. Figure 4 illustrates this difference. In Figure 4(a), In-Close2 is closing the bicluster ($A_r = \{g_2, g_3, g_4, g_8, g_9, g_{10}, g_{11}, g_{15}\}, B_r = \{m_1, m_3, m_7\}$). When it tries to add the attribute m_8, bicluster (A_r, B_r) gives rise to a new bicluster ($A = \{g_3, g_4, g_9, g_{15}\}, B = \{m_1, m_3, m_7, m_8\}$). In Figure 4(b), RIn-Close_CVC_P is closing the same bicluster (A_r, B_r), but when it tries to add the attribute m_8, bicluster (A_r, B_r) gives rise to four new perfect CVC biclusters *without overlap between their extents*:

- (d1) ($A = \{g_{11}\}, B = \{m_1, m_3, m_7, m_8\}$),
- (d2) ($A = \{g_2, g_{15}\}, B = \{m_1, m_3, m_7, m_8\}$),
- (d3) ($A = \{g_8, g_9, g_{10}\}, B = \{m_1, m_3, m_7, m_8\}$), and
- (d4) ($A = \{g_3, g_4\}, B = \{m_1, m_3, m_7, m_8\}$).

Algorithm 5.1 shows the pseudocode of RIn-Close_CVC_P. Notice that it is almost the same as In-Close2. There are basically two differences. The first one is that the current attribute j is added to the current intent B_r if all values of attribute j and objects A_r are equal. And the second one occurs by the fact that the bicluster (A_r, B_r) can generate multiple descendants. So, RIn-Close_CVC_P computes all the possible extents and loops across them.

Let A be the current extent, and j be the current attribute, the possible extents are given by

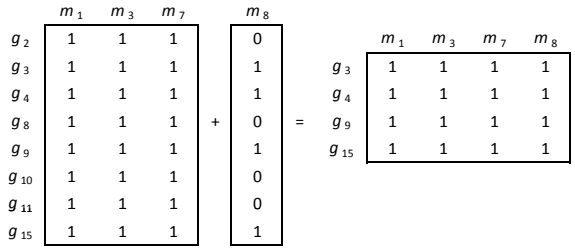
$$\{RW | [RW \subseteq A] \ \wedge \ [\max_{i \in RW}(\{d_{ij}\}) - \min_{i \in RW}(\{d_{ij}\}) \leq \epsilon] \ \wedge \ [RW \text{ is maximal}]\}. \tag{5.1}$$

In the case of mining perfect CVC biclusters, we have $\epsilon = 0$. Note in the example of Figure 4(b) that the elements of the current attribute, m_8, were sorted in order to easily identify all possible new extents.

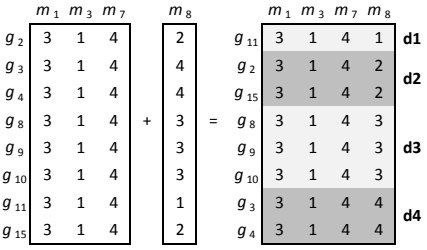
The test of canonicity is also essentially the same as in In-Close2. Let B be the current intent, j be the current attribute, and RW be a possible extent of a bicluster, it is not canonical if

$$\exists k \in M \setminus B | [k < j] \ \wedge \ [\max_{i \in RW}(d_{ik}) - \min_{i \in RW}(d_{ik}) = 0], \tag{5.2}$$

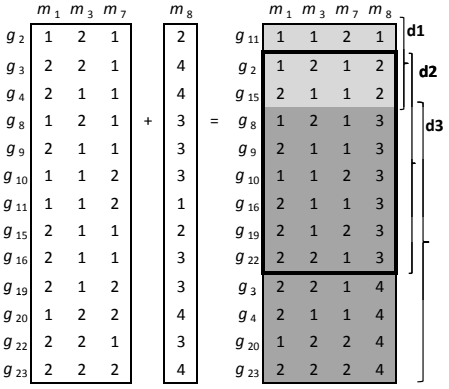
i.e., if there is an attribute $k < j$ that we can add to the bicluster (RW, B) and it remains a valid perfect CVC bicluster.



(a) Generation of a single descendant by In-Close2.



(b) Generation of multiple descendants, d1, d2, d3 and d4, by RIn-Close_CVC_P.



(c) Generation of multiple descendants, d1, d2 and d3, by RIn-Close_CVC (considering $\epsilon = 1$).

Figure 4 – Examples of the generation of descendants by (a) In-Close2, (b) RIn-Close_CVC_P, and (c) RIn-Close_CVC.

The worst-case time of RIn-Close_CVC_P is almost the same as the one for In-Close2: $O(knm(\log n + m))$. The difference is due to the use of a sorting algorithm to compute the possible extents.

5.1.2 Non-Perfect Biclusters

This adaptation of In-Close2, called RIn-Close_CVC, is significantly more elaborate than RIn-Close_CVC_P because, besides a bicluster (A_r, B_r) being able to generate multiple descendants per attribute, there may be overlaps between their extents. For instance, in Figure 4(c), RIn-Close_CVC is closing the bicluster

$$(A_r = \{g_2, g_3, g_4, g_8, g_9, g_{10}, g_{11}, g_{15}, g_{16}, g_{19}, g_{20}, g_{22}, g_{23}\}, B_r = \{m_1, m_3, m_7\}),$$

Algorithm 5.1 RIn-Close_CVC_P

Input: Data matrix $\mathbf{D}_{n\times m}$, minimum number of rows $minRow$, index of the bicluster to be closed r, index of the initial attribute y

```
1: J ← {}
2: R ← {}
3: for j ← y to m do
4:    if j ∉ B_r then
5:       if max_{i∈A_r}(d_ij) − min_{i∈A_r}(d_ij) = 0 then
6:          B_r ← B_r ∪ {j}
7:       else
8:          Compute the possible extents // Eq. 5.1
9:          for each possible extent RW do
10:            if |RW| ≥ minRow and RW is canonical then
11:               r_new ← r_new + 1
12:               R ← R ∪ {r_new}
13:               J ← J ∪ {j}
14:               A_{r_new} ← RW
15:            end if
16:         end for
17:      end if
18:   end if
19: end for
20: Store (A_r, B_r) in the solution 𝔅
21: for k ← 1 to |J| do
22:    B_{R_k} ← B_r ∪ {J_k} // R_k and J_k are the k-th elements of R and J, respectively
23:    RIn-Close_CVC_P(D, minRow, R_k, J_k + 1)
24: end for
```

when it tries to add the current attribute m_8, bicluster (A_r, B_r) gives rise to three new biclusters *with overlap between their extents* (considering $\epsilon = 1$):

1. $(A = \{g_{11}, g_2, g_{15}\}, B = \{m_1, m_3, m_7, m_8\})$,
2. $(A = \{g_2, g_{15}, g_8, g_9, g_{10}, g_{16}, g_{19}, g_{22}\}, B = \{m_1, m_3, m_7, m_8\})$, and
3. $(A = \{g_8, g_9, g_{10}, g_{16}, g_{19}, g_{22}, g_3, g_4, g_{20}, g_{23}\}, B = \{m_1, m_3, m_7, m_8\})$.

Notice again that the elements of the current attribute were sorted to facilitate the identification of all possible extents.

In this scenario, since a bicluster (A_r, B_r) is able to generate multiple descendants per attribute, with overlap between them, it is necessary to take some actions to avoid the generation of duplicate and non-maximal biclusters. In fact, these challenging issues can occur if two descendant biclusters share $minRow$ rows or more in their extents.

Assuming $minRow = 3$, in our example in Figure 4(c), biclusters (d1) and (d2) cannot generate duplicate biclusters because they share only 2 rows in their extents. But biclusters (d2) and (d3) can generate duplicate biclusters with extent

$$A \subseteq \{g_8, g_9, g_{10}, g_{16}, g_{19}, g_{22}\}$$

and $|A| \geq minRow$, when adding a new attribute. To solve this problem, we added one more verification on the test of canonicity. This new verification is based on the fact that two distinct CVC biclusters must have two distinct extents in order to be maximal. So, we track the extents that have already been generated using efficient symbol table implementations, such as hash tables (HTs) or balanced search trees (BSTs). So, symbol table's keys are given by the extents, in such way that the rows in an extent are in their ascending (or descending) order. The worst-case time to insert and search in a BST is $O(\log k)$, where k is its number of elements. The worst-case time to insert and search in a HT is $O(1)$ and $O(k)$, respectively. However, under reasonable assumptions, the average time to search in a HT is $O(1)$. The remainder of the test of canonicity is again essentially the same as in In-Close2. Supposing that B is the current intent, j is the current attribute, and RW is a possible extent of a bicluster, it is not canonical if

$$\exists k \in M \setminus B | [k < j] \ \wedge \ [\max_{i \in RW}(d_{ik}) - \min_{i \in RW}(d_{ik}) \leq \epsilon], \tag{5.3}$$

i.e., if there is an attribute $k < j$ that we can add to the bicluster (RW, B) and it remains a valid CVC bicluster.

But even with this new verification on the test of canonicity, we still have the undesirable possibility of generating non-maximal biclusters. For instance, in Figure 4(c), bicluster (d3) can give rise to the bicluster

$$(A = \{g_4, g_8, g_9, g_{10}\}, B = \{m_1, m_3, m_7, m_8, m_{11}, m_{16}\}),$$

and bicluster (d2) can give rise to the bicluster

$$(A = \{g_8, g_9, g_{10}\}, B = \{m_1, m_3, m_7, m_8, m_{11}, m_{16}\}),$$

which is clearly non-maximal. So, when two biclusters share $minRow$ rows or more in their extents, we need to verify if their descendants are maximal in their extents (row-maximal). Therefore, the descendants of biclusters (d2) and (d3), whose extents are contained in the shared rows, need to check if they are row-maximal. Suppose that RM is the set of rows that the bicluster (A, B) must check to find out if it is row-maximal. The bicluster (A, B) is not row-maximal if there is an object $g \in RM$ that we can add to the bicluster and it remains a valid CVC bicluster, i.e.,

$$\exists g \in RM | [\max_{i \in \{A \cup \{g\}\}}(d_{ij}) - \min_{i \in \{A \cup \{g\}\}}(d_{ij}) \leq \epsilon], \forall \ j \in B. \tag{5.4}$$

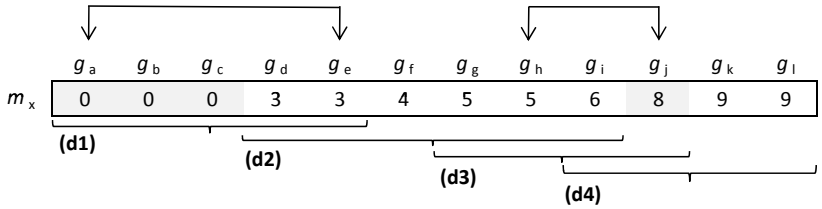


Figure 5 – Example of how to find RM (considering $\epsilon = 3$ and $minRow = 2$).

To explain how to compute RM, let us see the example in Figure 5, which considers $\epsilon = 3$ and $minRow = 2$. Suppose that when adding attribute m_x, a bicluster generated four biclusters: (d1), (d2), (d3) and (d4), whose extents are highlighted in Figure 5. Let us compute the set of rows RM that the descendants of the bicluster (d2) must check to verify their maximality, i.e., $RM_{(d2)}$. As the problem occurs when biclusters share $minRow$ rows or more in their extents, the pivot elements to compute $RM_{(d2)}$ are g_e and g_h because they are the $minRow$-th first and last elements of (d2), respectively. Their values are $g_e = 3$ and $g_h = 5$. Rows with values greater than or equal to 0 ($g_e - \epsilon$) or less than or equal to 8 ($g_h + \epsilon$) must comprise $RM_{(d2)}$, so $RM_{(d2)} = \{g_a, g_b, g_c, g_j\}$.

Back to our example in Figure 4(c), $RM_{(d2)} = \{g_3, g_4, g_{20}, g_{23}\}$. So, all descendants of the bicluster (d2) with extent $A \subseteq \{g_8, g_9, g_{10}, g_{16}, g_{19}, g_{22}\}$ must test the rows in $RM_{(d2)}$ to verify if they are row-maximal. For simplicity, the result will be correct if we implement this verification for all descendants of a bicluster.

It is very important to note that biclusters also need to inherit the set RM of their parents. For instance, suppose that the bicluster (d2) of Figure 4(c) gives rise to a bicluster $(A_x = \{g_8, g_9, g_{19}, g_{22}\}, B_x)$. So, we must have $RM_x \supseteq RM_{(d2)}$ because the descendants of (A_x, B_x) must test the rows in $RM_{(d2)}$ to verify if they are maximal.

Algorithm 5.2 shows the pseudocode of RIn-Close_CVC. The worst-case time of RIn-Close_CVC is $O(kmn(mn + x))$, where x is the worst-case time of searching in the symbol table, so $x = O(\log k)$ for BSTs and $x = O(k)$ for HTs. But recall that HTs have a much better computational cost on average: $O(1)$.

5.2 Finding biclusters with coherent values (CHV biclusters)

Once again, we will first show how to enumerate perfect CHV biclusters. We named this algorithm RIn-Close_CHV_P. It it is very similar to RIn-Close_CVC_P. Secondly, we will show how to enumerate non-perfect CHV biclusters. We named this algorithm RIn-Close_CHV.

Algorithm 5.2 RIn-Close_CVC

Input: Data matrix $\mathbf{D}_{n\times m}$, minimum number of rows $minRow$, index of the bicluster to be closed r, index of the initial attribute y, similarity constraint ϵ

```
1: J ← {}
2: R ← {}
3: for j ← y to m do
4:    if j ∉ B_r then
5:       if max_{i∈A_r}(d_ij) − min_{i∈A_r}(d_ij) ≤ ε then
6:          B_r ← B_r ∪ {j}
7:       else
8:          Compute the possible extents // Eq. 5.1
9:          for each possible extent RW do
10:            if |RW| ≥ minRow and RW is canonical and RW is row-maximal then
11:               Sort the elements of RW
12:               r_new ← r_new + 1
13:               R ← R ∪ {r_new}
14:               J ← J ∪ {j}
15:               A_{r_new} ← RW
16:               Set RM_{r_new}
17:               Update the symbol table
18:            end if
19:         end for
20:      end if
21:   end if
22: end for
23: Store (A_r, B_r) in the solution 𝔅
24: for k ← 1 to |J| do
25:    B_{R_k} ← B_r ∪ {J_k} // R_k and J_k are the k-th elements of R and J, respectively
26:    RIn-Close_CVC(D, minRow, R_k, J_k + 1, ε)
27: end for
```

5.2.1 Perfect Biclusters

When we are looking for CVC or CVR biclusters, we look directly to the values of the data matrix. But when we are looking for CHV biclusters, we need to check if there is coherence (additive or multiplicative) between each pair of columns (or rows) of the data matrix. For this, in RIn-Close_CHV_P, a bicluster starts with one column in its intent, which we call *pivot column*. Then, RIn-Close_CHV_P mines all biclusters that have this pivot column in their intents. Algorithm 5.3 shows this procedure. At the first iteration of the loop, RIn-Close_CHV_P will find all biclusters that have column 1 in their intents; at the second iteration, RIn-Close_CHV_P will find all biclusters that have column 2 and do not have column 1 in their intents; at the third iteration, RIn-Close_CHV_P will find all biclusters that have column 3 and do not have columns 1 and 2 in their intents; and so on.

RIn-Close_CHV_P exploits the fact that for mining perfect CHV biclusters it is not necessary to check if there is coherence between all pair of columns in an intent. Note

that in RIn-Close_CHV_P pseudocode, Algorithm 5.4, we just compute the difference between the current attribute j and the pivot column of the current intent B_r, i.e., B_{r_1}. If the current attribute j matches perfectly the pivot column, it will match perfectly the other columns of the intent B_r as well.

Algorithm 5.3 Main_RIn-Close_CHV_P

Input: Data matrix $\mathbf{D}_{n\times m}$, minimum number of rows $minRow$
Output: A set of biclusters $\mathfrak{B}$

```
1: r_new ← 0 // global variable
2: for atr ← 1 to m − 1 do
3:     r_new ← r_new + 1
4:     A_{r_new} ← {1, ..., n}
5:     B_{r_new} ← {atr}
6:     RIn-Close_CHV_P(D, minRow, r_new, atr + 1)
7: end for
```

Algorithm 5.4 RIn-Close_CHV_P

Input: Data matrix $\mathbf{D}_{n\times m}$, minimum number of rows $minRow$, index of the bicluster to be closed r, index of the initial attribute y

```
1: J ← {}
2: R ← {}
3: for j ← y to m do
4:     if j ∉ B_r then
5:         Z ← {d_{iB_{r_1}} − d_{ij}}_{i∈A_r}
6:         if max(Z) − min(Z) = 0 then
7:             B_r ← B_r ∪ {j}
8:         else
9:             Compute the possible extents // Eq. 5.5
10:            for each possible extent RW do
11:                if |RW| ≥ minRow and RW is canonical then
12:                    r_new ← r_new + 1
13:                    R ← R ∪ {r_new}
14:                    J ← J ∪ {j}
15:                    A_{r_new} ← RW
16:                end if
17:            end for
18:        end if
19:    end if
20: end for
21: Store (A_r, B_r) in the solution 𝔅
22: for k ← 1 to |J| do
23:     B_{R_k} ← B_r ∪ {J_k} // R_k and J_k are the k-th elements of R and J, respectively
24:     RIn-Close_CHV_P(D, minRow, R_k, J_k + 1)
25: end for
```

The computation of the possible extents is essentially the same as in the case of mining CVC biclusters (Eq. 5.1). The only difference is that we check if there is coherence

between the pivot column and the current column. So, let A be the current extent, and j be the current attribute, the possible extents are given by

$$\{RW | [RW \subseteq A] \ \wedge \ [\max_{i \in RW}(\{z_i\}) - \min_{i \in RW}(\{z_i\}) = 0] \ \wedge \ [RW \text{ is maximal}]\}, \tag{5.5}$$

where $z_i = d_{ip} - d_{ij}$ and p is the current pivot column.

The test of canonicity is also essentially the same as in In-Close2. Let B be the current intent, j be the current attribute, and RW be a possible extent of a bicluster. It is not canonical if

$$\exists k \in M \setminus B | [k < j] \ \wedge \ [\max(Z) - \min(Z) = 0], \tag{5.6}$$

where $Z \leftarrow \{d_{ip} - d_{ik}\}_{i \in RW}$ and p is the current pivot column, i.e., if there is an attribute $k < j$ that we can add to the bicluster (RW, B) and it remains a valid perfect CHV bicluster.

The worst-case time of RIn-Close_CHV_P is the same as that of RIn-Close_CVC_P: $O(knm(\log n + m))$.

5.2.2 Non-Perfect Biclusters

Now, we will explain how to perform a complete, correct and non-redundant enumeration of all maximal perturbed CHV biclusters, given a similarity constraint determined by the user-defined parameter $\epsilon > 0$, as defined in Definition 2.5.

To achieve this goal, we cannot simply apply to RIn-Close_CHV_P the same adaptations that we have made in RIn-Close_CVC_P to achieve RIn-Close_CVC. First of all, if RIn-Close_CHV computed the set Z considering only the current pivot column B_{r_1} and the current column j, it could occur a difference up to 2ϵ between any other two columns of B_r. Besides, the order of choice of the pivot columns interferes in the outcome in this scenario. In this way RIn-Close_CHV would yield just an approximate result of an actual enumeration (as SeqClus [105] and CPT [44] do), and this is not the case here. Also, RIn-Close_CHV could not simply verify if the current attribute j fits all the attributes in the current intent B_r. An example of a problem that could happen is that: if the data matrix has two biclusters with the same extent A, but different intents $B_x = \{m_1, m_3, m_5\}$ and $B_y = \{m_1, m_3, m_6, m_8\}$, a naive procedure would find just the first one because it loops through the attributes in its sequential order and the attribute m_5 is not coherent with attributes m_6 and m_8 (considering the rows in extent A). In addition, it would be quite difficult to define the possible new extents. Another challenging issue of this approach is to determine when a bicluster is canonical or not. For instance, if the data matrix has

two biclusters with the same extent A, but different intents $B_x = \{m_1, m_2, m_3, m_4\}$ and $B_y = \{m_2, m_3, m_4, m_5\}$, a naive procedure would discard the second because the attributes m_2, m_3 and m_4 are coherent with attribute m_1. Moreover, a non-canonical bicluster could give rise to canonical biclusters, so it could not be discarded until all its descendant were generated.

To avoid all these undesired scenarios, RIn-Close_CHV uses the following procedure with three steps:

- Compute the augmented matrix of the data matrix $\mathbf{D}$, denoted $\mathbf{D}_a$. $\mathbf{D}_a$ is a matrix with the difference between all pairs of columns of $\mathbf{D}$. For instance, the augmented matrix $\mathbf{D}_a$ of the data matrix $\mathbf{D}$ in Table 9 is illustrated in Table 10. The first column of $\mathbf{D}_a$ is the difference between columns 1 and 2 of $\mathbf{D}$, the second column of $\mathbf{D}_a$ is the difference between columns 1 and 3 of $\mathbf{D}$, the third column of $\mathbf{D}_a$ is the difference between columns 1 and 4 of $\mathbf{D}$, and so on for all combinations of pairs of columns.
- Apply RIn-Close_CVC to the augmented matrix $\mathbf{D}_a$. To illustrate, Figure 6(a) shows all maximal CVC biclusters found by RIn-Close_CVC when applied to the data matrix of Table 10 (using $minRow = 2$ and $\epsilon = 1$).
- Extract all maximal CHV biclusters from the maximal CVC biclusters found by RIn-Close_CVC (see the pseudocode in Algorithm 5.5).

Table 9 – Example of a numerical dataset.

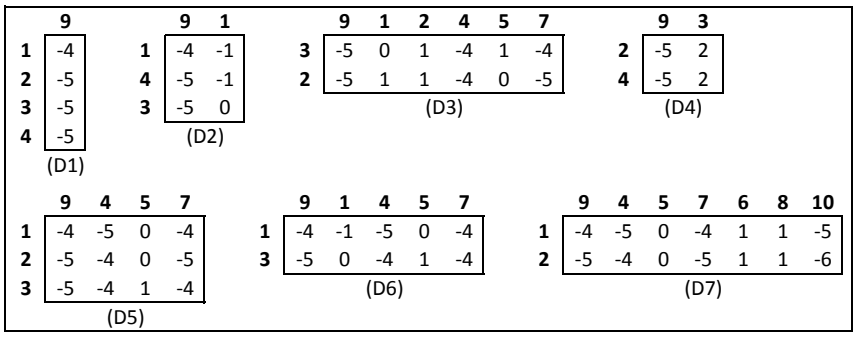
	m_1	m_2	m_3	m_4	m_5
g_1	1	2	2	1	6
g_2	2	1	1	0	6
g_3	2	2	1	7	6
g_4	8	9	2	6	7

Table 10 – Augmented matrix of the data matrix in Table 9.

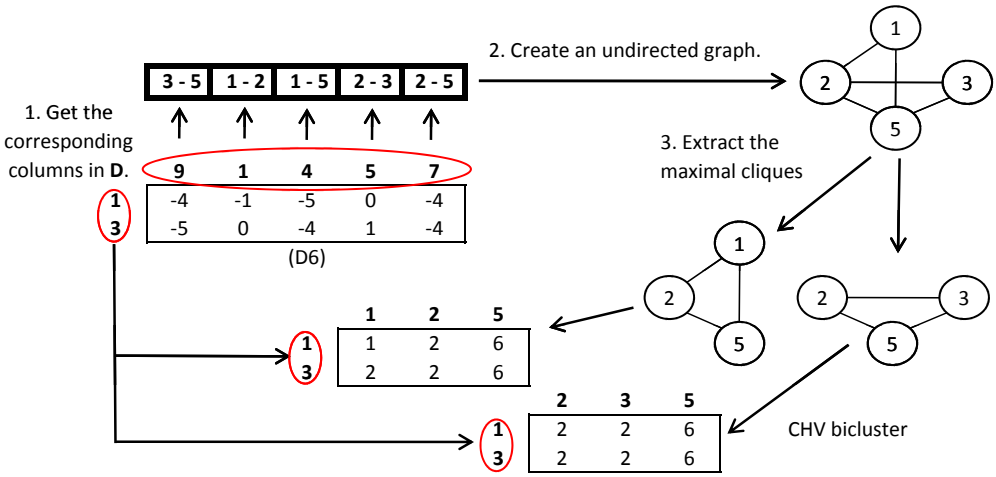
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	-1	-1	0	-5	0	1	-4	1	-4	-5
2	1	1	2	-4	0	1	-5	1	-5	-6
3	0	1	-5	-4	1	-5	-4	-6	-5	1
4	-1	6	2	1	7	3	2	-4	-5	-1

Algorithm 5.5 shows the pseudocode of the procedure to process each CVC bicluster. Steps in lines 2 to 5 are illustrated in Figure 6(b). In this illustrative scheme, we are extracting CHV biclusters from the CVC bicluster (D6) of Figure 6(a).

The first step is to get the corresponding columns in $\mathbf{D}$ of the intent of bicluster (D6). For instance, column 9 of $\mathbf{D}_a$ corresponds to the difference between columns 3 and



(a) All CVC biclusters found in the data matrix of Table 10 (using $minRow = 2$ and $\epsilon = 1$).



(b) Illustrative scheme to show how a CVC bicluster is processed by RIn-Close_CHV.

Figure 6 – RIn-Close_CHV's framework.

5 of **D**. Therefore, columns 3 and 5 are coherent with each other considering the extent $\{1, 3\}$ and $\epsilon = 1$. Let us name this corresponding set of columns as $B2$. In this example, we have $B2 = \{1, 2, 3, 5\}$.

The second step is to create an undirected graph, in which the nodes represent $B2$, and the edges represent the columns that the CVC bicluster indicated that are coherent with each other. So, we have 5 edges in this example: $3-5$, $1-2$, $1-5$, $2-3$, and $2-5$.

Since all pairs of columns of a CHV bicluster must be coherent with each other (see Definition 2.5), the third step is to find all maximal cliques from this undirected graph. A clique is a subset of vertices of an undirected graph such that every two distinct vertices in the clique are adjacent. Thus, each clique indicates the subsets of $B2$ in which all columns are coherent with each other considering the rows determined by the CVC bicluster and the user-defined parameter ϵ. So, the CHV biclusters generated by the CVC bicluster (D6) are: $(\{1, 3\}, \{1, 2, 5\})$ and $(\{1, 3\}, \{2, 3, 5\})$.

Lines 7 and 8 of the pseudocode in Algorithm 5.5 verify if the CHV bicluster (A, D) is new and, if so, store it in the set of CHV biclusters. A CHV bicluster (A, D) is new if (i) its intent D is equal to $B2$, or (ii) (A, D) is row-maximal (there is no object g that we can add to its extent A), i.e.,

$$\nexists g \in G \setminus A | [\max(Z^{jl}) - \min(Z^{jl}) \leq \epsilon], \forall j, l \in D, \tag{5.7}$$

where $Z^{jl} = \{d_{ij} - d_{il}\}_{i \in A \cup \{g\}}$. When the intent D is equal to $B2$, (A, D) is mandatorily row-maximal. We can affirm it because we are manipulating maximal CVC biclusters, and we are using all columns indicated by it.

The worst-case time of verifying if a CHV bicluster is row-maximal is $O(mn)$. Makino and Uno [81] proved that all maximal cliques of a graph containing v vertices can be enumerated with time delay $O(M(v))$, where $M(v)$ is the cost of multiplying two $v \times v$ matrices. As stated in Section 5.1.2, each CVC bicluster can be enumerated with time delay $O(m_a n(m_a n + x))$, where $m_a = m(m-1)/2$ is the number of columns of the augmented matrix $\mathbf{D}_a$.

Algorithm 5.5 Mining CHV biclusters from CVC biclusters

Input: Set of CVC biclusters *lbCVC*
Output: Set of CHV biclusters

```
1: for each (A, B) in lbCVC do
2:     Compute B2
3:     Create an undirected graph
4:     Find all maximal cliques
5:     Compute the CHV biclusters
6:     for each CHV bicluster (A, D) do
7:         if D = B2 or (A, D) is row-maximal then
8:             Store (A, D) in the set of CHV biclusters
9:         end if
10:    end for
11: end for
```

5.3 Finding other types of biclusters

The procedure of three steps used by RIn-Close_CHV, described in Subsection 5.2.2, can be easily adapted to mine other types of biclusters, such as the OPSM biclusters. The only required adaptation is on the computation of the augmented matrix $\mathbf{D}_a$.

To mine OPSM biclusters, the augmented matrix $\mathbf{D}_a$ must be a matrix with a comparison between all pairs of columns of the matrix $\mathbf{D}$. This comparison must return:

- 1, if $a_{ij} < a_{il}$,
- 2, if $a_{ij} = a_{il}$, and

- 3, if $a_{ij} > a_{il}$,

for $j < l$. For instance, the augmented matrix $\mathbf{D}_a$ of the data matrix $\mathbf{D}$ in Table 9 is illustrated in Table 11.

Table 11 – Augmented matrix of the data matrix in Table 9 with a CVC bicluster highlighted.

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	1	1	2	1	2	3	1	3	1	1
2	3	3	3	1	2	3	1	3	1	1
3	2	3	1	1	3	1	1	1	1	3
4	1	3	3	3	3	3	3	1	1	1

In this way, we just use RIn-Close_CVC with $\epsilon = 1$ in the second step of the procedure. As the data matrix has only 3 different values, we can implement RIn-Close_CVC in a more efficient way. There are only two possible new extents: one with the values 1 and 2, and other with the values 2 and 3. We do not need to sort the elements to compute these two extents (line 8 of the Algorithm 5.2) or to compute the set RM (line 16 of the Algorithm 5.2).

As an example, we have a CVC bicluster highlighted in Table 11. It tells us that, considering the set of rows {1, 2, 3}, there is an order relation between the following columns of the data matrix $\mathbf{D}$ in Table 9: $m_1 \leq m_5$, $m_3 \leq m_2$, $m_2 \leq m_5$, $m_3 \leq m_5$. Thus, the maximal cliques (in the third step of the procedure) correspond to the column-sets of column-maximal OPSM biclusters. It means that we can assemble a permutation so that all columns are rearranged from the column with the lowest values to the column with the highest values (see Lemma 2.6).

Other example of biclusters that can be mined using the procedure of three steps used by RIn-Close_CHV is the error-tolerated OPSM biclusters [24]. To mine them, the augmented matrix $\mathbf{D}_a$ must also be a matrix with a comparison between all pairs of columns of the data matrix $\mathbf{D}$. This comparison must return:

- 1, if $a_{ij} + \epsilon < a_{il}$,
- 2, if $|a_{ij} - a_{il}| \leq \epsilon$, and
- 3, if $a_{ij} - \epsilon > a_{il}$,

for $j < l$ and where $\epsilon > 0$ is a user-defined parameter that controls the tolerated error in the OPSM biclusters.

In these two examples, the values 1, 2, and 3 are not mandatory. We can use any three consecutive values, such as 10, 11, and 12. We can even use three values in a series

with step other than 1, such as 5, 7, and 9, and set ϵ equal to the step (in this example, $\epsilon = 2$).

Thus, we have shown that this procedure of 3 steps can be used for mining other types of biclusters. The main challenge is that the more flexible the definition of biclusters, the greater tends to be the number of biclusters in the dataset, and it can lead the enumeration to become impractical.

5.4 Handling missing values

There are some strategies that we can apply to treat missing values. The simplest one is to remove the rows or columns (usually the ones with smaller dimension) containing missing values, at the cost of information loss. Another strategy, which is simple in the biclustering's point of view, is the previous estimation of the missing values using some technique, such as k-NN [19] or rSVD [93]. The problem with this approach is that it introduces additional noise to the dataset which may significantly reduce the biclusters' homogeneity.

Henriques and Madeira [49] proposed the use of additional attributes handled according to a level of relaxation defined by the user. Our proposal to handle the missing data is similar to this when considering its more restrictive way, where the missing items are removed (not the entire rows or columns). In our case, we will look for biclusters in the regions of the dataset without missing values, ignoring the regions with missing values. In the following, we describe the required adaptations in RIn-Close algorithms to handle the missing values in this way.

In the Algorithms 5.1, 5.2 and 5.4, we must do the following adaptations:

- If the attribute j has missing values in the objects A_r, it cannot be added to the intent B_r.
- When computing the possible new extents, we must simply ignore the objects with missing values.
- When verifying if RW is not canonical (see Eqs. 5.2, 5.3 and 5.6), we also test if the attribute $k \in M \setminus B$ does not have missing values in the objects RW.

When testing if a bicluster is row-maximal (see Eq. 5.4) in the Algorithm 5.2, we also must test if the object g has missing values in one or more of the attributes B. If the object g has missing values in one or more of the attributes B, it cannot be added to the bicluster.

In Algorithm 5.3, A_{rnew} must have only the objects where the attribute atr does not have missing values.

For the algorithm RIn-Close_CHV, when computing the augmented matrix, we must simply compute the difference between the non-missing data. A missing value minus a non-missing value, or vice versa, or a missing value minus a missing value, results in a missing value in the augmented matrix $\mathbf{D}_a$.

When testing if a bicluster is row-maximal (see Eq. 5.7) in the Algorithm 5.5, we also must test if the object g has missing values in one or more of the attributes D. If the object g has missing values in one or more of the attributes D, it cannot be added to the bicluster.

Thereby, we are mining the biclusters in all the regions of the dataset without missing values. According to Property 2.4, the coverage of our bicluster solution (without restrictions in the minimum number of rows and columns of the biclusters) will be equal to the percentage of non-missing values in the dataset.

5.5 Chapter Overview

This chapter presented the main contribution of this thesis, i.e., our family of enumerative biclustering algorithms, which is called RIn-Close. The first two algorithms, RIn-Close_CVC_P and RIn-Close_CVC, are able to perform an efficient, complete, correct, and non-redundant enumeration of all maximal perfect and perturbed, respectively, CVC biclusters. The definition of a CVR bicluster is the equivalent transpose of the definition of a CVC bicluster. So, we can mine CVR biclusters by transposing the original data matrix and using an algorithm to mine CVC biclusters. Our proposed algorithm to enumerate all maximal perfect CHV biclusters, RIn-Close_CHV_P, has these four properties too. Our algorithm to enumerate all maximal perturbed CHV biclusters, RIn-Close_CHV, has the last three properties, i.e., it is able to perform a complete, correct, and non-redundant enumeration, but not a enumeration with polynomial delay. As far as we know, RIn-Close are the most complete biclustering algorithms from the literature.

We also showed how the procedure of three steps used by RIn-Close_CHV can be adapted to enumerate other types of biclusters, such as OPSM and error-tolerated OPSM biclusters.

Finally, this chapter presented our proposal to handle missing data using RIn-Close. Basically, the idea is to mine the biclusters in regions of the dataset without missing values, ignoring the regions with missing values.

In the next chapter, we will present the performed experiments as well as the obtained results.

6 Experimental Results

We evaluated the RIn-Close family of algorithms on both synthetic and real datasets. We highlight various practicalities in the usage of RIn-Close, and outline the advantages and distinct aspects of (i) an enumerative algorithm when compared to heuristics, and (ii) having an enumerative algorithm with the key properties of being correct, complete and non-redundant. We also apply RIn-Close in two real-world problems: (i) the gene ontology enrichment analysis, and (ii) the analysis and identification of biomarkers.

We implemented RIn-Close using C++ [1]. For the third step of RIn-Close_CHV, we implemented the BK algorithm [17] with the I. Koch's pivot selection strategy [61], because it is the best one in practice [20]. We compile RIn-Close using the following command: g++ -std=gnu++11 -O3. For the heuristics, we used the MTBA toolbox [45]. The only exception was ROCC's implementation that was obtained from the authors. MicroCluster's implementation were obtained from the authors' website. The experiments were carried out on a PC Intel(R) Core(TM) i7-4770K CPU @ 3.5 GHz, 32 GB of RAM, and running under Ubuntu 14.04.

6.1 RIn-Close: scalability issues

The results of this section were first reported in [103].

This experiment aims to test RIn-Close's performance when varying (i) the number n of rows of the dataset, (ii) the number m of columns of the dataset, (iii) the number of biclusters in the dataset, (iv) the bicluster row size, (v) the bicluster column size, (vi) the overlap, and (vii) the noise in the dataset. For this purpose, we created synthetic datasets with controlled number, size, shape and level of noise of the existing biclusters and, then, we tested how RIn-Close performs when varying each one of the parameters in isolation.

The default parameters used in the synthetic data generator were: $n = 5000$, $m = 60$, number of biclusters = 10, bicluster row size = 200, bicluster column size = 8, overlap = 0.2, and Gaussian noise with $\mu = 0$ and $\sigma^2 = 0.01$. The synthetic data generator creates the biclusters and assigns random values to the other regions of the dataset. Then, it adds the Gaussian noise and shuffles the rows and columns of the dataset. Therefore, the generator creates arbitrarily positioned overlapping biclusters, so that the resulting biclusters are usually non-contiguous. The amount of noise was chosen in such a way that the original biclusters were preserved. For each configuration, we created 50 synthetic datasets to compute the average runtimes. RIn-Close_CVC_P and RIn-Close_CHV_P,

[1] `https://sourceforge.net/projects/rinclose/`

that mine perfect biclusters, were applied to datasets without noise.

Figures 7 to 10 show, respectively, the sensitivity to distinct datasets' configurations of RIn-Close_CVC_P, RIn-Close_CVC, RIn-Close_CHV_P, and RIn-Close_CHV. The runtime growth rates were defined visually by the shape of the curve, being easy to discover the coefficients of the curve using some fit tool.

The results revealed that the runtime increased linearly with n, except for RIn-Close_CHV, for which it increased logarithmically. But for all algorithms, without exception, the runtime growth rate was significantly more favorable than their worst-case time complexities (see Chapter 5). For variable m, the runtime increased linearly for RIn-Close_CVC_P and RIn-Close_CVC, which is better than their worst-case time complexities. For RIn-Close_CHV_P and RIn-Close_CHV, the runtime increased polynomially with m, which coincides with their worst-case time complexities. For all algorithms, the runtime decreased linearly with the overlap, and the noise level did not greatly affect the runtime.

Except for RIn-Close_CHV, the runtime increased linearly with the number of biclusters. For RIn-Close_CHV, we can notice a tendency of a logarithmic growth, which is good news because, due to its worst-case time complexity, we expected a linear growth too. For the bicluster row size, we had a linear growth of the runtime, except for RIn-Close_CHV, for which we had a smooth polynomial growth. RIn-Close_CVC_P and RIn-Close_CVC had the same behavior for the bicluster column size: the runtime increased logarithmically, but saturates and started to decrease linearly. With the increase in the bicluster column size, the coverage of the dataset increases, and it seemed to help these algorithms to find the biclusters more quickly. With more columns in the biclusters, more the inheritance of the columns tends to have a positive effect, and also less columns tend to be tested in the canonicity function. For the RIn-Close_CHV_P and RIn-Close_CHV, the runtime increased logarithmically and polynomially with the bicluster column size, respectively.

The variations presented in the boxplots are due to two main reasons: (i) an intrinsic variation due to the machine; and (ii) some characteristics of the synthetic datasets, more specifically, due to the arrangement of the biclusters and the noise level in the datasets. The arrangement of the biclusters in the datasets impacts in the heritage of the columns, in the number of times that the canonicity function is called, and in the number of columns that are tested in the canonicity function. The noise in the datasets influences the computation of the possible extents as well as the number of times that the canonicity function is called. A note on these results is that the runtime of Steps 1 and 3 of RIn-Close_CHV was negligible. Therefore, the results for RIn-Close_CHV are basically the runtime of the Step 2 of the algorithm, which is devoted to mine the CVC biclusters from the augmented matrix $\mathbf{D}_a$.

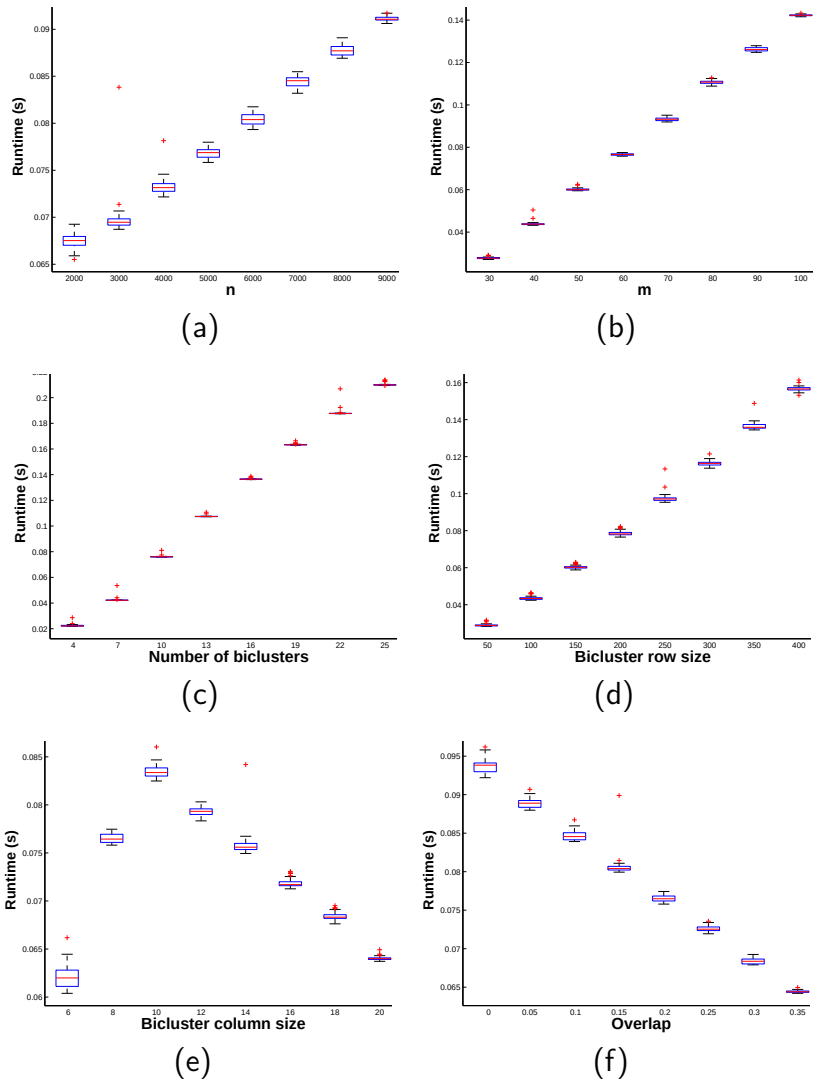


Figure 7 – Results of the performance of RIn-Close_CVC_P when varying (a) the number n of rows of the dataset, (b) the number m of columns of the dataset, (c) the number of biclusters in the dataset, (d) the bicluster row size, (e) the bicluster column size, and (f) the overlap.

6.2 RIn-Close: sensitivity analysis

The results of this section were first reported in [103].

This experiment aims to test RIn-Close's sensitivity to the parameters ϵ and *minRow*. We ran RIn-Close for four microarray gene expression datasets: Yeast[2] [25], GDS232[3] [78], GDS750[3] [70] and GDS4085[3] [54]. The former dataset, Yeast, was preprocessed by Cheng and Church [23], we just threw away the two genes with missing values. The last three datasets were preprocessed by us. For each one of them, we remove the empty spots; we threw away the data for any genes where one or more expression levels

[2] http://arep.med.harvard.edu/biclustering, last accessed 04/09/2015
[3] http://www.ncbi.nlm.nih.gov, last accessed 04/09/2015

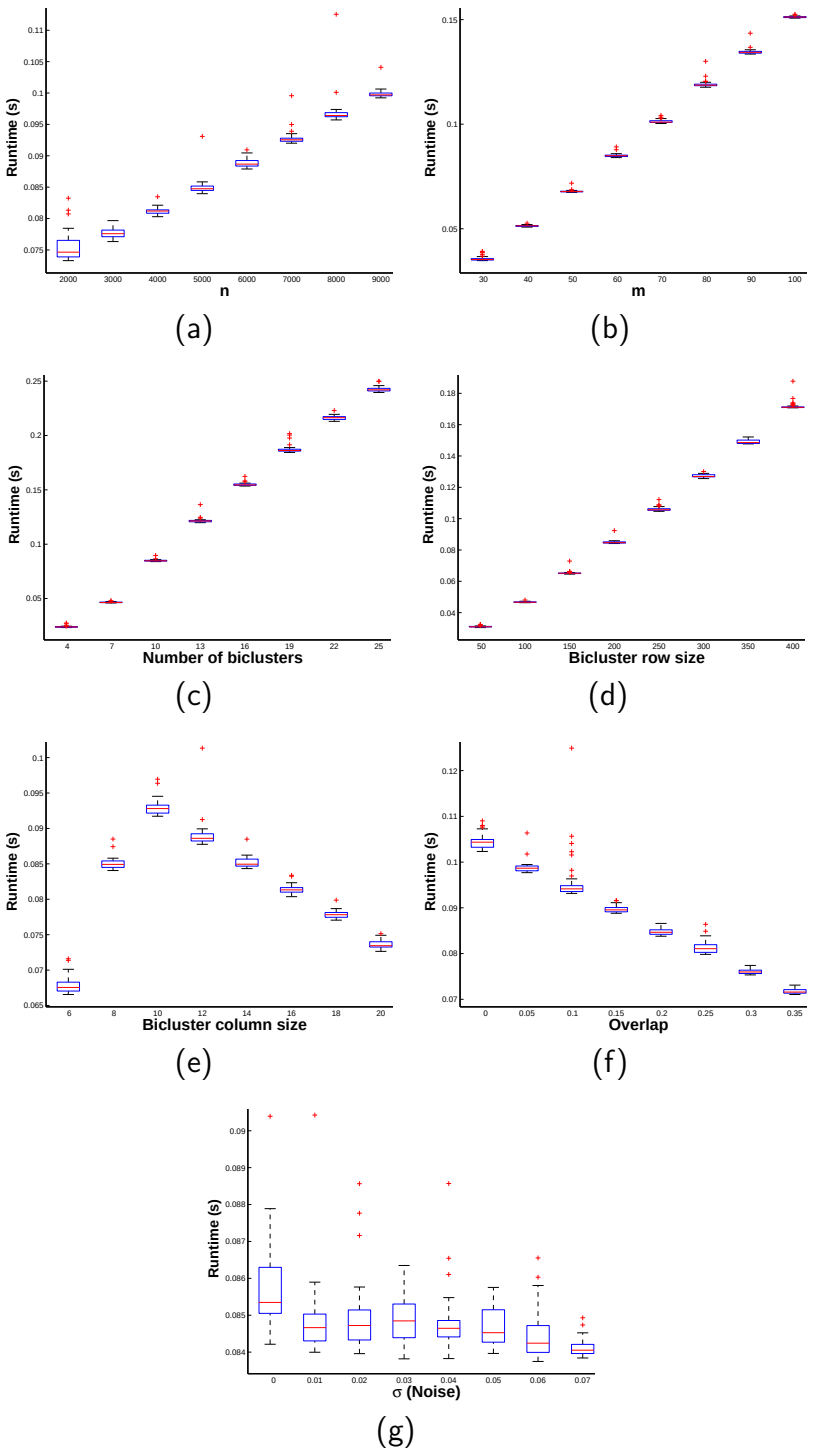


Figure 8 – Results of the performance of RIn-Close_CVC when varying (a) the number n of rows of the dataset, (b) the number m of columns of the dataset, (c) the number of biclusters in the dataset, (d) the bicluster row size, (e) the bicluster column size, (f) the overlap, and (g) the noise level.

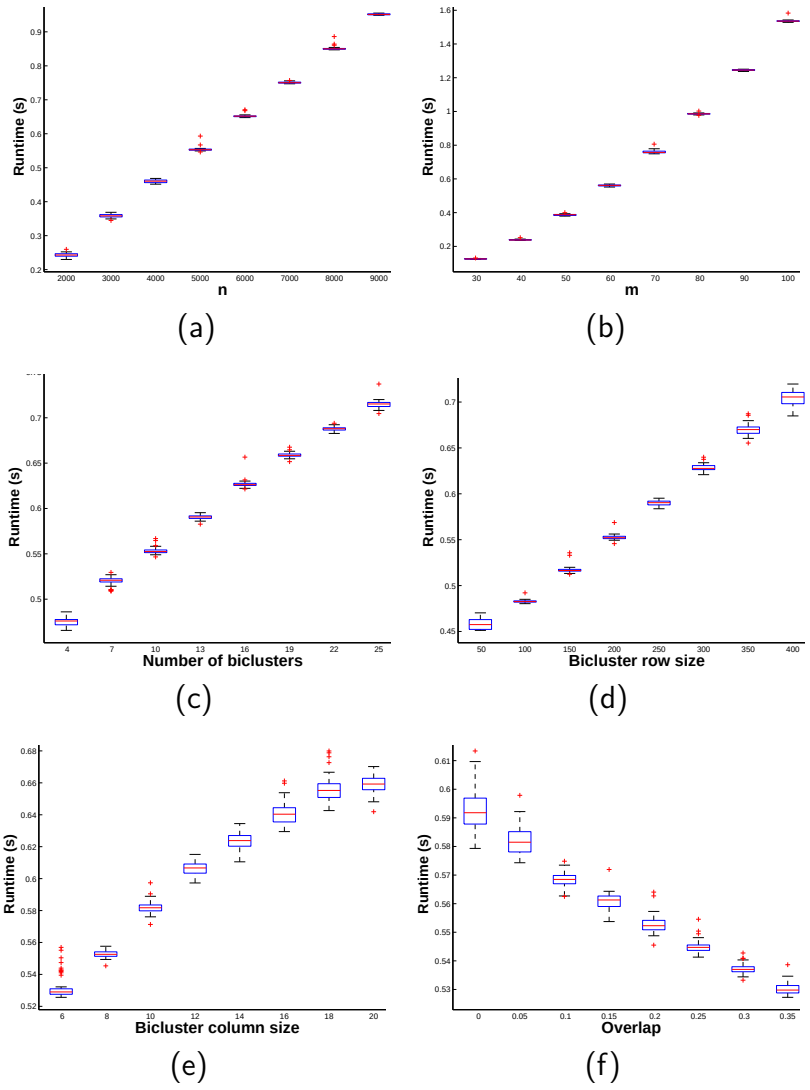


Figure 9 – Results of the performance of RIn-Close_CHV_P when varying (a) the number n of rows of the dataset, (b) the number m of columns of the dataset, (c) the number of biclusters in the dataset, (d) the bicluster row size, (e) the bicluster column size, and (f) the overlap.

were not measured; we filtered out genes with small variance over time; and we scale the data of each column to integers between 0 and 1000. Table 12 shows more information about these datasets. We ran RIn-Close 50 times to compute the average runtime, and we looked for biclusters with at least 3 columns.

The parameters ϵ and $minRow$ were set empirically according to the characteristics of each dataset, so that the number of biclusters were not so high and considering the relation between coverage and global overlap.

Figures 11 and 12 show respectively the sensitivity of RIn-Close_CVC and RIn-Close_CHV to the parameter ϵ. Usually, the number of biclusters, the runtime, the coverage, and the global overlap increases with ϵ. The only exception is the global overlap

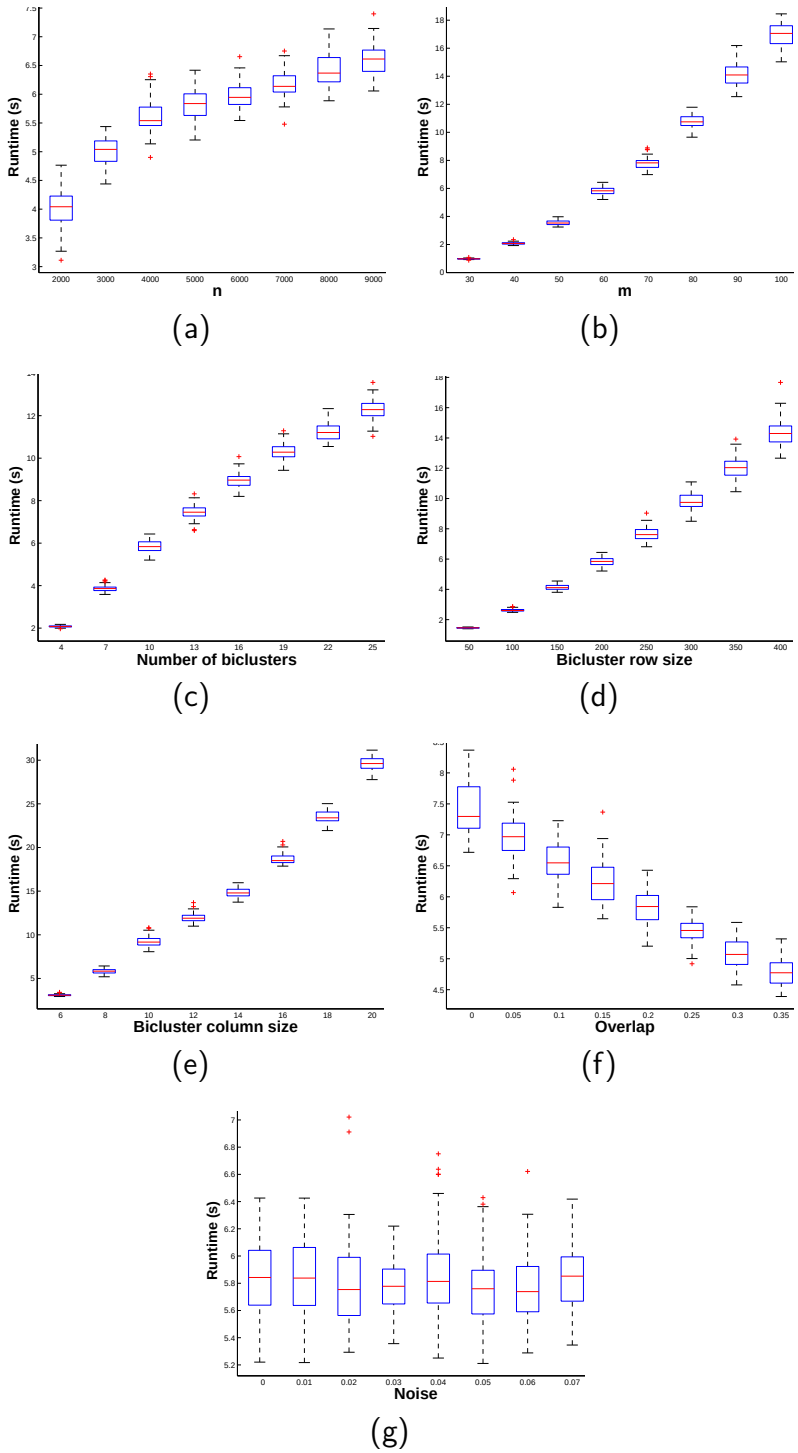


Figure 10 – Results of the performance of RIn-Close_CHV when varying (a) the number n of rows of the dataset, (b) the number m of columns of the dataset, (c) the number of biclusters in the dataset, (d) the bicluster row size, (e) the bicluster column size, (f) the overlap, and (g) the noise level.

Table 12 – Datasets description.

Name	Dimension	Organism
Yeast	2882×17	Saccharomyces cerevisiae
GDS232	589×23	Homo sapiens
GDS750	3456×13	Saccharomyces cerevisiae
GDS4085	1133×19	Homo sapiens

of the dataset Yeast in Figure 11(d). The coverage will always increase with ϵ, because all portions of the dataset explored with $\epsilon = x$, will be explored with $\epsilon > x$, as stated in Property 2.5 (see Chapter 2.3). However, as we stated in Chapter 2.3, the number of biclusters will not always increase with ϵ. The global overlap depends on the coverage and on the number of biclusters. If we increase ϵ and find more biclusters, but we explore pretty much the same portions of the dataset, the global overlap will increase. On the other hand, if these new biclusters bring a significant gain in coverage, the global overlap tends to decrease. Therefore, when the global overlap's growth rate is higher than the coverage's growth rate, it indicates that we are finding new biclusters in portions of the dataset explored with lower values of ϵ. Oliveira *et al.* [89] illustrated how the noise is responsible for fragmenting each true bicluster into many with high overlapping. As it complicates the analysis of the results, the aggregation of these biclusters is recommended [89, 111]. Given that the runtime is proportional to the number of biclusters, the choice of ϵ must consider the gain in the coverage and the gain in the global overlap. If we only have a considerable gain in the global overlap, then it might be more convenient to use a lower value of ϵ. Moreover, the smaller the value ϵ, the lower the accepted perturbation in the biclusters. In practical applications, usually we look for biclusters with little perturbation, which favors saving resources.

Figures 13 and 14 shows respectively the sensitivity of RIn-Close_CVC and RIn-Close_CHV to the parameter *minRow*. The number of biclusters, the runtime, the coverage, and the global overlap decreased with the increase in the value of *minRow* in all cases. We observe that the parameter *minRow* has also a strong influence in the computational cost of RIn-Close. The higher its value, the smaller the search space for enumerating biclusters. Again, we see that the choice of *minRow* must consider the relation between coverage and global overlap. If a larger value of *minRow* has a small impact on the coverage and a significant impact on the global overlap, keeping the current value seems to be a reasonable choice.

As we observed with this experiment, RIn-Close parameters can be set to avoid or at least to better control the explosion in the number of biclusters, with similar impact on the computational cost.

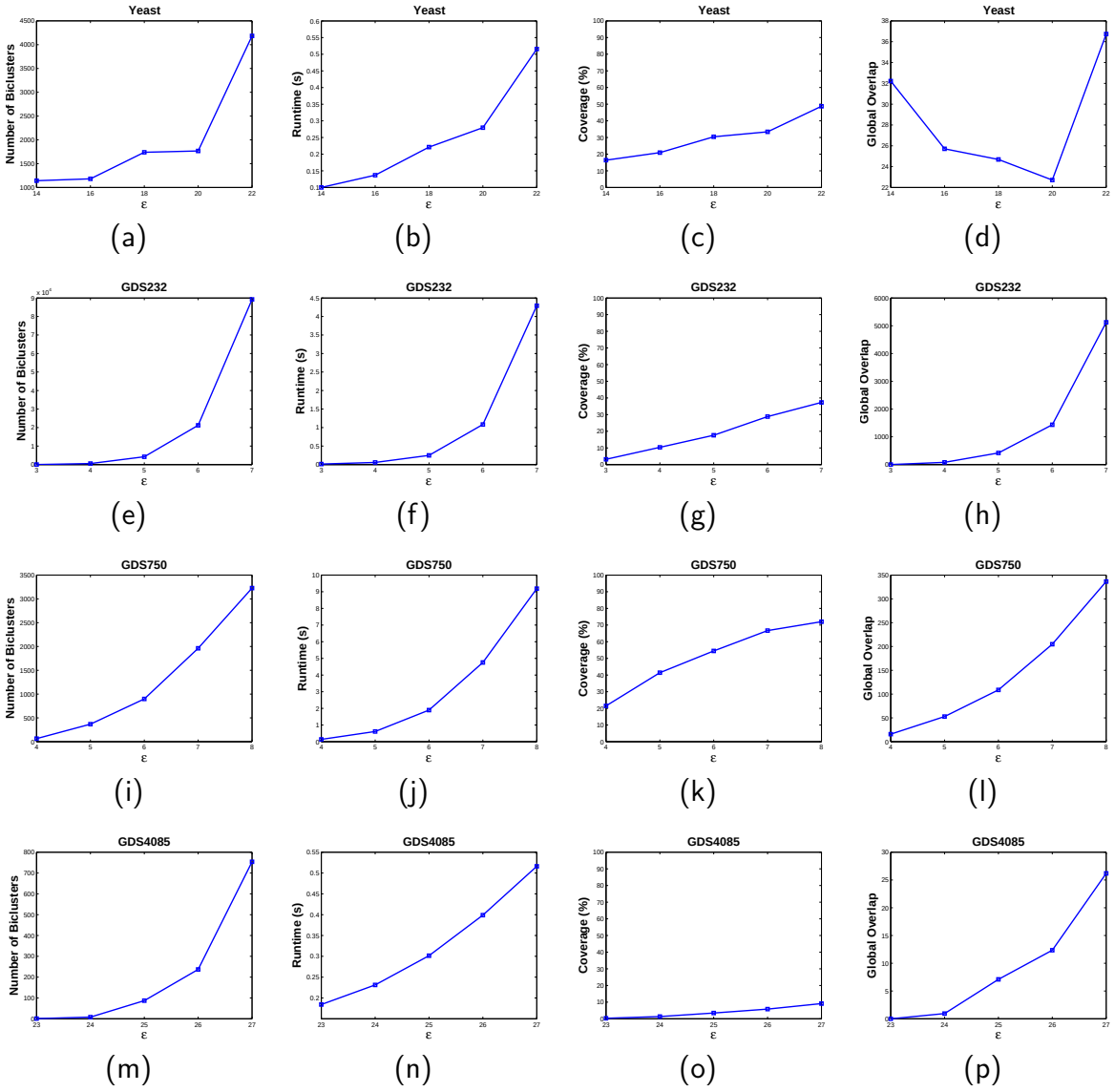


Figure 11 – Results of RIn-Close_CVC's sensitivity to the parameter ϵ. The parameter *minRow* was set to: 57 for Yeast; 59 for GDS232; 795 for GDS750; and 23 for GDS4085.

6.3 RIn-Close: handling missing values

In Section 5.4, we propose a strategy to handle datasets with missing values. Now, we will provide an insight about the impact of this strategy in the enumeration results.

We again used the four microarray gene expression datasets described in Table 12. We imputed artificially missing values on them in different percentages, creating various datasets with missing values. For each percentage of missing values, we created 50 datasets based on each one of the four datasets. So, each one of our results is the mean of 50 executions of RIn-Close algorithms. The parameters used in this experiment are reported in Table 13.

Figures 15 and 16 show the results for RIn-Close_CVC and RIn-Close_CHV,

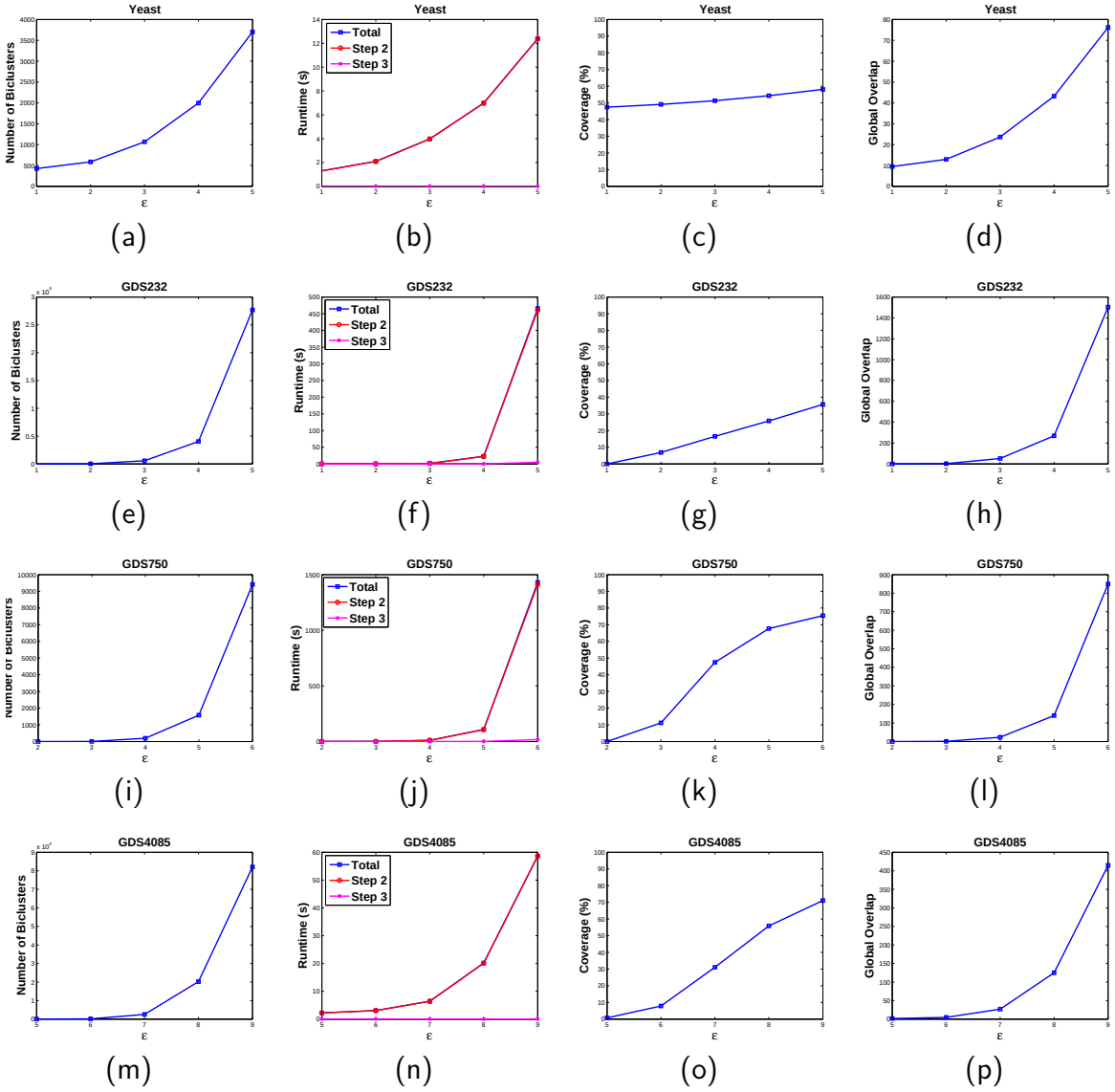


Figure 12 – Results of RIn-Close_CHV's sensitivity to the parameter ϵ. The parameter *minRow* was set to: 144 (5%) for Yeast; 59 (10%) for GDS232; 795 (23%) for GDS750; and 23 (2%) for GDS4085.

Table 13 – Parameters of the experiments with missing values.

	CVC			CHV		
	minRow	*minCol*	ϵ	*minRow*	*minCol*	ϵ
Yeast	57	3	20	144	3	4
GDS232	59	3	6	59	3	5
GDS750	795	3	10	795	3	6
GDS4085	23	3	36	23	3	9

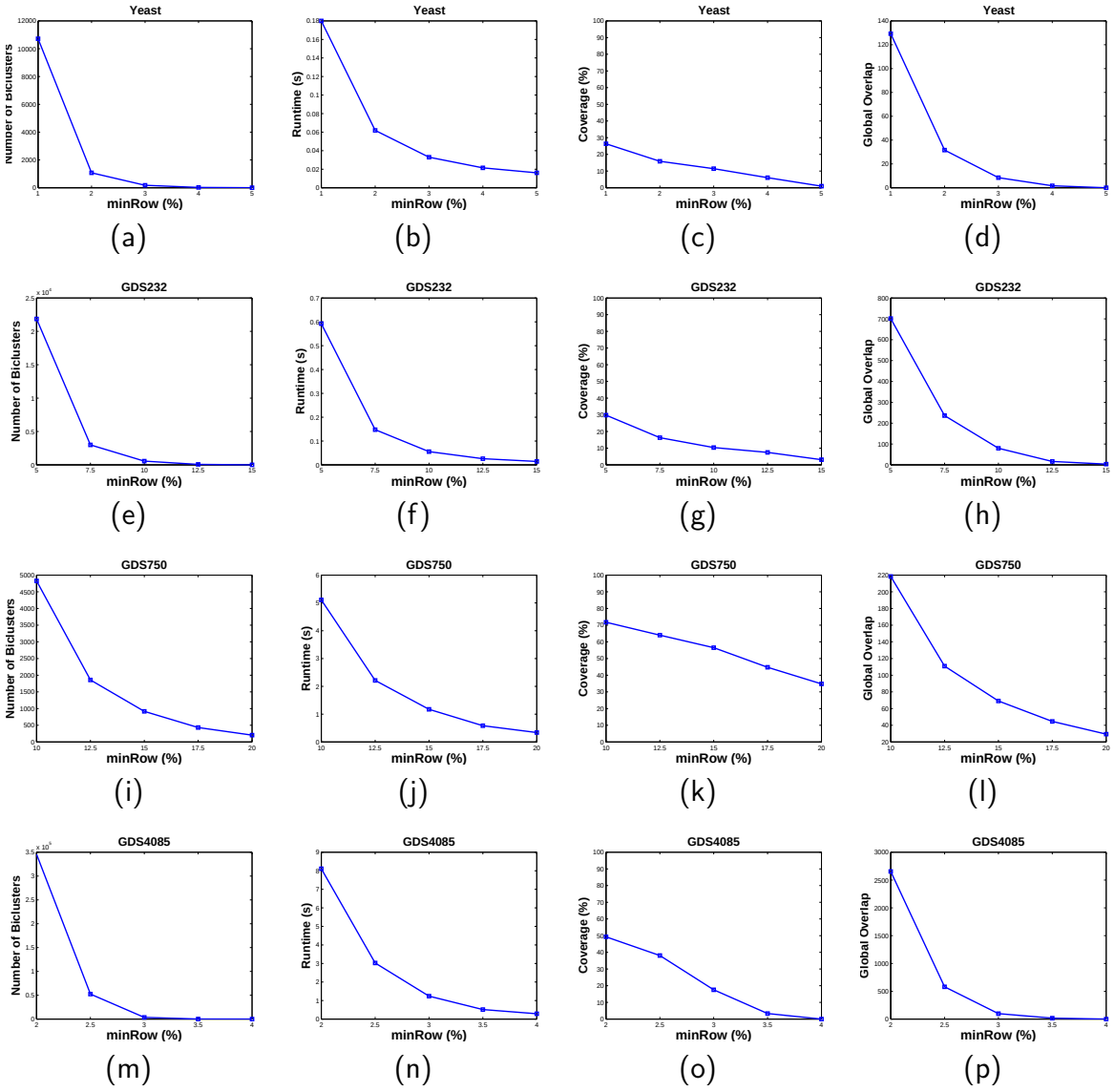


Figure 13 – Results of RIn-Close_CVC's sensitivity to the parameter *minRow*. The parameter ϵ was set to: 5 for Yeast; 4 for GDS232; 4 for GDS750; and 37 for GDS4085.

respectively. Invariably, the number of biclusters, the runtime, the coverage, and the global overlap decreased with the increase of missing values in the datasets. These results meet completely our expectations because of two reasons: (*i*) higher percentages of missing data means that we have fewer portions of the dataset to look for biclusters; and (*ii*) biclusters that meets the restrictions of minimum number of rows and columns cannot meet one or both of these restrictions with more missing values embedded in the dataset. The less the number of biclusters in a solution, the smaller tends to be the runtime, coverage, and global overlap. Logically, a missing value can partition a bicluster in two, increasing the number of biclusters in the solution. But the more the biclusters are close to these lower bounds (minimum number of rows and columns), the less it happens. It was certainly our case in this experiment.

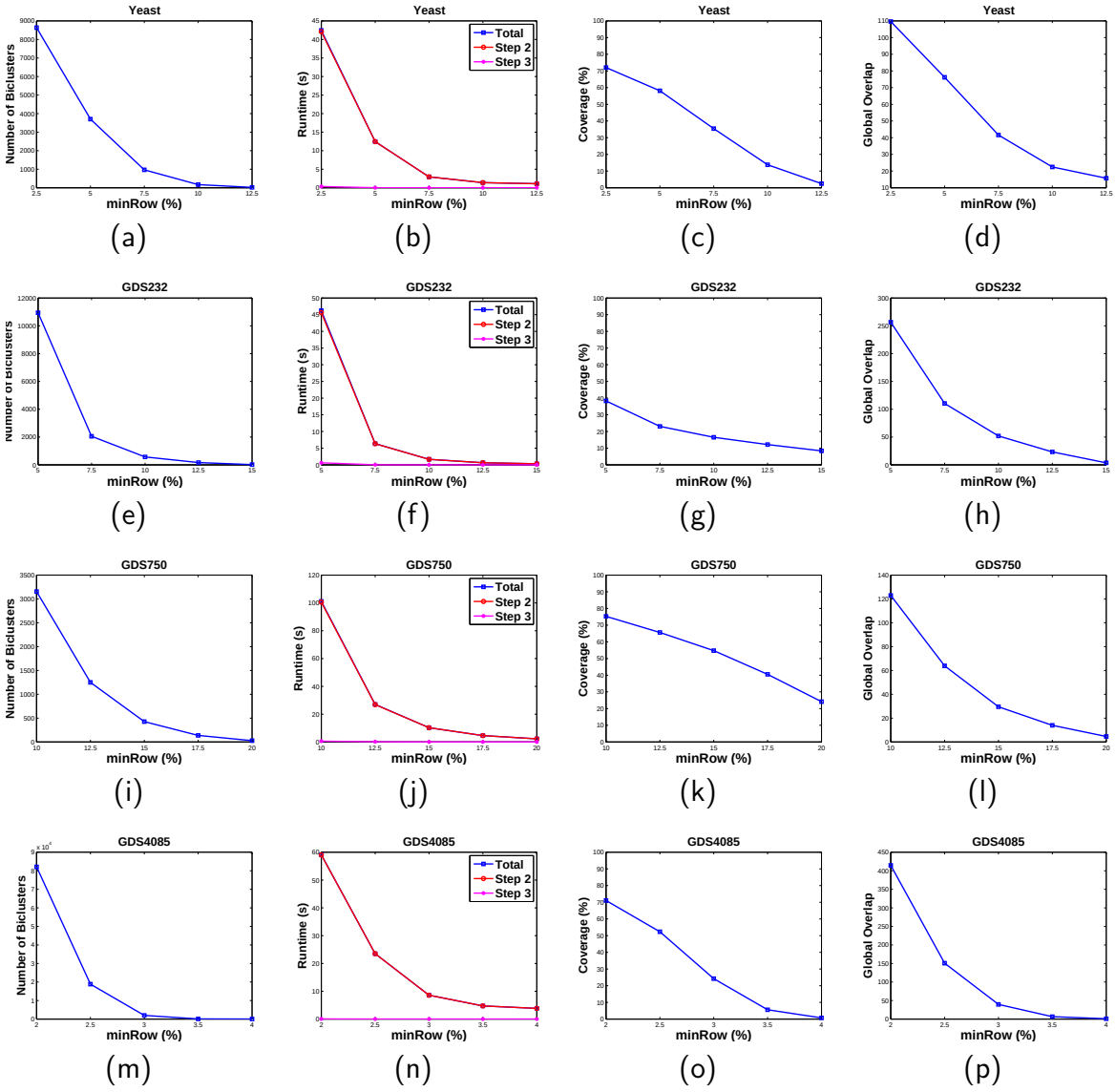


Figure 14 – Results of RIn-Close_CHV's sensitivity to the parameter *minRow*. The parameter ϵ was set to: 5 for Yeast; 3 for GDS232; 3 for GDS750; and 9 for GDS4085.

6.4 Comparison with Heuristics

The results of this section were first reported in [103].

With this experiment, we are going to demonstrate that well-known heuristic-based approaches can fail noticeably when trying to identify the existing biclusters in a dataset. We claim that the results to be presented turn to be a strong motivation for adopting enumerative algorithms, such as the ones proposed in this work. The dataset is also carefully designed so that we can propose a suitable set of parameters for the heuristic-based approaches, given that we are aware of the main attributes of the existing biclusters. So, the disastrous behavior of the heuristic-based approaches cannot be attributed to an unfortunate parameterization. We tested three heuristics that are specialized to mine CHV

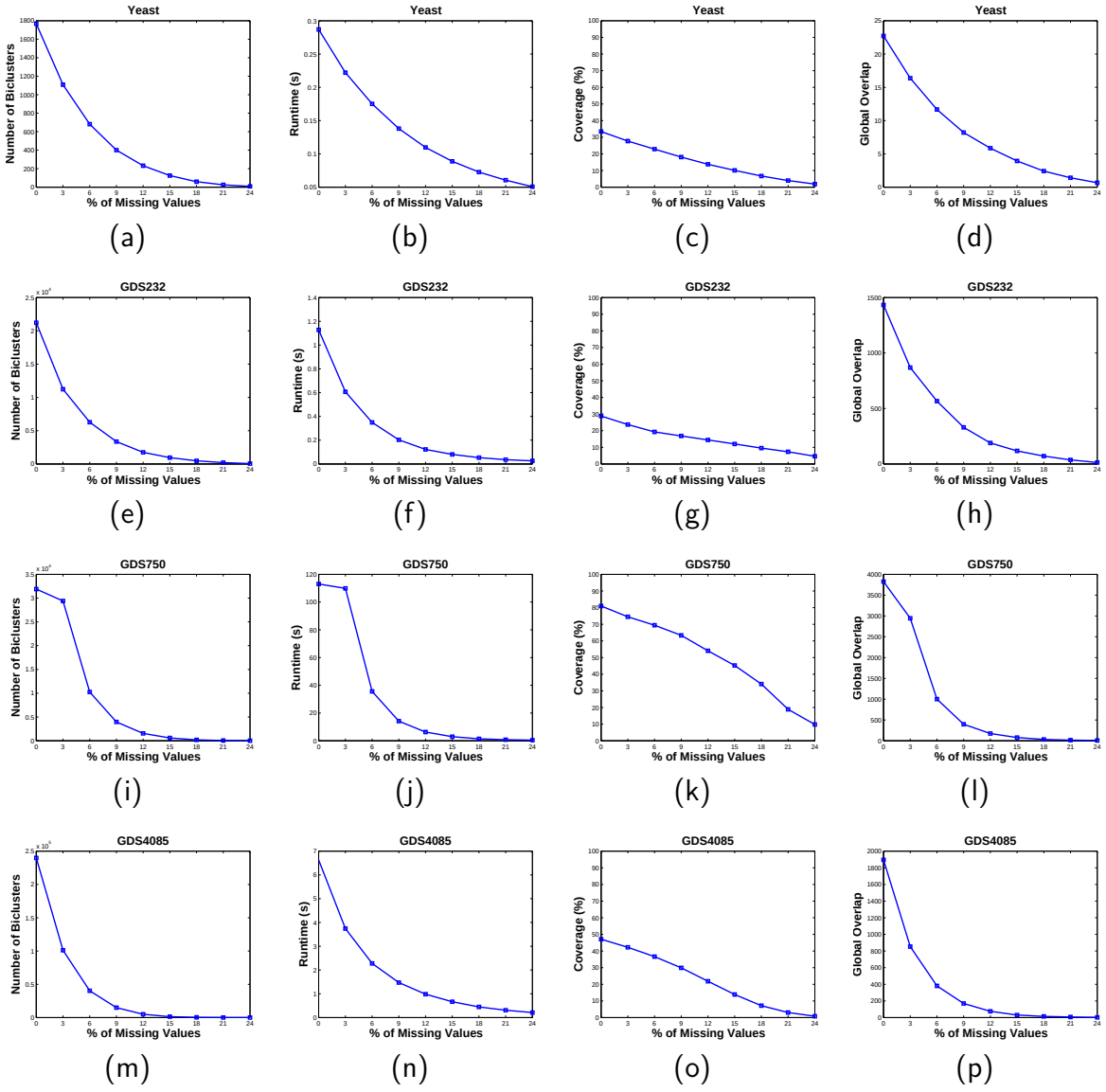


Figure 15 – Results of RIn-Close_CVC's sensitivity to the percentage of missing values in the dataset.

biclusters: CC, FLOC, and ROCC. These contenders were briefly described in Chapter 4. We chose to perform this experiment with the CHV type of biclusters because this is the most general type addressed in this work.

For this experiment, we used the 50 synthetic datasets described in Section 6.1 with the default parameters (i.e., $n = 5000$, $m = 60$, number of biclusters $= 10$, bicluster row size $= 200$, bicluster column size $= 8$, overlap $= 0.2$, and Gaussian noise with $\mu = 0$ and $\sigma^2 = 0.01$). These datasets represent a particular and controlled scenario, i.e., there is a very clear boundary between what should and what should not be part of a bicluster. Possibly, the boundaries are not so accurate in real-world applications. But in this way these dataset allows us to clearly determine the parameters of the biclustering algorithms, and find out what they are able to mine when looking for the original biclusters. For CC

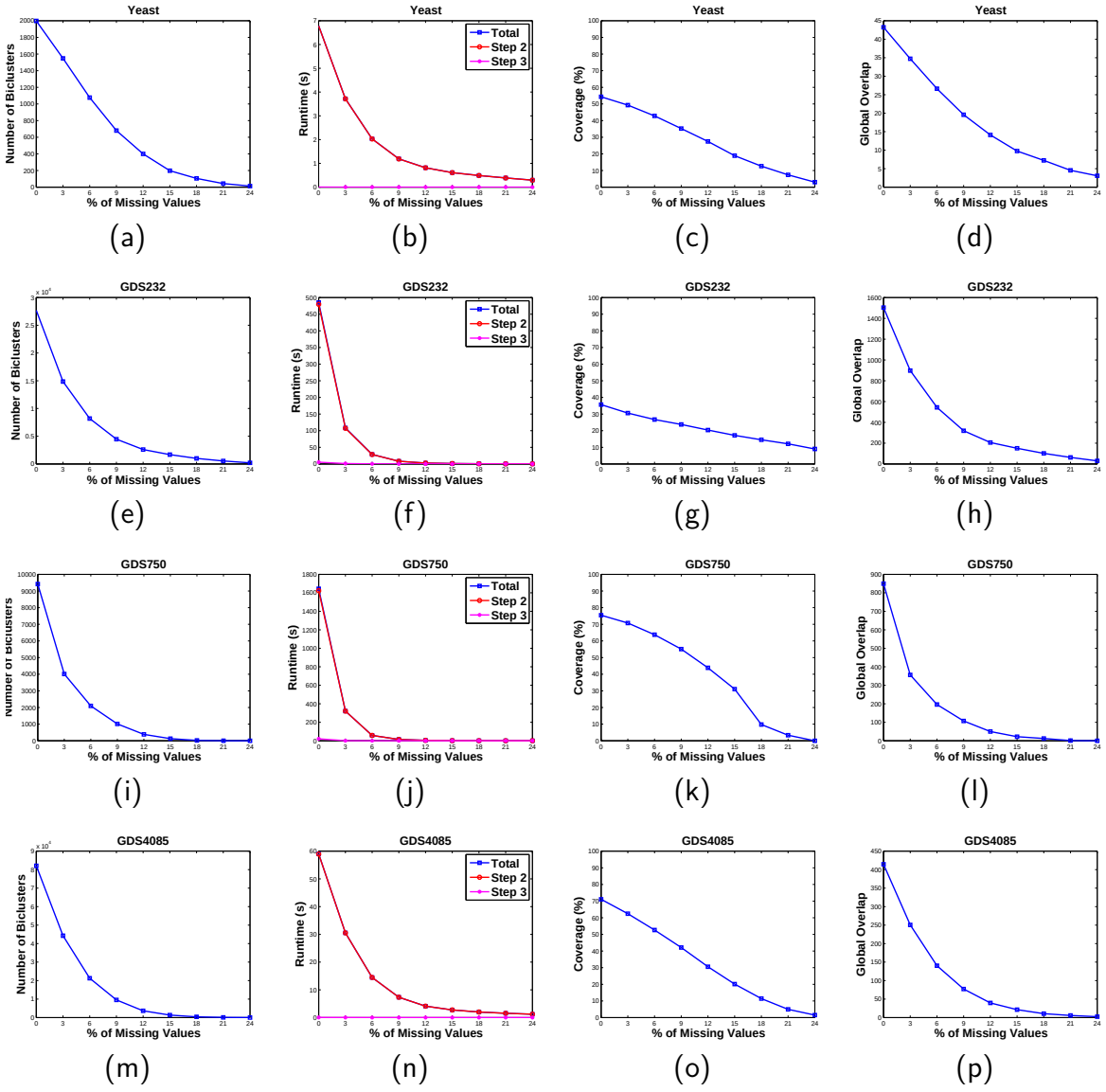


Figure 16 – Results of RIn-Close_CHV's sensitivity to the percentage of missing values in the dataset.

and FLOC, we set the value of δ for each dataset considering its largest bicluster MSR. For both, the number of biclusters to be mined were set to 10. CC's threshold for multiple node deletion α was set to 1.2 (the value suggested by the authors). For FLOC, we set the probability to add a row/column to a seed (initial) bicluster based on the proportion of the minimum number of rows/columns of a bicluster and the total number of rows/columns in the dataset. So, we set these parameters to 0.04 and 0.13, respectively. For ROCC, we set $sr = 1586$ and $sc = 17$ because these are the number of distinct rows/columns covered by the biclusters. Based on the fraction of the number of rows/columns of a dataset over the number of rows/columns of a bicluster, we set $k = 25$ and $l = 8$.

Table 14 shows the results of this experiment. CC completely failed to find the original biclusters. As the values of the parameter δ were very low, though adequate for

Table 14 – Results of the comparison with heuristics.

	Precision	Recall
RIn-Close	1.0000 (0.0000)	1.0000 (0.0000)
CC	-	0.0000 (0.0000)
FLOC	0.0768 (0.0241)	0.0874 (0.0252)
ROCC	0.1831 (0.0355)	0.3845 (0.0655)

the task, CC was unable to find so accurate submatrices. FLOC had a very poor result with low Precision and Recall. As the initial biclusters are generated at random, it is very unlike that FLOC can improve them to the original ones. There is a very clear boundary between what should and what should not be part of a bicluster, thus guiding to a very low MSR for the existing biclusters. ROCC produced better results than CC and FLOC. ROCC starts with a checkboard biclustering structure containing all rows and columns, so it is expected that ROCC would achieve a better Recall than the others. But its routine to refine this initial solution was not accurate, which led to a low Precision, even though its Precision was better than the one produced by the others.

As we have seen, although we knew how to choose suitable parameters for the heuristic-based algorithms, this case study was very challenging to them. On the other hand, RIn-Close easily accomplished this task.

6.5 Comparison with a baseline - Applying enumerative biclustering algorithms to gene ontology enrichment analysis

This experiment aims to demonstrate the usefulness of performing a complete, correct, and non-redundant enumeration. RIn-Close_CHV algorithm is the only one in the literature with these three key properties to enumerate perturbed CHV biclusters. Its competitors, such as MicroCluster, miss some biclusters and do not find the maximal version of others. Our motivation in this section is to answer the question: does the absence of these three properties together lead to information loss to the data analyst? Note that our goal is not to provide a biological analysis of the biclustering solutions, but only to emphasize that the solutions provided by actual enumerative algorithms have more information that should be considered in a biological analysis. We intend to establish partnerships with researchers of the biology field to make us able to provide biological analysis of our biclustering solutions in the future.

We compared RIn-Close_CHV and MicroCluster algorithms in a real-world application, more specifically in *Gene Ontology enrichment analysis* (GOEA). The GOEA indicates if the gene-sets delivered by some method are enriched with respect to some gene ontology (GO) aspect. Given that the concept behind the biclustering approach is

appealing in biosciences, biclustering became a popular tool for discovering local patterns on gene expression data. Also for this reason, GOEA is a usual way for the comparison of biclustering algorithms.

The GO project aims to unify the representation of gene and gene product attributes in a wide variety of organisms. The process of assigning GO terms to gene products is called *annotation*. The use of a consistent vocabulary allows genes from different species to be compared based on their GO annotations. GO covers three aspects: *Molecular Function*, *Biological Process*, and *Cellular Component*. The terms are structured as a directed acyclic graph, where each term has defined relationships to other terms in the same aspect, and sometimes in other aspects.

For this experiment, we used the Yeast dataset and verified whether the gene-sets of the biclusters delivered by the algorithms show significant enrichment with respect to *Biological Process* and *Molecular Function* annotations. The gene annotations were downloaded from the GO Website[4]. For the *Saccharomyces cerevisiae* organism, the number of annotated genes related to biological process and molecular function is 6380; the number of unique GO terms associated with annotated genes is 4483; and the number of gene-GO term associations is 58812.

We used the Matlab Bioinformatics Toolbox [5] to verify the biological significance of the biclustering results. The first step is to count the GO terms. So, for each gene of the microarray dataset, we get the GO terms associated with it, and add 1 to its count. We also have a count of the GO terms for each bicluster. We followed the instructions of the toolbox, and propagated the counting to the neighboring GO terms. It is done to alleviate the problem wherein the gene sets are not accurately annotated. The second step is, for each bicluster, to look at the probabilities that the GO terms are counted by chance. It is done by means of a hypergeometric probability distribution function, which calculates the statistical significance of having drawn a specific number of successes out of a total number of draws from a population. So, this function returns the p-value associated with each GO term, which is the probability of obtaining such test statistics. The most significant GO terms of a bicluster are the ones with the lowest p-values. A bicluster was considered enriched if at least one of its GO terms has a p-value less or equal to $\alpha = 0.05$.

MicroCluster looks for scaling biclusters by default, so we chose to mine this type of bicluster. As RIn-Close_CHV was implemented to mine shifting biclusters, we took the logarithm of the values of the dataset. The adopted parameters are described in Table 15. As SeqClus and CPT, MicroCluster returns biclusters that violate the user-defined parameter ϵ, so that its biclusters have residue below 2ϵ. Thereby, we set the

[4] http://geneontology.org/page/download-annotations, last accessed 05/13/2015

[5] http://www.mathworks.com/help/bioinfo/examples/gene-ontology-enrichment-in-microarray-data.html, last accessed 05/13/2015

Table 15 – Parameters of the algorithms in the experiment of GOEA.

	ϵ	$minRow$	$minCol$
RIn-Close	0.02	100	5
MicroCluster	0.01	100	5

Table 16 – Results of the algorithms in the experiment of GOEA.

	RIn-Close	MicroCluster
Number of biclusters	19	11
Number of maximal biclusters	19	4
Number of enriched biclusters	19	11

parameter ϵ to $\log(2 \times 0.01 + 1)$ for RIn-Close_CHV.

Table 16 shows the results of this experiment. MicroCluster misses 8 of the 19 biclusters that could have been found in the Yeast dataset with this parametrization, which represents a loss of more than 40% of the biclusters. All these 8 missed biclusters are enriched and are, therefore, of great importance to the data analyst. Only 4 biclusters from MicroCluster are maximal, i.e., more than 60% of its biclusters did not include all possible genes in their gene-sets. Moreover, all the maximal versions found by RIn-Close are also enriched. Logically, these 4 maximal biclusters found by MicroCluster are identical to 4 biclusters found by RIn-Close, but it represents only 21% of the total number of biclusters.

We made a more detailed analysis of one of the biclusters found only by RIn-Close, and of one of the maximal versions provided by RIn-Close. Their gene-sets are exhibited in Table 17.

Bicluster *bic_max* has 2 more genes than its non-maximal version found by MicroCluster, that are:

- YDR091C: Essential Fe-S protein; required for ribosome biogenesis, translation initiation/termination; facilitates binding of multifactor complex (MFC) of initiation factors to small ribosomal subunit; Dom34-Hbs1 complex and Rli1p work in dissociating inactive ribosomes, thereby facilitating translation restart; forms complex with Lto1p and Yae1p; dependency on ROS-labile FeS clusters, activity in nuclear ribosomal-subunit export impaired by mild oxidative stress.

- YGR214W: Ribosomal 40S subunit protein S0A; required for maturation of 18S rRNA along with Rps0Bp; deletion of either RPS0 gene reduces growth rate, deletion of both genes is lethal; homologous to human ribosomal protein SA and bacterial S2; RPS0A has a paralog, RPS0B, that arose from the whole genome duplication

Table 17 – Gene-sets of one of the biclusters found only by RIn-Close (*bic_novel*), and of one of the maximal versions provided by RIn-Close (*bic_max*)

Bicluster	Genes
bic_novel	YAL065C, YAL067C, YAR061W, YBL065W, YBL066C, YBR021W, YBR032W, YBR051W, YBR072W, YBR090C, YBR116C, YBR240C, YBR250W, YBR298C, YCL026C-A, YCR022C, YCR032W, YCR063W, YCR081W, YCR091W, YCR107W, YDL071C, YDL109C, YDL113C, YDL206W, YDL208W, YDL210W, YDL216C, YDL239C, YDL247W, YDR118W, YDR171W, YDR241W, YDR247W, YDR256C, YDR259C, YDR274C, YDR401W, YEL004W, YEL006W, YEL016C, YEL052W, YEL070W, YEL073C, YER054C, YER060W, YER096W, YER142C, YER179W, YER187W, YFL030W, YFL043C, YFL054C, YFR023W, YFR032C, YGL006W, YGL063W, YGL096W, YGL155W, YGL170C, YGL229C, YGL230C, YGL249W, YGL250W, YGR043C, YGR058W, YGR065C, YGR088W, YGR122W, YGR150C, YGR197C, YGR249W, YGR271W, YGR288W, YHL042W, YHR014W, YHR044C, YHR048W, YHR157W, YIL013C, YIL015C-A, YIL097W, YIL120W, YIR007W, YJL038C, YJL045W, YJL083W, YJL147C, YJR119C, YJR123W, YKL086W, YKL107W, YKL173W, YKL222C, YKR076W, YKR097W, YLL016W, YLR054C, YLR081W, YLR411W, YLR434C, YML066C, YMR034C, YMR234W, YMR251W, YNL034W, YNL191W, YNL269W, YNL279W, YNR063W, YOL023W, YOL091W, YOL104C, YOR160W, YOR173W, YPL194W, YPR001W, YPR007C, YPR064W
bic_max	YAL028W, YAL065C, YAL067C, YAR061W, YBL065W, YBL066C, YBR032W, YBR051W, YBR090C, YBR174C, YBR240C, YBR250W, YBR298C, YCL003W, YCR022C, YCR056W, YCR063W, YCR081W, YCR091W, YCR107W, YDL071C, YDL109C, YDL113C, YDL206W, YDL210W, YDL216C, YDL239C, YDL242W, YDL247W, YDR091C, YDR118W, YDR204W, YDR247W, YDR256C, YDR259C, YDR273W, YDR274C, YDR362C, YDR401W, YDR436W, YDR506C, YEL004W, YEL006W, YEL052W, YEL070W, YEL073C, YER054C, YER060W, YER096W, YER179W, YFL030W, YFL054C, YFR023W, YGL006W, YGL063W, YGL170C, YGL229C, YGL249W, YGL250W, YGR058W, YGR065C, YGR088W, YGR122W, YGR197C, YGR212W, YGR214W, YGR249W, YGR288W, YHL042W, YHR014W, YHR015W, YHR044C, YHR048W, YHR157W, YIL013C, YIL015C-A, YIL097W, YIL120W, YIR007W, YJL038C, YJL045W, YJL053W, YJL083W, YJL147C, YJR119C, YKL086W, YKL107W, YKL222C, YKR076W, YLL016W, YLR312C, YML066C, YMR034C, YMR251W, YNL034W, YNL035C, YNL083W, YNL146W, YNL191W, YNL194C, YNL203C, YNL269W, YNL279W, YNR062C, YNR063W, YOL023W, YOL101C, YOL163W, YOR173W, YOR181W, YOR292C, YOR300W, YOR358W, YPL194W, YPR001W, YPR007C, YPR026W, YPR064W, YPR095C

Tables 18 and 19 show a list of the ten most significant GO terms associated with the bilcusters *bic_novel* and *bic_max*. To find out more about the terms that appear on these lists, Figures 17 and 18 show a sub-ontology including the ancestors of these terms. The sub-ontology on the left refers to the biological process and the sub-ontology on the right refers to the molecular function. The nodes were colored in a way where bright red is the most significant and bright green is the least significant. These sub-ontology was created simply by getting the ancestors of the ten most significant GO terms, and getting the relationships between them. It is easily done because each GO term has fields for *is_a* and *part_of*. These fields represent the relationships between the GO terms. As GO terms can be seen as nodes in an acyclic graph, we can traverse such relationships looking for the ancestors, descendants, and relatives of a term.

The ancestors of a GO term are its parents in the graph. So, these two figures are showing the relations between less specific GO terms that represent the terms highlighted in Tables 18 and 19. It is usual to use less specific GO terms to facilitate the understanding of the relationships exhibited by the gene-set.

The sub-ontology in Figure 17 has 90 nodes and 176 edges, and the sub-ontology in Figure 18 has 88 nodes and 174 edges. Both of the sub-ontologies are very connected, mainly in the subgraph representing biological process. Thus, the GO terms provided by the two biclusters, *bic_novel* and *bic_max*, may contain relevant information about biological processes and molecular functions of the yeast organism.

Table 18 – The ten most significant GO terms of a bicluster found only by RIn-Close.

GO Term	p-value	Counts	Definition
GO:0000122	0.0010	3 / 5	Any process that stops, prevents, or reduces the frequency, rate or extent of transcription from an RNA polymerase II promoter.
GO:0006366	0.0010	3 / 5	The synthesis of RNA from a DNA template by RNA polymerase II, originating at an RNA polymerase II promoter. Includes transcription of messenger RNA (mRNA) and certain small nuclear RNAs (snRNAs).
GO:0045892	0.0010	3 / 5	Any process that stops, prevents, or reduces the frequency, rate or extent of cellular DNA-templated transcription.
GO:0006357	0.0031	3 / 7	Any process that modulates the frequency, rate or extent of transcription from an RNA polymerase II promoter.
GO:0000083	0.0130	2 / 4	Any process that regulates transcription such that the target genes are involved in the transition between G1 and S phase of the mitotic cell cycle.
GO:0000117	0.0130	2 / 4	Any process that regulates transcription such that the target genes are transcribed as part of the G2/M transition of the mitotic cell cycle.
GO:0000182	0.0130	2 / 4	Interacting selectively and non-covalently with DNA sequences encoding ribosomal RNA.
GO:0000429	0.0130	2 / 4	A transcription regulation process in which the presence of one carbon source leads to the modulation of the frequency, rate, or extent of transcription, from an RNA polymerase II promoter, of specific genes involved in the metabolism of other carbon sources.
GO:0000976	0.0130	2 / 4	Interacting selectively and non-covalently with a specific sequence of DNA that is part of a regulatory region that controls transcription of that section of the DNA. The transcribed region might be described as a gene, cistron, or operon.
GO:0000981	0.0130	2 / 4	Interacting selectively and non-covalently with a specific DNA sequence in order to modulate transcription by RNA polymerase II. The transcription factor may or may not also interact selectively with a protein or macromolecular complex.

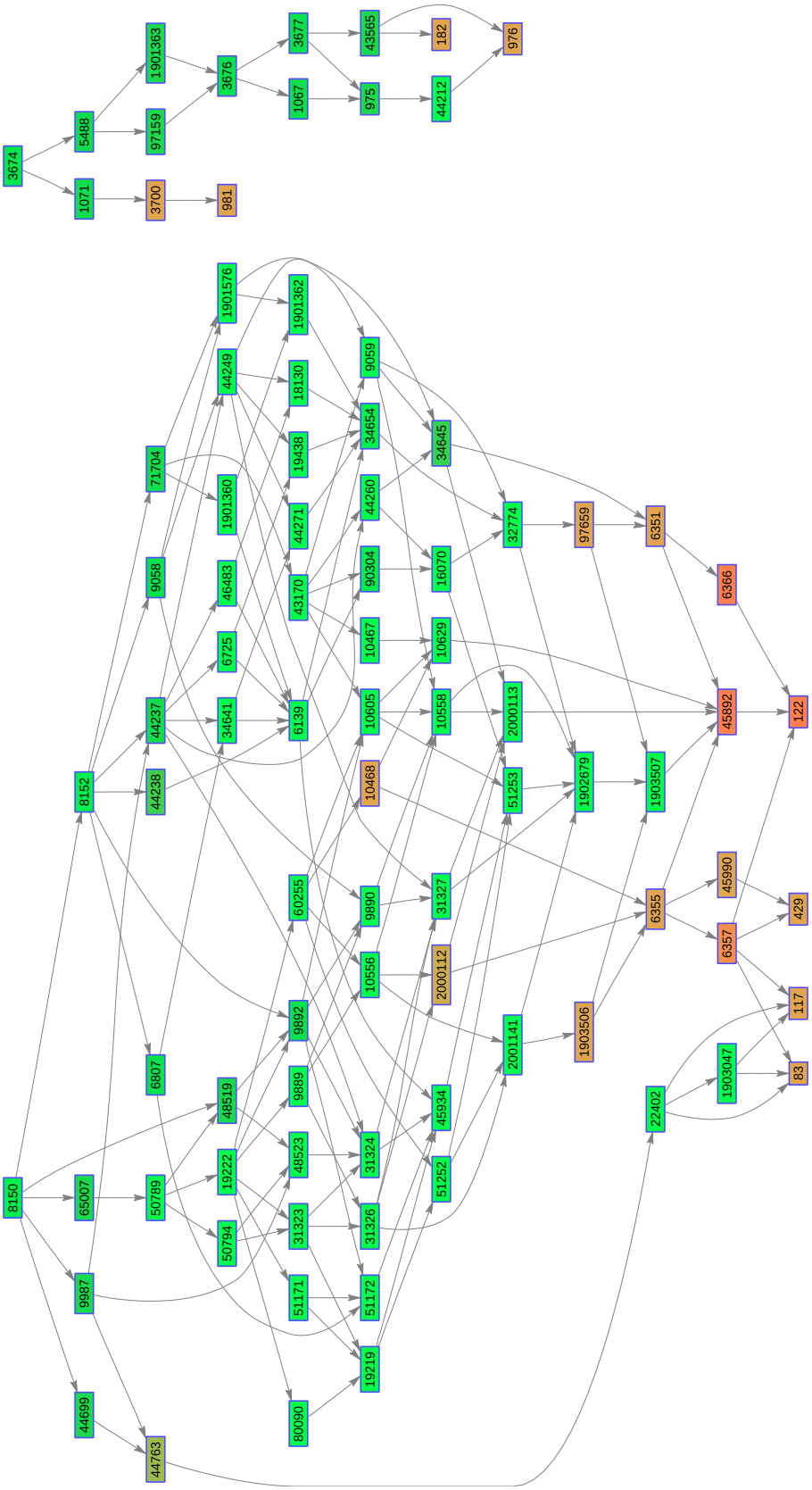


Figure 17 – Ancestors of the ten most significant GO terms of a bicluster found only by RIn-Close.

Table 19 – The ten most significant GO terms of a maximal version of a MicroCluster's bicluster.

GO Term	p-value	Counts	Definition
GO:0003824	0.0000	19 / 298	Catalysis of a biochemical reaction at physiological temperatures. In biologically catalyzed reactions, the reactants are known as substrates, and the catalysts are naturally occurring macromolecular substances known as enzymes. Enzymes possess specific binding sites for substrates, and are usually composed wholly or largely of protein, but RNA that has catalytic activity (ribozyme) is often also regarded as enzymatic.
GO:0000122	0.0023	3 / 5	Any process that stops, prevents, or reduces the frequency, rate or extent of transcription from an RNA polymerase II promoter.
GO:0006366	0.0023	3 / 5	The synthesis of RNA from a DNA template by RNA polymerase II, originating at an RNA polymerase II promoter. Includes transcription of messenger RNA (mRNA) and certain small nuclear RNAs (snRNAs).
GO:0045892	0.0023	3 / 5	Any process that stops, prevents, or reduces the frequency, rate or extent of cellular DNA-templated transcription.
GO:0006357	0.0072	3 / 7	Any process that modulates the frequency, rate or extent of transcription from an RNA polymerase II promoter.
GO:0000083	0.0227	2 / 4	Any process that regulates transcription such that the target genes are involved in the transition between G1 and S phase of the mitotic cell cycle.
GO:0000117	0.0227	2 / 4	Any process that regulates transcription such that the target genes are transcribed as part of the G2/M transition of the mitotic cell cycle.
GO:0000182	0.0227	2 / 4	Interacting selectively and non-covalently with DNA sequences encoding ribosomal RNA.
GO:0000429	0.0227	2 / 4	A transcription regulation process in which the presence of one carbon source leads to the modulation of the frequency, rate, or extent of transcription, from an RNA polymerase II promoter, of specific genes involved in the metabolism of other carbon sources.
GO:0000976	0.0227	2 / 4	Interacting selectively and non-covalently with a specific sequence of DNA that is part of a regulatory region that controls transcription of that section of the DNA. The transcribed region might be described as a gene, cistron, or operon.

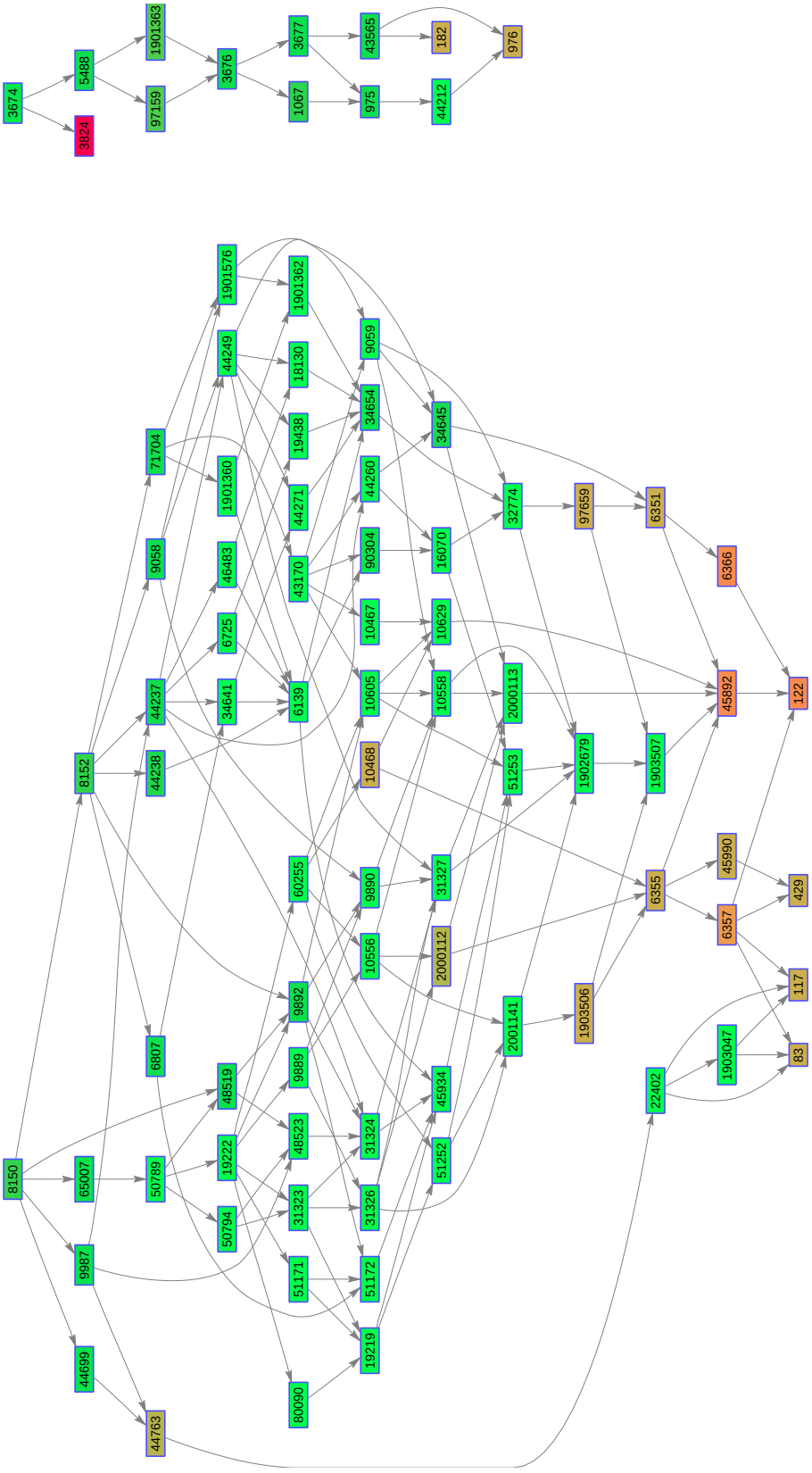


Figure 18 – Ancestors of the ten most significant GO terms of a maximal version of a MicroCluster's bicluster.

As we have seen, the RI-Close_CHV algorithm is able to (*i*) find biclusters that its competitors are unable to find, and (*ii*) find the maximal versions of the biclusters found by its competitors. Given that biclusters not found by MicroCluster and the maximal versions of MicroCluster's biclusters are enriched, we may conclude that there is loss of useful information when we are using biclustering algorithms without the three key properties of being correct, complete and non-redundant.

6.6 Applying enumerative biclustering algorithms to the analysis and identification of biomarkers

In genetics, biomarkers are defined as a set of genes that are associated with a disease or are associated with the susceptibility to develop a specific disease. To improve treatments of cancer, several biomarkers have been proposed for risk prognosis and treatment response [5, 29]. The disturbances in central processes in cancer cells are often due to abnormalities in gene expression. It is known that tumor cells show different gene expression profiles compared to normal tissue but also to tissue obtained from metastases [84]. Recent published biomarkers in many types of cancer contain numerous genes and are mainly based on gene expression, generated using microarray profiling or RNA-Seq technologies [5]. The identification of biomarkers, i.e, set of genes whose expression change is highly correlated with the disease, poses a great challenge to data analysis approaches [84].

Usually, identified biomarkers are developed to a specific cancer tissue and subtypes. In breast cancer, for example, more than 40 biomarkers have been proposed containing between 3 and 512 genes and whose prognostic or predictive performance depends on therapy, hormone receptor status, and the number of genes [5]. Here, we will analyze biomarkers in *the small, round blue cell tumors* (SRBCTs) of childhood, which include neuroblastoma (NB), rhabdomyosarcoma (RMS), non-Hodgkin lymphoma (NHL) and the Ewing family of tumors (EWS). Accurate diagnosis of SRBCTs is essential because the treatment options, responses to therapy and prognoses vary widely depending on the diagnosis [60]. These cancers are difficult to distinguishing by light microscopy, and currently no single test can precisely distinguish these cancers [60]. The biomarkers may be useful in distinguish between these four types of cancer. We use an enumerative biclustering algorithm, more specifically RIn-Close_CVC, to (*i*) analyze two sets of biomarkers proposed in the literature to classify these four types of cancer, and (*ii*) propose a set of biomarkers from scratch. Our proposals lead to simple human interpretable rules for distinguishing between these four cancers.

Our rules are based on *class association rules* (CARs), defined in Subsection 3.2.1, with one major difference: those concepts considers a binary relation $I \subseteq G \times M$ between

the objects G and attributes M of the data matrix. So, the rules are extracted from CTV biclusters of ones. But, we are handling numerical datasets in our case, and we will extract the rules from CVC biclusters. Therefore, the rules require one more information in their antecedent: a range of values associated with each attribute of the antecedent. So, the antecedent of a rule $X \Rightarrow c$ will be a set of attributes associated with ranges: $X = \{x_1[l_1, u_1], x_2[l_2, u_2], ...\}$. For instance, a CAR $x_1[l_1, u_1], x_2[l_2, u_2] \Rightarrow 1$ means that if an object has the attribute $x_1 \in [l_1, u_1]$ and the attribute $x_2 \in [l_2, u_2]$, then the class label of the object is equal to 1. Despite this extra information, the concepts described in Subsection 3.2.1 remains valid when we consider a closure operator adapted to handle CVC biclusters in numerical datasets:

$$X' = \{g \in G | \forall x[l, u] \in X : g_x \in [l, u]\}, \tag{6.1}$$

where $X = \{x_1[l_1, u_1], x_2[l_2, u_2], ...\}$ is a set of attributes associated with ranges, and g_x means the value of the object g on the attribute x.

We perform our analysis on the same dataset used by Khan *et al.* [60] and by Pal *et al.* [90]. The dataset consists of expression levels of 2,308 genes, which were obtained from glass-slide cDNA microarrays, prepared in accordance with the standard protocol of the National Human Genome Research Institute [90]. The dataset has 88 samples (EWS:29, BL:11, NB:18, RMS:25, and 5 non-SRBCT samples). Rows of the dataset refer to the samples and columns refer to the genes. Each gene has an *Image Id* associated with it. We will report the *Image Id* to ease the reading of our results. The description of the genes is available on the web [6].

Khan *et al.* [60] proposed a set of 96 genes as biomarkers, and Pal *et al.* [90] proposed a set of only 7 genes as biomarkers to identify SRBCTs, see Table 20. There are only four genes that are common to these two sets of biomarkers: 325182, 784224, 812105, 814260. Therefore, these two biomarkers are very different in size and composition. The smaller the set of genes, the easier it is to be assessed in real scenarios. However, few genes may not be sufficient to distinguish between the four types of SRBCTs.

We will analyze these two sets of biomarkers and provide some insights about them. First of all, we will answer the following questions:

- Are the proposed biomarkers sufficient to generate rules that discriminate all the SRBCTs?

  If yes, are all the genes necessary to generate such rules?

  If no, how many genes do we need to add to generate such rules?

[6] `http://research.nhgri.nih.gov/microarray/Supplement/`, last accessed 05/01/2016

Table 20 – Biomarkers proposed by Khan *et al.* [60] and by Pal *et al.* [90] to identify SRBCTs.

Ref.	Proposed Biomarkers
[60]	21652 39093 41591 42558 43733 44563 45291 45542 52076 80109 80338 82225 122159 135688 183337 200814 204545 207274 208718 212542 233721 241412 244618 245330 246377 289645 291756 292522 293500 295985 296448 297392 298062 308163 308231 308497 323371 324494 **325182** 357031 364934 365826 377048 377461 377671 377731 383188 395708 416959 417226 461425 486110 486787 504791 563673 609663 629896 714453 745343 755599 755750 756556 767183 768370 769959 770394 782503 **784224** 784257 784593 788107 796258 809901 809910 811000 **812105** 812965 813266 **814260** 814526 824602 839736 840942 841620 841641 854899 859359 866702 868304 878280 897788 1409509 1416782 1435862 1469292 1473131
[90]	143306 **325182** 745019 770868 **784224 812105 814260**

Then, we will provide new gene-sets and human interpretable rules based on our analysis of the biomarkers proposed by Khan *et al.* [60] and by Pal *et al.* [90] to identify SRBCTs. After that, we will propose biomarkers from scratch and compare then with these two proposals. Note that once more, our goal is not to provide a biological analysis of the results, but we intend to be able to provide such analysis in the future.

RIn-Close_CVC were applied in this experiment. We set $minRow = 3$ and $minCol = 1$. We scale the values of each column of the data matrix to the interval [0, 10,000] in order to use the same value of ϵ for all columns. Our analysis considers only the biclusters composed of samples from the same class, named here *pure biclusters*. We discarded the biclusters that have samples from more than one class. Nevertheless, the number of pure biclusters was high. Therefore, we filtered our results in order to have a small set of biclusters that represents the original result without loss. The purpose of our filter is to get few biclusters that cover the same samples as the original result. With few biclusters, we can produce a small set of rules for distinguishing between EWS, BL, NB, and RMS tumors, making it more simple and interpretable for humans. As our biclusters are maximal and pure, **all the rules have 100% of confidence**. We could have resorted to other filters, in consonance with the main purpose of the research, for example, obtaining the smallest set of biomarkers. In our case, we choose to have few biclusters capable of describing the four types of cancers. Briefly, our filter works as described in Algorithm 6.1.

Algorithm 6.1 Filter of biclustering solutions

```
while not reaching the original coverage do
    Select the bicluster yielding the greatest coverage
end while
```

We start by analyzing the biomarkers proposed in [60]. For it, we only consider the 96 columns of the dataset that correspond to the proposed biomarkers, so we ran RIn-Close_CVC in a dataset with 88 rows and 96 columns. Table 21 shows the results.

Table 21 – Analysis of the RIn-Close_CVC's solution considering the biomarkers proposed in [60].

		$\epsilon = 100$	$\epsilon = 200$	$\epsilon = 300$	$\epsilon = 400$	$\epsilon = 500$
Original solution	# of bics	3,641	11,166	24,355	50,237	86,904
	Avg. Ext.	4.03 ± 1.20	4.37 ± 1.39	4.80 ± 1.52	5.44 ± 1.78	5.92 ± 1.92
	Avg. Int.	7.79 ± 5.39	11.20 ± 8.81	11.98 ± 8.61	12.98 ± 8.14	14.24 ± 7.67
	Avg. Vol.	31.58 ± 22.08	50.09 ± 45.49	55.88 ± 43.38	67.27 ± 42.48	79.83 ± 40.29
	# of genes	96	96	96	96	96
Filtered solution	# of bics	17	14	13	10	9
	Avg. Ext.	6.47 ± 2.27	8.93 ± 2.34	10.69 ± 3.01	11.90 ± 3.21	14.44 ± 3.57
	Avg. Int.	2.88 ± 1.50	4.57 ± 3.86	6.69 ± 6.88	7.90 ± 8.99	10.56 ± 13.21
	Avg. Vol.	20.06 ± 16.53	43.43 ± 44.41	73.77 ± 78.07	92.80 ± 98.42	133.78 ± 138.51
	# of genes	27	29	41	44	57
% of Coverage		100.00	100.00	100.00	100.00	100.00

[# of bics.] Number of biclusters; [Avg. Ext] Average and standard deviation of the size of the biclusters' extents; [Avg. Int.] Average and standard deviation of the size of the biclusters' intents; [Avg. Vol.] Average and standard deviation of the biclusters' volume; [# of genes] Number of genes presented in at least one bicluster.

For all values of ϵ that we tested, all the genes were present in at least one bicluster of the solution. The size of the extents, the size of the intents, and the volume of the biclusters increased with the value of ϵ. All solutions cover 100% of the SRBCT samples. The filtered solution has a much smaller number of biclusters, being much easier to be analyzed by humans. The filtered biomarkers are of a smaller size when compared to the original ones, which indicates that we need less than 96 genes to distinguish between the four types of SRBCTs. The filtered biomarkers are exhibited in Table 22. As in the original solution, the smaller the value of ϵ, the smaller the size of the extents and intents, which has a direct impact on the resulting rules. Tables 23 and 24 show the rules corresponding to the filtered solutions with $\epsilon = 200$ and $\epsilon = 500$, respectively. With smaller values of ϵ, the trend is to have smaller biclusters, requiring a larger number of biclusters to cover all SRBCT samples. On the other hand, the number of genes in the biomarker is smaller. The best options is a trade-off between number of rules and number of genes acting as biomarkers.

To exemplify one of the human interpretable rules presented here, let us observe the third rule in Table 23: 41591[0.27,0.35], 714453[0.10,0.32] $\Rightarrow$ EWS. It indicates that if the expression of the gene 41591 is in the interval $[0.27, 0.35]$ and the expression of the gene 714453 is in the interval $[0.10, 0.32]$, then the sample is from the class EWS. The rules are exhibiting the original values of the dataset in the intervals.

Table 25 shows the results corresponding to the analysis of the seven biomarkers proposed in [90]. We only consider the 7 columns of the dataset that correspond to the proposed biomarkers, so we ran RIn-Close_CVC in a dataset with 88 rows and 7 columns. With only these 7 genes, we were not able to cover all SRBCT samples, although we have a high coverage. The highest coverages were achieved considering $\epsilon = 600$ and $\epsilon = 1000$. We tested values of $\epsilon < 600$, but the coverage was even smaller. With $\epsilon = 1000$, we have less biclusters than with $\epsilon = 900$, and the size of the extents was smaller in average. It happens because with a too loose value of ϵ, it becomes increasingly more difficult to find

Table 22 – Biomarkers corresponding to the filtered solutions of Table 21.

ϵ	Biomarkers
100	41591 42558 44563 80109 80338 82225 204545 233721 246377 292522 295985 324494 325182 364934 377461 377671 383188 486787 714453 756556 769959 770394 784224 784593 840942 841641 878280
200	41591 43733 44563 45542 80109 82225 183337 233721 245330 246377 292522 324494 325182 357031 364934 377461 377671 461425 486787 714453 756556 769959 770394 782503 784593 812105 814526 859359 1435862
300	41591 43733 44563 45542 52076 80109 82225 183337 207274 233721 246377 292522 295985 298062 308163 323371 324494 357031 364934 377461 461425 486787 563673 629896 714453 756556 770394 782503 784224 784593 809910 811000 812105 814526 840942 841641 854899 859359 868304 878280 1435862
400	21652 41591 43733 44563 45542 52076 80109 82225 183337 200814 207274 233721 245330 246377 292522 297392 298062 308163 323371 324494 325182 357031 364934 377461 461425 486787 563673 609663 629896 714453 755750 756556 769959 770394 782503 784224 809910 811000 814526 824602 840942 854899 859359 1435862
500	21652 41591 43733 44563 45542 52076 80109 82225 122159 183337 207274 233721 244618 245330 246377 291756 292522 293500 295985 297392 298062 308163 323371 324494 325182 357031 364934 365826 377461 417226 461425 486787 563673 609663 629896 714453 745343 755750 756556 768370 769959 770394 782503 784224 784593 809910 811000 814526 824602 840942 854899 859359 866702 868304 878280 1409509 1435862

Table 23 – Rules corresponding to the filtered solution of Table 21 considering $\epsilon = 200$.

#	Rule	$Comp(R)$
1	1714453[0.05,0.31], 82225[0.05,0.13], 44563[0.05,0.40], 324494[0.10,0.28] ⇒ EWS	44.83%
2	770394[0.10,0.31], 714453[0.05,0.30], 82225[0.10,0.18], 324494[0.10,0.24], 325182[0.09,0.18], 292522[0.10,0.24] ⇒ EWS	31.03%
3	41591[0.27,0.35], 714453[0.10,0.32] ⇒ EWS	34.48%
4	377461[0.09,0.66], 714453[0.05,0.31], 82225[0.01,0.09], 44563[0.03,0.28], 324494[0.06,0.28], 756556[0.13,0.30] ⇒ EWS	34.48%
5	82225[0.05,0.13], 324494[0.10,0.32], 756556[0.13,0.32] ⇒ EWS	44.83%
6	486787[0.05,0.12], 770394[0.05,0.25], 233721[0.08,0.25], 80109[0.07,0.42], 782503[0.06,0.13], 377461[0.08,0.36], 183337[0.04,0.14], 714453[0.05,0.23], 44563[0.04,0.17], 324494[0.06,0.25], 461425[0.05,0.21], 859359[0.10,0.20], 45542[0.10,0.25], 1435862[0.07,0.17], 364934[0.13,0.24], 246377[0.07,0.31], 292522[0.05,0.24] ⇒ BL	100.00%
7	461425[0.08,0.26], 784593[0.04,0.16], 756556[0.11,0.27], 364934[0.09,0.39], 292522[0.06,0.21] ⇒ NB	55.56%
8	324494[0.06,0.29], 756556[0.13,0.32], 325182[0.16,0.23], 364934[0.31,0.65] ⇒ NB	38.89%
9	812105[0.12,0.18], 714453[0.03,0.19], 324494[0.06,0.16], 461425[0.10,0.27], 245330[0.08,0.13] ⇒ NB	38.89%
10	814526[0.17,0.46], 246377[0.34,0.63] ⇒ RMS	32.00%
11	486787[0.17,0.34], 770394[0.15,0.29], 43733[0.33,0.48] ⇒ RMS	28.00%
12	770394[0.14,0.32], 357031[0.16,0.22], 769959[0.09,0.18] ⇒ RMS	24.00%
13	377671[0.12,0.19], 769959[0.17,0.29] ⇒ RMS	24.00%
14	814526[0.42,0.72], 246377[0.36,0.66] ⇒ RMS	32.00%

Table 24 – Rules corresponding to the filtered solution of Table 21 considering $\epsilon = 500$.

#	Rule	$Comp(R)$
1	714453[0.05,0.31], 82225[0.03,0.19], 324494[0.10,0.43], 756556[0.14,0.62], 325182[0.09,0.35] ⇒ EWS	75.86%
2	183337[0.05,0.34], 714453[0.05,0.41], 324494[0.06,0.32], 756556[0.13,0.35], 293500[0.44,0.69] ⇒ EWS	48.28%
3	365826[0.15,0.41], 770394[0.11,0.67], 714453[0.08,0.41], 298062[0.18,0.44], 324494[0.10,0.39], 756556[0.18,0.49], 292522[0.06,0.45] ⇒ EWS	55.17%
4	207274[0.03,0.12], 486787[0.05,0.12], 770394[0.05,0.25], 244618[0.09,0.49], 233721[0.08,0.25], 629896[0.08,0.23], 840942[0.04,0.21], 80109[0.07,0.42], 41591[0.05,0.14], 782503[0.06,0.13], 377461[0.08,0.36], 52076[0.06,0.19], 811000[0.02,0.13], 308163[0.07,0.55], 183337[0.04,0.14], 714453[0.05,0.23], 298062[0.04,0.17], 44563[0.04,0.17], 324494[0.06,0.25], 563673[0.03,0.09], 417226[0.03,0.33], 122159[0.05,0.34], 609663[0.06,0.25], 461425[0.05,0.21], 809910[0.13,0.53], 824602[0.04,0.21], 245330[0.08,0.24], 1409509[0.08,0.25], 756556[0.12,0.34], 21652[0.06,0.36], 745343[0.05,0.45], 859359[0.10,0.20], 45542[0.10,0.25], 323371[0.10,0.21], 293500[0.08,0.30], 1435862[0.07,0.17], 814526[0.05,0.39], 364934[0.13,0.24], 246377[0.07,0.31], 291756[0.07,0.25], 769959[0.08,0.33], 854899[0.06,0.16], 755750[0.06,0.22], 292522[0.05,0.24], 768370[0.12,0.32] ⇒ BL	100.00%
5	714453[0.03,0.61], 324494[0.06,0.51], 461425[0.08,0.47], 756556[0.13,0.42], 784224[0.25,0.51], 325182[0.11,0.33], 814526[0.79,1.59], 364934[0.09,0.65], 292522[0.03,0.21] ⇒ NB	72.22%
6	486787[0.08,0.50], 295985[0.05,0.21], 866702[0.07,0.21], 714453[0.05,0.25], 324494[0.07,0.51], 878280[0.05,0.14], 122159[0.05,0.28], 756556[0.11,0.31], 297392[0.28,0.88], 1435862[0.03,0.17], 364934[0.17,0.50], 868304[0.06,0.35] ⇒ NB	55.56%
7	486787[0.13,0.55], 770394[0.12,0.40], 43733[0.14,0.55], 814526[0.17,0.81], 246377[0.04,0.68] ⇒ RMS	68.00%
8	770394[0.12,0.65], 357031[0.08,0.23], 769959[0.14,0.44] ⇒ RMS	52.00%
9	486787[0.13,0.55], 770394[0.14,0.40], 43733[0.14,0.55], 784593[0.16,0.47] ⇒ RMS	56.00%

Table 25 – Analysis of the RIn-Close_CVC's solution considering the biomarkers proposed in [90]

		$\epsilon = 600$	$\epsilon = 700$	$\epsilon = 800$	$\epsilon = 900$	$\epsilon = 1000$
Original solution	# of bics	194	230	264	318	285
	Avg. Ext.	3.84 ± 1.28	4.17 ± 1.51	4.44 ± 1.47	5.11 ± 2.31	4.87 ± 2.30
	Avg. Int.	3.45 ± 0.88	3.63 ± 0.90	3.84 ± 0.90	4.01 ± 0.94	4.19 ± 0.97
	Avg. Vol.	12.79 ± 3.43	14.75 ± 5.02	16.59 ± 5.40	19.87 ± 8.73	19.79 ± 9.04
	# of genes	7	7	7	7	7
Filtered solution	# of bics	24	27	23	10	20
	Avg. Ext.	4.62 ± 2.02	4.74 ± 2.14	4.83 ± 1.80	5.71 ± 2.35	6.25 ± 2.86
	Avg. Int.	2.96 ± 0.75	3.11 ± 0.75	3.13 ± 0.76	3.19 ± 0.87	3.45 ± 1.28
	Avg. Vol.	13.04 ± 4.35	14.44 ± 6.20	14.65 ± 5.31	17.90 ± 8.28	20.65 ± 10.16
	# of genes	7	7	7	7	7
% of Coverage		97.59	96.39	95.18	95.18	97.59

[# of bics.] Number of biclusters; [Avg. Ext] Average and standard deviation of the size of the biclusters' extents; [Avg. Int.] Average and standard deviation of the size of the biclusters' intents; [Avg. Vol.] Average and standard deviation of the biclusters' volume; [# of genes] Number of genes presented in at least one bicluster.

pure biclusters. The filtered biomarkers has the same number of genes when compared to the original solutions, being possible only to significantly reduce the number of biclusters (consequently also the number of rules).

Analyzing the samples not covered in the solution of Table 25 with $\epsilon = 600$, and the important genes to cover these samples in the solutions of Table 21, we decided to add the gene of *Image Id.* 324494 to the set of genes proposed in [90]. The result is showed in Table 26. With only this extra gene, we were able to cover all SRBCT samples. Table 27 shows the rules corresponding to this solution. We note that these rules have a small completeness when compared to the rules in Tables 23 and 24. It is expected, given that we have small biclusters due to a very restrict number of genes to classify the cancers.

We also worked in proposing biomarkers from scratch. So, we applied our biclustering algorithm in the entire dataset (with 88 samples and 2,308 genes). To reduce the number of biclusters in the final solution, we parametrized RIn-Close_CVC as follows:

Table 26 – Analysis of the RIn-Close_CVC's solution considering the biomarkers proposed in [90] plus the gene 324494.

		$\epsilon = 600$
Original solution	# of bics	286
	Avg. Ext.	3.85 ± 1.23
	Avg. Int.	3.77 ± 1.10
	Avg. Vol.	14.09 ± 4.69
	# of genes	8
Filtered solution	# of bics	23
	Avg. Ext.	5.00 ± 2.15
	Avg. Int.	3.00 ± 0.95
	Avg. Vol.	14.43 ± 6.14
	# of genes	8
% of Coverage		100.00

[# of bics.] Number of biclusters; [Avg. Ext] Average and standard deviation of the size of the biclusters' extents; [Avg. Int.] Average and standard deviation of the size of the biclusters' intents; [Avg. Vol.] Average and standard deviation of the biclusters' volume; [# of genes] Number of genes presented in at least one bicluster.

Table 27 – Rules corresponding to the filtered solution of Table 26.

#	Rule	$Comp(R)$
1	143306[0.19,0.61], 814260[0.27,0.61]⇒ EWS	37.93%
2	812105[0.01,0.18], 770868[0.10,0.28], 325182[0.34,0.41], 143306[0.35,0.53]⇒ EWS	24.14%
3	770868[0.14,0.33], 325182[0.36,0.45], 143306[0.48,0.92]⇒ EWS	20.69%
4	770868[0.11,0.26], 325182[0.25,0.35], 143306[0.25,0.61]⇒ EWS	20.69%
5	143306[0.18,0.51], 324494[2.98,3.38]⇒ EWS	13.79%
6	770868[0.04,0.22], 143306[0.35,0.79], 324494[1.47,1.75]⇒ EWS	17.24%
7	784224[0.45,0.72], 770868[0.15,0.33], 143306[0.46,0.92]⇒ EWS	13.79%
8	812105[0.06,0.14], 814260[0.10,0.26], 324494[0.29,0.51]⇒ BL	72.73%
9	812105[0.08,0.23], 770868[0.11,0.16], 143306[0.50,0.77], 324494[0.78,1.14]⇒ BL	27.27%
10	812105[0.01,0.07], 770868[1.09,1.27], 143306[0.25,0.51], 324494[0.48,0.63]⇒ NB	33.33%
11	812105[0.05,0.14], 325182[0.28,0.37], 745019[0.30,0.43], 143306[0.17,0.60]⇒ NB	22.22%
12	812105[0.07,0.21], 143306[0.36,0.40], 814260[3.74,4.00], 324494[0.58,0.90]⇒ NB	22.22%
13	812105[0.06,0.15], 143306[0.17,0.59], 814260[2.87,3.08]⇒ NB	27.78%
14	784224[0.51,0.72], 812105[0.07,0.21], 770868[0.90,1.09], 143306[0.37,0.51], 324494[0.31,0.62]⇒ NB	16.67%
15	812105[0.06,0.17], 745019[0.43,0.52], 143306[0.36,0.60], 324494[0.87,1.24]⇒ NB	16.67%
16	812105[0.13,0.21], 325182[0.76,0.86], 143306[0.38,0.59]⇒ NB	16.67%
17	814260[0.33,0.68], 324494[0.47,1.02]⇒ RMS	36.00%
18	745019[0.27,0.35], 814260[0.86,1.10], 324494[0.38,0.92]⇒ RMS	16.00%
19	784224[1.62,1.89], 324494[0.31,0.82]⇒ RMS	24.00%
20	784224[2.79,2.89], 324494[0.19,0.66]⇒ RMS	12.00%
21	745019[1.02,1.13], 814260[0.70,0.86], 324494[0.35,0.77]⇒ RMS	12.00%
22	770868[0.38,0.54], 143306[1.01,1.43]⇒ RMS	12.00%
23	812105[1.38,1.45]⇒ RMS	20.00%

Table 28 – Analysis of the RIn-Close_CVC's solution considering the entire dataset.

		$\epsilon = 300$
Original solution	# of bics	76,554
	Avg. Ext.	8.73 ± 1.03
	Avg. Int.	11.52 ± 4.00
	Avg. Vol.	99.39 ± 32.18
	# of genes	641
Filtered solution	# of bics	9
	Avg. Ext.	12.67 ± 3.28
	Avg. Int.	11.00 ± 14.32
	Avg. Vol.	130.33 ± 154.62
	# of genes	62
% of Coverage	100.00	

[# of bics.] Number of biclusters; [Avg. Ext] Average and standard deviation of the size of the biclusters' extents; [Avg. Int.] Average and standard deviation of the size of the biclusters' intents; [Avg. Vol.] Average and standard deviation of the biclusters' volume; [# of genes] Number of genes presented in at least one bicluster.

Table 29 – Biomarkers corresponding to the filtered solution of Table 28.

ϵ	Biomarkers
300	**39093** 40643 **43733** **44563** **45542** 78353 **82225** 110503 **122159** 124605 **135688** 138672 196992 2**04545** **207274** **208718** **212542** 214990 **233721** **244618** **245330** **246377** 263716 283751 284001 292171 **295985** **296448** **298062** 300051 307532 307660 **308163** 345553 361974 **365826** 460487 **461425** 491692 502055 628357 714106 741831 **755750** **756556** 768443 769890 **769959** ***784224*** 796475 809694 **809901** 812033 813254 **813266** 813823 **839736** **841641** **878280** 1323448 **1409509** 1474174

$minRow = 8$, $minCol = 1$, and $\epsilon = 300$. Table 28 shows the result. The number of filtered biclusters are much smaller than the number of original biclusters, and the same can be said about the number of genes in the biomarkers. The genes in the biomarkers is shown in Table 29. It has 31 genes in common with the biomarker proposed by Khan *et al.* [60], and 1 gene in common with the biomarker proposed by Pal *et al.* [90] (this gene is also part of the biomarkers proposed by Khan *et al.* [60]). These 31 genes are highlighted in Table 29. The rules from this solution are presented in Table 30.

The 31 genes in common to the solution in [60], being 1 in common to the solution in [90] too, indicates that our methodology is coherent in defining biomarkers. It is difficult to know, without a biological analysis, what is the best gene-set, among those presented here, to discriminate the SRBCTs, but different methodologies share 31 genes. It is a evidence for us of the importance of these genes, mainly the gene 784224 that appears in [60], [90], and in our solution from scratch. Moreover, these 31 genes are sufficient to discriminate between all the SRBCT samples.

Table 30 – Rules corresponding to the filtered solution of Table 28.

#	Rule	$Comp(R)$
1	138672[0.07,0.40], 245330[0.05,0.41], 298062[0.10,0.39], 461425[0.20,0.49], 628357[0.15,0.42] ⇒ EWS	65.52%
2	207274[0.09,1.05], 245330[0.05,0.41], 295985[0.01,0.13], 296448[0.03,0.46], 298062[0.06,0.32], 461425[0.13,0.35] ⇒ EWS	58.62%
3	44563[0.15,0.36], 110503[0.10,0.21], 245330[0.05,0.32], 298062[0.10,0.43], 502055[0.15,0.55], 769890[0.19,0.32], 784224[0.25,0.49] ⇒ EWS	37.93%
4	39093[0.03,0.12], 40643[0.05,0.20], 43733[0.05,0.12], 44563[0.05,0.25], 45542[0.08,0.25], 82225[0.04,0.21], 122159[0.07,0.42], 124605[0.09,0.20], 135688[0.05,0.14], 196992[0.04,0.13], 204545[0.06,0.13], 207274[0.08,0.36], 208718[0.06,0.19], 212542[0.02,0.13], 214990[0.10,0.37], 233721[0.07,0.55], 244618[0.04,0.14], 245330[0.05,0.23], 246377[0.04,0.17], 263716[0.19,0.45], 284001[0.10,0.29], 292171[0.33,0.62], 296448[0.04,0.17], 298062[0.06,0.25], 307660[0.05,0.14], 308163[0.03,0.09], 345553[0.07,0.20], 361974[0.04,0.18], 365826[0.05,0.21], 460487[0.09,0.16], 461425[0.12,0.34], 491692[0.09,0.16], 502055[0.09,0.30], 628357[0.07,0.17], 714106[0.05,0.14], 741831[0.16,0.38], 755750[0.10,0.20], 756556[0.10,0.25], 769959[0.10,0.21], 809694[0.09,0.33], 809901[0.07,0.17], 813254[0.04,0.14], 813266[0.05,0.39], 813823[0.05,0.15], 839736[0.13,0.24], 841641[0.07,0.31], 878280[0.06,0.16], 1409509[0.05,0.24], 1474174[0.14,0.34] ⇒ BL	100.00%
5	40643[0.24,0.43], 245330[0.03,0.25], 307660[0.04,0.21], 345553[0.05,0.24], 365826[0.08,0.34], 460487[0.06,0.22], 796475[0.25,0.62], 1323448[0.12,0.51] ⇒ NB	72.22%
6	292171[0.21,0.64], 300051[0.07,0.50], 460487[0.06,0.22], 768443[0.04,0.13], 796475[0.29,0.65], 812033[0.04,0.72], 839736[0.17,0.65] ⇒ NB	66.67%
7	43733[0.16,0.35], 44563[0.14,0.39], 135688[0.12,0.21], 307532[0.13,0.36], 809694[0.20,0.48] ⇒ RMS	48.00%
8	44563[0.12,0.40], 78353[0.23,0.37], 292171[0.16,0.53], 841641[0.22,0.68] ⇒ RMS	40.00%
9	138672[0.20,0.47], 283751[0.13,0.32], 307532[0.23,0.42], 345553[0.10,0.28], 460487[0.03,0.17], 502055[0.11,0.46], 628357[0.10,0.24], 813266[0.18,0.66] ⇒ RMS	36.00%

6.7 Chapter Overview

This chapter presented the experimental results and discussed them.

The first experiment was devoted to analyzing RIn-Close's scalability when varying several characteristics of the dataset. It provides to the user an insight into when it is feasible to use an enumerative biclustering algorithm.

The second experiment analyzed RIn-Close's sensitivity to its main parameters, helping the user to set them.

The third experiment showed the difficulty that heuristics have to find out biclusters even in simple and controlled scenarios, encouraging the use of enumerative algorithms.

In the fourth and fifth experiments, we applied RIn-Close to two real-world problems involving gene expression data: the GOEA, and the analysis and identification of biomarkers, respectively. The fourth experiment also showed how a pseudo-enumerative algorithm misses useful information when analyzing the data, encouraging the use of enumerative algorithms. In our fifth experiment, we also showed how to extract CARs from CVC biclusters of an enumerative solution.

In the next chapter, we will conclude our thesis and outline limitations and suggestions of future works.

7 Conclusion

Biclustering is a very powerful data mining technique that overcomes several drawbacks of the well-known clustering technique. Due to its complexity, most of the proposed biclustering algorithms are heuristic-based. Nonetheless, there are several algorithms able to perform (*i*) efficient, (*ii*) complete, (*iii*) correct, and (*iv*) non-redundant enumeration of all maximal CTV biclusters of ones from a binary data matrix. These enumerative algorithms proved to be very useful and have been applied in various application domains. However, the raw data matrix admits integer and/or real values in several other application domains, and to transform it into binary data leads to loss of information. Hence, there are some proposals capable of dealing directly with numerical data matrices. Unfortunately, these already available algorithms to enumerate CVC, CVR, or CHV biclusters in numerical datasets are not capable of keeping one or more of these four properties.

In this thesis, we proposed a family of algorithms, called RIn-Close, capable of preserving these four properties when enumerating perfect CVC (or CVR) biclusters, perturbed CVC (or CVR) biclusters, and perfect CHV biclusters. Additionally, in the case of perturbed CHV biclusters, the last three of the aforementioned properties are preserved.

7.1 Discussion

The four key properties of an enumerative bicluster algorithm are: (1) efficiency, (2) completeness, (3) correctness, and (4) non-redundancy.

Regarding the enumeration of CVC and CVR biclusters in numerical data matrices, our algorithms, RIn-Close_CVC_P and RIn-Close_CVC, are the only ones in the literature with these four properties. As shown in Table 7, their competitors have no more than two of these four properties. Most of their competitors do not have the first and the fourth properties. The only exception is the PPS algorithm, which has the fourth property, but does not have the second one. Thus, our main contribution with respect to the enumeration of CVC and CVR biclusters is to propose algorithms that, in addition to performing a complete and correct enumeration, they do not return redundant biclusters and are computationally efficient.

RIn-Close_CHV_P is the only algorithm to enumerate CHV biclusters with these four properties. Its limitation is that it is only able to enumerate perfect CHV biclusters. Our algorithm to enumerate perturbed CHV biclusters, RIn-Close_CHV, has the last three properties. Although some authors claim that their proposals are enumerative

biclustering algorithms, all RIn-Close_CHV's competitors are pseudo-enumerative algorithms because they are not able to perform a complete and correct enumeration (see Table 8). Thus, our main contribution with respect to the enumeration of CHV biclusters is to propose algorithms that are fully enumerative biclustering algorithms.

Our experimental results provided a valuable insight into the scalability of RIn-Close and its sensitivity to the user-defined measure of similarity and minimum number of rows allowed in a bicluster. As the dataset becomes larger, the greater tends to be the number of biclusters, so that these parameters are critical to the feasibility of the biclustering solution. We suggest to start the parametrization with the intended value for $minRow$ and a small value for the parameter ϵ, or even with the algorithms for enumerating perfect biclusters. Analyzing the number of encountered biclusters, coverage and global overlap, we can refine the parameter values.

RIn-Close algorithms are capable of enumerating thousands of biclusters within minutes. The runtime growth rates were, in general, more favorable than their worst-case time complexities. Our most time consuming algorithm is RIn-Close_CHV, which had a polynomially growth in the number of columns of the dataset in our experiments. Besides that, RI-Close_CHV requires the construction of an augmented matrix that has $\frac{m-1}{2}$ times more columns than the original data matrix. Thus, this is the algorithm with more limitations to be used in real-world applications, thus requiring more attention when setting its parameters.

With our experimental results, we also showed that well-known heuristic-based algorithms can have a poor performance when trying to identify the existing biclusters in a simple and controlled scenario, thus emphasizing the necessity of having efficient enumerative biclustering algorithms. Moreover, in our experiment with GOEA, we showed how a pseudo-enumerative algorithm may miss useful information when analyzing the data, thus showing the importance and distinct aspects of a fully enumerative algorithm. In GOEA and other studies, such as gene-network construction or association-rules discovery, we highlighted the importance of finding all biclusters in their maximal versions.

In addition to GOEA, we also applied RIn-Close in other biological application: the analysis and identification of biomarkers. We showed that, by means of our biclustering solutions, we can test the discriminatory capability of proposed biomarkers, pointing when we have more or less genes than necessary. In addition, we were able to easily point extra or missing genes to discriminate all the types of cancer. Moreover, we showed that we can use RIn-Close to easily find biomarkers from scratch.

In our experiments with biomarkers, we also indicated how we can extend the idea of mining CARs from binary data. With simple modifications, it is possible to extract CARs from CVC biclusters of an enumerative solution, opening the possibility to use our biclustering solutions to provide sophisticated classifiers based on rules.

7.2 Suggestions of Further Research

Even though we proposed a number of improvements when we talk about the enumeration of biclusters in numerical datasets, there are still two major improvements that would bring many benefits to the users of our algorithms. The first one is with respect to the RIn-Close_CVC algorithm. This algorithm is very efficient in terms of runtime, but has a high computational cost in terms of memory usage. It must keep a symbol table in memory, whose keys are the sets of rows of each found bicluster, to prevent a maximal bicluster to be found more than once. Thus, the first challenge is to find some alternative to this symbol table, characterized by a lower computational cost in terms of memory, but without losing the algorithm efficiency. The second major improvement is associated with the RIn-Close_CHV algorithm, the only one without polynomial delay in our family of enumerative biclustering algorithms. An efficient, complete, correct, and non-redundant algorithm to enumerate all maximal perturbed CHV biclusters remains an open problem. Anyway, it is possible to improve the runtime of all RIn-Close algorithms by implementing parallelized versions of them.

Another initiative that may promote gain in computational performance is to incorporate the distinctive aspects of the recently proposed In-Close3 algorithm [10] into our RIn-Close family of algorithms.

The application of biclustering is fully disseminated, but the majority of the published works uses heuristic-based algorithms. As we showed in Section 6.4, heuristics can provide poor biclusters solutions. It motivates the use of enumerative algorithms in successfully conducted applications of biclustering of the literature. Moreover, new applications can be conceived with the use of enumerative biclustering algorithms, especially the RIn-Close family of algorithms that deal directly with numerical datasets.

In this thesis, we have extended some algorithms initially devoted to enumerate all maximal biclusters in binary datasets, so that they could treat numerical datasets as well. There are many other proposals in FCA and FPM areas restricted to binary datasets but that can be generalized to work with numerical datasets, such as:

- To mine only the top k biclusters in terms of volume [47, 46], where k is a user-defined parameter;
- To mine biclusters in data streams [42, 64, 67, 102];
- To mine fault-tolerant biclusters [75, 94], in an attempt to mitigate the effects of noise and missing values in the enumeration, thus avoiding the necessity of post-processing the bicluster solution;
- To mine high utility biclusters that directly support a given business objective [21, 100];

- To mine only the neighborhood of a specified bicluster in the iceberg concept lattice [15];
- Matrix Factorization [85];
- Classification based on association (also known as associative classification) [22, 73, 77]; and
- To mine triclusters in triadic data [71, 51].

We also intend to establish partnerships with researchers of the Biology field to promote a functional and integrated biological analysis of our biclustering solutions in the future.

Bibliography

[1] J. Abello, A. J. Pogel, and L. Miller. Breadth first search graph partitions and concept lattices. *Journal of Universal Computer Science*, 10(8):934–954, 2004. Cited in page 44.

[2] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. In *Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data*, volume 27, pages 94–105, 1998. Cited in page 18.

[3] R. Agrawal, T. Imieliński, and A. Swami. Mining association rules between sets of items in large databases. In *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, pages 207–216, 1993. Cited in page 43.

[4] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proceedings of the 20th International Conference on Very Large Data Bases*, volume 1215, pages 487–499, 1994. Cited 2 in pages 47 and 49.

[5] R. Aguirre-Gamboa, H. Gomez-Rueda, E. Martínez-Ledesma, A. Martínez-Torteya, R. Chacolla-Huaringa, A. Rodriguez-Barrientos, J.G. Tamez-Pena, and V. Trevino. Survexpress: an online biomarker validation tool and database for cancer gene expression data using survival analysis. *PloS One*, 8(9):e74250, 2013. Cited in page 91.

[6] G. Alexe, S. Alexe, Y. Crama, S. Foldes, P. L Hammer, and B. Simeone. Consensus algorithms for the generation of all maximal bicliques. *Discrete Applied Mathematics*, 145(1):11–21, 2004. Cited in page 44.

[7] F. Alqadah, C. K Reddy, J. Hu, and H. F. Alqadah. Biclustering neighborhood-based collaborative filtering method for top-n recommender systems. *Knowledge and Information Systems*, pages 1–17, 2014. Cited in page 18.

[8] S. Andrews. In-Close, a fast algorithm for computing formal concepts. In *International Conference on Conceptual Structures*, 2009. Cited 2 in pages 19 and 39.

[9] S. Andrews. In-Close2, a high performance formal concept miner. In *International Conference on Conceptual Structures*, pages 50–62, 2011. Cited 4 in pages 19, 20, 39, and 40.

[10] S. Andrews. A 'best-of-breed' approach for designing a fast algorithm for computing fixpoints of galois connections. *Information Sciences*, 295:633–649, 2015. Cited in page 102.

[11] G. Atluri, J. Bellay, G. Pandey, C. Myers, and V. Kumar. Discovering coherent value bicliques in genetic interaction data. In *Proceedings of 9th International Workshop on Data Mining in Bioinformatics (BIOKDD'10)*. Citeseer, 2000. Cited in page 47.

[12] J. Baixeries, M. Kaytoue, and A. Napoli. Characterizing functional dependencies in formal concept analysis with pattern structures. *Annals of Mathematics and Artificial Intelligence*, 72(1-2):129–149, 2014. Cited in page 49.

[13] A. Banerjee, I. Dhillon, J. Ghosh, S. Merugu, and D. S. Modha. A generalized maximum entropy approach to Bregman co-clustering and matrix approximation. *Journal of Machine Learning Research*, 8:1919–1986, 2007. Cited in page 18.

[14] A. Ben-Dor, B. Chor, R. Karp, and Z. Yakhini. Discovering local structure in gene expression data: the order-preserving submatrix problem. *Journal of Computational Biology*, 10(3-4):373–384, 2003. Cited 3 in pages 19, 35, and 47.

[15] A. Berry, J.P. Bordat, and A. Sigayret. A local approach to concept generation. *Annals of Mathematics and Artificial Intelligence*, 49(1-4):117–136, 2007. Cited in page 103.

[16] J. Besson, C. Robardet, L. De Raedt, and J.F. Boulicaut. Mining bi-sets in numerical data. In *Knowledge Discovery in Inductive Databases*, pages 11–23. Springer, 2007. Cited 4 in pages 20, 35, 47, and 48.

[17] C. Bron and J. Kerbosch. Algorithm 457: finding all cliques of an undirected graph. *Communications of the ACM*, 16(9):575–577, 1973. Cited in page 69.

[18] S. Busygin, O. Prokopyev, and P.M. Pardalos. Biclustering in data mining. *Computers & Operations Research*, 35(9):2964–2987, 2008. Cited in page 19.

[19] L. Candillier, K. Jack, F. Fessant, and F. Meyer. State-of-the-art recommender systems. *Collaborative and Social Information Retrieval and Access Techniques for Improved User Modeling*, 2009. Cited in page 67.

[20] F. Cazals and C. Karande. A note on the problem of reporting maximal cliques. *Theoretical Computer Science*, 407(1):564–568, 2008. Cited 2 in pages 45 and 69.

[21] Raymond Chan Chan, Qiang Yang, and Yi-Dang Shen. Mining high utility itemsets. In *Data Mining, 2003. ICDM 2003. Third IEEE International Conference on*, pages 19–26. IEEE, 2003. Cited in page 102.

[22] F. Chen, Y. Wang, M. Li, H. Wu, and J. Tian. Principal association mining: An efficient classification approach. *Knowledge-Based Systems*, 67:16–25, 2014. Cited in page 103.

[23] Y. Cheng and G.M. Church. Biclustering of expression data. In *Proceedings of the 8th International Conference on Intelligent Systems for Molecular Biology*, volume 1, pages 93–103, 2000. Cited 5 in pages 19, 24, 35, 47, and 71.

[24] L. Cheung, D.W. Cheung, B. Kao, K.Y. Yip, and M.K. Ng. On mining micro-array data by order-preserving submatrix. *International Journal of Bioinformatics Research and Applications*, 3(1):42–64, 2006. Cited in page 66.

[25] R.J. Cho, M.J. Campbell, E.A. Winzeler, L. Steinmetz, A. Conway, L. Wodicka, T.G. Wolfsberg, A.E. Gabrielian, D. Landsman, and D.J. Lockhart. A genome-wide transcriptional analysis of the mitotic cell cycle. *Molecular Cell*, 2(1):65–74, 1998. Cited in page 71.

[26] V. Codocedo and A. Napoli. Bicluster enumeration using formal concept analysis. In *What formal concept analysis can do for artificial intelligence?(FCA4AI 2014) Workshop at ECAI 2014*, 2014. Cited in page 49.

[27] V. Codocedo and A. Napoli. Lattice-based biclustering using partition pattern structures. In *ECAI 2014: 21st European Conference on Artificial Intelligence*, volume 263, page 213. IOS Press, 2014. Cited in page 49.

[28] A. Colantonio, R. Di Pietro, A. Ocello, and N.V. Verde. ABBA: Adaptive bicluster-based approach to impute missing values in binary matrices. In *Proceedings of the 2010 ACM Symposium on Applied Computing*, pages 1026–1033, 2010. Cited in page 18.

[29] P. Dao, R. Colak, R. Salari, F. Moser, E. Davicioni, A. Schönhuth, and M. Ester. Inferring cancer subnetwork markers using density-constrained biclustering. *Bioinformatics*, 26(18):i625–i631, 2010. Cited in page 91.

[30] P.A.D. de Castro, F.O. de França, H.M. Ferreira, and F.J. Von Zuben. Applying biclustering to perform collaborative filtering. In *7th International Conference on Intelligent Systems Design and Applications*, pages 421–426, 2007. Cited in page 18.

[31] O. de França, F, G. P. Coelho, and F.J. Von Zuben. Predicting missing values with biclustering: A coherence-based approach. *Pattern Recognition*, 46(5):1255–1266, 2013. Cited in page 18.

[32] M. Deodhar, G. Gupta, J. Ghosh, H. Cho, and I. Dhillon. A scalable framework for discovering coherent co-clusters in noisy data. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 241–248, 2009. Cited 2 in pages 19 and 47.

[33] I.S. Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. In *Proceedings of the seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 269–274, 2001. Cited in page 18.

[34] D. Eppstein. Arboricity and bipartite subgraph listing algorithms. *Information Processing Letters*, 51(4):207–211, 1994. Cited in page 44.

[35] D. Eppstein, M. Löffler, and D. Strash. Listing all maximal cliques in sparse graphs in near-optimal time. In *Algorithms and Computation*, pages 403–414. Springer, 2010. Cited in page 19.

[36] J.L. Flores, I. Inza, P. Larrañaga, and B. Calvo. A new measure for gene expression biclustering based on non-parametric correlation. *Computer Methods and Programs in Biomedicine*, 112(3):367–397, 2013. Cited in page 35.

[37] B. Ganter. Two basic algorithms in concept analysis. Technical report, FB4-Preprint No. 831, 1984. Cited in page 39.

[38] B. Ganter and S.O. Kuznetsov. Pattern structures and their projections. In *Conceptual Structures: Broadening the Base*, pages 129–142. Springer, 2001. Cited in page 48.

[39] B. Ganter and R. Wille. *Formal concept analysis: mathematical foundations*. Springer-Verlag New York, Inc., 1997. Cited 4 in pages 19, 37, 38, and 48.

[40] B. Gaume, E. Navarro, and H. Prade. A parallel between extended formal concept analysis and bipartite graphs analysis. In *Computational Intelligence for Knowledge-Based Systems Design*, pages 270–280. Springer, 2010. Cited in page 44.

[41] A. Gély, L. Nourine, and B. Sadi. Enumeration aspects of maximal cliques and bicliques. *Discrete Applied Mathematics*, 157(7):1447–1459, 2009. Cited in page 44.

[42] C. Giannella, J. Han, J. Pei, X. Yan, and P.S. Yu. Mining frequent patterns in data streams at multiple time granularities. *Next Generation Data Mining*, 212:191–212, 2003. Cited in page 102.

[43] B. Goethals. Survey on frequent pattern mining. *Univ. of Helsinki*, 2003. Cited in page 19.

[44] J. Guan, Y. Gan, and H. Wang. Discovering pattern-based subspace clusters by pattern tree. *Knowledge-Based Systems*, 22(8):569–579, 2009. Cited 2 in pages 50 and 62.

[45] J.K. Gupta, S. Singh, and N.K. Verma. MTBA: Matlab toolbox for biclustering analysis. pages 94–97. IEEE, 2013. Cited in page 69.

[46] J. Han, H. Cheng, D. Xin, and X. Yan. Frequent pattern mining: current status and future directions. *Data Mining and Knowledge Discovery*, 15(1):55–86, 2007. Cited in page 102.

[47] J. Han, J. Wang, Y. Lu, and P. Tzvetkov. Mining top-k frequent closed patterns without minimum support. In *Data Mining, 2002. ICDM 2003. Proceedings. 2002 IEEE International Conference on*, pages 211–218. IEEE, 2002. Cited in page 102.

[48] J.A. Hartigan. Direct clustering of a data matrix. *Journal of the American Statistical Association*, pages 123–129, 1972. Cited in page 24.

[49] R. Henriques and S.C. Madeira. BicPAM: Pattern-based biclustering for biomedical data analysis. *Algorithms for Molecular Biology*, 9(1):27, 2014. Cited 4 in pages 20, 47, 52, and 67.

[50] D. Horta and R. Campello. Similarity measures for comparing biclusterings. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 11(5):942–954, 2014. Cited in page 34.

[51] D.I. Ignatov, D.V. Gnatyshak, S.O. Kuznetsov, and B.G. Mirkin. Triadic formal concept analysis and triclustering: searching for optimal patterns. *Machine Learning*, 101(1-3):271–302, 2015. Cited in page 103.

[52] J. Ihmels, G. Friedlander, S. Bergmann, O. Sarig, Y. Ziv, and N. Barkai. Revealing modular organization in the yeast transcriptional network. *Nature genetics*, 31(4):370–377, 2002. Cited 2 in pages 19 and 47.

[53] A.K. Jain and R.C. Dubes. *Algorithms for clustering data*. Prentice-Hall, Inc., 1988. Cited in page 18.

[54] S. Julien, A. Ivetic, A. Grigoriadis, D. QiZe, B. Burford, D. Sproviero, G. Picco, C. Gillett, S.L. Papp, L. Schaffer, A. Tutt, J. Taylor-Papadimitriou, S.E. Pinder, and J.M. Burchell. Selectin ligand sialyl-lewis x antigen drives metastasis of hormone-dependent breast cancers. *Cancer research*, 71(24):7683–7693, 2011. Cited in page 71.

[55] G. Karam and J.Z. Mohammed. Genmax: An efficient algorithm for mining maximal frequent itemsets. *Data Mining and Knowledge Discovery: An International Journal*, 11(3):223–242, 2005. Cited in page 43.

[56] M. Kaytoue, Z. Assaghir, A. Napoli, and S.O. Kuznetsov. Embedding tolerance relations in formal concept analysis: an application in information fusion. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 1689–1692. ACM, 2010. Cited in page 48.

[57] M. Kaytoue, V. Codocedo, J. Baixeries, and A. Napoli. Three related fca methods for mining biclusters of similar values on columns. In *Proceedings of the Eleventh International Conference on Concept Lattices and Their Applications, Kosice, Slovakia, October 7-10, 2014*, 2014. Cited 2 in pages 20 and 49.

[58] M. Kaytoue, S.O. Kuznetsov, J. Macko, and A. Napoli. Biclustering meets triadic concept analysis. *Annals of Mathematics and Artificial Intelligence*, pages 1–25, 2013. Cited 2 in pages 47 and 48.

[59] M. Kaytoue, S.O. Kuznetsov, and A. Napoli. Biclustering numerical data in formal concept analysis. In *9th International Conference on Formal Concept Analysis*, pages 135–150, 2011. Cited 4 in pages 42, 47, 48, and 49.

[60] J. Khan, J.S. Wei, M. Ringner, L.H. Saal, M. Ladanyi, F. Westermann, F. Berthold, M. Schwab, C.R. Antonescu, C. Peterson, and P.S. Meltzer. Classification and diagnostic prediction of cancers using gene expression profiling and artificial neural networks. *Nature medicine*, 7(6):673–679, 2001. Cited 6 in pages , 91, 92, 93, 94, and 98.

[61] I. Koch. Enumerating all connected maximal common subgraphs in two graphs. *Theoretical Computer Science*, 250(1):1–30, 2001. Cited in page 69.

[62] P. Krajca, J. Outrata, and V. Vychodil. Advances in algorithms based on cbo. In *Proceedings of the 8th International Conference on Concept Lattices and Their Applications*, volume 672, pages 325–337, 2010. Cited 2 in pages 19 and 39.

[63] H.P. Kriegel, P. Kröger, and A. Zimek. Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 3(1):1, 2009. Cited in page 19.

[64] V. Kumar and S.R. Satapathy. A novel technique for mining closed frequent itemsets using variable sliding window. In *Advance Computing Conference (IACC), 2014 IEEE International*, pages 504–510. IEEE, 2014. Cited in page 102.

[65] S. Kuznetsov. Learning of simple conceptual graphs from positive and negative examples. In *Principles of Data Mining and Knowledge Discovery*, pages 384–391, 1999. Cited 2 in pages 19 and 39.

[66] S.O. Kuznetsov. Mathematical aspects of concept analysis. *Journal of Mathematical Sciences*, 80(2):1654–1698, 1996. Cited in page 39.

[67] P.T. La, B. Le, and B. Vo. Incrementally building frequent closed itemset lattice. *Expert Systems with Applications*, 41(6):2703–2712, 2014. Cited in page 102.

[68] L. Lakhal and G. Stumme. Efficient mining of association rules based on formal concept analysis. In *Formal concept analysis: Foundations and Applications*, pages 180–195. Springer, 2005. Cited in page 43.

[69] L. Lazzeroni and A. Owen. Plaid models for gene expression data. *Statistica Sinica*, 12(1):61–86, 2002. Cited 2 in pages 19 and 47.

[70] J.H. Leber, S. Bernales, and P. Walter. Ire1-independent gain control of the unfolded protein response. *PLoS biology*, 2(8):e235, 2004. Cited in page 71.

[71] F. Lehmann and R. Wille. *A triadic approach to formal concept analysis*. Springer, 1995. Cited 2 in pages 49 and 103.

[72] J. Li, G. Liu, H. Li, and L. Wong. Maximal biclique subgraphs and closed pattern pairs of the adjacency matrix: A one-to-one correspondence and mining algorithms. *IEEE Transactions on Knowledge and Data Engineering*, 19(12):1625–1637, 2007. Cited in page 44.

[73] W. Li, J. Han, and J. Pei. Cmar: Accurate and efficient classification based on multiple class-association rules. In *Data Mining, 2001. ICDM 2001, Proceedings IEEE International Conference on*, pages 369–376. IEEE, 2001. Cited in page 103.

[74] B. Liu, W. Hsu, and Y. Ma. Integrating classification and association rule mining. In *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*, 1998. Cited 2 in pages 20 and 51.

[75] S. Liu and C.K. Poon. On mining proportional fault-tolerant frequent itemsets. In *Database Systems for Advanced Applications*, pages 342–356. Springer, 2014. Cited in page 102.

[76] Y. Liu, Q. Gu, J.P. Hou, J. Han, and J. Ma. A network-assisted co-clustering algorithm to discover cancer subtypes based on gene expression. *BMC Bioinformatics*, 15(1):37, 2014. Cited 2 in pages 20 and 51.

[77] B.L.W.H.Y. Ma. Integrating classification and association rule mining. In *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*, 1998. Cited in page 103.

[78] T.J. MacDonald, K.M. Brown, B. LaFleur, K. Peterson, C. Lawlor, Y. Chen, R.J. Packer, P. Cogen, and D.A. Stephan. Expression profiling of medulloblastoma: Pdgfra and the ras/mapk pathway as therapeutic targets for metastatic disease. *Nature genetics*, 29(2):143–152, 2001. Cited in page 71.

[79] S.C. Madeira and A.L. Oliveira. Biclustering algorithms for biological data analysis: a survey. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 1(1):24–45, 2004. Cited 8 in pages 19, 20, 24, 25, 28, 32, 43, and 47.

[80] S.C. Madeira and A.L. Oliveira. A polynomial time biclustering algorithm for finding approximate expression patterns in gene expression time series. *Algorithms for Molecular Biology*, 4(1):8, 2009. Cited 3 in pages 19, 20, and 47.

[81] K. Makino and T. Uno. New algorithms for enumerating all maximal cliques. In *Proceedings of the 9th Scandinavian Workshop on Algorithm Theory*, pages 260–272, 2004. Cited 2 in pages 19 and 65.

[82] R. Martinez, C. Pasquier, and N. Pasquier. GenMiner: mining informative association rules from genomic data. In *Bioinformatics and Biomedicine, 2007. BIBM 2007. IEEE International Conference on*, pages 15–22. IEEE, 2007. Cited 2 in pages 20 and 47.

[83] B. Mirkin. *Mathematical Classification and Clustering*. Springer, 1996. Cited in page 24.

[84] S. Motameny, B. Versmold, and R. Schmutzler. Formal concept analysis for the identification of combinatorial biomarkers in breast cancer. In *Formal Concept Analysis*, pages 229–240. Springer, 2008. Cited in page 91.

[85] E. Nenova, D.I. Ignatov, and A.V. Konstantinov. An FCA-based boolean matrix factorisation for collaborative filtering. *arXiv preprint arXiv:1310.4366*, 2013. Cited in page 103.

[86] A. Oghabian, S. Kilpinen, S. Hautaniemi, and E. Czeizler. Biclustering methods: Biological relevance and application in gene expression analysis. *PloS One*, 9(3):e90801, 2014. Cited in page 18.

[87] Y. Okada, W. Fujibuchi, and P. Horton. A biclustering method for gene expression module discovery using a closed itemset enumeration algorithm. *IPSJ Digital Courier*, 3:183–192, 2007. Cited 2 in pages 20 and 47.

[88] Y. Okada, K. Okubo, P. Horton, and W. Fujibuchi. Exhaustive search method of gene expression modules and its application to human tissue data. *IAENG international journal of computer science*, 34(16), 2007. Cited 2 in pages 20 and 47.

[89] S. Oliveira, R. Veroneze, and F.J. Von Zuben. On bicluster aggregation and its benefits for enumerative solutions. In *11th International Conference on Machine Learning and Data Mining*, pages 135–150, 2015. Cited in page 75.

[90] N.R. Pal, K. Aguan, A. Sharma, and S. Amari. Discovering biomarkers from gene expression data for predicting cancer subgroups using neural networks and relational fuzzy clustering. *BMC Bioinformatics*, 8(1):5, 2007. Cited 7 in pages , 92, 93, 94, 96, 97, and 98.

[91] G. Pandey, G. Atluri, M. Steinbach, C.L. Myers, and V. Kumar. An association analysis approach to biclustering. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 677–686, 2009. Cited 3 in pages 20, 29, and 49.

[92] T. Pang-Ning, M. Steinbach, and V. Kumar. *Introduction to Data Mining*. Pearson, 2005. Cited in page 20.

[93] A. Paterek. Improving regularized singular value decomposition for collaborative filtering. In *Proceedings of KDD Cup and Workshop*, volume 2007, pages 5–8, 2007. Cited in page 67.

[94] J. Pei, A.K.H. Tung, and J. Han. Fault-tolerant frequent pattern mining: Problems and challenges. *DMKD*, 1:44, 2001. Cited in page 102.

[95] J. Pei, X. Zhang, M. Cho, H. Wang, and P.S. Yu. Maple: A fast algorithm for maximal pattern-based clustering. In *3th IEEE International Conference on Data Mining*, pages 259–266, 2003. Cited 2 in pages 20 and 50.

[96] B. Pontes, R. Giráldez, and J. Aguilar-Ruiz. Biclustering on expression data: A review. *Journal of Biomedical Informatics*, 57:163–180, 2015. Cited in page 19.

[97] A. Serin and M. Vingron. DeBi: Discovering differentially expressed biclusters using a frequent itemset approach. *Algorithms for Molecular Biology*, 6(1):18, 2011. Cited 2 in pages 20 and 47.

[98] P. Symeonidis, A. Nanopoulos, A. Papadopoulos, and Y. Manolopoulos. Nearest-biclusters collaborative filtering with constant values. *Advances in Web Mining and Web Usage Analysis*, pages 36–55, 2007. Cited in page 18.

[99] P. Symeonidis, A. Nanopoulos, A.N. Papadopoulos, and Y. Manolopoulos. Nearest-biclusters collaborative filtering based on constant and coherent values. *Information Retrieval*, 11(1):51–75, 2008. Cited in page 18.

[100] Vincent S Tseng, Cheng-Wei Wu, Bai-En Shie, and Philip S Yu. Up-growth: an efficient algorithm for high utility itemset mining. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 253–262. ACM, 2010. Cited in page 102.

[101] T. Uno, M. Kiyomi, and H. Arimura. LCM ver. 2: Efficient mining algorithms for frequent/closed/maximal itemsets. In *Workshop on Frequent Itemset Mining Implementations*, volume 19, page 30, 2004. Cited 2 in pages 19 and 44.

[102] P. Valtchev, R. Missaoui, and R. Godin. A framework for incremental generation of closed itemsets. *Discrete Applied Mathematics*, 156(6):924–949, 2008. Cited in page 102.

[103] R. Veroneze, A. Banerjee, and F.J. Von Zuben. Enumerating all maximal biclusters in real-valued datasets. *submitted to Information Sciences*. Cited 3 in pages 69, 71, and 79.

[104] R. Veroneze, F.O. de Franca, and F.J. Von Zuben. Assessing the performance of a swarm-based biclustering technique for data imputation. In *IEEE Congress on Evolutionary Computation*, pages 386–393, 2011. Cited in page 18.

[105] H. Wang, F. Chu, W. Fan, P.S. Yu, and J. Pei. A fast algorithm for subspace clustering by pattern similarity. In *Proceedings of the 16th International Conference on Scientific and Statistical Database Management*, pages 51–60, 2004. Cited 2 in pages 50 and 62.

[106] H. Wang, W. Wang, J. Yang, and P.S. Yu. Clustering by pattern similarity in large data sets. In *Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data*, pages 394–405, 2002. Cited 2 in pages 18 and 49.

[107] J. Yang, H. Wang, W. Wang, and P. Yu. Enhanced biclustering on expression data. In *Proceedings of the 3th IEEE Symposium on Bioinformatics and Bioengineering*, pages 321–327, 2003. Cited 2 in pages 19 and 47.

[108] M.J. Zaki. Generating non-redundant association rules. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 34–43. ACM, 2000. Cited in page 20.

[109] M.J. Zaki and C.J. Hsiao. CHARM: An efficient algorithm for closed itemset mining. In *Proceedings of the 2002 SIAM International Conference on Data Mining*, volume 2, pages 457–473, 2002. Cited 2 in pages 19 and 43.

[110] M.J. Zaki, S. Parthasarathy, M. Ogihara, and W. Li. New algorithms for fast discovery of association rules. In *3rd International Conference on Knowledge Discovery and Data Mining*, 1997. Cited in page 43.

[111] L. Zhao and M.J. Zaki. Microcluster: Efficient deterministic biclustering of microarray data. *Intelligent Systems*, 20(6):40–49, 2005. Cited 4 in pages 18, 20, 50, and 75.

Appendix

APPENDIX A – Exemplifying the key concepts of biclustering to newcomers

The application of biclustering is fully disseminated. So, we are able to use many real-world problems to illustrate its application. However, to simplify the understanding of the main concepts of biclustering to newcomers in the area, we believe it is a good idea to choose an example that is part of our daily life. Nowadays, it is usual to watch a video or a movie on-line and give an opinion about it, which can be done by means of a "like" or "dislike" opinion, by choosing a certain number of stars, among other ways. This information that we give to the companies about our taste and opinion can be used for several purposes, such as movies recommendations or personalized marketing campaigns. Thus, aiming at converting those raw and unstructured information into useful knowledge, the data must be analyzed with the goal of finding interesting patterns on it.

We will give some examples of the usefulness of applying biclustering techniques to analyze this kind of data. Moreover, we will try to exemplify some limitations of using traditional clustering techniques when dealing with the same problems.

Tables 31 and 32 show an example of a data matrix $\mathbf{A}_{60\times38}$, where rows represent users and columns represent movies. Each $a_{ij} \in \mathbf{A}$ is the rating given by user u_i to the movie m_j, ranging from 1 to 10. The symbol "?" represents missing values. The data matrix $\mathbf{A}$ is exhibited in two tables because of space limitation. All examples of this aapendix will be based on this data matrix.

A.1 Traditional clustering vs. Biclustering

One of the first questions we might want to answer is: do we have groups of users sharing similar tastes?

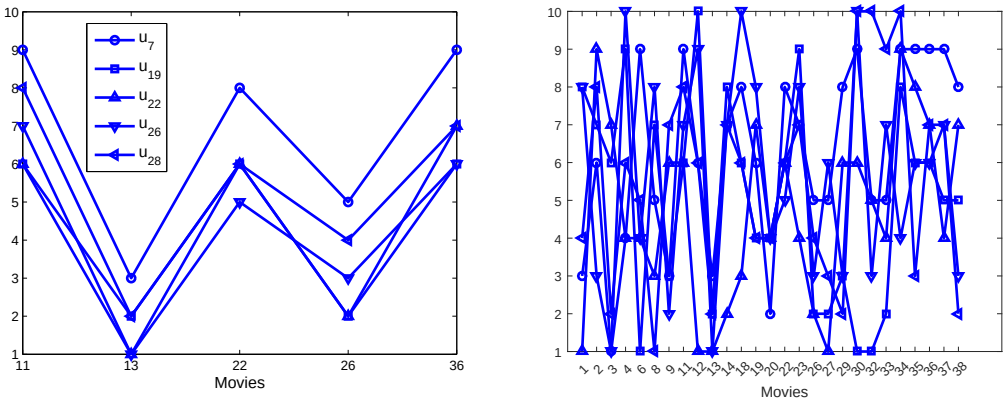
One limitation of using traditional clustering techniques to answer this is to compute the similarity among the users based on all movies. For instance, Figure 19(b) shows the ratings given by five users for all the movies that they have watched. No patterns among the five users are visibly explicit. However, if we pick a subset of the movies and plot the ratings of these five users on these movies (see Figure 19(a)), it is easy to see that they present similar patterns. Note that the kind of pattern exhibited on Figure 19(a) does not depend on the distance between any two users. This is due to the fact that biclustering methods are able to consider coherence measures which are more general than distance functions.

Table 31 – First 20 columns of a data matrix $\mathbf{A}_{60\times38}$, where rows represent users and columns represent movies. The next 18 columns are in Table 32.

	m_1	m_2	m_3	m_4	m_5	m_6	m_7	m_8	m_9	m_{10}	m_{11}	m_{12}	m_{13}	m_{14}	m_{15}	m_{16}	m_{17}	m_{18}	m_{19}	m_{20}
u_1	9	8	2	6	1	1	1	8	7	4	9	2	2	?	2	10	9	3	5	6
u_2	10	3	10	3	3	7	6	5	7	?	6	5	4	6	8	2	8	4	7	5
u_3	2	6	1	8	?	?	9	1	?	6	10	9	1	4	3	3	2	1	8	10
u_4	10	7	8	?	?	1	7	3	10	?	7	9	6	1	7	4	6	7	6	9
u_5	7	9	9	7	10	6	2	2	3	1	6	?	4	8	10	1	4	2	6	5
u_6	?	10	9	2	8	1	4	3	8	8	?	3	2	4	5	7	6	?	6	9
u_7	3	6	1	4	5	9	?	5	3	3	9	6	3	7	7	5	4	8	6	2
u_8	6	2	4	7	6	9	?	6	2	5	10	7	10	8	8	10	5	4	1	4
u_9	10	2	3	8	3	8	?	?	7	7	?	5	7	2	?	5	2	7	8	10
u_{10}	10	3	9	1	5	2	9	9	?	4	9	3	?	2	7	7	3	4	10	10
u_{11}	2	?	5	10	10	7	7	6	5	8	7	10	10	6	2	2	1	7	4	8
u_{12}	10	3	10	8	6	6	4	10	7	4	10	1	2	5	10	4	10	1	10	7
u_{13}	10	9	2	5	6	10	2	7	8	7	?	2	8	9	?	2	7	10	4	4
u_{14}	?	3	3	5	3	7	5	?	4	8	?	2	?	8	3	8	10	9	9	10
u_{15}	9	10	?	5	5	9	5	3	7	5	9	2	6	8	8	9	2	8	5	2
u_{16}	2	4	2	?	7	5	2	7	5	?	3	7	2	?	5	4	10	9	5	?
u_{17}	5	2	9	6	7	5	6	3	9	4	5	6	6	1	8	7	?	4	3	7
u_{18}	10	3	6	?	4	9	3	7	9	5	10	1	?	?	4	3	6	7	2	9
u_{19}	8	7	6	9	4	1	?	7	3	3	6	10	2	8	?	?	5	6	4	4
u_{20}	10	?	2	8	10	2	6	1	7	2	?	?	3	10	1	9	3	6	8	8
u_{21}	7	4	9	?	1	2	?	?	6	9	8	8	9	7	7	6	8	3	8	9
u_{22}	1	9	7	4	9	4	3	3	6	5	6	1	1	2	5	4	3	3	7	4
u_{23}	?	6	4	9	10	9	7	7	9	9	3	9	3	8	5	3	1	5	1	6
u_{24}	10	?	6	6	8	?	3	9	?	4	7	10	1	2	7	5	8	3	9	10
u_{25}	7	10	5	4	1	1	9	4	4	8	1	10	5	2	1	5	7	9	10	?
u_{26}	8	3	1	10	3	4	10	8	2	?	7	9	1	7	4	4	8	10	8	4
u_{27}	8	8	3	9	4	6	8	7	?	9	7	8	9	4	8	6	7	1	1	7
u_{28}	4	8	2	6	?	5	4	1	7	8	8	6	2	7	?	8	?	6	4	4
u_{29}	7	4	?	?	2	7	6	7	5	4	9	2	1	8	2	?	4	1	8	8
u_{30}	2	6	?	6	8	7	2	4	7	3	10	4	4	6	2	5	9	9	?	5
u_{31}	8	1	5	3	2	3	10	10	6	8	8	2	5	8	1	2	4	?	3	5
u_{32}	1	1	1	4	7	5	9	1	7	10	6	1	?	3	1	1	9	1	3	7
u_{33}	3	?	10	5	5	1	?	5	6	4	?	10	10	8	5	3	8	10	7	10
u_{34}	1	8	10	3	8	10	3	5	8	7	6	4	4	10	7	4	9	1	5	4
u_{35}	1	10	5	9	8	2	6	5	6	5	1	3	3	9	8	7	6	7	7	9
u_{36}	9	2	5	2	10	?	1	8	10	9	2	?	1	1	6	10	7	8	3	8
u_{37}	7	6	4	3	9	4	5	4	3	?	9	5	3	4	2	10	10	6	2	10
u_{38}	4	5	10	2	?	2	4	8	2	2	?	7	1	4	7	5	5	9	?	1
u_{39}	10	1	4	3	7	5	2	5	2	9	9	1	6	7	2	3	1	9	8	?
u_{40}	1	4	2	5	2	4	2	1	1	10	3	9	8	6	2	8	9	7	10	7
u_{41}	?	2	8	?	1	10	5	2	5	?	6	6	7	8	1	8	7	?	2	3
u_{42}	4	8	4	10	8	10	1	8	5	9	7	9	1	4	2	8	4	?	2	3
u_{43}	?	4	?	?	6	1	6	5	4	6	1	4	?	3	2	8	10	2	1	8
u_{44}	8	6	5	2	5	?	5	2	8	2	7	5	8	1	2	2	3	1	5	7
u_{45}	2	2	1	10	10	3	7	?	7	2	?	1	10	8	4	7	7	2	2	6
u_{46}	5	7	2	10	7	5	7	7	8	?	1	2	6	3	4	5	7	7	9	?
u_{47}	5	3	10	5	7	6	7	2	10	8	?	7	?	4	3	3	4	10	1	1
u_{48}	7	7	10	2	9	10	1	8	10	9	2	4	9	6	3	1	2	4	1	4
u_{49}	8	7	6	3	9	5	1	3	2	8	?	9	4	3	9	9	1	5	6	?
u_{50}	8	8	1	5	6	10	4	10	2	4	3	?	3	7	8	2	5	?	8	3
u_{51}	3	5	3	?	2	4	6	3	7	6	2	10	8	5	6	2	?	10	?	8
u_{52}	7	1	4	3	3	8	7	8	1	1	2	6	1	2	2	7	8	7	2	9
u_{53}	7	?	?	7	9	7	5	2	6	2	1	8	1	8	3	9	4	10	4	3
u_{54}	2	?	1	8	1	6	9	?	6	2	7	10	7	2	1	6	9	?	3	8
u_{55}	2	?	1	3	5	7	8	1	9	7	3	3	7	3	10	8	8	4	6	2
u_{56}	5	9	?	2	2	7	10	6	5	5	6	?	6	3	8	?	6	?	10	9
u_{57}	10	?	?	3	10	2	6	7	4	?	7	5	8	6	6	10	2	3	7	2
u_{58}	4	10	8	4	8	?	4	6	7	5	5	8	8	1	?	?	10	3	2	6
u_{59}	6	1	7	5	?	10	?	?	8	2	6	9	8	5	2	7	3	7	4	4
u_{60}	3	5	?	6	5	?	7	7	6	1	5	2	3	2	?	1	10	6	1	9

Table 32 – Last 18 columns of a data matrix $\mathbf{A}_{60\times38}$, where rows represent users and columns represent movies. The previous 20 columns are in Table 31.

	m_{21}	m_{22}	m_{23}	m_{24}	m_{25}	m_{26}	m_{27}	m_{28}	m_{29}	m_{30}	m_{31}	m_{32}	m_{33}	m_{34}	m_{35}	m_{36}	m_{37}	m_{38}
u_1	6	6	5	10	?	2	6	8	6	8	2	5	9	8	3	10	2	8
u_2	5	7	3	6	6	3	8	4	9	10	2	7	?	?	?	6	5	10
u_3	9	4	9	6	3	1	2	6	9	2	4	10	6	4	9	10	5	?
u_4	4	?	8	4	6	5	4	2	8	8	9	?	?	?	7	8	6	3
u_5	5	5	9	5	2	3	2	1	4	2	5	8	1	6	10	5	9	2
u_6	10	4	3	5	7	3	4	10	5	?	5	6	9	?	3	7	8	9
u_7	?	8	7	?	6	5	5	3	8	9	2	5	5	9	9	9	9	8
u_8	10	8	7	9	1	2	6	6	2	5	10	9	1	?	7	2	1	3
u_9	2	7	2	1	1	5	1	10	2	8	5	4	8	6	3	4	3	3
u_{10}	7	2	5	5	2	?	6	2	3	5	9	2	2	?	5	10	?	4
u_{11}	6	1	3	9	1	?	3	?	6	9	7	7	2	7	4	5	10	9
u_{12}	7	6	8	4	?	8	?	4	10	4	4	7	7	8	6	?	8	?
u_{13}	4	4	3	?	9	1	?	10	8	1	?	4	3	8	9	10	5	6
u_{14}	7	?	9	9	7	4	2	4	4	6	8	9	4	2	2	10	4	6
u_{15}	9	10	9	9	6	?	10	3	3	10	5	?	5	?	9	7	1	3
u_{16}	1	3	4	10	9	3	?	2	9	2	9	10	5	1	9	2	8	7
u_{17}	1	9	?	2	1	1	9	4	10	5	9	2	4	?	4	1	6	8
u_{18}	10	9	7	3	10	2	8	4	?	8	5	3	7	8	5	7	2	5
u_{19}	7	6	9	9	?	2	2	?	3	1	4	1	2	8	6	6	5	5
u_{20}	3	9	7	6	6	3	4	5	?	?	6	7	2	4	8	10	2	5
u_{21}	5	10	6	6	2	2	2	1	9	8	10	2	1	8	8	?	8	3
u_{22}	2	6	4	7	6	2	1	7	6	6	8	5	4	9	8	7	4	7
u_{23}	3	8	5	9	1	6	4	1	10	2	4	9	8	3	4	6	10	10
u_{24}	3	6	8	6	8	10	7	6	1	?	8	6	?	2	5	3	?	10
u_{25}	4	1	9	3	9	10	?	8	6	6	10	4	8	3	10	10	9	8
u_{26}	2	5	8	5	10	3	6	?	3	10	?	3	7	4	6	6	7	3
u_{27}	4	7	1	5	10	5	5	5	9	9	6	5	9	3	9	1	6	7
u_{28}	?	6	7	10	6	4	3	6	2	10	?	10	9	10	3	7	7	2
u_{29}	9	4	5	7	3	6	6	1	5	7	1	2	3	1	7	6	8	2
u_{30}	1	10	5	7	2	3	?	5	4	5	2	?	4	6	6	1	1	2
u_{31}	10	9	2	8	6	1	8	7	9	10	?	?	6	2	10	?	?	2
u_{32}	4	9	9	4	6	5	6	3	7	5	?	1	4	9	1	4	1	6
u_{33}	1	4	4	6	8	2	8	9	3	3	1	7	9	2	6	3	3	1
u_{34}	4	6	3	?	1	1	7	10	4	4	10	10	9	6	6	2	2	9
u_{35}	8	9	4	2	7	10	?	9	?	8	7	3	6	10	1	4	10	8
u_{36}	8	10	4	6	6	5	6	?	7	6	1	2	3	4	10	3	?	10
u_{37}	6	7	6	7	2	10	?	3	6	8	1	7	7	1	9	7	6	5
u_{38}	7	3	6	5	?	8	1	8	2	10	3	10	3	?	?	1	7	7
u_{39}	9	7	4	9	6	1	2	3	?	10	5	7	5	?	8	3	7	9
u_{40}	1	1	?	8	5	7	2	10	5	6	2	9	?	4	2	?	9	6
u_{41}	4	5	6	4	10	8	7	7	10	10	1	2	6	?	7	9	1	3
u_{42}	1	7	7	5	7	7	5	?	6	?	8	?	?	10	3	?	?	10
u_{43}	?	10	10	4	5	6	7	2	1	1	4	4	8	7	5	?	?	1
u_{44}	8	9	8	8	9	?	3	1	1	4	8	5	10	10	?	7	7	4
u_{45}	?	5	5	8	6	8	1	3	9	6	5	2	3	5	2	8	3	?
u_{46}	?	8	9	5	6	3	6	9	5	6	5	?	6	10	4	2	6	4
u_{47}	6	5	2	7	7	4	?	10	4	10	1	7	1	1	5	10	8	4
u_{48}	1	10	1	10	4	9	10	7	?	?	1	8	8	7	?	9	10	2
u_{49}	10	10	1	8	3	9	10	8	4	5	1	9	7	9	8	?	?	10
u_{50}	9	9	2	8	6	?	?	?	6	6	6	?	9	3	10	5	1	2
u_{51}	3	4	4	2	9	4	1	6	8	8	3	10	10	10	2	4	1	4
u_{52}	?	5	4	4	5	7	7	9	9	1	9	5	?	8	3	7	7	9
u_{53}	10	3	?	6	2	10	9	5	4	9	9	8	5	9	?	3	8	5
u_{54}	8	8	6	5	5	10	10	10	7	2	10	8	1	6	4	6	7	?
u_{55}	9	9	1	1	?	6	1	1	?	5	5	9	6	8	10	7	4	6
u_{56}	5	10	2	3	5	4	5	?	1	?	3	2	3	4	?	6	?	7
u_{57}	5	6	7	9	?	9	6	6	6	?	3	5	3	3	7	5	6	?
u_{58}	6	6	9	1	5	5	7	1	5	7	?	7	4	4	9	1	10	6
u_{59}	3	2	10	?	4	10	8	8	4	2	8	?	1	6	5	6	1	1
u_{60}	8	9	6	1	4	1	7	6	?	3	4	9	8	9	7	5	3	?



(a) Biclustering (example of a CHV bicluster considering $\epsilon = 2$).

(b) Clustering.

Figure 19 – Example of patterns that can be found by biclustering and cannot be found by traditional clustering techniques.

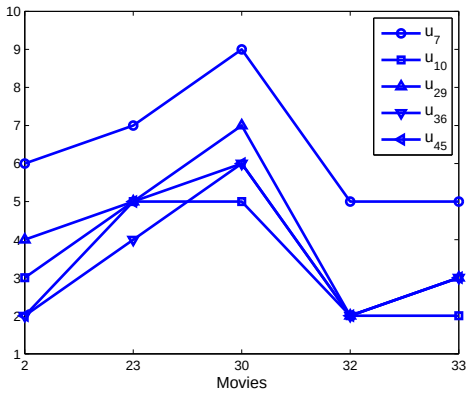


Figure 20 – Another example of a CHV bicluster considering $\epsilon = 2$.

It is important to emphasize that clustering may also be applied to a subset of movies. However, this subset should be known beforehand and distinct groups of users may require completely distinct subsets of movies.

Another important aspect in answering this question is that a user may belong to more than one group. For instance, let us suppose that the movies presented in Figure 19(a) are comedies. So, these users present a similar taste when considering comedy movies, not necessarily exhibiting similar tastes to other genres. In fact, a user from this group may belong to other groups when considering other genres. For instance, user u_7 belongs to the groups exhibited in Figure 19(a) and Figure 20. Traditional clustering techniques would assign user u_7 just to a single group.

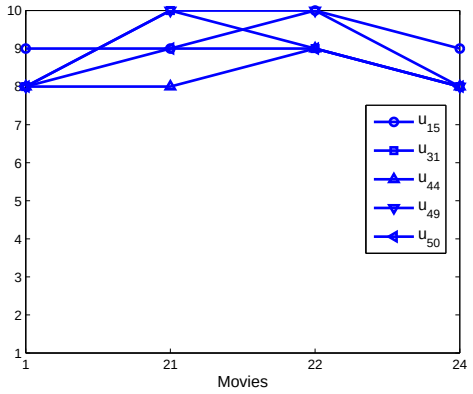


Figure 21 – Example of a CTV bicluster.

A.2 Biclusters with constant values (CTV biclusters)

So far, we have seen that biclustering can help us to find interesting patterns in this type of dataset. But what kind of bicluster is more useful? It depends on the kind of question we want to answer, i.e, our goals with the data analysis.

Suppose we want to find groups of users who have appreciated the same collection of movies. The first thing is to define what is to appreciate. Let us determine that if a user u_i has appreciated a movie m_j, then $a_{ij} \geq 8$. Thus, we can solve this problem by means of looking for perturbed CTV biclusters containing ratings from 8 to 10 (10 is the maximum rating), so the CTV biclusters should have maximum perturbation $\epsilon = 2$. As we are not interested in CTV biclusters with values other than 8 to 10, we could simplify the problem and binarize the dataset such that the binary value b_{ij} would be equal to 1 if $a_{ij} \geq 8$, and 0 otherwise.

Figure 21 shows an example of a CTV bicluster that groups together five users that have given high ratings for four movies. This bicluster represents the kind of pattern that we would obtain when solving this problem in this way.

A.3 Biclusters with constant values on columns or rows (CVC or CVR biclusters)

In the previous example, we considered that two users u_i and u_k are similar to each other if they have given similar ratings to a representative set of movies. More specifically, the same representative score is expected to occur in every element of the bicluster, thus characterizing a CTV bicluster. This definition of similarity between users is very strict. So, let us loose this definition and consider that a user u_i is similar to a user u_k if they have given similar ratings to a common set of movies such that the users may have liked some movies, and disliked others. Let us consider that the ratings for a movie m_j given by

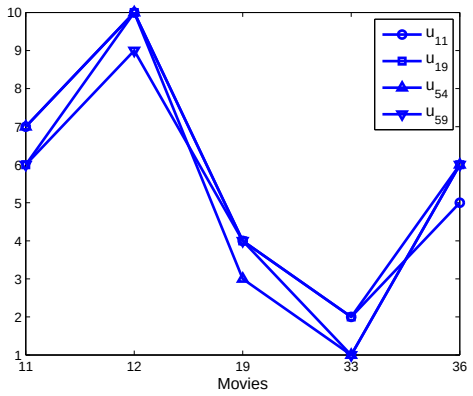
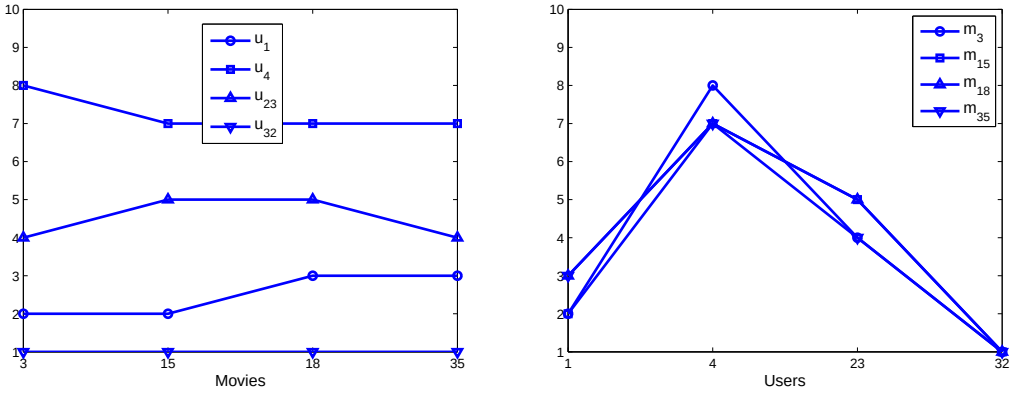


Figure 22 – Example of a CVC bicluster.



(a) Considering the original matrix. (b) Considering the transposed matrix.

Figure 23 – Example of a CVR bicluster.

users u_i and u_k are similar if $|a_{ij} - a_{kj}| \leq 1$, i.e., the difference between the ratings given by users u_i and u_k to movie m_j is at most 1. In this scenario, we can group the users by means of mining perturbed CVC biclusters with maximum perturbation $\epsilon = 1$. Figure 22 shows an example of a CVC bicluster with maximum perturbation $\epsilon = 1$. Observe how the users share a pattern along a wide range of scores.

The transposed version of this problem is to assume that a movie m_j is similar to a movie m_l if they have received similar ratings of a common set of users. In a similar way, let us consider that the ratings given by a user u_i to the movies m_j and m_l are similar if $|a_{ij} - a_{il}| \leq 1$. In this scenario, we can group the movies by means of mining perturbed CVR biclusters with maximum perturbation $\epsilon = 1$. Figure 23 shows an example of a CVR bicluster with maximum perturbation $\epsilon = 1$. We plotted the graphics in two perspectives: considering the original matrix (Figure 23(a)) and its transposed version (Figure 23(b)). In Figure 23(a), it is easy to observe that a user has given similar ratings to the movies of the group. In Figure 23(b), it is easy to observe how the movies share a pattern about being liked and disliked.

A.4 Biclusters with coherent values (CHV biclusters)

We can loose even more the definition of similarity between the users. For instance, in Figure 20, the users present the same behavior in evaluating the movies, however their ratings for a movie are not necessarily similar to each other. So, this kind of pattern is a generalization of the kind of pattern presented in Figure 22. In this case, each pair of users of a group must present a similar behavior in their evaluations. More specifically, if the ratings given by users u_i and u_k to a set of movies $\mathbf{m} = \{m_a, m_b, m_c, ...\}$ are similar, then the ratings given by user u_i must be explained by a shifting (or scaling) in the ratings given by user u_k. So, in this case, we can group the users by means of mining CHV biclusters.

Figures 19(a) and 20 show examples of CHV biclusters with maximum perturbation $\epsilon = 2$. It is clear in these examples that numerical proximity of elements does not matter in this kind of pattern.

A.5 Heuristics or Enumeration?

In the scenarios described here, what are the differences in the biclustering solutions when using heuristics or a enumeration (having the four key properties listed in Chapter 1: efficient, complete, correct, and non-redundant)?

When we use an actual enumerative algorithm, we find all groups of users (or movies) that attend our definition of similarity. Moreover, all the biclusters will be maximal, in the sense of incorporating all the possible users and movies without violating the similarity criterion.

Heuristics do not give us neither of these two guarantees. On the contrary, it is likely that a heuristic will find a small and incomplete subset of all possible biclusters, and the biclusters themselves will be an incomplete subset of their possible maximal versions. So, heuristics usually give us a small and twofold incomplete subset of the enumerative solution.

The drawback of enumerative algorithms is that, even being efficient, they do not scale well for all dataset sizes. So, enumerative algorithms become computationally unfeasible in many cases. Therefore, the usage of enumerative algorithms is recommended in all cases that the runtime is not prohibitive.