Enumerating all maximal biclusters in real-valued datasets

Rosana Veroneze, Aridam Banerjee, and Fernando J. Von Zuben

arXiv:1403.3562v2 [cs.DM] 8 Apr 2014

Abstract

Biclustering is a powerful data mining technique which simultaneously finds cluster structure over both objects and attributes in a data matrix. The main advantages of biclustering are twofold: first, a single object/attribute can belong to none, one, or more than one bicluster, allowing biclusters arbitrarily positioned in the data matrix; and second, biclusters can be defined using coherence measures which are substantially more general than distance measures generally used in clustering. In spite of the preliminary advances in non-partitional biclustering, the existing literature is only capable of efficiently enumerating biclusters with constant values for integer or real-valued data matrices. In this paper, we present a general family of biclustering algorithms for enumerating all maximal biclusters with (i) constant values on rows, (ii) constant value on columns, or (iii) coherent values. The algorithms have three key properties: they are efficient (take polynomial time between enumerating two consecutive biclusters), non-redundant (do not enumerate the same bicluster twice), and complete (enumerate all maximal biclusters). The proposed algorithms are based on a generalization of an efficient formal concept analysis algorithm denoted In-Close2. Experimental results with artificial and real-world datasets highlight the main advantages of the proposed methods in comparison to the state-of-the-art based on heuristics.

Index Terms

Biclustering, formal concept analysis, frequent pattern mining, maximal bicliques, data mining.

1 Introduction

Clustering is one of the most popular data mining techniques for knowledge discovery in datasets [1]. It consists of grouping the data into clusters, such that the data points inside a

- *R. Veroneze and F. J. Von Zuben are with the Department of Computer Engineering and Industrial Automation of the School of Electrical and Computer Engineering, University of Campinas (Unicamp), Campinas, São Paulo, Brazil. E-mail: {veroneze, vonzuben}@dca.fee.unicamp.br.*
- *A. Banerjee is with the Department of Computer Science and Engineering, University of Minnesota, Twin Cities, Minnesota, USA. E-mail: banerjee@cs.umn.edu.*

cluster are more similar to each other 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. 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 [2], [3], [4], [5].

Biclustering is a data mining technique that addresses these challenges by simultaneously clustering the set of objects and attributes of a data matrix. Biclustering finds coherent blocks constituted of subsets of objects and attributes, and allows an object/attribute to belong to none, one, or more than one bicluster. Thus, there may be overlap between the biclusters, and the biclusters can be arbitrarily positioned in the data matrix, visually revealed only with a suitable row-column permutation of the matrix. Further, biclustering methods can consider coherence measures which are more general than distance functions, such as Euclidean and Manhattan distances, and hence can find biclusters beyond numerical proximity of elements.

Due to the flexibility and wide ranging applications, biclustering methods have gained considerable attention over the past decade. For surveys, please refer to Madeira and Oliveira [6] and Busygin et al. [7]. Further, researches are realizing the connection of biclustering to several other important problems in large scale data analysis, including subspace clustering, frequent pattern mining, association analysis, and formal concept analysis.

The problem of counting the number of maximal biclusters (see Section 3) in a given dataset is a $\#P$-complete problem [6], and biclustering considers the enumeration version of the problem. The best one can computationally do in such a scenario is to develop an algorithm which (i) takes polynomial time to enumerate the first bicluster, (ii) takes polynomial time between enumerating two consecutive biclusters, (iii) is complete, i.e., enumerates all biclusters, and (iv) does a non-redundant enumeration, i.e., does not enumerate any bicluster more than once. If done properly, such an algorithm will have time complexity linear in the number of biclusters and polynomial in the input size. We will refer to any such algorithm as an *efficient enumeration* algorithm. Unfortunately, the biclustering algorithms in the literature are heuristics, and do not satisfy the above criteria of being an efficient enumeration algorithm [6].

In the special case of binary data, with biclusters defined as having constant value, the enumeration problem is equivalent to the problem of listing all maximal bipartite cliques in a bipartite graph over the rows and columns, for which efficient enumeration algorithms exist [8], [9]. Such algorithms have also been studied in the context of Formal Concept Analysis (FCA) for several years [10], leading to algorithms such as Close-by-One (CbO) [11], In-Close [12], In-Close2 [13], and FCbO [14]. These are efficient enumeration algorithms because they

can generate the concept lattice with polynomial time delay (i.e., the time taken to generate the first formal concept, and the time taken between any two consecutive formal concepts are both bounded by a polynomial in the input size), and space linear in the number of all formal concepts (modulo some factor polynomial in the input size) [15]. In particular, if the number of formal concepts is polynomial in the input size, the overall algorithm will be a polynomial time algorithm. FCA-based methods have been applied to various application domains, such as association mining, software mining, web mining, text mining, and bioinformatics. For a survey on applications refer to Poelmans et al. [16].

Recently, FCA-based algorithms have been developed for efficient enumeration of all maximal biclusters with constant values (CTV) in a numerical (integer or real-valued) data matrix [17], [18], [19]. CTV biclusters represent one of the four major types of biclusters defined in [6]. In addition, Madeira and Oliveira [6] discuss three other major types of biclusters: (i) biclusters with constant values on rows (CVR) or columns (CVC), (ii) biclusters with coherent values (CHV), and (iii) biclusters with coherent evolutions (CHE) (definitions in Section 3).

In this paper, we propose efficient enumeration algorithms for enumerating all maximal CVR, CVC, and CHV biclusters in a numerical data matrix. These types of biclusters are a generalization of CTV biclusters. We call our family of algorithms *RIn-Close* because they are generalizations of the FCA algorithm In-Close2 [13]. To the best of our knowledge, the proposed family of algorithms are the first ones to be able to perform a complete, correct, and non-redundant enumeration of all maximal CVR, CVC, and CHV biclusters in numerical data matrices. In particular, the computational cost of the RIn-Close family is proportional to the number of biclusters to be enumerated, and the algorithms are guaranteed to enumerate all maximal biclusters correctly.

The rest of the paper is organized as follows. Section 2 reviews related works. Section 3 introduces definitions and problem formulations for biclustering. Section 4 reviews FCA, related areas, and the algorithm In-Close2. Section 5 presents the main contributions of this paper, more specifically the RIn-Close family of efficient enumeration algorithms for CVR, CVC, and CHV biclusters. Experimental results are discussed in Section 6, and we conclude in Section 7.

2 Related Works

In FCA and related areas (see Subsection 4.1), there are efficient algorithms to mine all maximal CTV biclusters of ones from a binary dataset. Also, there are in FCA three proposals able to mine all maximal CTV biclusters from a numerical dataset [17], [18], [19]. In [17], the algorithm starts with a lattice containing all possible biclusters, and then explores recursively

its sublattices pruning unpromising branches. Kaytoue *et al.* [18] proposed two equivalent FCA-based methods. The first method is based on the discretization of the numerical dataset, which is called *scaling* [20]. The second one is based on *interval pattern structures* [21]. In [19], the authors also used an approach based on scaling, but instead of using standard FCA, they proposed an algorithm based on Triadic Concept Analysis (TCA) [22], which is an extension of FCA.

In subspace clustering, where biclustering is called *clustering by pattern similarity* [3], there are some algorithms that have an enumeration approach to mine CHV biclusters, as pCLuster [3], Maple [23], SeqClus [24] and CPT [5].

pCLuster was the first deterministic algorithm with an enumeration approach to mine CHV biclusters. However, pCluster has several shortcomings. It does not find all biclusters, finds biclusters that do not attend the user-defined measure of similarity, and returns redundant / non-maximal biclusters. Furthermore, pCluster's computational complexity is exponential with regard to the number of attributes.

Maple is an improved version of pCluster. It returns only maximal 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 / non-maximality. Besides, there are two scenarios where Maple fails in performing 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 (violating the user-defined measure of similarity). Maple may miss biclusters due to its routine of pruning unpromising biclusters. If a bicluster has a subset of objects and a superset of attributes of another bicluster, and its extra attributes are subsequent considering Maple's attribute-list, Maple would prune it incorrectly. In addition to these problems, the worst-case time of Maple's search is also exponential with regard to the number of attributes.

To reduce computational cost, SeqClus and CPT relaxed the definition of CHV biclusters (see Subsection 3.4). Instead of looking for every pair of attributes (or objects), they use a pivot attribute to compute the CHV biclusters. We can say that this strategy yields an approximate result for the actual enumeration. Unfortunately, when resorting to the pivot attribute, those techniques are prone to lose one fundamental property: the ability to obtain the same set of biclusters when dealing with the original matrix or with its transpose version. Notice that the CHV biclusters are fully preserved when rows become columns and columns become rows of the matrix.

3 BICLUSTERING

The term *biclustering* was introduced in [25] 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 in [2]. Cheng and Church [2] were responsible for the popularization of biclustering techniques with their algorithm called CC. However, Hartigan [26] was the first one to propose an algorithm capable of simultaneously clustering both rows and columns of a data matrix, using the term *direct clustering*. Other terms that are found in literature are: co-clustering, two-way clustering, bidimensional clustering, among others [6].

While clustering techniques look for similar rows (or columns) of a data matrix, biclustering techniques look for blocks (called *biclusters*) of rows and columns that are inter-related [27]. Biclustering is often more informative than clustering, and moreover it is able to effectively intertwine row and column information.

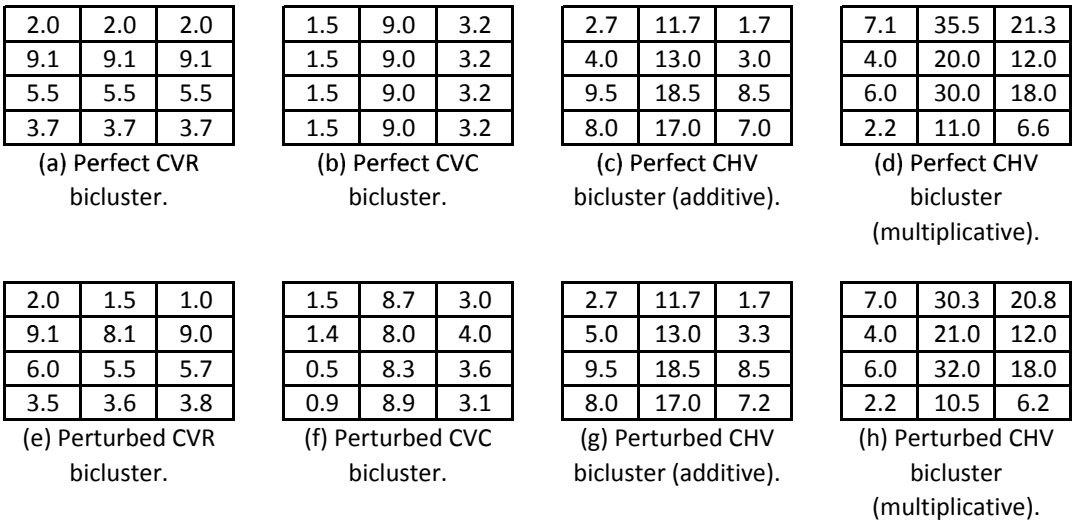
In the literature, most biclustering applications are in the analysis of gene expression data, but its application is already fully disseminated and not limited to biological data. For instance, we can mention: electoral data analysis [26], dimensionality reduction [28], text mining [29], [30], collaborative filtering [31], [32], and treatment of missing data [27], [33], [34]. 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 analysis; and (*ii*) unveiling the connection between biclustering and several other important problems, including subspace clustering, frequent pattern mining, and association analysis. Thus, the ability to efficiently find all the existing maximal biclusters in a dataset becomes a desirable goal.

3.1 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.

A bicluster is a submatrix $\mathbf{A}_{IJ} = (I, J)$ where 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. A biclustering algorithm looks for a set of biclusters $B_l = (I_l, J_l)$, such that each bicluster B_l satisfies some specific characteristics of homogeneity [6].

Considering these characteristics of homogeneity, there are four major types of biclusters [6]:



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

(a) Perfect CVR bicluster.

1.5	9.0	3.2
1.5	9.0	3.2
1.5	9.0	3.2
1.5	9.0	3.2

(b) Perfect CVC bicluster.

2.7	11.7	1.7
4.0	13.0	3.0
9.5	18.5	8.5
8.0	17.0	7.0

(c) Perfect CHV bicluster (additive).

7.1	35.5	21.3
4.0	20.0	12.0
6.0	30.0	18.0
2.2	11.0	6.6

(d) Perfect CHV bicluster (multiplicative).

2.0	1.5	1.0
9.1	8.1	9.0
6.0	5.5	5.7
3.5	3.6	3.8

(e) Perturbed CVR bicluster.

1.5	8.7	3.0
1.4	8.0	4.0
0.5	8.3	3.6
0.9	8.9	3.1

(f) Perturbed CVC bicluster.

2.7	11.7	1.7
5.0	13.0	3.3
9.5	18.5	8.5
8.0	17.0	7.2

(g) Perturbed CHV bicluster (additive).

7.0	30.3	20.8
4.0	21.0	12.0
6.0	32.0	18.0
2.2	10.5	6.2

(h) Perturbed CHV bicluster (multiplicative).

Fig. 1. Examples of different types of biclusters. Biclusters (e) to (h) consider a similarity constraint $\epsilon = 1$.

1) Biclusters with constant values (CTV).
2) Biclusters with constant values on rows (CVR) or columns (CVC).
3) Biclusters with coherent values (CHV).
4) Biclusters with coherent evolutions (CHE).

In this paper, we will propose biclustering algorithms for enumerating all maximal CVR, CVC and CHV biclusters in a numerical data matrix $\mathbf{A}_{n\times m}$. Fig. 1 exhibits examples of the different types of biclusters that we will address in this paper. We will explain them in the following subsections, and we will also explain what are CTV biclusters because there are some recent papers in FCA that focus on their enumeration [17], [18], [19]. For definitions and examples of all types of biclusters, refer to [6].

3.2 Biclusters with constant values (CTV)

A *perfect CTV bicluster* is a submatrix (I, J) such that $a_{ij} = a_{kl}$, $\forall i, k \in I$ and $\forall j, l \in J$. In real-world datasets, CTV biclusters are usually masked by noise. Therefore, we redefine a *CTV bicluster* as a submatrix (I, J) such that $|a_{ij} - a_{kl}| \leq \epsilon$, $\forall i, k \in I$ and $\forall j, l \in J$, i.e.,

$$\max_{i\in I, j\in J}(a_{ij}) - \min_{i\in I, j\in J}(a_{ij}) \leq \epsilon. \tag{1}$$

This definition of a CTV bicluster was first proposed in [17], where the authors came up with the algorithm NBS-Miner to enumerate all maximal CTV biclusters.

3.3 Biclusters with constant values on rows or columns (CVR or CVC)

A *perfect CVR bicluster* is a submatrix (I, J) such that $a_{ij} = a_{il}$, $\forall i \in I$ and $\forall j, l \in J$. See Fig. 1(a) for an example. In real-world datasets, CVR biclusters are usually masked by noise. Therefore, we redefine a *CVR bicluster* as a submatrix (I, J) such that $|a_{ij} - a_{il}| \leq \epsilon$, $\forall i \in I$ and $\forall j, l \in J$, i.e.,

$$\max_{j \in J}(a_{ij}) - \min_{j \in J}(a_{ij}) \leq \epsilon, \forall i \in I. \tag{2}$$

Fig. 1(e) shows an example of a perturbed version of a CVR bicluster with $\epsilon = 1$.

Similarly, a *perfect CVC bicluster* is a submatrix (I, J) such that $a_{ij} = a_{kj}$, $\forall i, k \in I$ and $\forall j \in J$. See Fig. 1(b) for an example. In real-world datasets, CVC biclusters are usually masked by noise. Therefore, we redefine a *CVC bicluster* as a submatrix (I, J) such that $|a_{ij} - a_{kj}| \leq \epsilon$, $\forall i, k \in I$ and $\forall j \in J$, i.e.,

$$\max_{i \in I}(a_{ij}) - \min_{i \in I}(a_{ij}) \leq \epsilon, \forall j \in J. \tag{3}$$

Fig. 1(f) shows an example of a perturbed version of a CVC bicluster with $\epsilon = 1$.

3.4 Biclusters with coherent values (CHV)

There are two perspectives for CHV biclusters: (*i*) additive model, and (*ii*) multiplicative model. In the additive model, any row (column) of a perfect CHV bicluster can be obtained by adding a constant value to any other row (column) of the bicluster. For instance, in Fig. 1(c): 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, in the multiplicative model, any row (column) of a perfect CHV bicluster can be obtained by multiplying a constant value to any other row (column) of the bicluster. For instance, in Fig. 1(d): 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.

Consider $Z^{jl} = \{a_{ij} - a_{il}\}_{i \in I}$, $j, l \in J$, in the case of additive coherence, and $Z^{jl} = \{a_{ij}/a_{il}\}_{i \in I}$, $j, l \in J$, in the case of multiplicative coherence. For instance, consider the bicluster of Fig. 1(g), and let $j = 1$ and $l = 3$, then $Z^{13} = \{1, 1.7, 1, 0.8\}$. A *perfect CHV bicluster* is a submatrix (I, J) such that all elements of the set Z^{jl}, $\forall j, l \in J$, are equal, i.e., $z = w$, $\forall z, w \in Z^{jl}$, $\forall j, l \in J$. Again, due to noise in real-world datasets, we will define a *CHV bicluster* as a submatrix (I, J) such that $|z - w| \leq \epsilon$, $\forall z, w \in Z^{jl}$, $\forall j, l \in J$, i.e.,

$$\max(Z^{jl}) - \min(Z^{jl}) \leq \epsilon, \forall j, l \in J. \tag{4}$$

Figs. 1(g) and (h) are examples of CHV biclusters using additive and multiplicative coherence, respectively.

Note that we could also define CHV biclusters using $Z^{ik} = \{a_{ij} - a_{kj}\}_{j \in J}$, $i, k \in I$ in the case of additive coherence, and $Z^{ik} = \{a_{ij}/a_{kj}\}_{j \in J}$, $i, k \in I$ in the case of multiplicative coherence.

3.5 Maximal biclusters

Given the desired characteristics of homogeneity, a bicluster (I, J) is called a *maximal bicluster* iff:

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

It means that a bicluster is maximal if we can not add any object/attribute to it without violating the desired characteristics of homogeneity.

4 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 [20]. Here, we will explain the basic principles of FCA. For more details refer to [20].

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 1 shows an example of a formal context represented by a binary matrix.

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

$$A' = \{m \in M | \forall g \in A : gIm\} \tag{5}$$

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

$$B' = \{g \in G | \forall m \in B : gIm\} \tag{6}$$

(the set of objects common to the attributes in B). These derivation operators $\{(.)', (.)'\}$ define a Galois connection between the power sets of G and M [20].

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 $B' = A$. Therefore, besides many values of A can generate the same value of B, only the largest (closed) value of A is part of a formal concept (and vice versa). Considering the dataset in Table 1, with $A = \{g_3, g_6\}$, we have $A' = B = \{m_3, m_4, m_7\}$, but this pair (A, B) is not a

TABLE 1
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

formal concept because $B' = \{g_1, g_3, g_6\} \supset A$. The part A of a formal concept (A, B) is called *extent*, and the part B is called *intent*.

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) [11], In-Close [12], In-Close2 [13], and FCbO [14]. As the algorithms that we will propose are based on In-Close2, subsection 4.2 is devoted to its formalization.

4.1 Related areas of research in the literature

Table 1 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. Thus, 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.

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 from these itemsets. As the first sub-problem 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 $X \subseteq M$ such that the support of X ($supp(X) = |X'|$) is above a user-defined parameter [35]. Examples of algorithms that perform this task are Apriori [36] and Eclat [37]. To reduce the computational cost of the frequent pattern mining problem, some algorithms (as GenMax [38]) 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 *frequent closed itemsets*. A frequent itemset X is called closed if there exists no superset $Y \supset X$ with $X' = Y'$. The frequent closed itemsets are also called *frequent concept intents*. For any itemset X, its concept intent is given by X''. 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 [35]. Indeed, the set of frequent closed itemsets uniquely determines the exact frequency of all itemsets, and it can be orders of magnitude smaller than the set of all frequent itemsets [39]. Moreover, this approach drastically reduces the number of rules that have to be presented to the user, without any information loss [35]. CHARM [39] is a well-known algorithm to mine all frequent closed 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 [39].

The problem of enumerating all maximal bicliques from a bipartite graph is also closely related to FCA. Madeira and Oliveira [6] stated that in the simplest case of biclustering, where we are looking for CTV biclusters of ones in a binary data matrix $\mathbf{D}$, a bicluster corresponds to a biclique in the corresponding bipartite graph. Rows and columns of the matrix $\mathbf{D}$ correspond, respectively, to the first and second sets of vertices of a bipartite graph. For example, in Table 1, 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 vertice i is connected to vertice j, and 0 otherwise. Thus, the binary matrix $\mathbf{D}$ is the adjacency matrix of a bipartite graph. In this scenario, a formal concept from the binary matrix $\mathbf{D}$ 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, as [40], [41], [42]. Moreover, Gély *et al.* [42] 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 [43]. In fact, Li *et al.* [43] 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.* [43] also modified the LCM [44] (an algorithm for finding frequent closed 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 specific nature of particular classes of graphs, leading to highly efficient solutions [45]. For example, Eppstein's algorithm [46] 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 the FCA algorithms to efficiently find all maximal cliques of an undirected graph. For this, 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.

4.2 In-Close2

In-Close2 [13] and its precursor In-Close [12] are based conceptually on Close-By-One [11]. These algorithms use the lexicographic approach for mining formal concepts avoiding the overheads of computing repeated ones. In-Close2 [13] is a computationally faster version of In-Close [12]. 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 explain it here.

Ganter [10] 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 (*canon*), for instance, {1, 2, 3} comes before {1, 2, 4}, and {1, 2, 6} comes before {1, 3} [12]. Ganter's algorithm [10], and subsequent algorithms, maintain a *current object* [12]. The concept next generated is new (canonical) if its extent contains no object preceeding the current object. In-Close and In-Close2 maintain a *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

$$\exists k \in M \setminus B | [k < j] \ \wedge \ [\forall g \in RW : gIk]. \tag{7}$$

i.e., if there is an attribute $k < j$ where $k \notin B$ and $RW \subseteq \{k\}'$. The concept of canonicity was introduced in [47].

Algorithm 1 shows In-Close2 pseudocode. 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 (R_k) it means the element of the set J (R) at position k, for instance, if $J = \{2, 5, 7, 8, 13\}$ and $k = 2$, $J_k = 5$. In the main function of In-Close2, 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), beginning 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. If the current attribute j is not an inherited attribute, In-Close2 computes the candidate to a new extent RW. If the extent RW is the same as the current extent A_r, then attribute j is added to the current intent B_r. If the extent RW is not the same as the current extent A_r, In-Close2 tests if RW is canonical. If yes, the current formal concept (A_r, B_r) will give rise to a child formal concept. After the closure of the current formal concept (A_r, B_r), In-Close2 starts to close their children.

Algorithm 1 In-Close2

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

$J \leftarrow \{\}$
$R \leftarrow \{\}$
for $j \leftarrow y$ to m **do**
  if $j \notin B_r$ **then**
    $RW \leftarrow A_r \cap \{j\}'$
    if $|RW| \geq minRow$ **then**
      if $|RW| = |A_r|$ **then**
        $B_r \leftarrow B_r \cup \{j\}$
      else if RW is canonical **then**
        $r_{new} \leftarrow r_{new} + 1$ // global variable
        $J \leftarrow J \cup \{j\}$
        $R \leftarrow R \cup \{r_{new}\}$
        $A_{r_{new}} \leftarrow RW$
for $k \leftarrow 1$ to $|J|$ **do**
  $B_{R_k} \leftarrow B_r \cup \{J_k\}$
  In-Close2$(\mathbf{D}, minRow, R_k, J_k + 1)$

The worst-case time of In-Close2 is $O(knm^2)$, where k is the number of biclusters. 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* [48].

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

TABLE 2
Closure of the supremum formal concept (A_1, B_1) for the data in Table 1.

	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_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 save computational resources because it avoids generating descendants that do not meet the restriction $minCol$.

4.2.1 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 1. The supremum is initialized as $(A_1 = \{1, 2, 3, 4, 5, 6\}, B_1 = \{\})$. Supposing that $minRow = 2$, Table 2 shows all the extents RW computed during the closure of the supremum formal concept (A_1, B_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).

To illustrate, let's compute the closure of the formal concept (A_2, B_2). Table 3 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 $minRow$).

5 RIn-Close

In-Close2 has been specifically designed to extract all maximal constant biclusters of ones from a binary data matrix. Now, we will propose adaptations of In-Close2 to enumerate other types of biclusters from numerical matrices. We call our family of algorithms RIn-Close.

TABLE 3
Closure of the formal concept (A_2, B_2) for the data in Table 1.

	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

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 real-valued data matrices. For instance, for finding CVC biclusters, given the current attribute, we can look for the subsets of rows of the extent that accomplish the similarity restriction ϵ.

RIn-Close operation is basically the same as In-Close2 operation. The major differences are in the fact that: in In-Close2, each bicluster (A_r, B_r) can generate just one descendant per attribute, whereas in RIn-Close, each bicluster (A_r, B_r) can generate multiple descendants per attribute (possibly with overlap between their extents). Therefore, RIn-Close must deal with this new situation. When there is no overlap, the adaptations are straightforward. Otherwise, the test of canonicity is not sufficient to avoid redundant biclusters. As a consequence, RIn-Close must use new resources to avoid redundancy.

5.1 Biclusters with constant values on columns

The algorithms that we will present now compute a 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 ϵ ($\epsilon > 0$), we will show how to extract non-perfect CVC biclusters.

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,

	m_1	m_3	m_7		m_8
g_2	1	1	1		0
g_3	1	1	1		1
g_4	1	1	1		1
g_8	1	1	1	+	0
g_9	1	1	1		1
g_{10}	1	1	1		0
g_{11}	1	1	1		0
g_{15}	1	1	1		1

=

	m_1	m_3	m_7	m_8
g_3	1	1	1	1
g_4	1	1	1	1
g_9	1	1	1	1
g_{15}	1	1	1	1

Fig. 2. Example of the generation of a single descendant by In-Close2.

	m_1	m_3	m_7		m_8
g_2	3	1	4		2
g_3	3	1	4		4
g_4	3	1	4		4
g_8	3	1	4	+	3
g_9	3	1	4		3
g_{10}	3	1	4		3
g_{11}	3	1	4		1
g_{15}	3	1	4		2

=

	m_1	m_3	m_7	m_8	
g_{11}	3	1	4	1	**(a)**
g_2	3	1	4	2	**(b)**
g_{15}	3	1	4	2	
g_8	3	1	4	3	
g_9	3	1	4	3	**(c)**
g_{10}	3	1	4	3	
g_3	3	1	4	4	**(d)**
g_4	3	1	4	4	

Fig. 3. Example of the generation of multiple descendants by RIn-Close_CVC_P.

each bicluster (A_r, B_r) can generate multiple descendants per attribute. It happens because In-Close2 just looks for blocks of 1s, whereas RIn-Close_CVC_P looks for any blocks of constant values on columns. Fig. 2 and Fig. 3 illustrate this difference. In Fig. 2, 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 Fig. 3, 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: (a) $(A = \{g_{11}\}, B = \{m_1, m_3, m_7, m_8\})$, (b) $(A = \{g_2, g_{15}\}, B = \{m_1, m_3, m_7, m_8\})$, (c) $(A = \{g_8, g_9, g_{10}\}, B = \{m_1, m_3, m_7, m_8\})$, and (d) $(A = \{g_3, g_4\}, B = \{m_1, m_3, m_7, m_8\})$. Note that we sort the elements of the current attribute (m_8 in our example) in order to easily identify all subsets of objects with constant values.

Algorithm 2 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.

Algorithm 2 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
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: for k ← 1 to |J| do
16:    B_{R_k} ← B_r ∪ {J_k}
17:    RIn-Close_CVC_P(D, minRow, R_k, J_k + 1)
```

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 the extent of the new bicluster, it is not canonical if

$$\exists k \in M \setminus B | [k < j] \ \wedge \ [\max_{i \in RW}(d_{ik}) - \min_{i \in RW}(d_{ik}) = 0], \tag{8}$$

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.

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

5.1.2 Non-Perfect Biclusters

Now, we will explain how to perform a complete, correct and non-redundant enumeration of all maximal CVC biclusters, given a similarity constraint determined by the user-defined parameter ϵ ($\epsilon > 0$), as presented in (3). This adaptation of In-Close2, called RIn-Close_CVC,

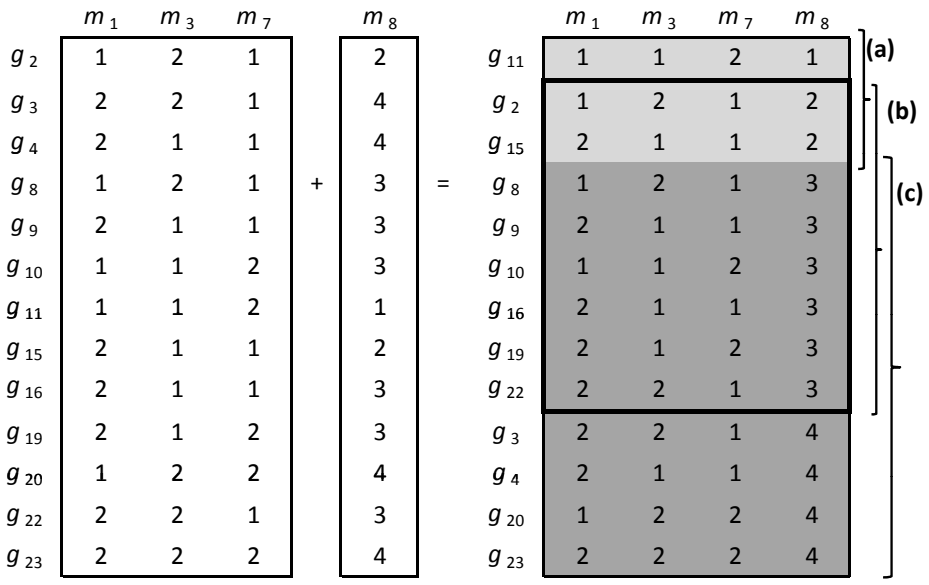


Fig. 4. Example of the generation of multiple descendants by RIn-Close_CVC (considering $\epsilon = 1$).

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 occur overlapping between their extents. For instance, in Fig. 4, RIn-Close_CVC is closing the bicluster $(A_r = \{g_2, g_3, g_4, g_8, g_9, g_{10}, g_{11}, g_{15}, g_{16}, g_{19}, g_{20}, g_{22}, g_{23}\}, B_r = \{m_1, m_3, m_7\})$, when it tries to add the current attribute m_8, bicluster (A_r, B_r) gives rise to three new biclusters with overlapping between their extents (considering $\epsilon = 1$). Notice again that we sort the elements of the current attribute, m_8 in our example, to facilitate the identification of all possible extents.

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

Assuming $minRow = 3$, in our example in Fig. 4, biclusters (a) and (b) can not generate redundant biclusters because they share only 2 rows in their extents. But biclusters (b) and (c) can generate redundant biclusters with extent $A \subseteq \{g_8, g_9, g_{10}, g_{16}, g_{19}, g_{22}\}$ and $|A| > minRow$, 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. 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 the extent of the new bicluster, it is not canonical if

$$\exists k \in M \setminus B | [k < j] \ \wedge \ [\max_{i \in RW}(d_{ik}) - \min_{i \in RW}(d_{ik}) \leq \epsilon], \tag{9}$$

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 Fig. 4, bicluster (c) 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 (b) 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 (b) and (c), whose extents is 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.,

$$\exists g \in RM | [\max_{i \in \{A \cup \{g\}\}}(d_{ij}) - \min_{i \in \{A \cup \{g\}\}}(d_{ij}) \leq \epsilon], \forall \ j \in B. \tag{10}$$

To explain how to compute RM, let us see the example in Fig. 5, which considers $\epsilon = 3$ and $minRow = 2$. Suppose that when adding attribute m_x, a bicluster generated four biclusters: (a), (b), (c) and (d), whose extents are highlighted in Fig. 5. Let us compute the set of rows RM that the descendants of the bicluster (b) must check to verify their maximality, i.e., $RM_{(b)}$. As the problems occurs when biclusters share $minRow$ rows or more in their extents, the pivot elements to compute $RM_{(b)}$ are g_e and g_h because they are the $minRow$-th first and last element of (b), 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 $RM_{(b)}$, so $RM_{(b)} = \{g_a, g_b, g_c, g_j\}$.

Back to our example in Fig. 4, $RM_{(b)} = \{g_3, g_4, g_{20}, g_{23}\}$. So, all descendants of the bicluster (b) with extent $A \subseteq \{g_8, g_9, g_{10}, g_{16}, g_{19}, g_{22}\}$ must test the rows in $RM_{(b)}$ to verify if they are

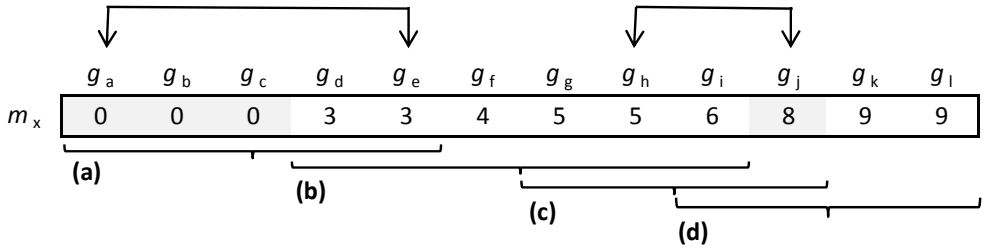


Fig. 5. Example of how to find RM (considering $\epsilon = 3$ and $minRow = 2$).

row-maximal. For simplicity, the result will be correct if you 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 (b) of Fig. 4 give rise to a bicluster $(A_x = \{g_8, g_9, g_{19}, g_{22}\}, B_x)$. So, we must have $RM_x \supseteq RM_{(b)}$ because the descendants of (A_x, B_x) must test the rows in $RM_{(b)}$ to verify if they are maximal.

Algorithm 3 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 Biclusters with constant values on rows

To compute a complete, correct and non-redundant enumeration of all maximal CVR biclusters, we have two simple options:

1) Run RIn-Close_CVC (RIn-Close_CVC_P) with the transpose of $\mathbf{D}$.
2) Change RIn-Close_CVC (RIn-Close_CVC_P) to (*i*) start with a bicluster with all attributes in B and no objects in A, (*ii*) keep the current object, and (*iii*) loop through the objects.

5.3 Biclusters with coherent values

The algorithms that we will present now compute a complete, correct and non-redundant enumeration of all maximal CHV biclusters. Once again, we will first show how to enumerate perfect biclusters. We named this algorithm RIn-Close_CHV_P. It is very similar to RIn-Close_CVC_P. Secondly, given a user-defined parameter ϵ ($\epsilon > 0$), we will show how to enumerate non-perfect CHV biclusters. We named this algorithm RIn-Close_CHV. It is not similar to RIn-Close_CVC, but it uses RIn-Close_CVC as an internal procedure. We begin with the **additive model**.

Algorithm 3 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, user-defined parameter 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
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: for k ← 1 to |J| do
19:    B_{R_k} ← B_r ∪ {J_k}
20:    RIn-Close_CVC(D, minRow, R_k, J_k + 1, ε)
```

5.3.1 Perfect Biclusters

The adaptation of In-Close2 to enumerate all maximal perfect CHV biclusters, called RIn-Close_CHV_P, is also very simple and very close to RIn-Close_CVC_P.

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 4 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 column 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, we just compute the difference between the current attribute j and the pivot column of the current intent Br, 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 4 Main_RIn-Close_CHV_P

Input: Data matrix $\mathbf{D}_{n\times m}$, minimum number of rows $minRow$

1: $r_{new} \leftarrow 0$ // global variable
2: **for** $atr \leftarrow 1$ to $m-1$ **do**
3: $\quad r_{new} \leftarrow r_{new} + 1$
4: $\quad A_{r_{new}} \leftarrow \{1, ..., n\}$
5: $\quad B_{r_{new}} \leftarrow \{atr\}$
6: $\quad$ RIn-Close_CHV_P($\mathbf{D}, minRow, r_{new}, atr + 1$)

Algorithm 5 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: $J \leftarrow \{\}$
2: $R \leftarrow \{\}$
3: **for** $j \leftarrow y$ to m **do**
4: $\quad$ **if** $j \notin B_r$ **then**
5: $\quad\quad Z \leftarrow \{d_{iB_{r_1}} - d_{ij}\}_{i\in A_r}$
6: $\quad\quad$ **if** $\max(Z) - \min(Z) = 0$ **then**
7: $\quad\quad\quad B_r \leftarrow B_r \cup \{j\}$
8: $\quad\quad$ **else**
9: $\quad\quad\quad$ Compute the possible extents
10: $\quad\quad\quad$ **for** each possible extent RW **do**
11: $\quad\quad\quad\quad$ **if** $|RW| \geq minRow$ **and** RW is canonical **then**
12: $\quad\quad\quad\quad\quad r_{new} \leftarrow r_{new} + 1$
13: $\quad\quad\quad\quad\quad R \leftarrow R \cup \{r_{new}\}$
14: $\quad\quad\quad\quad\quad J \leftarrow J \cup \{j\}$
15: $\quad\quad\quad\quad\quad A_{r_{new}} \leftarrow RW$
16: **for** $k \leftarrow 1$ to $|J|$ **do**
17: $\quad B_{R_k} \leftarrow B_r \cup \{J_k\}$
18: $\quad$ RIn-Close_CHV_P($\mathbf{D}, minRow, R_k, J_k + 1$)

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 the extent of the new bicluster, it is not canonical if

$$\exists k \in M \setminus B | [k < j] \ \wedge \ [\max(Z^{B_{r_1} k}) - \min(Z^{B_{r_1} k}) = 0], \tag{11}$$

where $Z^{B_{r_1} k} \leftarrow \{d_{iB_{r_1}} - d_{ik}\}_{i \in RW}$, i.e., if there is an attribute $k < j$ that we can add to the bicluster 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: $O(knm(\log n + m))$.

5.3.2 Non-Perfect Biclusters

Now, we will explain how to perform a complete, correct and non-redundant enumeration of all maximal CHV biclusters, given a similarity constraint determined by the user-defined parameter ϵ, as presented in (4). This adaptation is called RIn-Close_CHV.

To achieve this goal, we can not simply apply in RIn-Close_CHV_P the same adaptations that we 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 pivot column B_{r_1} and the current attribute 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 interfere in the outcome in this scenario. In this way RIn-Close_CHV would yield just an approximate result of an actual enumeration (as SeqClus [24] and CPT [5] do). 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 $B_y = \{m_1, m_3, m_6, m_8\}$, RIn-Close_CHV could 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 serious problem with 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 $B_y = \{m_2, m_3, m_4, m_5\}$, RIn-Close_CHV would discard the second because the attributes m_2, m_3 and m_4 are coherent with attribute m_1. Not to mention that a non-canonical bicluster could give rise to a canonical bicluster.

To avoid this undesired behavior, RIn-Close_CHV uses the following procedure:

1) 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 two columns of $\mathbf{D}$. For instance, the augmented matrix $\mathbf{D}^a$ of the data matrix $\mathbf{D}$ in Table 4 is illustrated in Table 5. 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.

TABLE 4

Example of a numerical dataset (extracted from [17])

	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 5

Augmented matrix $\mathbf{D}^a$ of the data matrix in Table 4.

	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

2) Apply RIn-Close_CVC in the augmented matrix $\mathbf{D}^a$. To illustrate, Fig. 6 shows all maximal CVC biclusters found by RIn-Close_CVC applied in the data matrix of Table 5 (using $minRow = 2$ and $\epsilon = 1$).
3) Extract all maximal CHV biclusters from the maximal CVC biclusters found by RIn-Close_CVC (see the pseudocode in Algorithm 6).

Algorithm 6 shows the pseudocode of the procedure to process each CVC bicluster. Steps in

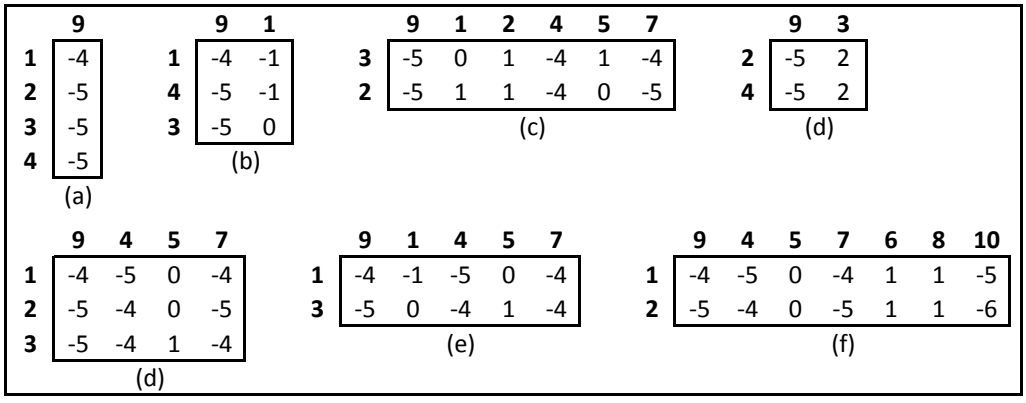


Fig. 6. CVC biclusters found in the data matrix of Table 5 (using $minRow = 2$ and $\epsilon = 1$).

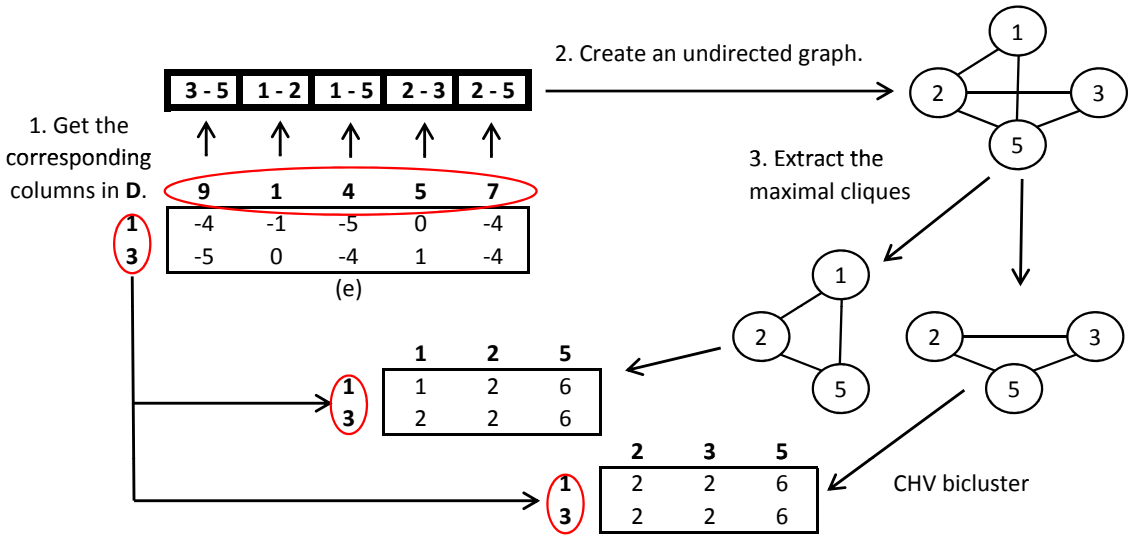


Fig. 7. Illustrative scheme to show how a CVC bicluster is processed by RIn-Close_CHV.

lines 2 to 5 are illustrated in Fig. 7. In this illustrative scheme, we are extracting CHV biclusters from the CVC bicluster (e) of Fig. 6. The first step is to get the corresponding columns in $\mathbf{D}$ of the intent of bicluster (e). For instance, column 9 of $\mathbf{D}^a$ corresponds to the difference between columns 3 and 5 of $\mathbf{D}$. Therefore, columns 3 and 5 are coherent with each order considering the extent $\{1, 3\}$ and $\epsilon = 1$. Let's name this corresponding set of columns as $B2$. The second step is to create an undirected graph, in which the nodes represent $B2$, and the edges represent the columns that are coherent with each other. The third step is to find all maximal cliques from this undirected graph. Each one of these cliques indicates the subsets of $B2$ in which all columns are coherent with each other considering the user-defined parameter ϵ. Thus, the CHV biclusters generated by the CVC bicluster (e) are: $(\{1, 3\}, \{1, 2, 5\})$ and $(\{1, 3\}, \{2, 3, 5\})$. Lines 7 and 8 of the pseudocode in Algorithm. 6 verify if the CHV bicluster (C, D) is new and, if so, store it in the list of CHV biclusters. A CHV bicluster (C, D) is new if (*i*) its intent D is equal to $B2$, or (*ii*) (C, D) is row-maximal (there is no object g that we can add to its extent C), i.e.,

$$\nexists g \in G \setminus C | [\max(Z^{jl}) - \min(Z^{jl}) \leq \epsilon], \forall j, l \in D, \tag{12}$$

where $Z^{jl} = \{d_{ij} - d_{il}\}_{i \in C \cup \{g\}}$.

Algorithm 6 Mining CHV biclusters from CVC biclusters

Input: List of CVC biclusters $lbCVC$

Output: List of CHV biclusters

```
for each (A, B) in lbCVC do
    Compute B2
    Create an undirected graph
    Find all maximal cliques
    Compute the CHV biclusters
    for each CHV bicluster (C, D) do
        if D = B2 or (C, D) is row-maximal then
            Store (C, D) in the list of CHV biclusters
```

5.3.3 *Multiplicative model*

To compute a complete, correct and non-redundant enumeration of all maximal CHV biclusters using a multiplicative model, we have two simple options:

1) Run RIn-Close_CHV (RIn-Close_CHV_P) using the log of the values of $\mathbf{D}$.
2) Change RIn-Close_CHV (RIn-Close_CHV_P) to compute divisions instead of differences between the attributes.

6 Experimental Results

In this section, we will show the experiments that we performed with the RIn-Close family of algorithms just proposed in this paper. Our goals are to highlight various practicalities in the usage of RIn-Close, and to outline the advantages and distinct aspects of an enumeration algorithm when compared to heuristics.

Metrics. Now, we will define some metrics that are necessary to understand the results of our experiments. In a perfect CHV bicluster (I, J), each element a_{ij}, $i \in I$, $j \in J$, is given by (considering an additive model):

$$a_{ij} = a_{iJ} + a_{Ij} - a_{IJ}, \tag{13}$$

where $a_{iJ} = \frac{1}{|J|}\sum_{j\in J} a_{ij}$, $a_{Ij} = \frac{1}{|I|}\sum_{i\in I} a_{ij}$, and $a_{IJ} = \frac{1}{|I||J|}\sum_{i\in I, j\in J} a_{ij}$. Thus, the quality of a CHV bicluster can be measured by the *mean squared residue* (MSR) [2]:

$$H(I, J) = \frac{1}{|I||J|}\sum_{i\in I, j\in J}(a_{ij} - a_{iJ} - a_{Ij} + a_{IJ})^2. \tag{14}$$

If the bicluster (I, J) is perfect, $H(I, J) = 0$, otherwise $H(I, J) > 0$. Therefore, the lower the value of $H(I, J)$, the better the bicluster quality in terms of residue. Another metric is the

volume of a bicluster (I, J), which is given by $|I| \times |J|$. Biclusters with very small volumes tend to be poorly informative. Volume and MSR are two conflicting metrics. The *overlap* between two biclusters (A, B) and (C, D) is given by:

$$\frac{|A \cap C| \times |B \cap D|}{\min(|A| \times |B|, |C| \times |D|)}. \tag{15}$$

The overlap ranges from 0 to 1, where 0 means no overlap between the biclusters, and 1 means that there is total overlap between the biclusters. Finally, the *coverage* of a biclustering result is given by the percentage of cells of the data matrix covered by at least one bicluster. The coverage is also conflicting with the MSR.

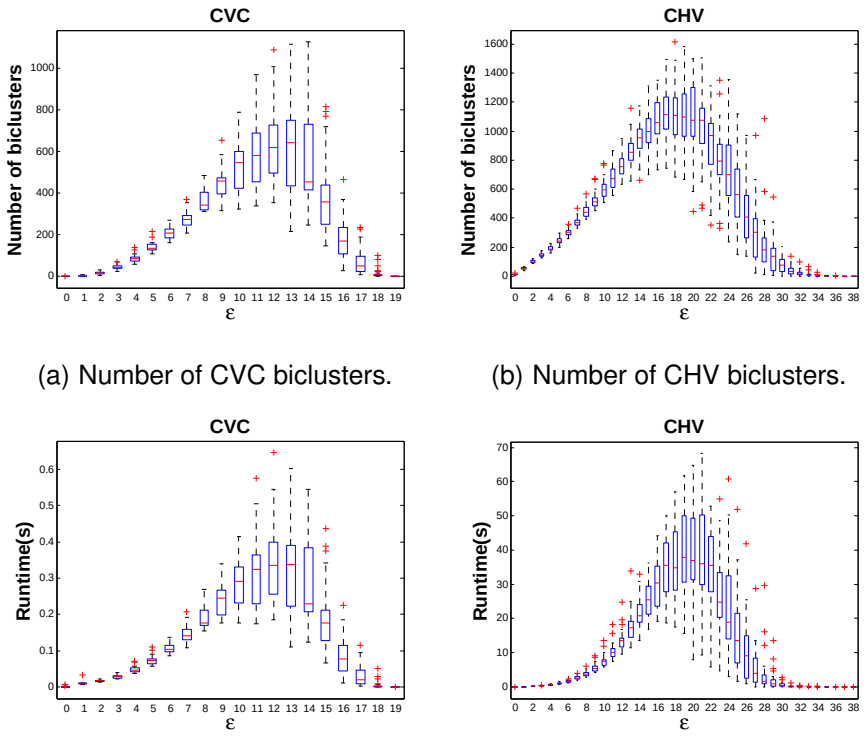
Note. When we report results considering $\epsilon = 0$, they were achieved by means of the algorithms for finding perfect biclusters, RIn-Close_CVC_P and RIn-Close_CHV_P. All algorithms were implemented in MATLAB, except pCluster whose executable was obtained from the author's website: http://haixun.olidu.com/system.html. The experiments were carried out on a PC Intel(R) Core(TM) i5-2400 CPU @ 3.1 GHz, 4 GB of RAM, and running under Windows 7 64 bits.

6.1 Experiments with Synthetic Data

This experiment aims to illustrate the influence of the parameter ϵ on the outputs of RIn-Close algorithms. We will show that even with a very small dataset, the amount of biclusters can be huge depending on the value of ϵ.

We ran RIn-Close algorithms in 30 instantiations of a randomly generated dataset with 15 objects and 8 attributes. Each dataset entry is a random integer drawn from a discrete uniform distribution on the set $\{1, 2, ..., 20\}$. With this simple process, small biclusters are randomly generated. So, we set the parameters $minRow$ and $minCol$ to 3 and 2, respectively.

Figs. 8 and 9 show the results of this experiment. Figs. 8(a) and (b) show the number of CVC and CHV biclusters, respectively. The number of biclusters varied from one instantiation to another, but for all instantiations the number of biclusters dizzily grew with the value of ϵ and started to decrease when the value of ϵ was large enough (reaching a single bicluster that corresponds to the entire dataset). As the largest possible difference between two elements of a dataset's column is 19, $\epsilon = 19$ provided just 1 biclusters for all instantiations in the CVC case, but $\epsilon = 18$ was enough for some instantiations (when the largest difference was 18). In the CHV case, this happened with ϵ ranging from 30 to 37. As CVC biclusters are special cases of CHV biclusters, it was expected that the number of CHV biclusters was equal or larger than the number of CVC biclusters for the same value of ϵ. The running time of RIn-Close



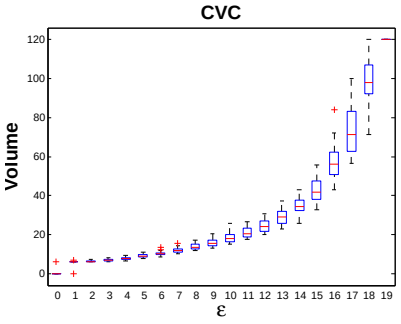
(a) Number of CVC biclusters. (b) Number of CHV biclusters.

(c) Running time of RIn-Close_CVC. (d) Running time of RIn-Close_CHV.

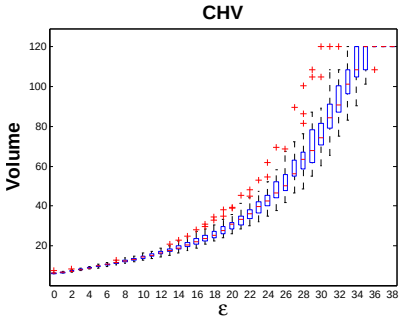
Fig. 8. Results on synthetic data - the influence of the similarity constraint ϵ on the number of biclusters and running time. All box plots taken as a function of ϵ.

algorithms was proportional to the number of biclusters, as illustrated in Figs. 8(c) and (d). The average volume, shown in Fig. 9(a) and (b), increased with the value of ϵ, reaching 120 when a bicluster corresponds to the entire dataset. Fig. 9(c) and (d) help us to explain why the number of biclusters increased that much. The increase in the number of biclusters is related to the increase in the overlap between them. The overlap rate goes directly to zero when the number of biclusters collapses to one. Fig. 9(e) shows the impact of the value of ϵ on the MSR of the CHV biclusters. Evidently, the increase in the value of ϵ degrades the quality of the biclusters, since it is relaxing the similarity constraint.

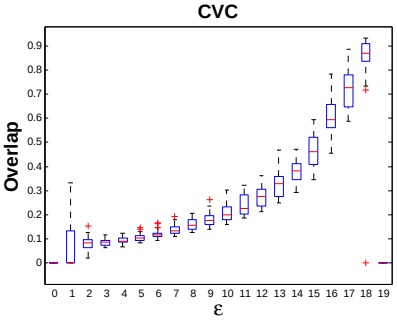
On the one hand, the choice of ϵ must consider that small values can prevent interesting biclusters from being found due to noise in the dataset. On the other hand, large values for ϵ may deteriorate too much the quality of the biclusters. Moreover, the value of ϵ plays an important role in the algorithm running time (see Fig. 8).



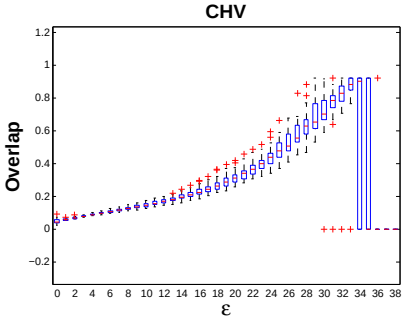
(a) Volume of the CVC biclusters.



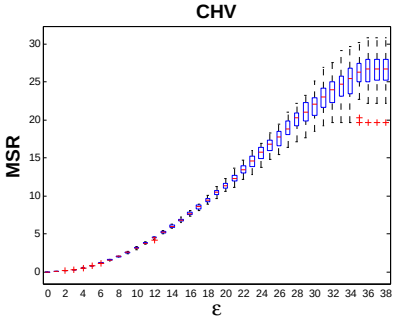
(b) Volume of the CHV biclusters.



(c) Overlap of the CVC biclusters.



(d) Overlap of the CHV biclusters.



(e) MSR of the CHV biclusters.

Fig. 9. Results on synthetic data - the influence of the similarity constraint ϵ on the volume, overlap and MSR of the biclusters. All box plots taken as a function of ϵ.

6.2 Experiments on Real Datasets

In this experiment, we ran RIn-Close in two real-world datasets: *Yeast*[1] [49] and *Food*[2]. We performed 20 executions of RIn-Close to compute the average running times.

Yeast is a gene expression dataset, which contains 2,884 genes under 17 conditions. Each entry in the matrix is an integer value that represents the logarithm of the ratio of the actual strength of the gene under the test condition and the expected strength of the gene [50]. *Yeast* has already been used in many papers in the area of biclustering [7]. To perform our experiments, we replaced the 0.07% of *Yeast* missing entries by random values in the same range as the known expression values.

Food is a dataset with 7 nutritional information of 961 different foods. The nutritional information are: grams of fat, calories of food energy, grams of carbohydrate, grams of protein, milligrams of cholesterol, grams of saturated fat, and the weight of the food item in grams. Some foods appear in different serving sizes, for example: a piece of apple pie, or an entire apple pie. This dataset has already been used to evaluate a well-know biclustering algorithm called Plaid Models [48]. Once the ranges of the values of the attributes are very different, RIn-Close was applied in a rescaled version of this dataset. Each attribute was mapped to values that are at the range $[0, 1000]$. However, the metrics were computed based on the original matrix.

As CVC and CVR biclusters are special cases of CHV biclusters [6], we focus only on CHV biclusters in this experiment. We emphasize that the purpose of this experiment is not to perform a qualitative evaluation of RIn-Close results. For example, we will not provide biological interpretations of the results obtained for the *Yeast* dataset.

Fig. 10 shows RIn-Close results for *Yeast* dataset using $minRow = 144$ ($\approx$ 5% of *Yeast* rows) and $minCol = 2$, and Fig. 11 shows RIn-Close results for *Food* dataset using $minRow = 48$ ($\approx$ 5% of *Food* rows) and $minCol = 2$. Number of biclusters, average running time, coverage, and average MSR increased with the value of ϵ, as expected. The average MSR increases with the value of ϵ because we are relaxing the similarity constraint, but even with the largest values of ϵ, the average MSR of the resulting biclusters were very low in both cases. In [2], [51], the authors reported results using MSR = 300 as a reference to find 100 biclusters in *Yeast* dataset. Here, in our experiments, we have found more than 60,000 biclusters with average MSR $\cong$ 1. The average volume did not increase monotonically with the value of ϵ. It is possible to ensure that the volume of the largest bicluster will monotonically increase with the value of ϵ. And more than that, it is possible to ensure, for instance, that all biclusters found with $\epsilon = 3$

1. http://arep.med.harvard.edu/biclustering
2. http://www.ntwrks.com/chart1a.htm

will be found with $\epsilon = 4$, but with volumes greater than or equal to the ones achieved using $\epsilon = 3$. However, nothing can be said about the average volume since the number of biclusters is varying. Analyzing the running time of our algorithm, we realize that finding the CVC biclusters is the most expensive step of RIn-Close_CHV, as we can see clearly in Figs. 10(b) and 11(b).

Again, we observed how the parameter ϵ plays a crucial role in the results of RIn-Close. But also, this experiment gives us a valuable clue to the choice of the value of ϵ. For the *Yeast* dataset, note that the growth in the coverage is too small for $\epsilon > 5$, but the increase in the number of bicluster is very large. For the *Food* dataset, the growth in the coverage is also much lower than the growth in the number of biclusters for $\epsilon > 1.25$. This indicates that few new regions of the datasets are being explored, i.e., these values of ϵ are generating biclusters in many regions that have already been explored with lower values of ϵ, thus essentially contributing to enlarge overlap.

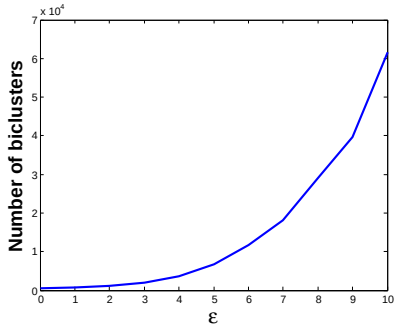
The parameter $minRow$ also has a strong influence in the computational cost of RIn-Close. The higher its value, the smaller the search space of biclusters. Table 6 illustrates this influence. It shows results using the parameter $minRow$ set to approximately 5% and 10% of *Yeast* and *Food* rows. We observe that with a stronger restriction on the minimum number of rows, the number of biclusters was considerably smaller, as well as the running time. The average volume considering a minimum support of 10% is considerably larger, since we are discarding biclusters with less than 288 and 96 rows for the *Yeast* and *Food* datasets, respectively. For the *Yeast* dataset, the coverage and the average MSR considering $minRow = 288$ was very similar to the results considering $minRow = 144$. The same is not true for the *Food* dataset, despite the average MSR remained low and coverage remained considerably high for $minRow = 96$. Anyway, the reduction in the number of bicluster and, consequently, in the computational cost was quite impressive with an increase from 5% to 10% in the minimum support of rows.

As we observed with this experiment, RIn-Close parameters can be set in a way that mitigates the explosion in the number of biclusters, with similar impact on the computational cost.

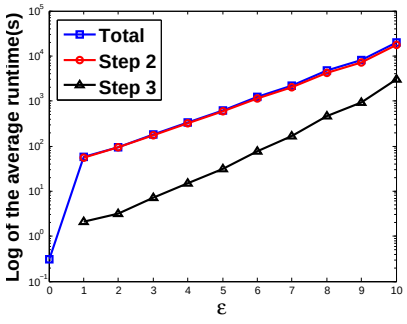
6.3 Comparison with Baselines

This experiment aims to illustrate advantages and distinct aspects of RIn-Close when compared to heuristic-based biclustering algorithms.

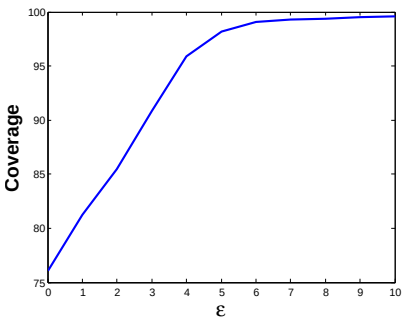
For this, we used an artificial dataset composed of 4000 objects and 30 attributes, named as *ArtData*. For its creation, initially, each entry was a random integer drawn from the discrete uniform distribution on the set $\{100, 101, ..., 199\}$. Then, 15 perfect CHV biclusters were deployed in this matrix. Some biclusters are completely disjoint from each other, and some of them overlap



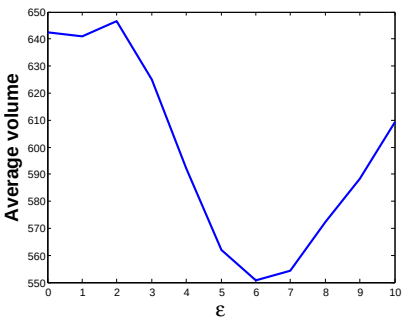
(a) Number of biclusters.



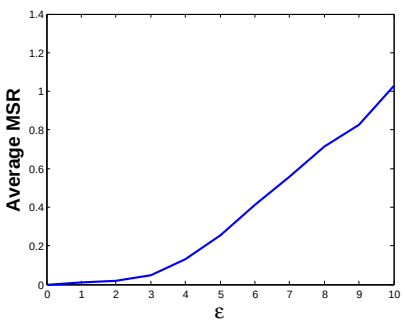
(b) Log of the average running time.



(c) Coverage.

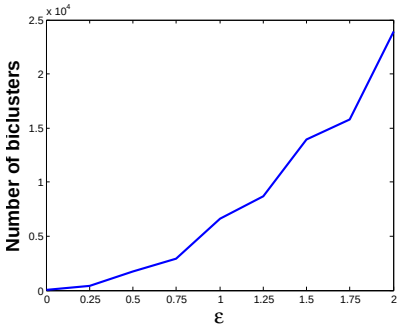


(d) Average Volume.

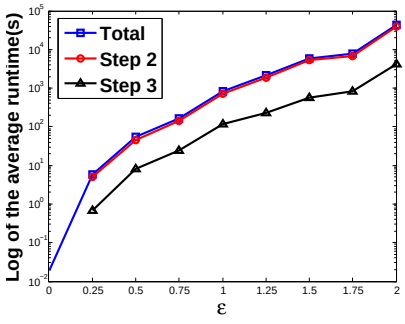


(e) Average MSR.

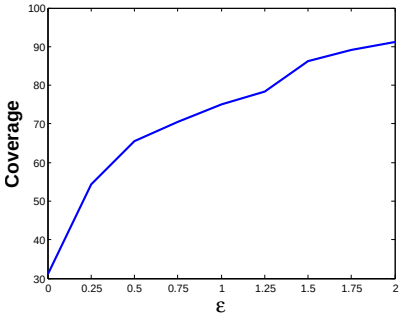
Fig. 10. Results for *Yeast* dataset using $minRow = 144$ ($\approx$ 5% of *Yeast* rows) and $minCol = 2$. All curves taken as a function of the similarity constraint ϵ. Number of biclusters, average running time, coverage, and average MSR tends to increase with the value of ϵ. The average volume varies since the algorithm can find new biclusters with higher values of ϵ. The most expensive step of RIn-Close_CHV is to find the CVC biclusters.



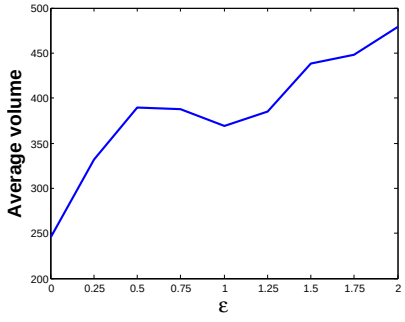
(a) Number of biclusters.



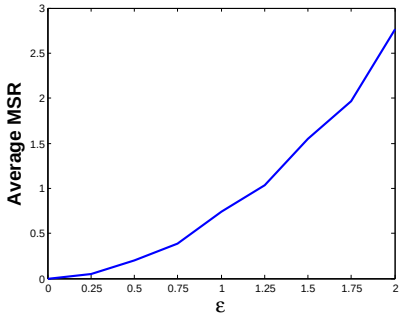
(b) Log of the average running time.



(c) Coverage.



(d) Average Volume.



(e) Average MSR.

Fig. 11. Results for *Food* dataset using $minRow = 48$ ($\approx$ 5% of *Food* rows) and $minCol = 2$. All curves taken as a function of the similarity constraint ϵ. Number of biclusters, average running time, coverage, and average MSR tends to increase with the value of ϵ. The average volume varies since the algorithm can find new biclusters with higher values of ϵ. The most expensive step of RIn-Close_CHV is to find the CVC biclusters.

TABLE 6

Results for *Yeast* and *Food* datasets varying the parameter $minRow$. The higher the value of $minRow$, the smaller the search space of biclusters, so the computational cost.

	Yeast, $\epsilon = 10$, $minCol = 2$		*Food*, $\epsilon = 2$, $minCol = 2$	
	$minRow = 144$	$minRow = 288$	$minRow = 48$	$minRow = 96$
# of bics [1]	61,544	6,200	23,906	15,035
Avg. MSR [2]	1.03(0.70)	1.02(0.73)	0.07(0.02)	2.47(0.64)
Avg. Vol. [3]	609.24(179.36)	967.32(207.77)	479.17(326.03)	631.80(321.54)
Coverage	99.63	96.34	91.27	74.42
RT [4]	23,928.65	294.04(1.84)	43,629.01	8,427.25

[1]Number of biclusters [2]Average MSR [3]Average Volume [4]Running Time (in seconds).

each other. Each bicluster has a number of rows drawn from the set $\{50, 51, ..., 70\}$, and a number of columns drawn from the set $\{5, 6, ..., 10\}$. Also, rows and columns of each bicluster are arbitrary, so that the resulting biclusters tend to be non contiguous. To create each bicluster, we fixed the values of its first column, and the other columns were obtained by adding a constant value to the first one. The constant values were draw from the set $\{-10, -9, ..., -1\}$ or $\{1, 2, ..., 10\}$ with equal probability. Finally, we added a Gaussian noise of mean 0 and variance 0.1 to all elements of the matrix and rounded them to one decimal point. The noise was small to ensure that the biclusters were preserved, and the other elements of the matrix did not contribute to the biclusters.

This dataset represents an extreme 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. Thus, it can be seen as a simpler problem than the real-world problems. Moreover, this dataset allows us to clearly determine the parameters of biclustering algorithms, and find out what they are able to mine when looking for the original biclusters.

Given an appropriate value for the similarity constraint ϵ, it would be a very simple task for RIn-Close to find all the original biclusters. In contrast, this task is not easy for the heuristics that we tested, as we will see in the following.

The algorithms that we tested were: CC [2], FLOC [51], ROCC [4], pCluster [3], and Maple [23]. CC is the most well-known algorithm to mine CHV biclusters. It mines one bicluster at each iteration, and performs random perturbations to the data to mask the discovered bicluster (avoiding that the same bicluster is found more than once). FLOC is based on the bicluster definition used by Cheng and Church [2], but performs simultaneous bicluster identification, and avoids the introduction of random interference. It generates a set of initial biclusters and then iteratively improves their quality considering the trade-off between MSR and volume.

FLOC terminates when it failed to make further improvements. Briefly, the goal of CC and FLOC is to mine a set of biclusters with high average volume given a user-defined parameter δ, which is related to the MSR of the biclusters. CC looks for biclusters with MSR less or equal to δ, and FLOC tries to mine biclusters with MSR close to δ. Meanwhile, ROCC does not work directly with the trade-off between MSR and volume. Basically, ROCC 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 error associated with them, and (*ii*) filter out the biclusters with the largest error values, and merge similar biclusters. ROCC enables scalability to large and high-dimensional datasets, and can be used to mine several types of biclusters. pCluster computes all row-maximal biclusters with two columns and all column-maximal biclusters with two objects, 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. Maple is an improved version of pCluster and it is closer to an actual enumerator algorithm.

Table 7 shows the results of this experiment.

CC was the algorithm with the best results among the heuristics. We set $\delta = 0.0111$ (equal to the largest MSR of an *ArtData*'s bicluster) and the threshold for multiple node deletion $\alpha = 1.2$ (the value suggested by the authors). Initially, we did not use restrictions on the number of rows and columns of each bicluster, but CC was finding tiny biclusters (even with only a single row or column) that were not contained in the original ones. We tried $minRow = 50$ and $minCol = 5$, but after more than 3 hours of execution, CC was unable to find even one bicluster with these restrictions. So, we decided to put a restriction on the number of rows/columns just enough to prevent CC from finding biclusters unrelated to the original ones. The values were $minRow = 4$ and $minCol = 3$. The stopping criterion was defined as follows. CC stops when it finds 15 biclusters that attend the restrictions $minRow = 4$ and $minCol = 3$, or after a maximum of 150 attempts (to avoid many executions without success). We run 20 executions of CC with this configuration and the results, in average, were as follows. CC found 46.67% of the original biclusters, and its coverage was 23% of the expected one. The biclusters found by CC had, in average, 27.34% of the original volumes.

FLOC and ROCC had a quite poor result in this experiment because none of the 15 biclusters was properly identified by them. We believed that their poor results in this case study happened mainly because they do not look for biclusters with a residue below a threshold (as CC does).

For the algorithm FLOC, we also used $\delta = 0.0111$, and 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 an *ArtData*'s bicluster and its total number of rows/columns. So, we set

these parameters to 0.013 and 0.17, respectively. As we did with CC, we used $minRow = 4$ and $minCol = 3$ to avoid tiny bicluster with low residue but not contained in the original ones. We run 20 executions of FLOC, but in none of them FLOC was able to find even one bicluster that can be interpreted as being formed by parts of a single original bicluster (notice that we have 15 original biclusters). All biclusters had a residue far greater than δ. As the initial biclusters are generated at random, it is very unlike that FLOC can improve them to the original ones, that have very low MSR.

For the algorithm ROCC, we set $sr = 680$ and $sc = 28$ because these are the number of distinct rows/columns present in the *ArtData*'s biclusters. Based on the fraction of the total number of rows/columns over the minimum number of rows/columns of an *ArtData*'s bicluster, we set $k = 800$ and $l = 6$. In 20 ROCC executions, it was unable to find even one bicluster that can be interpreted as being formed by parts of a single original bicluster. As FLOC, ROCC could not improve their initial biclusters sufficiently.

For the algorithm pCluster, we set $\epsilon = 0.9$ (the ideal value to mine all *ArtData*'s biclusters). Initially, we set minRow = 50 and minCol = 5 because these values are the minimum number of rows and columns, respectively, in a *ArtData*'s bicluster. But pCluster did not return any bicluster. As pCluster has an enumerative approach, we could not relax too much these constraints (notice that pCluster can return non-maximal biclusters). So, we set $minRow = 30$ and $minCol = 3$. With this parametrization, pCluster returned 30 biclusters (all non-maximal) that correspond to 33.33% of the original biclusters. The coverage was 21.12% of the expected one, and the biclusters had, in average, 15.95% of the original volumes.

ArtData does not have the two scenarios described in Section 2 where Maple fails. So, it was able to find the perfect result, i.e., 100% of the original biclusters with their original volume and coverage. However, it took more than 8 hours to accomplish this task while RIn-Close took less than 5 minutes.

As we have seen, although we knew how to choose good parameters for the heuristic-based algorithms, this case study was very challenging to them. On the other hand, RIn-Close is able to easily accomplish this task, finding 100% of the original biclusters with their original volume and coverage.

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 FCA-based algorithms able to enumerate all maximal CTV biclusters of ones from a binary data matrix. These enumeration

TABLE 7

Results of the comparison with baselines. The heuristics had a poor result in finding the original biclusters. RIn-Close and Maple were the only ones able to accomplish this task perfectly.

	RIn-Close/Maple	pCluster	CC	FLOC/ROCC
%bics[1]	100.00	33.33	46.67	0.00
%cov[2]	100.00	21.12	23.00	0.00
%vol[3]	100.00	15.95	27.34	0.00

[1]% of biclusters correctly recovered [2]% of the original coverage [3]% of the original volume.

algorithms proved to be very useful and have been applied in various application domains. Nonetheless, the raw data matrix admits integer and/or real values in several application domains, and to transform it into binary data leads to loss of information. Hence, there are some FCA-based proposals capable of dealing directly with numerical (integer or real-valued) data matrices. Among these proposals, there are three recent algorithms able to enumerate all maximal CTV biclusters in a numerical data matrix [17], [18], [19].

In this paper, we expanded the possibilities of enumerating maximal biclusters in the case of integer or real-valued data matrices (binary data matrices are obviously included). We proposed a family of algorithms, called RIn-Close, able to perform a complete, correct and non-redundant enumeration of all maximal CVC, CVR and CHV biclusters. RIn-Close algorithms are adaptations of the FCA algorithm called In-Close2 [13]. As far as we know, our algorithms are the first ones able to perform these enumerations.

Due to the inherent computational cost of the enumeration algorithms, the usage of RIn-Close is more suitable for small and medium size datasets. RIn-Close can be impractical for large data matrices, since the larger the dataset, the greater tends to be the number of biclusters. However, our experiments showed that an appropriate choice of the RIn-Close parameters (ϵ and minimum number of rows and columns) can dramatically reduce the number of biclusters, together with the RIn-Close computational cost. Thus, it is possible to consider the usage of RIn-Close on several scenarios in which an enumeration algorithm is recommended.

In future works, we intend to investigate how to handle data matrices with missing values, and how to enumerate biclusters with coherent evolutions. We intend to investigate the extension of RIn-Close to enumerate only the top k biclusters in terms of volume, thus reducing RIn-Close computational cost. Still looking at the reduction of their computational cost, we will also implement parallelized versions of RIn-Close algorithms.

ACKNOWLEDGMENTS

R. Veroneze and F. J. Von Zuben would like to thank CAPES and CNPq for the financial support. A. B. acknowledges the support of NSF via IIS-0953274, IIS-1029711, IIS- 0916750, IIS-0812183, NASA via NASA grant NNX-12AQ39A, and the technical support from the Minnesota Supercomputing Institute (MSI).

REFERENCES

[1] A. K. Jain and R. C. Dubes, *Algorithms for clustering data*. Prentice-Hall, Inc., 1988.

[2] Y. Cheng and G. Church, "Biclustering of expression data," in *Proceedings of the eighth international conference on intelligent systems for molecular biology*, vol. 1, 2000, pp. 93–103.

[3] 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*. ACM, 2002, pp. 394–405.

[4] 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*. ACM, 2009, pp. 241–248.

[5] J. Guan, Y. Gan, and H. Wang, "Discovering pattern-based subspace clusters by pattern tree," *Knowledge-Based Systems*, vol. 22, no. 8, pp. 569–579, 2009.

[6] S. Madeira and A. Oliveira, "Biclustering algorithms for biological data analysis: a survey," *Computational Biology and Bioinformatics, IEEE/ACM Transactions on*, vol. 1, no. 1, pp. 24–45, 2004.

[7] S. Busygin, O. Prokopyev, and P. M. Pardalos, "Biclustering in data mining," *Computers & Operations Research*, vol. 35, no. 9, pp. 2964–2987, 2008.

[8] K. Makino and T. Uno, "New algorithms for enumerating all maximal cliques," in *Algorithm Theory-SWAT 2004*. Springer, 2004, pp. 260–272.

[9] D. Eppstein, M. Löffler, and D. Strash, "Listing all maximal cliques in sparse graphs in near-optimal time," in *Algorithms and Computation*. Springer, 2010.

[10] B. Ganter, "Two basic algorithms in concept analysis," *Technical Report FB4-Preprint No. 831,*, 1984.

[11] S. Kuznetsov, "Learning of simple conceptual graphs from positive and negative examples," in *Principles of Data Mining and Knowledge Discovery*. Springer, 1999, pp. 384–391.

[12] S. Andrews, "In-close, a fast algorithm for computing formal concepts," *ICCS 2009*, 2009.

[13] ——, "In-close2, a high performance formal concept miner." *ICCS 2011*, 2011.

[14] 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 (CLA)*, vol. 672, 2010, pp. 325–337.

[15] S. O. Kuznetsov and S. A. Obiedkov, "Comparing performance of algorithms for generating concept lattices," *Journal of Experimental & Theoretical Artificial Intelligence*, vol. 14, no. 2-3, pp. 189–216, 2002.

[16] J. Poelmans, D. I. Ignatov, S. O. Kuznetsov, and G. Dedene, "Formal concept analysis in knowledge processing: A survey on applications," *Expert Systems with Applications*, vol. 40, no. 16, pp. 6538–6560, 2013.

[17] J. Besson, C. Robardet, L. De Raedt, and J.-F. Boulicaut, "Mining bi-sets in numerical data," in *Knowledge Discovery in Inductive Databases*. Springer, 2007, pp. 11–23.

[18] M. Kaytoue, S. O. Kuznetsov, and A. Napoli, "Biclustering numerical data in formal concept analysis," in *Formal Concept Analysis*. Springer, 2011, pp. 135–150.

[19] M. Kaytoue, S. O. Kuznetsov, J. Macko, and A. Napoli, "Biclustering meets triadic concept analysis," *Annals of Mathematics and Artificial Intelligence*, pp. 1–25, 2013.

[20] B. Ganter, R. Wille, and C. Franzke, *Formal concept analysis: mathematical foundations*. Springer-Verlag New York, Inc., 1997.

[21] B. Ganter and S. O. Kuznetsov, "Pattern structures and their projections," in *Conceptual Structures: Broadening the Base*. Springer, 2001, pp. 129–142.

[22] F. Lehmann and R. Wille, *A triadic approach to formal concept analysis*. Springer, 1995.

[23] J. Pei, X. Zhang, M. Cho, H. Wang, and P. S. Yu, "Maple: A fast algorithm for maximal pattern-based clustering," in *Data Mining, 2003. ICDM 2003. Third IEEE International Conference on*. IEEE, 2003, pp. 259–266.

[24] H. Wang, F. Chu, W. Fan, P. S. Yu, and J. Pei, "A fast algorithm for subspace clustering by pattern similarity," in *Scientific and Statistical Database Management, 2004. Proceedings. 16th International Conference on*. IEEE, 2004, pp. 51–60.

[25] B. Mirkin, *Mathematical Classification and Clustering*. Springer, 1996.

[26] J. Hartigan, "Direct clustering of a data matrix," *Journal of the American Statistical Association*, pp. 123–129, 1972.

[27] A. Banerjee, I. Dhillon, J. Ghosh, S. Merugu, and D. S. Modha, "A Generalized Maximum Entropy Approach to Bregman Co-clustering and Matrix Approximation," *J. Mach. Learn. Res.*, vol. 8, pp. 1919–1986, 2007.

[28] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan, "Automatic subspace clustering of high dimensional data for data mining applications," *SIGMOD Rec.*, vol. 27, pp. 94–105, Jun. 1998.

[29] 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*, ser. KDD '01. New York, NY, USA: ACM, 2001, pp. 269–274.

[30] P. de Castro, F. de França, H. Ferreira, G. Coelho, and F. Von Zuben, "Query expansion using an immune-inspired biclustering algorithm," *Natural Computing*, vol. 9, no. 3, pp. 579–602, 2010.

[31] P. Symeonidis, A. Nanopoulos, A. Papadopoulos, and Y. Manolopoulos, "Nearest-biclusters collaborative filtering with constant values," *Advances in web mining and web usage analysis*, pp. 36–55, 2007.

[32] P. de Castro, F. de França, H. Ferreira, and F. Von Zuben, "Applying biclustering to perform collaborative filtering," in *Intelligent Systems Design and Applications, 2007. ISDA 2007. Seventh International Conference on*. IEEE, 2007, pp. 421–426.

[33] A. Colantonio, R. Di Pietro, A. Ocello, and N. Verde, "Abba: Adaptive bicluster-based approach to impute missing values in binary matrices," in *Proceedings of the 2010 ACM Symposium on Applied Computing*. ACM, 2010, pp. 1026–1033.

[34] R. Veroneze, F. de Franca, and F. Von Zuben, "Assessing the performance of a swarm-based biclustering technique for data imputation," in *Evolutionary Computation (CEC), 2011 IEEE Congress on*. IEEE, 2011, pp. 386–393.

[35] L. Lakhal and G. Stumme, "Efficient mining of association rules based on formal concept analysis," in *Formal concept analysis: Foundations and Applications*. Springer, 2005, pp. 180–195.

[36] R. Agrawal, T. Imieliński, and A. Swami, "Mining association rules between sets of items in large databases." in *ACM SIGMOD Record*, vol. 22, no. 2. ACM, 1993, pp. 207–216.

[37] 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 (KDD)*, Aug 1997.

[38] K. Gouda and M. J. Zaki, "Genmax: An efficient algorithm for mining maximal frequent itemsets," *Data Mining and Knowledge Discovery: An International Journal*, vol. 11, no. 3, pp. 223–242, Nov 2005.

[39] M. J. Zaki and C.-J. Hsiao, "Charm: An efficient algorithm for closed itemset mining." in *SDM*, vol. 2, 2002, pp. 457–473.

[40] J. Abello, A. J. Pogel, and L. Miller, "Breadth first search graph partitions and concept lattices." *Journal of Universal Computer Science*, vol. 10, no. 8, pp. 934–954, 2004.

[41] 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*. Springer, 2010, pp. 270–280.

[42] A. Gély, L. Nourine, and B. Sadi, "Enumeration aspects of maximal cliques and bicliques," *Discrete Applied Mathematics*, vol. 157, no. 7, pp. 1447–1459, 2009.

[43] 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," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 19, no. 12, pp. 1625–1637, 2007.

[44] T. Uno, M. Kiyomi, and H. Arimura, "Lcm ver. 2: Efficient mining algorithms for frequent/closed/maximal itemsets," in *FIMI*, vol. 19, 2004, p. 30.

[45] 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*, vol. 145, no. 1, pp. 11–21, 2004.

[46] D. Eppstein, "Arboricity and bipartite subgraph listing algorithms," *Information Processing Letters*, vol. 51, no. 4, pp. 207–211, 1994.

[47] S. Kuznetsov, "Mathematical aspects of concept analysis," *Journal of Mathematical Sciences*, vol. 80, no. 2, pp. 1654–1698, 1996.

[48] L. Lazzeroni and A. Owen, "Plaid models for gene expression data," *Statistica Sinica*, vol. 12, no. 1, pp. 61–86, 2002.

[49] R. J. Cho, M. J. Campbell, E. A. Winzeler, L. Steinmetz, A. Conway, L. Wodicka, T. G. Wolfsberg, A. E. Gabrielian, D. Landsman, D. J. Lockhart *et al.*, "A genome-wide transcriptional analysis of the mitotic cell cycle," *Molecular cell*, vol. 2, no. 1, pp. 65–74, 1998.

[50] J. Yang, W. Wang, H. Wang, and P. Yu, "δ-clusters: Capturing subspace correlation in a large data set," in *Data Engineering, 2002. Proceedings. 18th International Conference on*. IEEE, 2002, pp. 517–528.

[51] J. Yang, H. Wang, W. Wang, and P. Yu, "Enhanced biclustering on expression data," in *Bioinformatics and Bioengineering, 2003. Proceedings. Third IEEE Symposium on*. IEEE, 2003, pp. 321–327.