BICLUSTERING BASED ON ASSOCIATION ANALYSIS

BASILIS BOUTSINAS

Dept. of Business Administration, MIS & Business Intelligence Lab,
University of Patras, Rio, 26500, Greece
vutsinas@upatras.gr

Abstract
Clustering has been applied in a wide variety of disciplines and has also been utilized in many scientific areas. Usually, clustering algorithms construct either clusters of rows or clusters of columns of the input data matrix. Biclustering is a methodology where biclusters are formed by both a subset of rows and a subset of columns, such that objects represented by the first are the most similar to each other when compared over the latter. In this paper, we introduce a new biclustering technique, based on association rule mining, which can support different well-known biclustering models proposed in the literature. Experimental tests demonstrate the accuracy and efficiency of the proposed technique with respect to well known related ones.

***Keywords*:** Biclustering; clustering; association rule mining; data mining; high dimensionality.

1. Introduction

Clustering consists in partitioning a set of objects into clusters with respect to a set of features (attributes) that these objects possess. Objects within a cluster are more similar to each other, when compared over this set of features, with respect to objects belonging to other clusters. Clustering algorithms e.g.[7,12,13,16] (see [29] for a recent survey on clustering methods) construct either clusters of rows or clusters of columns of an input data matrix *D(R,C)*. Thus, they are of limited use for the analysis of large data sets containing several hundred thousands of rows and tens of columns with large domains. This is because rows are usually affected by a small subset of columns, such that several columns do not contribute in extracting knowledge but rather they increase noise. Intuitively, in such high dimensional spaces the average density of objects is quite low.

In recent years, biclustering is introduced in order to tackle the problem. Biclustering is a methodology where rows and columns are clustered simultaneously. A bicluster is defined as a submatrix spanned by both a subset of rows and a subset of columns, such that objects represented by the first are the most similar to each other when compared over the latter.

Since the work in [10], where the term ‘’bicluster” has been used for simultaneously clustering of gene expression data in DNA microarray analysis, there are several biclustering models and different algorithms for each model proposed in the literature with respect to: the type of biclusters they can find, the way multiple biclusters are treated and the bicluster structure produced, the type of the specific algorithm used to identify each bicluster, etc. (see [9,20,23,28] for surveys).

In this work, we introduce and evaluate a new biclustering technique, based on association rule mining, which can support different well-known biclustering models proposed in the literature.

2. A New Biclustering Algorithm

There are biclustering algorithms presented in the literature which, trying to exhaustively enumerate all candidate biclusters, are based on the key idea of association rule mining, as the proposed algorithm. These algorithms could be classified i) to those[19,22] that are based on a transformation of the input matrix that imposes the application of a dedicated variation of an association rule mining algorithm and ii) to those that properly transform the input matrix in order to directly apply association rule mining algorithms[2], as the proposed algorithm does.

The proposed biclustering algorithm adopts the approach of an exhaustive enumeration of possible biclusters in order to guarantee the detection of the best biclusters. To reduce the search space, it is based on a theory inspired by that of Apriori algorithm of Association Rule Mining for finding frequent itemsets.

At first, we will present the proposed algorithm by considering its application to a Boolean input data matrix and then (see subsection 3.1) to other well-known biclustering models. In a Boolean input data matrix $D(R,C)$, $R=\{r_1,r_2,...,r_o\}$, $C=\{c_1,c_2,...,c_a\}$, cell values $v_{ij}=1$ ($v_{ij}=0$) denote that the object represented by row r_i, $1\leq i\leq o$, satisfies (does not satisfy) the attribute represented by column c_j, $1\leq j\leq a$. The proposed algorithm extracts biclusters $S(R_s,C_s)$ where $R_s\subseteq R$, $C_s\subseteq C$ and $\forall i\in R_s\, j\in C_s\, v_{ij}=1$.

Table 1. An example Boolean input data matrix

	1st	2nd	3rd	4th	5th
A	1	1	1	0	1
B	0	0	1	1	0
C	0	1	0	0	1
D	1	1	0	0	1
E	1	0	1	0	0

A	1	2	3	5
B	3	4		
C	2	5		
D	1	2	5	
E	1	3		

Thus, the Boolean input data matrix shown in the left of Table 1 represents that, for instance, object A satisfies 1st, 2nd, 3rd and 5th attribute. To apply the proposed algorithm, the Boolean input data matrix is transformed to the matrix $D'(R,C')$, $C'=\{c'_1,c'_2,...,c'_b\}$, shown in the right of Table 1: each object is assigned the column indexes representing the attributes it satisfies, i.e., $D'(r_i,c'_j)=\{c''_j \mid D(r_i,c''_j)=1\}$, (or alternatively, depending on the application, the column indexes representing the attributes it does not satisfy). The key idea of the proposed algorithm is to process the latter matrix by an Association Rule Mining algorithm, as the Apriori algorithm.

It is obvious that the transformed input data matrix shown in the right of Table 1 can be processed as if it were a representation of transactions consisting of itemsets; rows represent transactions and columns itemsets. The application of an Association Rule Mining algorithm, as the Apriori algorithm, results in extracting a set of frequent itemsets (which are then used to extract association rules) of various length. Let $fr_{(s)}$ denote a frequent itemset, where $fr \subseteq I$ and s is its support. For example, $2/5_{(60\%)}$ is a frequent itemset obtained after the application of Apriori algorithm to the matrix shown in the right of Table 1, setting $ms=50\%$ as minimum support. The transactions containing $2/5_{(60\%)}$ are A,C and D. Thus, it is obvious that rows A,C and D along with the 2nd and 5th columns of the initial input data matrix form a bicluster, i.e. more than or equal to 50% (namely 60%) of the rows (namely A,C and D) are the most similar to each other (cell values equal to 1) when compared over the 2nd and 5th columns. Therefore, the detection of biclusters can be directly achieved by detecting frequent itemsets. Actually, every frequent itemset $fr_{(s)}$ defines a bicluster including a fraction s of the rows and $|fr|$ columns. The extracted biclusters are determined by the user defined minimum support. Setting $ms=0$ results in even one-cell biclusters. Thus, the minimum support determines the minimum number of rows a bicluster can have. Given a certain minimum support ms, the extracted frequent itemsets (from an input matrix with r rows) are included in at least $r*ms$ rows. The following Lemma 1 proves that the proposed algorithm extracts all biclusters having at least $r*ms$ rows

Lemma 1. The proposed technique is sound and complete with respect to a given minimum support ms.
Proof. To prove soundness, suppose that $c''_1, c''_2,...,c''_{q\ (s\%)}$ is an extracted frequent itemset. Then $\exists\ r_1,r_2,...,r_t\in R$, $t=|R|*s/100$, $s\geq ms$, where $D(r_i,c''_j)=1$, $1\leq i\leq t$, $1\leq j\leq q$. Clearly, the latter cells form a bicluster. To prove completeness, suppose that cells $D(r_i,c''_j)=1$, $1\leq i\leq t$, $1\leq j\leq q$ form a bicluster. Then, if $t=|R|*s/100$ and $s\geq ms$, $c''_1,c''_2,...,c''_q$ is a frequent itemset since it is included in more than ms rows. □

As shown above, Apriori-based algorithms adopt an exhaustive enumeration of possible frequent itemsets. To reduce the search space, they are based on the theory that any subset of a frequent itemset is also frequent. Therefore, the candidate frequent itemsets having k items can be generated by joining frequent itemsets having k-1 items, and deleting those that contain any subset that is not frequent. Thus, although the detection of frequent itemsets is exponential in the highest dimensionality as shown in [1], the candidate generation procedure produces the minimal number of candidates that can guarantee that all frequent itemsets will be found. More specifically, the time complexity of detecting the frequent itemsets from an input data matrix $D(R,C)$ is $O(|R|*|C|*|C_1\cup C_2\cup...\cup C_k|)$, where C_i is the set of candidate frequent i-itemsets generated by apriori-gen() and (k-1) is the maximum length of detected frequent itemsets.

The time complexity of the proposed algorithm is $O(|R|*|C|*|C_1 \cup C_2 \cup ... \cup C_k|)$, since it is dominated by the time complexity of transforming the input data matrix as shown in Table 1, which is $O(|R|*|C|)$, and the time complexity of detecting the frequent itemsets and hence the biclusters, which is $O(|R|*|C|*|C_1 \cup C_2 \cup ... \cup C_k|)$, where $C_1 ... C_k$ are the examined sets of candidate frequent itemsets. The latter is true if candidates are generated by the technique adopted in [1]. However, more efficient techniques could be used instead[8,14,15].

Note that the complexity of the proposed algorithm is comparable to that of fast similar biclustering algorithms. For instance, the time complexity of the proposed algorithm is similar to that in [2], which is also a sound and complete algorithm, since they are both dominated by the detection of frequent itemsets. However, the proposed algorithm is faster since the set of items in the proposed algorithm is $|C|$ while in [2] it is $\xi*|C|$, since every dimension is partitioned into $\xi>10$ intervals. Note that the performance of set of items is heavily relied on the number of items. Moreover, in [2] there is a need for extra processing in order to define the biclusters, after the detection of the frequent itemsets. This processing is actually an instance of the problem of finding connected components in a graph bounded by $O(2*k*n)$, where n is a fraction of the set of items ($\xi*|C|$) and k is the maximum number of dimensions of biclusters. Note also that the algorithm in [3] can only extract biclusters with the cell values in the columns being in the same range of values and/or constant.

In general, the main factor of time complexity in extracting frequent itemsets is the size of C_1 and C_2 sets of candidate frequent itemsets. The latter is heavily relied on the number of items, on support threshold, on the length of transactions, as well as on the structure of input data[30]. For instance, in a real life data set with about 500K rows, the number of frequent itemsets is reduced from 31*106 to 103 and the length of transactions is reduced from 14 to 2 if minimum support threshold is reduced from 0.01% to 1%[30].

The way the Boolean input data matrix is transformed by the proposed methodology guarantees that if the Boolean input data matrix is sparse (a small number of 1s), the size of C_1 and therefore that of C_2 is reduced while the length of transactions (entries in each row) is considerably smaller than the number of columns in the initial data matrix. Moreover, in the problem at hand, the minimum support threshold, i.e. the minimum number of rows of a bicluster, is certainly greater than 2.

2.1 Handling other biclustering models

Apart from handling Boolean input data matrix, the proposed algorithm can also support different other well-known biclustering models proposed in the literature. Such models are mentioned above and concern biclusters with constant cell values in the rows and/or columns, with coherence, order preserving, co-regulated or extensions/variations of them. The proposed algorithm can support different biclustering models if the theory behind reducing the search space it uses is also valid to these models.

Formally, for a bicluster B(Rb,Cb), Rb={r1,…,rp}, Cb={c1,…,cq} with:

1) constant cell values (or within a range) in the columns, $\forall i \in R_b\ j \in C_b$, $B(i,j)=a_j$ (or $B(i,j) \in [a_{1j},a_{2j}]$)

2) constant cell values (or within a range) in the rows, $\forall i \in R_b\ j \in C_b$, $B(i,j)=a_i$ (or $B(i,j) \in [a_{1i},a_{2i}]$)

3) coherence, $\forall i \in R_b\ j \in C_b$, $B(i,j)=\mu+\alpha_i+\beta_j$, where μ is the typical value within the bicluster, $\alpha_i \in \alpha=\{\alpha_1,...,\alpha_p\}$ is the adjustment for row i and $\beta_j \in \beta=\{\beta_1,...,\beta_q\}$ is the adjustment for column j

4) order preserving, $\forall i \in R_b\ j \in \{c_1,\ldots,c_{q-1}\}$, $B(i,j)<B(i,j+1)$

5) co-regulation, $\forall i \in R_b$ $sign(B(i,c_1)-B(i,c_2))=...=sign(B(i,c_{q-1})-B(i,c_q))$.

Lemma 2 theoretically proves that a whole variety of different biclustering models can be addressed by the proposed algorithm after modifying the transformation of the input matrix or the association analysis.

Lemma 2. Every submatrix of a bicluster $B(R_b,C_b)$, $R_b=\{r_1,...,r_p\}$, $C_b=\{c_1,...,c_q\}$, with (1) constant cell values in the columns, (2) and/or rows, (3) coherence, (4) order preserving or (5) co-regulation is also a bicluster.

Proof. Suppose that $S(R_s,C_s)$ is a submatrix of $B(R_b,C_b)$, where $R_s \subseteq R_b$, $C_s \subseteq C_b$. To prove (1) for constant cell values in the columns note that since cells $B(r_x,c_y)=a_y$ (or $B(r_x,c_y) \in [a_{1y},a_{2y}]$) $\forall\ r_x,c_y \quad r_x \in R_b\ c_y \in C_b$, then cells $S(r_i,c_j)=a_j$ (or $S(r_i,c_j) \in [a_{1j},a_{2j}]$) $\forall r_i,c_j \quad r_i \in R_s \subseteq R_b,\ c_j \in C_s \subseteq C_b$. To prove (2) for constant cell values in the rows note that since cells $B(r_x,c_y)=a_x$ (or $B(r_x,c_y) \in [a_{1x},a_{2x}]$) $\forall\ r_x,c_y\ r_x \in Rb\ c_y \in Cb$, then cells $S(r_i,c_j)=a_i$ (or $S(r_i,c_j) \in [a_{1i},a_{2i}]$) $\forall r_i,c_j \quad r_i \in R_s \subseteq R_b,\ c_j \in C_s \subseteq C_b$. In the specific case where $a_y=a$, $\forall y\ 1 \le y \le q$, (or $a_x=a$, $\forall x\ 1 \le x \le p$), the proof concerns biclusters with constant cell values in both columns and rows. To prove (3) note that cells

cells $S(i,j)=\mu+\alpha_i+\beta_j \ \forall i,j \ i\in R_s\subseteq R_b, j\in C_s\subseteq C_b, \alpha_i\in\alpha, \beta_j\in\beta$. To prove (4) note that $S(i,j)<S(i,j+1), \forall i,j \ i\in R_s\subseteq R_b, j\in\{j_1,...j_{u-1}\} \wedge C_s=\{j_1,...j_u\}\subseteq C_b$. To prove (5) note that $sign(S(i,j_x)-S(i,j_{x+1}))=...=sign(S(i,j_y)-S(i,j_{y+1})), \forall i,j_x,j_y \ i\in R_s\subseteq R_b \wedge i\equiv r$, where $j_x,j_{x+1}, j_{x+2},j_y,j_{y+1}\in C_s\subseteq C_b \wedge j_x<j_y \wedge j_x\equiv c_w \Rightarrow j_{x+1}\equiv c_{w+1} \wedge j_y\equiv c_z \Rightarrow j_{y+1}\equiv c_{z+1}$, where $c_w,c_{w+1},c_z,c_{z+1}\in Cb$. Suppose that $\neg j_{x+1}\equiv c_{w+1}$ and let also $c_{w+2},...,c_{w+v}\notin C_s$, while $j_{x+1}\equiv c_{w+v+1}\in Cs$. However, $sign(B(r,c_w)-B(r,c_{w+1}))=...=sign(B(r,c_{w+v})-B(r,c_{w+v+1}))=sign(B(r,c_w))-B(r,c_{w+v+1})) \Rightarrow sign(S(i,j_x)-B(r,c_{w+1}))=...=sign(B(r,c_{w+v})-S(i,j_{x+1}))=sign(S(i,j_x))-S(i,j_{x+1})) \Rightarrow sign(B(r,c_1)-B(r,c_2))=...=sign(B(r,c_{q-1})-B(r,c_q))=sign(S(i,j_1)-S(i,j_2))=...=sign(S(i,j_{u-1})-S(i,j_u))$. □

One straightforward technique that we propose is to properly transform the input data matrix in order to be able to check the constraints of each model during detecting frequent itemsets. For example, detecting biclusters with constant cell values in the columns and/or the rows, the input data matrix could be transformed as in Table 2, where the item in each row is a concatenation of corresponding column index and associated cell value in the input data matrix. Then, 1-4/3-4/5-$4_{(60\%)}$ is an example of a detected bicluster, representing the submatrix consisting of 1st, 3rd and 5th columns along with A, C and E rows, having its cell values equal to 4.

Table 2. An example input data matrix containing biclusters with constant cell values

	1st	2nd	3rd	4th	5th
A	4	2	4	5	4
B	1	2	3	4	3
C	4	5	4	5	4
D	3	5	3	5	6
E	4	2	4	3	4

A	1-4	2-2	3-4	4-5	5-4
B	1-1	2-2	3-3	4-4	5-3
C	1-4	2-5	3-4	4-5	5-4
D	1-3	2-5	3-3	4-5	5-6
E	1-4	2-2	3-4	4-3	5-4

Formally, the Boolean input data matrix $D(R,C)$, $R=\{r_1,r_2,...,r_o\}$, $C=\{c_1,c_2,...,c_a\}$ is transformed to the matrix $D'(R,C)$; each object is assigned the column indexes representing the attributes it satisfies along with corresponding cell values, i.e., $D'(r_i,c_j)=concatenate(c_j,"-",D(r_i,c_j))$. It is obvious that under such a transformation, the proposed algorithm can extract biclusters with constant cell values in the columns and/or rows, either numeric or categorical ones.

Lemma 3. The proposed technique for constant cell values in the columns is sound and complete with respect to a given minimum support *ms*.

Proof. To prove soundness, suppose that c''_1-v_1, c''_2-v_2, …, c''_q-$v_{q\ (s\%)}$ is an extracted frequent itemset. Then $\exists$ $r_1,r_2,...,r_t\in R$, $t=|R|*s/100$, $s\geq ms$, where $D(r_i,c''_j)=v_j$, $1\leq i\leq t$, $1\leq j\leq q$. Clearly, the latter cells form a bicluster having cell values of column c''_j equal to v_j. To prove completeness, suppose that cells $D(r_i,c''_j)=v_j$, $1\leq i\leq t$, $1\leq j\leq q$ form a bicluster. Then, if $t=|R|*s/100$ and $s\geq ms$, c''_1-v_1,c''_2-v_2,...,c''_q-v_q is a frequent itemset since it is contained in more than *ms* rows. Also, note that the case of biclusters with constant cell values in both columns and rows is contained in the above case by considering $v_1=v_2=...=v_q$ while the case of biclusters with constant cell values in rows is as the above case after rows swap places with columns. □

Table 3. Example input data matrix containing co-regulated biclusters

	1st	2nd	3rd	4th	5th
A	29	30	33	45	67
B	44	45	38	28	7
C	2	1	11	23	31
D	21	22	16	13	1
E	21	17	29	39	41

A	1,29	2,30	3,33	4,45	5,67
B	1,44	2,45	3,38	4,28	5,7
C	1,2	2,1	3,11	4,23	5,31
D	1,21	2,22	3,16	4,13	5,1
E	1,21	2,17	3,29	4,39	5,41

Another straightforward technique that we propose is to check the constraints of the model during calculating the support of frequent itemsets. As an example, consider the model of biclusters with co-regulated cell values on the columns for the input data matrix shown on Table 3, which is transformed to the right matrix of Table 3. The transformation is the same to that of Boolean input data matrix shown in Table 1; the item c_i in every cell is the index of its corresponding column. However, in each cell of the new matrix we assign the corresponding cell value of the input data matrix, denoted as $val(c_i)$. During calculating the support of every candidate itemset $c=c_{i1},c_{i2},...,c_{ik}$, every row is examined if it contains c, as shown in the following part of Apriori algorithm:

Forall transactions t ∈ D **do begin**

C_t=subset(C_k,t); % Candidates contained in *t*

Forall candidates $c=c_{i1},c_{i2},\ldots,c_{ik-1},c_{ik} \in C_t$ **do**

c.count++;

end

Then, in parallel, it is checked if *c* is up-regulated or down-regulated. Thus, the above part of the algorithm has to change to:

Forall transactions t ∈ D **do begin**

C_t=subset(C_k,t); % Candidates contained in *t*

Forall candidates $c=c_{i1},c_{i2},\ldots,c_{ik-1},c_{ik} \in C_t$ **and**

$sign(val(c_{i1})-val(c_{i2}))=\ldots=sign(val(c_{ik}-1)-val(c_{ik}))$ **do**

c.count++;

end

For instance, during calculating the support of the c=1/3/4/5 candidate itemset, it is checked that it is up-regulated since sign(val(1)-val(3))=sign(val(3)-val(4))=sign(val(4)-val(5)) for A, C and E rows (e.g for row A: sign(29-33)=sign(33-45)=sign(45-67)='minus'). Thus, c=1/3/4/5(60%) is an up-regulated frequent itemset representing the up-regulated submatrix consisting of 1st, 3rd, 4th and 5th columns along with A, C and E rows.

Lemma 4. The proposed technique for co-regulated cell values is sound and complete with respect to a given minimum support *ms*.

Proof. To prove soundness, suppose that $c''_1,c''_2,...,c''_{q-1},c''_{q\ (s\%)}$ is an extracted frequent itemset. Then $\exists$ $r_1,r_2,...,r_t \in R$, $t=|R|*s/100$, $s \geq ms$, where $\forall i, 1 \leq i \leq t$, $sign(D(r_i,c''_1)-D(r_i,c''_2))=...=sign(D(r_i,c''_{q-1})-D(r_i,c''_q))$. Clearly, the latter cells form a bicluster with co-regulated cell values. To prove completeness, suppose that the *t* cells, where $\forall i, 1 \leq i \leq t$, $sign(D(r_i,c''_1)- D(r_i,c''_2))=...=sign(D(r_i,c''_{q-1})-D(r_i,c''_q))$ form a bicluster. Then, if $t=|R|*s/100$ and $s \geq ms$, $c''_1,c''_2,...,c''_{q-1},c''_q$ is a frequent itemset since it is contained in more than *ms* rows.□

Also, another example concerns the detection of order preserving biclusters, in which cell values of all rows induce the same linear ordering of the columns[4]. In this case, the input data matrix could be also transformed as in Table 3 above. Then, the support of every candidate itemset $c=c_{i1}/c_{i2}/.../c_{ik}$, is calculated grouped by its different permutations. Then, if the most common permutation of itemset c appears in more that ms rows, it is considered as a frequent itemset. Of course, this permutation must be preserved during checking larger candidate itemsets.

Lemma 5. The proposed technique for order preserving cell values is sound and complete with respect to a given minimum support ms.

Proof. To prove soundness, suppose that $c_1,...,c_{q(s\%)}$ is an extracted frequent itemset under the permutation $rank(c_1)$-$rank(c_2)$-...-$rank(c_q)$. Then $\exists r_1,r_2,...,r_t \in R$, $t=|R|*s/100$, $s \geq ms$, where $D(r_i,c_1)$-$D(r_i,c_2)$-...-$D(r_i,c_q)$, $1 \leq i \leq t$, much the permutation $rank(c_1)$-$rank(c_2)$-...-$rank(c_q)$. Clearly, the latter cells form a bicluster where cell values of all of its rows induce the same linear ordering of its columns. To prove completeness, suppose that cells $D(r_i,c_j)$, $1 \leq i \leq t$, $1 \leq j \leq q$ form a bicluster. If $t=|R|*s/100$ and $s \geq ms$, then $c_1,...,c_{q(s\%)}$ is a frequent itemset since it is contained in more than ms rows, under the permutation $rank(D(r_i,c_1))$-$rank(D(r_i,c_2))$-...-$rank(D(r_i,c_q))$. □

All the above proposed extensions increase the time complexity. This is not due to the preprocessing phase of input data matrix transformation. The latter is $O(|R|*|C|)$ for the transformation in Table 2. Clearly, the complexity could be worse if extra processing were needed for each row or column. However, the time complexity of the preprocessing phase is bounded by the time complexity of detecting the biclusters. In the case of handling biclusters with constant cell values in rows and/or columns, there is an increase in time complexity due to the increase in the length of the set of items. The set of items after transforming a Boolean data matrix (see Table 1) is the set of seven different column indexes {1st,2nd,3rd,4th,5th}. However, the set of items in Table 2 is the set of 13 different combinations of indexes and cell values. Since the time complexity of the Apriori algorithm relies heavily on the number of items, the overall complexity of the extension will be increased. More specifically, the time complexity increased to $O(|R|*|C|*|V|*|C_1 \cup C_2 \cup ... \cup C_k|)$, where $|V|$ is the number of different cell values.

Also, in the case of handling co-regulated and order preserving biclusters, there is an increase in time complexity due to the extra processing in each row for detecting ordering (different permutations) and up/down-regulation ($sign(val(c_{i1})-val(c_{i2}))=...=sign(val(c_{ik-1})-val(c_{ik})$). More specifically, the time complexity increased to $O(|R|*|L|*|C|*|C_1 \cup C_2 \cup ... \cup C_k|)$, where $|L|$ is the average length of an itemset.

3. Experimental Confirmation

We performed empirical tests in order to confirm the above theoretical results concerning the efficiency and accuracy of the proposed algorithm. To this end, the quality of the extracted biclusters was calculated by using the match scores presented in [23]. They describe the degree of similarity between the extracted biclusters (M) and the biclusters implanted in synthetic input test data matrices (Mopt): the average bicluster relevance Sr*(M,Mopt), the average module recovery Sr*(Mopt,M), the match score S*(M,Mopt) and the match score S*(Mopt,M). All scores take the maximum value of 1, if M=Mopt.

Also, to evaluate comparative to co-clustering results, we used the well known Purity and Normalized Mutual Information (NMI) measures.

All the following experiments run on an Intel(R) Core(TM)2 Duo CPU 2.53 GHz system with 4GB RAM. An implementation of the proposed algorithm (BIB) is freely available for download at: http://150.140.136.40/lab/images/stories/uploads/files/BIB.rar

3.1 Experimental results for Boolean input data matrices

We first evaluate the proposed algorithm with respect to completeness. As an experimental test, we applied both the proposed and the Bimax[23] algorithms to various small (up to 20X20) Boolean input data matrices with implanted biclusters. Note that the Bimax algorithm has been evaluated against several other biclustering algorithms[23]. We used the implementation of Bimax included in BicAT V2.2 (http://www.tik.ethz.ch/sop/bicat) software tool[3]. Borgelt's implementation of Apriori algorithm (http://www.borgelt.net//software.html) is used, setting the argument "-tm" in order to extract only the maximal frequent itemsets. Exactly the implanted biclusters were extracted by both the proposed algorithm and Bimax in less than a second. Note that completeness is not guaranteed by all the biclustering algorithms, not even by all of those based on an exhaustive enumeration of all candidate biclusters.

We empirically test the time complexity of the proposed algorithm with respect to the Bimax algorithm using a large Boolean input test data matrix. The input test data matrix is a 698X72=50256 gene expression data file that was generated using the cDNA microarray technology. It is a partial dataset extracted from a yeast cell cycle dataset[26], which is supplied with Expander as a sample input file. It is transformed to Boolean by discretizing it using the BicAT and setting the dicretization threshold to 0, 1 and 2, (set cell value to 1 for cell values greater than the threshold and to 0 otherwise), in order to produce different sparse levels. Setting the dicretization threshold to 0, 1 and 2, then data matrices with 25263 1s – 24993 0s, 7583 1s – 42673 0s and 1087 1s – 49169 0s, are produced respectively.

The Bimax algorithm of BicAT is used setting the minimum number of rows (MNR) of a bicluster to 111, i.e., about 16% of the rows, and the minimum number columns of a bicluster to 1. Borgelt's implementation of Apriori algorithm is used, setting minimum support (MS) to 7%. Results are shown in Table 4.

Table 4. Test results for time complexity for Boolean input data matrix

Algorithm	Sparse	Preprocessing Time (sec)	Execution Time (sec)	Number of Biclusters
Bimax MNR=16%	0	---	5700	251785
Bimax MNR=1%	1	---	16	14574
Bimax MNR=1%	2	---	1	59
Proposed MS=7%	0	2	174.07	33085129
Proposed MS=1%	1	2	<1	20051
Proposed MS=1%	2	2	<1	70

Note that, in this testing environment, Borgelt's implementation of Apriori algorithm cannot extract frequent itemsets when it is applied to the 698X72 yeast cell cycle dataset and setting minimum support (MS) to less than 7%, due to memory limitations. Thus the extracted biclusters cannot have less than 49 rows. Also, note that the BicAT implementation of Bimax cannot handle biclusters in this testing environment with less than 111 rows (16%).

The proposed algorithm extracts all the different biclusters. However, the number of extracted biclusters could be huge (see Tables 4 & 9). The number of the extracted biclusters can be reduced by extracting only the maximal biclusters (argument "-tm" in Borgelt's implementation). A frequent itemset is called maximal if no superset is frequent, that is, has a support exceeding the minimum support. Alternatively, the user could access the whole set of extracted biclusters by applying some cluster[6] or bicluster[24] visualization techniques proposed in the literature.

Handling noise and overlapping is very important for biclustering algorithms. For instance, almost all the co-clustering algorithms cannot extract overlapping biclusters. Thus, we evaluated the proposed algorithm using the same synthetic data sets under the two different scenarios proposed in [23] for binary biclusters. The first scenario aims at studying the effects of noise and the second the effects of overlapping on the performance of the biclustering methods. For studying the effects of noise, we used both the discretization threshold *min+(max-min)/2* proposed in [23] and also the application of a clustering algorithm which can detect the number of clusters, e.g. [7], so that to assign 0s and 1s to the cell values of the same cluster.

Table 5. Test results for noise effects

Noise level	$S_r^*(M,Mopt)$	$S_r^*(Mopt,M)$	$S^*(M,Mopt)$	$S^*(Mopt,M)$
min+(max-min)/2				
0, 0.05, 0.10, 0.15	1	1	1	1
0.20	0,98	0,98	0,99	0,99
0.25	0,91	0,92	0,94	0,95
0,25 (-tc)	0,81	1	0,57	0,99
Clustering				
0, 0.05, 0.10, 0.15	1	1	1	1
0.20	0,97	0,98	0,98	0,98
0.25	0,94	0,95	0,96	0,97
0,25 (-tc)	0,79	1	0,55	1
0.25 (5%)	0,75	0,94	0,76	0,97
0.25 (2%)	0,12	0,27	0,19	0,47

Results are shown in Table 5 with respect to different noise levels. As far as overlapping is concerned, all scores are equal to 1, i.e. all the implanted biclusters are exactly extracted. The running-times varied between 0.01 and 4.10 seconds for overlapping biclusters and less than 0.01 seconds for non overlapping biclusters with noise, including the running-time of a filtering procedure for selecting the maximal biclusters (argument "-tm" in Borgelt's implementation). This procedure has similar effects with the greedy filtering procedure adopted in [23]: "in each step, the largest of the remaining biclusters is chosen that has less than *x*% of its cells in common with any previously selected bicluster", where *x* is a threshold. If such filtering is included in a post processing phase as in [23], then the running-times of the proposed algorithm varied between 0.04 and 5.05 seconds for non overlapping biclusters, since extra time is needed for writing the much more extracted biclusters to a file. Note, however, that if the number of optimal biclusters is not known beforehand, then the greedy filtering procedure adopted in [23] cannot be used. On the contrary, the proposed algorithm is not based on such apriori information. Also, the above tests were performed without assuming any prior knowledge about the sizes of biclusters. Minimum support was set to 8% due to memory constraints and thus even 8X1 biclusters were checked.

However, as mentioned above, the performance of the proposed algorithm is heavily relied on the minimum support: the lower the minimum support the larger the number of extracted biclusters. A large set of extracted biclusters possibly includes not true biclusters also, thus a comparison to an optimal bicluster set is not so successful. For instance, in Table 5, the match scores are shown for 0.25 noise level using clustering for discretization, setting minimum support to 5% and 2%. However, using any value between 7% and 10% the same match scores to 8% are obtained. Thus, the proposed algorithm need the definition of a range of minimum support values, (which is usually between 5% and 15%), for best results.

The dependence on the number of extracted biclusters can be tackled by the adopted filtering procedure. In all the above results, maximal biclusters were chosen, i.e., biclusters with maximal number of columns. Maximizing the number of columns is efficient in the absence of noise. However, the structure of maximal biclusters is destroyed by noise. In this case, one has to maximize both columns and rows for better results. Filtering only the frequent itemsets can handle noisy data more efficiently. A frequent itemset is called closed if no superset has the same support. For instance, in Table 5, the match scores are shown for 0.25 noise level using a filtering procedure for selecting the closed biclusters (argument "-tc" in Borgelt's implementation). Thus, optimal results are obtained for recovery. Note that Sr*(M,Mopt), the average bicluster relevance, reflects to what extent the generated biclusters represent true biclusters in rows. In contrast, Sr*(Mopt,M), the average module recovery, quantifies how well each of the true biclusters is recovered by the biclustering algorithm under consideration. Thus, the proposed algorithm applied to data with the effects of noise and overlapping either exhibits optimal results or it outperforms fast similar well-known biclustering algorithms.

3.2 Experimental results for biclusters with constant cell values

We applied the proposed and the CC[10], ISA[5], xMOTIF[21] and SAMBA[27] algorithms to an input data matrix with biclusters having constant cell values.

Table 6. Test input data matrix having constant biclusters

	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15	C16	C17	C18	C19	C20
R1	150	150	150	150	150	150	150	150	53	22	48	66	22	28	59	53	4	20	25	45
R2	150	150	150	150	150	150	150	150	66	27	11	23	10	20	45	60	4	21	58	27
R3	150	150	150	150	150	150	150	150	48	22	58	21	26	59	14	10	67	51	26	56
R4	150	150	150	150	150	150	150	150	21	27	56	60	56	3	26	25	9	13	9	1
R5	150	150	150	150	150	150	150	150	66	66	4	24	25	44	13	55	32	8	23	14
R6	17	27	16	12	4	22	16	26	100	100	100	100	100	100	4	17	1	42	43	16
R7	40	2	18	28	57	3	47	58	100	100	100	100	100	100	21	52	9	40	15	22
R8	13	42	1	23	20	29	48	29	100	100	100	100	100	100	28	17	17	45	19	41
R9	2	46	24	28	4	5	6	12	100	100	100	100	100	100	43	34	6	3	2	33
R10	55	20	51	26	3	44	58	15	100	100	100	100	100	100	11	33	46	43	47	54
R11	59	46	18	20	26	58	33	37	100	100	100	100	100	100	33	52	16	28	9	33
R12	54	39	24	1	22	39	24	21	100	100	100	100	100	100	34	9	34	18	44	27
R13	18	34	35	26	18	26	14	24	100	100	100	100	100	100	37	1	37	17	15	42
R14	25	44	28	9	10	27	25	48	59	41	15	3	20	10	200	200	200	200	200	200
R15	30	31	37	31	13	11	31	33	55	16	33	42	1	36	200	200	200	200	200	200
R16	55	56	42	20	16	51	34	58	11	8	16	14	27	8	200	200	200	200	200	200
R17	17	1	41	46	57	48	1	23	46	21	39	9	42	37	200	200	200	200	200	200
R18	59	49	40	34	42	50	47	51	29	4	46	25	15	17	200	200	200	200	200	200
R19	33	8	15	46	39	12	41	43	3	13	20	40	59	37	200	200	200	200	200	200
R20	56	16	56	42	50	28	17	39	25	58	50	44	26	23	200	200	200	200	200	200

The input data matrix 120X20 is formed by vertically repeating 6 times the matrix shown in Table 6. We considered Mopt={(R1-R5…R101-R105/C1-C8), (R6-R13…R106-R113/C9-C14), (R14-R20…R114-R120/C15-C20)} as the implanted biclusters having cell values equal to 150, 100 and 200 respectively. The rest cells values are randomly set from the range 1-60. We used the implementation of CC, ISA and xMOTIF algorithms included in BicAT V2.2, and also that of SAMBA algorithm included in Expander 5.1 software tool[25]. The results are shown in Table 7. Note that we performed several experiments in order to obtain the best results for CC (delta=0.9, number of output clusters = 3), ISA (t_g=1, t_c=1), xMOTIF (α=0.01, s_d=3) and SAMBA (Try covering all probes: true, Overlap prior factor: 0.9, Hashing kernel size: Minimal=5 Maximal=5) algorithms. On the contrary, the proposed algorithm does not need any parameter testing phase but just setting the minimum support, i.e. the minimum number of rows. The user can always set a low minimum support in order to extract even small biclusters, of course at a cost of a higher execution time.

Table 7. Test results for the quality of biclustering for biclusters with constant cell values

Algorithm Applied	Preprocessing Time (sec)	Execution Time (sec)	Number of Biclusters	$S^*(M,M_{opt})$	$S^*(M_{opt},M)$
CC	---	1 sec	3	0,75	0,75
ISA	---	<1 sec	3	1	1
xMotif	---	3 sec	3	1	1
SAMBA	---	8 sec	3	0,79	0,79
Proposed	<1 sec	<1 sec	3	1	1

To evaluate the proposed algorithm with respect to recent co-clustering algorithms, we apply it to the UCI test data sets used in [17] with categorical values, extracting biclusters with constant cell values in the columns. Note that the proposed algorithm can extract biclusters with constant cell values in the columns and/or rows, either numeric or categorical ones. Thus, there is no need to transform categorical attributes to a long list of binary attributes, as in [17]. The results (Purity/NMI) are shown in Table 8 (minimum support=30%). Note also that a naïve criterion is used to form the final classes; start from the biclusters with the greater number of columns and select the rows of each bicluster that have not previously included in a class in order to form a new class. Of course, significantly better results could be obtained by adopting more elaborated criteria to form the classes. However, the latter goes beyond the scope of this paper.

Table 8. Test results for the quality of biclustering for biclusters with categorical cell values

Algorithm	Kmodes[16]	NMF[18]	ONMTF[11]	SpecCo[17]	Proposed
Soybean small	97/94	100/100	100/100	100/100	97.87/80.50
Zoo	89/89	88/87	90/80	90/92	97.03/77.28
Soybean large	53/67	58/71	65/77	67/78	64.82/78.43
Congressional votes	86/45	81/48	87/48	87/47	89.66/28.89

The 698X72 yeast cell cycle dataset mentioned above is also used to test both the accuracy and the time complexity of the proposed algorithm for constant cell values. Cell values are in the range [-6,6], while the average values are in the range [-2,2]. To form the input data matrix, every cell within the ranges [-6,-2), [-2,0), [0,2] and (2,6] is replaced by the values 5, 10, 15 and 20 respectively. We applied the proposed and the CC, ISA, xMOTIF and SAMBA algorithms to the 698X72 yeast cell cycle dataset. Parameters of all the applied algorithms were properly set, (CC: delta=0.1, number of output clusters = 500, ISA: number of starting points = 1200, xMOTIF: α=0.05, n_s=50 and SAMBA: Try covering all probes: true, Overlap prior factor: 0, Hashing kernel size: Minimal=1 Maximal=3), in order to extract the same number of biclusters. Note that the used implementation of xMOTIF algorithm cannot handle input data matrices with more than 64 columns, thus the last 8 columns were deleted. Also, it was not possible to extract more than 130 biclusters by applying the SAMBA algorithm. The results are shown in Table 9, along with the quality of biclustering.

Table 9. Test results for time complexity for biclusters with constant cell values

Algorithm	Preprocessing Time (sec)	Execution Time (sec)	Number of Biclusters
CC	---	2319	500
ISA	---	64	519
xMOTIF(64 col.)	---	253	500
SAMBA	---	8	130
Proposed	1	<1	548

4. Conclusion

We proposed a new biclustering algorithm which guarantees the detection of all biclusters by enumerating all candidate ones. To reduce the infeasible for high dimensional data sets search space, it is based on a theory inspired by that of Association Rule Mining.

We also presented two general approaches for extending the proposed biclustering algorithm to other biclustering models. Of course, the detailed presentation of how the proposed algorithm can be properly modified for application to each different biclustering model goes far beyond the scope of this paper.

We are currently working in extending the proposed algorithm to all the different biclustering modes proposed in the literature.

Acknowledgments

The author wish to thank G. Ravasopoulos and D. Katsantas, PhD candidates, for their help during experimental tests.

References

R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, A.I. Verkamo, Fast discovery of association rules, in: Fayyad, U.M. Piatetsky-Shapiro, G. Smyth, P. Uthurusamy R. (Eds.), Advances in Knowledge Discovery and Data Mining, AAAI/MIT Press, 1996, pp. 307–328.

R. Agrawal, J. Gehrke, D. Gunopulos, P. Raghavan, Automatic subspace clustering of high dimensional data, Data Mining and Knowledge Discovery 11 (2005) 5–33.

S. Barkow, S. Bleuler, A. Prelic, P. Zimmermann, E. Zitzler, BicAT: a biclustering analysis toolbox, Bioinformatics 22(10) (2006) 1282-1283.

A. Ben-Dor, B. Chor, R. Karp, Z. Yakhini, Discovering local structure in gene expression data: The order–preserving submatrix problem, Proceedings of the 6th International Conference on Computacional Biology (RECOMB'02), (2002) 49–57.

S. Bergmann, J. Ihmels, N. Barkai, Iterative signature algorithm for the analysis of largescale gene expression data, PHYSICAL REVIEW E67031902 (2003) 1–18.

B. Boutsinas, Accessing Data Mining Rules through Expert Systems, International Journal of Information Technology & Decision Making 1(4) (2002) 657-672.
B. Boutsinas, D. Tasoulis, M.N. Vrahatis, Estimating the number of clusters using a windowing technique. Pattern Recognition.and Image Analysis, 16(2) (2006) 143-154.
B. Boutsinas, C. Siotos, A. Gerolymatos, Distributed mining of association rules based on reducing the support threshold, International Journal on Artificial Intelligence Tools, 17(6) (2008) 1109-1129.
S. Busygin, O. Prokopyev, P.M. Pardalos, Biclustering in data mining, Computers & Operations Research 35 (2008) 2964–2987.
Y. Cheng, G.M. Church, Biclustering of expression data, Proceedings of the 8th International Conference on Intelligent Systems for Molecular Biology (ISMB'00) (2000) 93–103.
C. Ding, T. Li, W. Peng, H. Park, Orthogonal nonnegative matrix tri-factorizations for clustering, ACM SIGKDD, 2006.
R.C. Dubes, A.K. Jain, Clustering methodologies in exploratory data analysis, Adv. Comput. 19 (1980) 113-228.
M. Ester, H.-P. Kriegel, J. Sander, X. Xu, A density-based algorithm for discovering clusters in large spatial databases with noise, in: E. Simoudis, J. Han, U.M. Fayyad (Eds.), Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96), AAAI Press, (1996) 226–231.
G. Grahne, J. Zhu, Efficiently Using Prefix-trees in Mining Frequent Itemsets, Proceedings of Workshop Frequent Item Set Mining Implementations (FIMI 2003), Germany, (2003).
J. Han, J. Pei, Y. Yin, R. Mao, Mining frequent patterns without candidate generation: A frequent-pattern tree approach, Data Mining and Knowledge Discovery 8 (2004) 53–87.
Z. Huang, Extensions to the k-means algorithm for clustering large data sets with categorical values, Data mining and Knowledge Discovery 2 (1998) 283-304.
L. Labiod, M. Nadif, Co-Clustering for binary and categorical data with maximum modularity, Proceedings of the 11th international conference on Data Mining (2011) 1140-1145.
D.D. Lee, H.S. Seung, Learning the parts of objects by non-negative matrix factorization, Nature, 401 (1999) 788-791.
J. Liu, J. Yang, W. Wang, Biclustering in Gene Expression Data by Tendency, Proceedings of the 2004 IEEE Computational Systems Bioinformatics Conference.
S.C. Madeira, A.L. Oliveira, Biclustering algorithms for bilogical data analysis: a survey, IEEE Transactions on Computational Biology and Bioinformatics 1 (2004) 24–45.
T.M. Murali, S. Kasif, Extracting conserved gene expression motifs from gene expression data. Proceedings of the Pacific Symposium on Biocomputing volume 8 (2003) 77–88.
G. Pandey, G. Atluri, M. Steinbach, C.L. Myers, V. Kumar, An association analysis approach to biclustering, Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining (2009) 677-686.
A. Prelić, S. Bleuler, P. Zimmermann, A. Wille, P. Bühlmann, W. Gruissem, L. Hennig, L. Thiele, E. Zitzler, Comparison of Biclustering Methods: A Systematic Comparison and Evaluation of Biclustering Methods for Gene Expression Data, Bioinformatics 22:9 (2006) 1122-1129.
R. Santamaria, R. Theron, L. Quintales, BicOverlapper: A tool for bicluster visualization, Bioinformatics, 24 (9) (2008) 1212-1213.
R. Sharan, A. Maron-Katz, R. Shamir, CLICK and EXPANDER: A system for clustering and visualizing gene expression data, Bioinformatics 14 (2003) 1787-1799.
P.T. Spellman, G. Sherlock, M.O. Zhang, V.R Iyer, K. Anders, M.B. Eisen, P.O. Brown, D. Botstein, B. Futcher, Comprehensive identification of cell cycle-regulated genes of the yeast Saccharomyces cerevisiae by microarray hybridization. Mol Biol Cell 9(12) (1998) 3273-97.
A. Tanay, R. Sharan, R. Shamir, Discovering statistically significant biclusters in gene expression data, Bioinformatics 18:1 (2002) S136-S144.
A. Tanay, R. Sharan, R. Shamir, Biclustering algorithms: a survey, in: Chapman, A.S. (Ed) Handbook of Computational Molecular Biology, 2004.
R. Xu, D. Wunsch, Survey of clustering algorithms. IEEE Transactions on Neural Networks 16:3 (2005) 645–678.
Z. Zheng, R. Kohavi, L. Mason, Real World Performance of Association Rule Algorithms, Proceedings of the KDD-2001, (2001).