A Co-occurrence Based Approach for Mining Overlapped Co-clusters in Binary Data

Yuri Santa Rosa *Nassar* dos Santos[1,2(✉)[0000−0001−9371−427X]], Rafael Santiago[1,2[0000−0001−7033−125X]], Raffaele Perego[3[0000−0001−7189−4724]], Matheus Henrique Schaly[1], Luis Otávio Alvares[1[0000−0003−2372−4995]], Chiara Renso[3[0000−0002−1763−2966]], and Vania Bogorny[1,2[0000−0002−0159−4643]]

[1] Universidade Federal de Santa Catarina, Florianópolis, Brazil
[2] INE, Programa de Pós-Graduação em Ciência da Computação, Florianópolis, Brazil
yuri.nassar@posgrad.ufsc.br
[3] ISTI-CNR, Pisa, Italy

Abstract. Co-clustering is a specific type of clustering that addresses the problem of simultaneously clustering objects and attributes of a data matrix. Although general clustering techniques find non-overlapping co-clusters, finding possible overlaps between co-clusters can reveal embedded patterns in the data that the disjoint clusters cannot discover. The overlapping co-clustering approaches proposed in the literature focus on finding *global* overlapped co-clusters and they might overlook interesting local patterns that are not necessarily identified as global co-clusters. Discovering such *local* co-clusters increases the granularity of the analysis, and therefore more specific patterns can be captured. This is the objective of the present paper, which proposes the new Overlapped Co-Clustering (OCoClus) method for finding overlapped co-clusters on binary data, including both *global* and *local* patterns. This is a non-exhaustive method based on the co-occurrence of attributes and objects in the data. Another novelty of this method is that it is driven by an objective cost function that can automatically determine the number of co-clusters. We evaluate the proposed approach on publicly available datasets, both real and synthetic data, and compare the results with a number of baselines. Our approach shows better results than the baseline methods on synthetic data and demonstrates its efficacy in real data.

Keywords: Co-clustering · Overlapped co-clusters · Binary data.

1 Introduction

Over the years, the task of clustering complex data has become more challenging since a high number of attributes can increase computational complexity and affect cluster consistency [17]. One way to deal with this complexity is to use the co-clustering approach, which simultaneously clusters objects (rows) and attributes (columns) in matrix data [5]. The focus of these methods relies on finding co-clusters, where each co-cluster is formed by a subset of objects and attributes that can represent a submatrix of a given matrix.

Co-clustering approaches use, in general, a non-overlapping strategy, which means that an *element* of a co-cluster can belong to only one co-cluster [1,14]. However, in many real situations, an *element* can participate simultaneously in more than one co-cluster. For example, a movie could be both *thriller* and *science-fiction*, a song can be both *rock* and *high-energy*, etc. Therefore, an overlapping strategy is important because it identifies *intersections* between co-clusters, revealing patterns that could be lost when using disjoint co-clustering. Besides, the detection of overlapping co-clusters has proven to be challenging since it is not trivial to evaluate the clustering quality [8].

We notice that most of the overlapping co-clustering approaches proposed in the literature have two characteristics: (1) they discover *global* clusters, and (2) they tend to fit a specific application. Examples can be found in text mining [2], bioinformatics [13], recommendation systems [15], and social network analysis [18], to name a few. In contrast, the works of Fu et al. [4], Li [7], Whang et al. [16], and Zhu et al. [19], not only focus on *global* clusters on binary datasets, but they were designed for generic purposes. The limitation of these works is that they do not detect *local* co-clusters, i.e., refined groups formed by objects and attributes that identify an overlap pattern on *global* co-clusters.

In this paper, we propose a new co-clustering method that combines simplicity of use with the capacity to extract overlapping global and local co-clusters. This method is named Overlapped Co-Clustering (OcoClus) and it is based on the co-occurrence of attributes and objects. The main novelty of this method is that, unlike the traditional overlapped co-clustering, it is able to infer a new type of patterns called *local* co-clusters. Furthermore, OCoClus is driven by an objective cost function that does not require the user-defined parameter of the number of co-clusters.

In summary, we make the following contributions: (i) propose an incremental co-clustering approach that is application-independent and an algorithm that can find both overlapped and non-overlapped *global* and *local* co-clusters; (ii) use a cost function that, finds the number of co-clusters automatically, that ranks the co-clusters from the most relevant to the less relevant, and that finds overlapped co-clusters.

The remainder of this work is organized as follows. The basic concepts definition of our work are presented in Section 2. Section 3 presents the works that are related to our proposal. Section 4 presents the details of our method. Section 5 presents the evaluation of the method with synthetic and real data. Finally, the conclusion and further research directions of our work are presented in Section 6.

2 Basic Concepts

In this section we present the basic concepts to guide the reader throughout this paper.

Let D be a *binary* matrix with N rows (objects) and M columns (attributes). The element d_{ij} of D, where i and j are integers that $1 \leq i \leq N$ and $1 \leq j \leq M$,

is equal to 1 if the j-th attribute occurs in the i-th object (*true element*); otherwise, it is 0. Co-clustering is the grouping task of finding K (*global*) co-clusters in D where each co-cluster is formed by a subset of objects and attributes [11]. The subset of objects I can be represented as a binary vector of length N, where $I_i = 1$ indicates that the i-th object is present in I. Similar to that, a subset of attributes J with $J_j = 1$ indicates that the j-attribute is present in J with length M. More formally, a co-cluster can be defined as follows:

Definition 1. ***Co-cluster:*** *Let D be a binary matrix, I be the subset of objects, J be the subset of attributes; a co-cluster C is defined as $C = \langle I, J \rangle$. The elements c_{ij} of co-cluster C are formed by the outer product of its subsets I and J ($C \in \{0,1\}^{|I| \times |J|}$). Thus, a co-cluster C can represent a submatrix of D.*

Such (*global*) co-cluster C can be formed with only the *true elements* in D or mixed with true elements and *noise elements* ($d_{ij} = 0$). In this paper, the terms *global* co-cluster and co-cluster are interchangeably used. The co-occurrence between objects and attributes can form a co-cluster C which can be simplified by searching elements $d_{ij} = 1$ in the matrix D. Furthermore, it can reduce the search space once the goal is to identify true co-occurrences. Inspired by [9], we adapted four concepts for co-clustering problem: cost function $\mathcal{F}_P$, *pure* co-cluster PC, noise thresholds ϵ_I and ϵ_J, and expanded *pure* co-cluster EC. The cost function $\mathcal{F}_P$ can be used to evaluate the process of forming a co-cluster. More formally, we can define the cost function $\mathcal{F}_P$ as follows:

Definition 2. ***Cost Function:*** *Let C^* be a co-cluster, $\prod$ be a set of global co-clusters, D be an input matrix, ρ be a weight of importance for the co-clusters cost, $\mathcal{N}$ be a noise matrix, ${\gamma_C}^*$ and $\gamma_{\mathcal{N}}$ be user-defined functions for measuring the costs of co-clusters and noise; a cost function $\mathcal{F}_P$ is defined as $\mathcal{F}_P(\prod, \boldsymbol{D}) = \rho \times \sum_{C^* \in \prod} {\gamma_C}^*(C^*) + \gamma_{\mathcal{N}}(\mathcal{N})$.*

The objective is to minimize $\mathcal{F}_P$ regarding ρ, ${\gamma_C}^*(C^*)$ and $\gamma_{\mathcal{N}}(\mathcal{N})$. Such noise matrix $\mathcal{N}$ used by [9] takes into account the *false positives*, *false negatives*, and the already covered elements in D. Regarding the set of co-clusters $\prod$ and matrix D, the *false positives* are elements $d_{ij} = 0$ covered by some pattern in $\prod$, while *false negatives* are elements $d_{ij} = 1$ not covered by any pattern in $\prod$. The concept of *pure* co-cluster simplifies the identification of a *global* co-cluster, which identifies a disjoint *global* co-cluster that contains only *true elements*. Thus, we can define a *Pure Co-cluster* PC as follows:

Definition 3. ***Pure Co-cluster:*** *Let D be a binary matrix, d_{ij} be an element of D; a pure co-cluster $PC = \langle PC_J, PC_I \rangle$ is formed by a subset of objects PC_I and a subset of attributes PC_J that identifies only the true elements. Thus, PC can represent a submatrix of D, which contains only the true elements $d_{ij} = 1$.*

A *pure* co-cluster PC can be expanded with noisy objects and attributes. We use two thresholds to control the amount of noise in a co-cluster: ϵ_I for objects and ϵ_J for attributes. Thus, the noise thresholds ϵ_I and ϵ_J can be defined as follows:

Definition 4. ***Noise Thresholds:*** *Let C^* be a co-cluster, C^*_J and C^*_I be the subsets of attributes/objects that define C^*; a maximum noise threshold for objects ϵ_I and attributes ϵ_J limit the amount of noise that can be included in C^*. Thus, each new object must be included in at least $(1-\epsilon_I) \times ||C^*_J||$ attributes of C^*, while each new attribute must be included in at least $(1-\epsilon_J) \times ||C^*_I||$ objects of C^*.*

The noise threshold value can range from $[0, 1]$, where 0 does not allow any noise while 1 allows the maximum amount. The number of objects and attributes of a given subset is measured by the L^1-norm $|| \cdot ||$ (or Hamming norm), which simply counts the number of bits 1 in the vector. From that, the expanded *pure* co-cluster EC represents a expanded version of PC with noise data. More formally, an *Expanded Pure Co-cluster EC* can be defined as follows:

Definition 5. ***Expanded Pure Co-cluster:*** *Let PC be the pure co-cluster, EC_I be a subset of objects, EC_J be a subset of attributes, ϵ_I be the noise object threshold, ϵ_J be the noise attribute threshold; an expanded pure co-cluster is defined as $EC = \langle EC_J, EC_I \rangle$, where EC_J and EC_I can contain new attributes and objects not present in PC regarding the noise thresholds ϵ_I and ϵ_J.*

3 Related Works

Because of the difficulty in finding co-clusters, there is no method widely accepted as the state-of-the-art; instead, there are algorithms that perform better in certain types of data than others. Since a complete review is out of the scope of this paper, we shall briefly discuss some reference algorithms. For a comprehensive review of co-clustering algorithms, we refer to [12].

Dhillon[3] used the matrix decomposition using the eigenvectors combined with bipartite graph to find *global* co-clusters in a real-valued matrix. It needs to know the number of co-clusters *a priori* and the order of the discovered co-clusters is not important. Furthermore, it uses a support matrix to include some attributes as noise data; however, it does not have any noise-parameter to control the number of objects or attributes as noise data. Kluger et al.[6] extended the Dhillon[3] approach by using the singular value decomposition to find *global* co-clusters. It assumes that the data have a checkerboard structure in the matrix. This approach includes each element of a matrix into one co-cluster without overlap; therefore, it cannot control the noise data.

Fu et al.[4] proposed a Bayesian-based overlapping co-clustering approach based on a multivariate distribution to find *global* co-clusters in a binary data matrix. It assumes that the number of co-clusters is known *a priori*. This approach does not indicate that the order of the discovered co-clusters is important. It tolerates noisy elements in the co-clusters; however, it does not have a noise-parameter to control the number of objects or attributes included in the co-cluster as noise. Zhu et al.[19] proposed an overlapping co-clustering approach to approximate a binary data matrix with the sum of identified *global* co-clusters. This method needs to have the number of co-clusters a priori. Furthermore, it

does not deal with noise data and does not associate any importance for the co-cluster that explains the discovered order.

Lucchese et al.[9] proposed a frequent pattern mining method for binary datasets. The patterns are formed by sets of attributes and objects, where they can represent a non-overlapped *global* co-cluster. It uses a generalized cost function to drive the mining process to find the number of patterns automatically. The discovered order of the patterns is relevant regarding the cost function; therefore, it can be seen as a ranking. Finally, two noise thresholds control the number of noisy attributes and objects included in a pattern.

Whang et al.[16] modeled the input data as a bipartite graph to find *global* overlapping co-clusters in binary data. This method allows to include noise objects in the co-clusters with a probability distribution function that models the noise. However, it does not define noise thresholds to control the number of noise objects and attributes. It can automatically infer the number of co-clusters, besides that, the method does not consider that the order of the discovered co-clusters is relevant in the process.

Li[7] presented a generalized overlapped co-clustering approach that uses singular value decomposition on the binary data matrix to identify *global* co-clusters. This method does not include noise automatically or by a user-defined noise threshold; it searches for homogeneous co-clusters without noise. Furthermore, it can infer the number of co-clusters; however, it is not guaranteed to converge to the optimum number. Finally, the method does not show that the discovered order of these co-clusters is relevant to it.

4 The Overlapped Co-clustering Approach

In this section we present a new method called *OCoClus* (Overlapped Co-Clustering) for finding overlapped co-clusters in a binary dataset by identifying both *global* and *local* co-clusters. OCoClus searches for the co-occurrences between attributes and objects to identify co-clusters where a cost function drives the co-clustering process. In the following we present the method definitions in Section 4.1 and the proposal details in Section 4.2.

4.1 Method definitions

Local co-clusters are patterns in the data related to specific characteristics that are overlooked by *global* co-clusters since it finds clusters which are in the intersection of objects and attributes of the *global* clusters. Thus, we formally define a *local* co-cluster LC as follows:

Definition 6. ***Local Co-cluster:*** *Let $\prod$ be the set of co-clusters, LC_I be a subset of objects, LC_J be a subset of attributes, C be the co-cluster in $\prod$; a local co-cluster is defined as $LC = \langle LC_I, LC_J \rangle$, where the object intersections of co-cluster C with the co-clusters in $\prod$ forms LC_I, and the union of the attributes between C and the intersected co-clusters in $\prod$ forms LC_J.*

We propose a new cost function $\mathcal{F}$ designed to make the overlapping and non-overlapping co-clusters equally important including both *global* and *local* co-cluster. The difference between the new cost function in Definition 7 and the cost function given in Definition 2 is that the new cost function considers just the size of the co-cluster and the quantity of noise that can be included in the co-cluster. However, the cost function in Definition 2 weights the relevance of the patterns regarding its size, it penalizes the patterns that cover an element already covered and does not include an element into the expected pattern. From that, we define the new cost function $\mathcal{F}$ as follows:

Definition 7. ***Cost Function:*** *Let $\prod$ be the set of co-clusters, D be the binary matrix, C^* be the co-cluster, $||C_J^*||$ and $||C_I^*||$ be the size of the subsets of attributes/objects that define C^*, $\mathcal{N}$ be the number of noise elements included in C^* ($d_{ij} = 0$), and H be the part that does not consider noise data; a cost function $\mathcal{F}$ is defined as $\mathcal{F}(\prod, \boldsymbol{D}) = H + \mathcal{N}$, where $H = \sum_{C^* \in \prod} (||C_I^*|| + ||C_J^*||) - (||C_I^*|| \times ||C_J^*||)$. Thus, the objective is to minimize $\mathcal{F}$ regarding C_I^*, C_J^* and $\mathcal{N}$.*

Regarding the new cost function $\mathcal{F}$, part H contributes to the cost function evaluating co-clusters without noise, while part $\mathcal{N}$ contributes by allowing some noise data regarding the maximum noise thresholds. Once we have already formalized the main definitions, it is simple to define the overlapped co-cluster used to represent the *global* and *local* patterns as follows:

Definition 8. ***Overlapped Co-cluster:*** *Let $\prod$ be the set of co-clusters, X be the co-cluster $\in \prod$ with its subset of attributes X_J and objects X_I, Op be the set of co-clusters $\in \prod$ that intersect X_I, and Op_J and Op_I be the subset of attributes and objects of Op; an overlapped co-cluster is formally defined as $OC = \langle X_J \cup Op_J, X_I \cap Op_I \rangle$.*

Considering the Definition 8, the subset of objects of OC_I is formed by the nested intersection of objects between X_I and Op_I ($X_I \cap Op_I$), and the subset of attributes OC_J by joining the attributes of X_J with Op_J ($X_J \cup Op_J$).

4.2 Method description

Algorithm 1 is the main algorithm that organizes our approach. It receives four input parameters: the matrix D, the number of co-clusters K, the object noise threshold ϵ_I and the attribute noise threshold ϵ_J. As a result, it outputs a set of co-clusters Φ which contain K co-clusters that can overlap.

In Algorithm 1, the set of co-clusters $\prod$ is set as empty (line 1), and the residual matrix D_r is initiated with D, which is used to find uncovered co-clusters in D (line 2). The algorithm iterates over *findPureCocluster* (line 4) and *expandPureCocluster* (line 5) methods at most K times, where K is the maximum number of co-clusters (line 3). In *findPureCocluster* method, the attributes in D_r are sorted in descending order (from the most frequent to the least) and stored in a list S to maximize the probability of forming large co-clusters. Therefore,

Algorithm 1 OCoClus

Input: D: input matrix
K: max number of clusters {optional}
ϵ_I: max object noise threshold {optional}
ϵ_J: max attribute noise threshold {optional}
Output: Φ: set of disjoint and overlapped co-clusters
OCoClus ($K, D, \epsilon_J, \epsilon_I$)
1: $\prod \leftarrow \emptyset$
2: $D_r \leftarrow D$ {residual matrix}
3: **for** $i = 1, \ldots, K$ **do**
4: PC,$E \leftarrow$ findPureCocluster($D, D_r, \prod$) {Definition 3}
5: $EC \leftarrow$ expandPureCocluster($PC, E, \prod, D, \epsilon_I, \epsilon_J$) {Definition 5}
6: **if** $\mathcal{F}(\prod, D) < \mathcal{F}(\prod \cup EC, D)$ **then**
7: **break**
8: **end if**
9: $\prod \leftarrow \prod \cup EC$
10: $D_r(i,j) \leftarrow 0\ \forall i,j$ where $EC_I(i) = 1 \wedge EC_J(j) = 1$
11: **end for**
12: $\Phi \leftarrow$ findOverlap($\prod$) {Definition 8}
13: **return** Φ

the attributes in S are evaluated for being added to a co-cluster without backtracking reducing the search space. Only the true elements in D forms the *pure* co-cluster PC regarding the attributes in S. With this, the number of objects and attributes that co-occur are used in the cost function $\mathcal{F}$ stated in Definition 7 to evaluate if the tested subsets of objects and attributes can minimize the cost function. The PC grows in the number of objects and attributes as long as the cost function $\mathcal{F}$ is minimized. Besides, some attributes cannot be used to form the PC, then these attributes are stored in an extension list E. The output is a *pure* co-cluster PC and an extension list of attributes E.

In *expandPureCocluster* method, OCoClus expands PC with new objects and attributes that allows noise data (line 5). With this, the *expanded* co-cluster EC is initiated with PC identified at line 4. Then, the process is similar to *findPureCocluster*; however, at this part, the method checks if the inclusion does not exceed the maximum noise thresholds (Definition 4) and improves the cost function $\mathcal{F}$. This inclusion occurs in two steps. First, the method tries to include new objects that are not present in EC and does not modify the current attributes. Second, it does not modify the current objects and tries to include the attributes stored in the extension list E one at a time without backtracking. If an attribute is included in EC, the process goes back to the first step and repeats both steps. We remark that each new object and attribute is included in EC if such inclusions respect both Definition 4 and Definition 7. This process is repeated while E is not empty. The *expandPureCocluster* returns an *expanded* co-cluster EC as the output.

Given the output of the *expandedPureCocluster*, if the new co-cluster EC minimizes the cost function $\mathcal{F}$ of the model (line 6), it is added to $\prod$ (line 9). However, if EC does not minimize the cost function $\mathcal{F}$ of the model, even though the parameter K does not reach its maximum value, the algorithm stops searching for new co-clusters (line 7). Besides, if the cost function $\mathcal{F}$ is improved, the residual matrix D_r is then updated with EC (line 10). The updated residual

matrix D_r is used in the next iteration to find new patterns in D that are not covered by any previous co-cluster. OCoClus can find the number of co-clusters automatically whenever K is not given. However, if the user misspecify the value of K, then the true number of co-clusters may not be discovered.

So far, $\prod$ covers non-overlapped patterns in the data (line 9); hence, it cannot show which co-clusters *share* characteristics. Therefore, OCoClus refines these non-overlapped co-clusters to identify *global* and *local* overlapped co-clusters as stated in Definition 8. With this, the *findOverlap* method (line 12) iterates over $\prod$ to identify possible overlapped co-clusters by taking the nested object intersections between the co-clusters in $\prod$. The nested intersection considers what is shared among all intersected co-clusters instead of a common intersection between pairs of co-clusters. If such a co-cluster intersection exists, the attributes of the co-clusters involved in the intersection are joined. The next step is to delete the redundant co-clusters, i.e., co-cluster totally covered by another co-cluster. From that, the *findOverlap* is a simple and effective method that allows OCoClus to find both overlapped co-cluster structures. Its simplicity and effectiveness become possible by exploring the nested intersections of objects and joining attributes separately regarding the co-clusters in $\prod$. This process looks simple once the cost function $\mathcal{F}$ already evaluated the disjoint co-clusters in the previous methods, but it effectively identifies overlapped co-clusters. At the end, *findOverlap* returns the set of non-overlapped and overlapped (if exist) co-clusters Φ. Finally, Algorithm 1 returns this set of non-overlapped and overlapped patterns including both *global* and *local* co-clusters (line 13).

Proposition 1. *Let K be the maximum number of non-overlapped co-clusters, N the total number of objects, M the total number of attributes, and P the number of overlapped co-clusters. The computational complexity of findPureCocluster method is $O(MN)$, expandPureCocluster method is $O(M(MN+N)) = O(M^2N)$, and findOverlap method is $O(K^2+P^2)$. Regarding the overall complexity of our algorithm, OCoClus calls findPureCocluster and expandPureCocluster methods, then builds D_r for each of the K (or less) non-overlapped co-clusters and finalizes with the findOverlap method. Thus, the computational complexity of the OCoClus Algorithm is $O(KM^2N + (K^2+P^2))$.*

5 Experimental Evaluation

We compare OCoClus[1] with four publicly available methods presented in the related works to use as the baseline methods, which are: Li[7], Lucchese et al.[9], Kluger et al.[6], and Dhillon[3]. We include the works of Dhillon and Kluger et al. because they are consolidated approaches in the literature and publicly available as a package by *scikit learn*[2]. Considering their stable implementation, we selected these works once the overlapping baseline methods fail to find the embedded overlapped co-clusters. Therefore, we include those non-overlapped

[1] https://github.com/bigdata-ufsc/ococlus
[2] https://scikit-learn.org/stable/modules/biclustering.html

Table 1: Datasets description.

Dataset	number of objects	number of attributes	Sparsity (%)	number of co-clusters
Synthetic-1	100	100	76.62	7
Synthetic-2	600	1000	77.97	10
Synthetic-3	100	100	68	4
CAL500	502	103	76.6	-
CV-19	5729	567	98.18	-

methods in the baseline to compare such a co-clustering result. We used three synthetic datasets named *synthetic-1* and *synthetic-2*, and *synthetic-3*, where we artificially embedded the co-clusters (patterns) to create the ground-truth datasets. Furthermore, we also evaluated OCoClus on two real-life datasets, named CAL500[3] and CV-19[4], to show its efficacy in the real application scenario. All the experiments data and source code are made public.

We performed the experiments in a machine with a processor Intel i7-7700 3.6GHz, 16 GB of memory, and OS Windows 10 64bits. Furthermore, we ran 15 independent simulations for all methods on each synthetic dataset to compute the average and standard deviation of the evaluation metrics score. Table 1 shows the main characteristics of the datasets used in the experiments. It shows the total number of objects and attributes, the sparsity in the data (percentage of zeros), and the number of co-clusters for the synthetic datasets.

We use four evaluation metrics to assess the quality of the OCoClus. First, we use the reconstruction error matrix to measure the difference between data input and found co-clusters given by $\text{Rec}_{error} = ||X \veebar Y||$ similar to [7]. In short, we take the sum of the element-wise *xor* ($\veebar$) between the input data matrix X (ground-truth) and the reconstructed matrix Y regarding the found co-clusters to measure the quantity of *false positives* and *false negatives*. The clustering quality is better when the result of the Rec_{error} is equal or close to zero. The other three metrics are *Omega* index (overlapped version of ARI measure), *Overlapped Normalized Mutual Information* (ONMI) and *overlapped F1* measure (F_{score}) [10]. For these three last measures, the clustering quality is better when the result is equal or close to one, where one is the maximum score. We decided to use these measures since our approach focuses on the overlapping problem and therefore the traditional measures like for example *Adjusted Rand Index* (ARI), *Normalized Mutual Information* (NMI), and F_{score} are not suitable to capture the overlapping behaviour.

5.1 Evaluation of OCoClus with Synthetic Data

To be fair with all methods, we set the number of co-clusters K according to the ground-truth shown in Table 1. Regarding the noise control used by Lucchese et

[3] http://mulan.sourceforge.net/datasets-mlc.html
[4] https://ti.saude.rs.gov.br/covid19/; just passed away people data were used.

Table 2: Score of the evaluation metrics for the synthetic datasets.

Dataset	Work	Rec_{error}	Omega	ONMI	F_{score}
Synthetic-1	Li	2086.27∓371.33	-0.0110∓0.0228	0.0060∓0.0103	0.0717∓0.1051
	Lucchese_{t_1}	0∓0	0.6640∓0	0.5419∓0	0.6617∓0
	Lucchese_{t_2}	160∓0	0.6125∓0	0.509∓0	0.6359∓0
	Lucchese_{t_3}	160∓0	0.6125∓0	0.509∓0	0.6359∓0
	Dhillon	1404∓0	0.1818∓0	0.1648∓0	0.4868∓0
	Kluger	3759∓0	0.2107∓0	0.1824∓0	0.3155∓0
	OCoClus	**0∓0**	**1∓0**	**1∓0**	**1∓0**
Synthetic-2	Li	19528.33∓1568.05	0.748∓0.022	0.2624∓0.0092	0.4968∓0.0177
	Lucchese_{t_1}	353∓0	0.7644∓0	0.5985∓0	0.711∓0
	Lucchese_{t_2}	40078∓0	0.7483∓0	0.326∓0	0.5425∓0
	Lucchese_{t_3}	40078∓0	0.7483∓0	0.326∓0	0.5425∓0
	Dhillon	31426∓0	0.9106∓0	0.3627∓0	0.5001∓0
	Kluger	34382∓0	0.9653∓0.001	0.1744∓0.004	0.2828∓0.0036
	OCoClus	**0∓0**	**1∓0**	**1∓0**	**1∓0**
Synthetic-3	Li	1530.4∓923.1	0.0152∓0.0554	0.0395∓0.0355	0.3466∓0.1857
	Lucchese_{t_1}	0∓0	0.1813∓0	0.4031∓0	0.6462∓0
	Lucchese_{t_2}	1000∓0	0∓0	0.0003∓0	0.4767∓0
	Lucchese_{t_3}	1000∓0	0∓0	0.0003∓0	0.4767∓0
	Dhillon	2200∓0	-0.0123∓0	0.1565∓0	0.3856∓0
	Kluger	1000∓0	0.0835∓0	0.2935∓0	0.4576∓0
	OCoClus	**0∓0**	**1∓0**	**1∓0**	**1∓0**

al.[9], we use three configurations (t_1, t_2 and t_3) of noise threshold parameters to assess its clustering result when the noise values change. The configuration t_1 uses the object noise threshold $\epsilon_I = 0$ and attribute noise threshold $\epsilon_J = 0$. For the last two configurations t_2 and t_3, the noise values are the same used in Lucchese et al.[9]. The configuration t_2 uses $\epsilon_I = 0.5$ and $\epsilon_J = 0.8$, while configuration t_3 uses $\epsilon_I = 1$ and $\epsilon_J = 1$.

Table 2 shows respectively the average and standard deviation from the evaluation metrics for each method and synthetic dataset. It can be seen that OCoClus obtained the best score result in all synthetic datasets; hence, it finds the embedded overlapped co-clusters. Lucchese_{t_1} obtained the second best result while the other two configurations obtained the same score values because they identified the same co-clusters. The method proposed by Li[7] obtained the worst result among the methods. This happens because the method sometimes does not converge to any co-cluster which makes its overall result worse than or close to the non-overlapped methods. Besides, it can be seen in Table 2 that the baseline methods do not find the real number of co-clusters once their evaluation scores do not reach the best value. Regarding the non-overlapped methods, the method proposed by Kluger et al.[6] shows the worse overall co-clustering result. Meanwhile, the method of Li[7] improved slightly its overall clustering result in *synthetic-2* dataset compared to *synthetic-1* and *synthetic-3*. However, it does not overcome the clustering results of the non-overlapped approaches at all.

In summary, it can be seen in Table 2 that OCuClus outperformed the baseline methods in all evaluation metrics. Such a result occurs because OCoClus identifies all *global* and *local* co-clusters while the baselines fail to find both co-cluster structures correctly in the data. The baselines focus on the *global* non-overlapped and overlapped structures. Regarding the baseline methods, using the constraint t_1 in the work of Lucchese et al.[9], this configuration generated the best clustering result. However, considering the other two constraints, we notice that they do not improve the clustering result. Furthermore, the method proposed by Li [7] shows an overall worse clustering result, even though it improved its performance in the synthetic-2 dataset but not enough to overcome all methods. The methods of Kluger et al.[6] and Dhillon[3], in general, obtained stable results in comparison with Li[7].

5.2 Real application scenario

In this section, we used OCoClus on two real datasets to demonstrate its general utility. We set the noise thresholds ϵ_I and ϵ_J to the minimum value, and this means that we are not allowing any attribute or object to be added as noise in the co-clusters. With this parameter control, it is possible to have a better understanding of the co-cluster structure. In fact, OCoClus finds *pure* co-clusters when the noise thresholds are set to the minimum value (zero); this means that all attributes that occur in all objects do not include the presence of noise.

Music Annotation. The left side of Figure 1 shows the bitmap of the CAL500 dataset, and the right side shows the bitmap of the OCoClus result. Similar to Li[7], the question is to identify song sets that share similar annotations. Moreover, we are interested in finding which are the common annotations that distinguish each song set. This task can enhance our perception of the relationship between songs and annotations and therefore it can be applied to music retrieval and recommendation system. We used OCoClus in the processed dataset and the main co-clusters are highlighted in the right side of Figure 1.

OCoClus identified three main levels which are within the red lines and four main co-clusters. The two larger *global* co-clusters have the size 150×13 and 100×12. Further, looking into these two co-clusters we found such annotations as "NOT-Song-Fast_Tempo", "NOT-Emotion-Angry-Agressive", "NOT-Song-Heavy_Beat" and "NOT-Emotion-Bizarre-Weird" for the first co-cluster, and the "Song-Fast_Tempo", "NOT-Emotion-Angry-Agressive", "Song-Heavy-_Beat" and "Song-High_Energy" for the second co-cluster. The first co-cluster includes songs such as "For you and I" by 10cc, "Three little birds" by Bob Marley and The Weilers, and "I'll be your baby tonight" by Bob Dylan. In the second co-cluster includes songs such as "Trapped" by 2pac, "Dirty deeds done dirt cheap" by AC/DC and "Livin on a prayer" by Bon Jovi. Considering the song attributes in the first cluster, it can be seen that songs are formed by a slow rhythm and without strong beats. The second cluster characterized songs with strong beats and a fast rhythm. Therefore, the method identified clusters with opposite characteristics, showing two groups of users with different preferences.

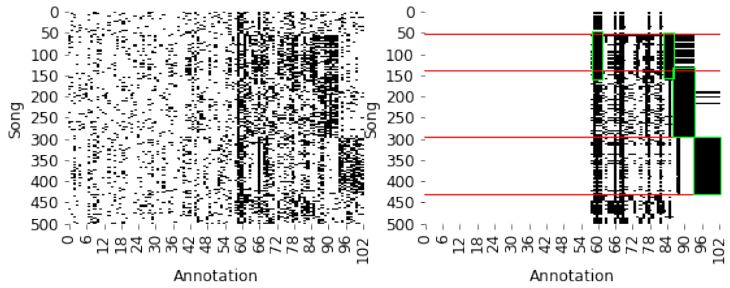

Fig. 1: CAL500 Music dataset. The left side shows the bitmap of the binary annotation matrix where objects represent songs and attributes represent annotations (black=presence; white=absence). The right side shows the bitmap of the identified co-clusters. Objects and attributes are ordered in the same way in both figures just for visualization purpose. (best seen in color)

Considering the *local* patterns, for instance, OCoClus finds a co-cluster of songs as "Summertime" by Dj Jazzy Jeff and The fresh prince, "Sunset 138 bpm remix" by Dj Markitos and "Teenage shutdown" by Electric Frankenstein to name few songs that share some annotations as "Song-Texture_Electric", "Song-Fast_Tempo" and "Song-High_Energy". The fourth *local* co-cluster consists of 45 songs and 7 music annotations. It characterizes a group formed by songs with drums, men on vocals, with electronic and acoustic parts. We identified the music genres like Rock, Pop music, Pop rock, and Alternative rock in this co-cluster. For instance, to name a few songs, this cluster has the "Soul and Fire" by Sebadoh, "Clocks" by Coldplay, "Tubthumping" by Chumbawamba, "Last Goodbye" by Jeff Buckley, "November Rain" by Guns N 'Roses, and "Wonderful Tonight" by Eric Clapton. Thus, it can be seen that OCoClus is useful for finding overlapped and non-overlapped co-clusters that identify the relationship between songs based on the semantic annotations.

Coronavirus Information. Table 3 shows the co-clusters with its attributes and the number of objects. The union of clusters G8 and G9 show the cluster of older people with a total of 1137. Attribute *Senior_3* aggregates people 80 years old or above, and *Senior_2* aggregates people from 70 to 79 years old. Once the *Senior_3* and heart disease attributes appear in G8, they are relevant to form this cluster with 913 people and no other attributes have improved the cost function for it. The G9 cluster has 224 people who died in August, which also happened in the G6 cluster (467 people) during July. These two months mark the peak of the winter season in the region where these people lived.

The clusters G2, G3, G4, G7, and G10 can be seen as a group of people who had at least one main symptom of Covid-19 associated with some comorbidity. The non-overlapped co-clusters are the first 10 groups in Table 3. We notice that the Male attribute is present in two clusters (G1 and G3) regarding the top four. Cluster G1 identifies 1174 men who experienced the three main symptoms

Table 3: Description of the clusters in the CV-19 dataset.

Clusters	Attributes	Number of objects
G1	Dyspnea, Cough, Male, Fever	1174
G2	Dyspnea, Other_Comorbidities, Female	1066
G3	Heart_Disease, Dyspnea, Male	1272
G4	Cough, Diabetes	1307
G5	Fever, PORTO ALEGRE - R10, PORTO ALEGRE	631
G6	Other_Comorbidities, Infected_July, Death_July	467
G7	Other_Symptoms, Dyspnea, Female	569
G8	Senior_3, Heart_Disease	913
G9	Senior_2, Death_August, Infected_August	224
G10	Fever, Other_Comorbidities, Male	769
G11	Dyspnea, Heart_Disease, Cough, Fever, Male	539
G12	Dyspnea, Cough, Fever, Diabetes, Male	410
G13	Dyspnea, Cough, Diabetes, Other_Comorbidities, Female	258
G14	Dyspnea, Other_Comorbidities, Other_Symptoms, Female	365
G15	Dyspnea, Cough, Fever, Heart_Disease, Diabetes, PORTO ALEGRE - R10, PORTO ALEGRE, Male	39
G16	Dyspnea, Cough, Fever, Heart_Disease, Diabetes, Male	241
G17	Dyspnea, Cough, Fever, Heart_Disease, Diabetes, Senior_3, Male	41
G18	Dyspnea, Cough, Fever, Heart_Disease, Senior_3, Male	135

of Covid-19. Meanwhile, the Female attribute is present in one cluster regarding the top four. It identifies 1066 women who presented Dyspnea and other comorbidities as main attributes for this group.

The identified overlapped co-clusters show details that are overlooked in disjoint co-clusters and these co-clusters are the last 8 groups (G11 - G18) in Table 3. For instance, cluster G11 identifies a group of 539 men that felt symptoms as dyspnea, fever, and cough and had heart disease problem. In the same way, cluster G12 identifies 410 men with symptoms as in G1, but now it has those with diabetic issues. In comparison, the overlapped cluster G13 identifies a group of 258 women with cough symptoms in combination with diabetes and other comorbidities. Cluster G14 represents another pattern since it identifies 365 women with dyspnea and other symptoms in combination with other comorbidites.

Groups G11 and G12 are examples of *global* overlapped co-clusters, while the groups G15 and G17 are two different examples of *local* co-clusters. Cluster G15 represents a group of 39 men from Porto Alegre region and lived in the capital, where they all felt the main symptoms of covid-19 and who had heart disease and diabetes problems. For group G17, characterizes a group of 41 older men over 80 years of age who had the main symptoms and who had heart disease and diabetes. Regarding the *local* co-clusters, it can be seen that such co-clusters identify detailed patterns that are overlooked by *global* co-clusters. The experiment with the Covid-19 dataset is an example of a real problem related

to data analysis complexity. Thus, it can be seen that each co-cluster reveals a meaning pattern according to the information granularity.

6 Conclusion and Future Works

We proposed *OCoClus*, a new non-exhaustive overlapped co-clustering method for binary data, designed for general purpose analysis. OCoClus is based on the detection of co-occurrence of objects and attributes, to identify *global* and *local* co-clusters that overlap. Besides that, when there are no overlapped patterns in the dataset, OCoClus can identify the non-overlapped co-clusters. Furthermore, it is driven by a cost function to automatically identify the number of co-clusters. We performed experiments on synthetic and real data that demonstrates the efficacy and utility of our proposed method.

OCoClus found all embedded co-clusters in the synthetic datasets used as ground-truth, proven by the fact that OCoClus obtained the maximum score in the evaluation metrics. Such a result shows that OCoClus outperformed the limitations of the baseline methods. Nevertheless, we prove the usefulness of our method in two real datasets where we show that OCoClus identified co-clusters that can represent meaningful patterns. We highlight the fact that the obtained results are interesting to propose new specialized systems that use the identified co-clusters as input to decision support systems.

Like any work in the literature, our approach also has space for improvements as future research. First, the number of co-clusters is driven by a cost function regarding the number of objects and attributes. Then, the method tends to find rectangular clusters which may generate patterns with few attributes for big data mining. Second, an interesting research direction is to adapt the method to deal with heterogeneous data. Third, it may be interesting to set the noise thresholds ϵ_I and ϵ_J in a data-driven way. Finally, identifying uncorrelated co-clusters in the data matrix is another interesting direction to improve the method.

Acknowledgements. This work has been partially supported by CAPES (Finance code 001), CNPQ, FAPESC (Project Match - co-financing of H2020 Projects - Grant 2018TR 1266), and the European Union's Horizon 2020 research and innovation programme under GA N. 777695 (MASTER). The views and opinions expressed in this paper are the sole responsibility of the author and do not necessarily reflect the views of the European Commission.

References

1. Affeldt, S., Labiod, L., Nadif, M.: Ensemble block co-clustering: a unified framework for text data. In: Proceedings of the 29th ACM International Conference on Information & Knowledge Management. pp. 5–14 (2020)
2. Brunialti, L.F., Peres, S.M., da Silva, V.F., de Moraes Lima, C.A.: The binovnmtf algorithm: Overlapping columns co-clustering based on non-negative matrix trifactorization. In: 2017 Brazilian Conference on Intelligent Systems (BRACIS). pp. 330–335. IEEE, Uberlandia, Brazil (2017)

3. Dhillon, I.S.: 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. pp. 269–274. ACM, Association for Computing Machinery, New York, NY, USA (2001)
4. Fu, Q., Banerjee, A.: Bayesian overlapping subspace clustering. In: 2009 Ninth IEEE International Conference on Data Mining. pp. 776–781. IEEE (2009)
5. Hartigan, J.A.: Direct clustering of a data matrix. Journal of the american statistical association **67**(337), 123–129 (1972)
6. Kluger, Y., Basri, R., Chang, J.T., Gerstein, M.: Spectral biclustering of microarray data: coclustering genes and conditions. Genome research **13**(4), 703–716 (2003)
7. Li, G.: Generalized co-clustering analysis via regularized alternating least squares. Computational Statistics & Data Analysis **150**, 106989 (2020)
8. Lucchese, C., Orlando, S., Perego, R.: A generative pattern model for mining binary datasets. In: Proceedings of the 2010 ACM Symposium on Applied Computing. pp. 1109–1110. ACM (2010)
9. Lucchese, C., Orlando, S., Perego, R.: A unifying framework for mining approximate top-k binary patterns. IEEE Transactions on Knowledge and Data Engineering **26**(12), 2900–2913 (2013)
10. Lutov, A., Khayati, M., Cudré-Mauroux, P.: Accuracy evaluation of overlapping and multi-resolution clustering algorithms on large datasets. In: 2019 IEEE International Conference on Big Data and Smart Computing (BigComp). IEEE, Kyoto, Japan (2019)
11. Madeira, S.C., Oliveira, A.L.: Biclustering algorithms for biological data analysis: a survey. IEEE/ACM transactions on computational biology and bioinformatics **1**(1), 24–45 (2004)
12. Padilha, V.A., Campello, R.J.: A systematic comparative evaluation of biclustering techniques. BMC bioinformatics **18**(1), 1–25 (2017)
13. Pio, G., Ceci, M., D'Elia, D., Loglisci, C., Malerba, D.: A novel biclustering algorithm for the discovery of meaningful biological correlations between micrornas and their target genes. BMC bioinformatics **14**(S7), S8 (2013)
14. Role, F., Morbieu, S., Nadif, M.: Coclust: a python package for co-clustering. Journal of Statistical Software **88**(1), 1–29 (2019)
15. Vlachos, M., Dünner, C., Heckel, R., Vassiliadis, V.G., Parnell, T., Atasu, K.: Addressing interpretability and cold-start in matrix factorization for recommender systems. IEEE Transactions on Knowledge and Data Engineering **31**(7), 1253–1266 (2018)
16. Whang, J.J., Rai, P., Dhillon, I.S.: Stochastic blockmodel with cluster overlap, relevance selection, and similarity-based smoothing. In: 2013 IEEE 13th International Conference on Data Mining. IEEE (2013)
17. Xu, R., Wunsch, D.: Survey of clustering algorithms. IEEE Transactions on neural networks **16**(3), 645–678 (2005)
18. Zheng, Y., Hu, R., Fung, S.f., Yu, C., Long, G., Guo, T., Pan, S.: Clustering social audiences in business information networks. Pattern Recognition **100**, 107126 (2020)
19. Zhu, H., Mateos, G., Giannakis, G.B., Sidiropoulos, N.D., Banerjee, A.: Sparsity-cognizant overlapping co-clustering for behavior inference in social networks. In: 2010 IEEE International Conference on Acoustics, Speech and Signal Processing. pp. 3534–3537. IEEE (2010)