Co-clustering Numerical Data under User-defined Constraints

Ruggero G. Pensa[1,3] Jean-Francois Boulicaut[2] Francesca Cordero[1,4]
Maurizio Atzori[3]

1. Dept. Computer Science, University of Torino, I-10149 Torino, Italy
2. INSA-Lyon, LIRIS CNRS UMR5205, F-69621 Villeurbanne, France
3. Pisa KDD Laboratory, ISTI-CNR, I-56124 Pisa, Italy
4. Dept. Clinical and Biological Sciences, University of Torino, I-10043 Orbassano, Italy

Abstract

In the generic setting of objects × attributes matrix data analysis, co-clustering appears as an interesting unsupervised data mining method. A co-clustering task provides a bi-partition made of co-clusters: each co-cluster is a group of objects associated to a group of attributes and these associations can support expert interpretations. Many constrained clustering algorithms have been proposed to exploit the domain knowledge and to improve partition relevancy in the mono-dimensional clustering case (e.g., using the must-link and cannot-link constraints on one of the two dimensions). Here, we consider constrained co-clustering not only for extended must-link and cannot-link constraints (i.e., both objects and attributes can be involved), but also for interval constraints that enforce properties of co-clusters when considering ordered domains. We describe an iterative co-clustering algorithm which exploits user-defined constraints while minimizing a given objective function. Thanks to a generic setting, we emphasize that different objective functions can be used. The added value of our approach is demonstrated on both synthetic and real data. Among others, several experiments illustrate the practical impact of this original co-clustering setting in the context of gene expression data analysis, and in an original application to a protein motif discovery problem.

1 Introduction

Large datasets that record numerical values of given attributes for given objects (say $objects \times attributes$ matrices) are quite common and supporting data mining processes on them remains challenging. For instance, we will consider gene expression datasets that record gene expression values for given genes in given biological samples (see, e.g., microarray data analysis in [2] and the example dataset X_r in Fig. 1). In X_r, attributes may denote biological samples and each object may be associated to one particular gene. For instance, the gene expression value for Gene 2 in Experiment 3 would be 5. Exploratory data analysis processes are often based on clustering methods to get insights about global patterns that hold in the data. A clustering task provides a partition of objects and/or of attributes such that a grouping quality measure is optimized. Many techniques however suffer from the lack of explicit characterization for clusters, and this has motivated the research on co-clustering [34, 19, 4]. The objective of co-clustering is to compute co-clusters that are associations of (possibly overlapping) sets of objects with sets of attributes. A co-clustering algorithm computes simultaneously linked partitions on both row and column dimensions. An example of a bi-partition in X_r would be $\{\{1,2,3,4,5\},\{6,7\}\}$ for objects, $\{\{1,4,5\},\{2,3\}\}$ for attributes. It indicates that the characterization of objects from $\{1,2,3,4,5\}$ is that they tend to share similar values for attributes from $\{1,4,5\}$. Also, attributes in $\{2,3\}$ can be used to characterize objects in $\{6,7\}$. From this perspective, it is clear that performing a separate clustering on objects and attributes is substantially different from performing simultaneous clusterings of objects and attributes. In the former case, each partition is computed independently from the other one. In the

$$X_r = \begin{bmatrix} 3 & 0 & 0 & 2 & 4 \\ 1 & 4 & 5 & 1 & 2 \\ 4 & 1 & 0 & 4 & 5 \\ 2 & 0 & 1 & 3 & 4 \\ 1 & 4 & 5 & 1 & 2 \\ 1 & 4 & 6 & 0 & 0 \\ 0 & 5 & 6 & 0 & 0 \end{bmatrix}$$

Figure 1: A toy example X_r

latter, the two partitions are linked. Two different (and suboptimal) partitions on one dimension of the matrix correspond, in general, to two different (and suboptimal) partitions on the other dimension.

Unconstrained co-clustering has been studied in the context of co-occurrence or contingency tables, e.g. in *documents* × *words* matrices, where the goal is to find similar documents and their interplay with word clusters [19]. Co-clustering and the related task of bi-clustering [12, 30] have been well studied in the context of gene expression data analysis. They indeed provide valuable information about putative regulation mechanisms and biological functions. Intuitively, a co-cluster extracted from a gene expression dataset denotes a set of genes with similar expression profiles along its associated set of biological samples.

We are interested in new co-clustering methods for enforcing the relevancy of computed bi-partitions in general and their application to gene expression data analysis in particular. Given a (co-)clustering algorithm, the analyst has generally a weak control on the clusters he/she obtains. Typically, he/she can decide for ad-hoc parameter settings which are quite operational and conceptually far from the declarative specification of desired properties. A co-clustering algorithm tries to optimize an objective function (e.g., Goodman-Kruskal's τ coefficient in [34] or the loss of mutual information in [19]) but it may also ensure that some user-defined constraints are satisfied (e.g., the fact that some objects and/or attributes have to be together or not). However, enforcing constraints can lead to lower values for the considered objective functions. Furthermore, it is clear that combining objective function optimization and the satisfaction of other user-defined constraints is challenging. The last 5 years, several researchers have studied "single-sided" constrained clustering for rather simple types of user-defined constraints, mainly the so-called must-link and cannot-link constraints [38, 7, 27, 8, 17, 16, 9]. To the best of our knowledge, constrained co-clustering has been rarely studied. We are only aware of [32, 33]. In [32] (whose an extended version is [33]), the authors address constrained co-clustering when at least one of the dimensions is ordered and when interval constraints are defined w.r.t. orders. A typical application for interval constraints concerns kinetic gene expression data analysis. In this case, objects denote gene expression level measurements performed for successive time points. For many organisms, we see that during their life cycle, groups of genes can be activated and then inhibited, being somehow characteristics of development stages. Using interval constraint supports the discovery of such groups from experimental data.

In this paper, we propose a constraint-based co-clustering approach which is different from [32, 33]. First, these articles concern only 0/1 data mining. More importantly, we propose here to work directly on the data, i.e., without any postprocessing of collections of local patterns that have to be computed beforehand (See Section 2). Our new method builds a bi-partition that satisfies the user-defined constraints while optimizing objective functions based on sum-squared residues [13]. Furthermore, this paper is a significant extension of [31]. The genericity of the method is here emphasized and different objective functions are considered. More details are given on the algorithms and the related unconstrained co-clustering framework from [4]. Finally, the empirical validation has been revisited and considerably extended. The rest of the paper is organized as follows. Section 2 discusses the related work. Section 3 is dedicated to the problem setting for constrained co-clustering. Section 4 describes our generic algorithms where the considered co-clustering objective functions are left undefined. Section 5 studies two useful objective functions based on residues and thus concrete instances of the algorithms introduced in Section 4. Section 6 is an empirical study about the added-value of our framework on both synthetic and real-life data. It includes gene expression data analysis tasks and on a protein motif discovery problem. Section 7 briefly concludes.

2 Related work

Constrained co-clustering is a new approach to gene expression data analysis. To the best of our knowledge, only the previous work [32] extended in [33] has addressed the problem of co-clustering under user-defined constraints. We consider that optimizing the objective function and enforcing the number of co-clusters are more or less implicit constraints. In other terms, we would say that an algorithm like CoCluster [19] just performs co-clustering and not constrained co-clustering.

Furthermore, co-clustering is related to bi-clustering [30], and the two terms are sometimes used interchangeably. Bi-clustering algorithms search for an a priori unspecified number of possibly overlapping bi-clusters. A bicluster is a set of objects and an associated set of attributes. Each object belonging to a bicluster is strongly correlated to any other object belonging to the same bicluster w.r.t. the set of associated attributes. A set of biclusters does not provide a model of the data, since they capture local associations rather than providing a global insight of the data. In other terms, we may say that it does not provide a clustering structure but this is related to subspace clustering techniques. On the other hand, such algorithms provide meaningful groups that may possibly diverge from the global trend of the data. Co-clustering algorithms, instead, look for a generally pre-specified number of co-clusters that form a bi-partition, i.e., a partition of objects which is strongly related to a partition of attributes. Each cluster of objects is such that each object belonging to it is strongly and differently related to any other objects belonging to the same cluster w.r.t. *all* clusters of attributes, and vice versa. Co-clustering provides a descriptive model of the data, where the partition of objects is described by the partition of attributes and vice versa. Alternatively, the result of a co-clustering can be viewed as a collection of bi-clusters that satisfy some properties, e.g., the condition that all bi-clusters constitute a bipartition in the sense of a co-clustering result.

Let us discuss further this question in the context of bioinformatics: using bi-clustering or co-clustering depends on the biological question the experimenter wants to answer. Considering gene expression analysis, if the final goal of one experiment is finding the module networks characterizing the cell activity [36], bi-clustering can be considered as an optimal strategy. Network modules consist in a set of co-expressed genes characterized by the same cis-regulatory binding sites (also called regulatory modules) in the upstream region of those genes. Each regulatory module contains a dispersed collection of short sequences, each of which specifically binds to a particular transcription factor protein. As a result, the network modules are composed by a set co-regulated genes and all their regulators. The main feature that links each gene in a module is the expression values that must be very similar in a specific experimental condition. What we expect here is not to cluster all the genes involved in this experiment, but to find groups of genes that are very similar among them. In other words, we look for local signatures of gene expression. On another hand, if we consider a microarray experiment for studying the development of a specific organism (see, e.g., [20]), the final goal is to obtain some expression pattern groups associated to each developmental stage. In such a case, the biologist looks for a global pattern of the gene expression. From this point of view, co-clustering provides a useful and global descriptive model of the data.

We now discuss the previous work on (constrained) co-clustering. Then, we consider the relationship to (constrained) bi-clustering. Finally, we briefly compare our approach to previous work on constrained clustering.

2.1 Co-clustering Many co-clustering methods have been developed, possibly dedicated to gene expression data analysis. Kluger et al. [28] propose a spectral co-clustering method. First, they perform an adequate normalization of the dataset to emphasize co-clusters if they exist. Then, they consider that the correlation between two columns is better estimated by the expression level mean of each column w.r.t. a partition of the rows. The bi-partition is computed by the algebraic eigenvalue decomposition of the normalized matrix. Their algorithm critically depends on the normalization procedure. Dhillon et al. [19] and Robardet et al. [34] have considered the two searched partitions as discrete random variables whose association must be maximized. Different measures can be used. Whereas COCLUSTER [19] uses the loss in mutual information, BI-CLUST [34] uses Goodman-Kruskal's τ coefficient to evaluate the link strength between the two variables. In both algorithms, a local optimization method is used to optimized the measure by alternatively changing a partition when the other one is fixed. The main difference between these two approaches is that the τ measure is independent of the number of co-clusters and thus BI-CLUST can automatically determine the number of co-clusters. Lazzeroni et al. [29] propose to consider each matrix value as a sum of variables. Each variable represents a particular phenomenon in the data and corresponds to a co-cluster. In each co-cluster, column or row values are linearly correlated. Then, the method consists in determining the model minimizing the Euclidean distance between the matrix and the modeled values. This method is similar to the eigenvalue decomposition used in [28] without the orthogonal constraint on the computed variables. Notice also that, the problem of matrix partitioning has been investigated in other contexts, such as for parallel processing purposes [23], where the goal is to provide a block diagonal structure of sparse matrices in order to parallelize some common and frequent operations like matrix-vector products. A new and significant result has been presented in [1]. The authors show that the co-clustering problem is NP-hard, and they propose a constant-factor approximation algorithm for any norm-based objective functions. Concerning recent contributions to co-clustering, they have focused on hierarchical [25], overlapping [18] and Bayesian [37] models. Recently, Banerjee et al. have proposed in [4] a co-clustering setting based on matrix approximation. The approximation error is measured using a large class of loss functions called Bregman divergences. They introduce a meta-algorithm whose special cases include the algorithms from [19] and [13]. We come back on such a meta-algorithm in Section 4 when introducing our iterative approach to constrained co-clustering.

Let us now consider the previous work on co-clustering under user-defined constraints. In [32, 33], a local-to-global approach has been designed to support the computation of bi-partitions given interval constraints. This is performed by postprocessing collections of local patterns (e.g., closed sets) extracted from 0/1 datasets. The basic idea is to translate the interval constraint into a relaxation which can be enforced in the collection of local patterns. Then, it uses a k-means-based approach to obtain a partition of local patterns. Finally, such a partition can be post-processed to determine a co-clustering structure over the data. It also discussed possibilities to process co-clustering counterparts of the popular must-link and cannot-link constraints introduced for standard clustering. The main difference of our contribution w.r.t. this previous work, is that we compute co-clustering directly by alternatively computing clusters on columns and rows w.r.t. a common objective function. A second difference is that we ensure, when needed, the satisfaction of the interval constraint on the computed bi-partition while this was not the case for the method in [32, 33]. Third, our proposal works on numerical data and it it is not limited to 0/1 data. Last, but not least, we significantly improve must-link and cannot-link constraint processing over both sets of objects and attributes.

2.2 Bi-clustering In the context of gene expression data analysis, several authors have considered the computation of potentially overlapping local patterns that they call bi-clusters (see [30] for a survey). Ihmels et al. [26] propose a simple algorithm which builds in two steps a single association called a bi-cluster starting from a column set. First, they consider that the rows having a high score (greater than a threshold on the normalized matrix) on these columns belong to the bi-cluster. Then, they use the same principle to increase the original column set. In [12], Cheng et al. propose a so-called bi-clustering approach for gene expression data. They define a bi-cluster as a subset of rows and subset of columns with a low mean squared residue. When the measure is equal to 0, the bi-cluster contains rows having the same value on the bi-cluster columns. When the measure is greater than 0, one can remove rows or columns to decrease the value. Thus the method consists in finding maximal size bi-clusters such that the measure is inferior to a threshold. Various heuristics can be used for this purpose. The same definition of residue is used in [13] to define the objective function which is also used in our current proposal. Authors propose two different residue measures, and show that the one proposed by Cheng et al. fits better to gene expression data analysis. Then, they introduce their co-clustering algorithm which optimizes the sum-squared residue functions. This approach has been the starting point for our contribution.

2.3 Constrained clustering Constrained clustering is a recent and active research domain (see, e.g., [9] for a state-of-the-art survey). It has been mainly studied in the context of semi-supervised learning for which an alternative approach is the so-called metric-based method (i.e., learning a metric considering labeled data before applying standard clustering). Semi-supervised clustering can support classification tasks when labeled data are limited and/or expensive to collect. A solution is to use the knowledge given by available labeled instances within a clustering algorithm. In [38], a simple adaptation of k-means which enforces must-link and cannot-link constraints during the clustering process is described. [7] proposes a constrained clustering approach which uses labeled data during the initialization and clustering steps. An example of metric-based approach is given in [27]. Notice that [10] integrates both constraint-based and metric-based approaches in a k-means-like algorithm. In [8], the authors propose a probabilistic model for semi-supervised clustering, which also combines the two approaches. Other related work focuses on constraint feasibility on a k-means-like scheme [17], and on an agglomerative hierarchical clustering scheme [16]. In such a related work, the goal is to improve accuracy in classification when only few instances are labeled, whereas our goal is not to support prediction: we are working within an unsupervised framework. For us, constraints are used to specify user expectation, also called subjective interestingness, and thus to improve the a priori relevancy of computed groups. Interval constraints are not pair-wise constraints (such as must-link and cannot-link): the analyst specifies whether he/she wants intervals or not, without knowing if a particular object x is in the same cluster than another object y. Moreover, constraints can be applied both on objects and attributes. Other applications of constrained clustering are the so-called sensor network and k-anonymity problems. In both applications, a possible solution is to find compact clusters containing a balanced number of objects. For instance, the discovery of balanced clusters is considered in [5, 6]. [21] describes an algorithm which finds an unspecified number of compact clusters under the combination of minimum significance constraints and minimum variance constraints. Similarly to such approaches, our algorithms enable to define constraints which are more related to the shape of the clusters rather than to pairs of objects.

3 A constrained co-clustering setting

Let $X \in \mathbb{R}^{m \times n}$ denote a data matrix. In the rest of the paper, the dataset to be mined is the matrix X, and we always talk about rows and columns instead of objects and attributes. Let x_{ij} be the element corresponding to row i and column j. For instance, x_{ij} might contain the expression level of gene i in the experimental condition j. Let $x_{i\cdot}$ and $y_{\cdot j}$ denote the vectors associated to, respectively, row i and column j.

A co-clustering $C^{k \times l}$ over X produces simultaneously a set of $k \times l$ co-clusters (a partition C^r into k groups of rows associated to a partition C^c into l groups of columns). To obtain a first quality criterion, we first try to optimize a certain objective function.

Definition 3.1. (optimization constraint) *Let us assume an objective function $f(X, C^{k \times l})$, an optimization constraint $c_{opt}(f, X, C^{k \times l})$* is satisfied iff $C^{k \times l} = \text{argmin}_{\phi \in \mathcal{L}_{\mathcal{C}^{k \times l}}} f(X, \phi)$ *where $\mathcal{L}_{\mathcal{C}^{k \times l}}$ is the the collection of all possible co-clusterings.*

Some examples of objective functions are the Goodman-Kruskal's τ coefficient and the loss of mutual information [34, 19]. In this paper, we use the sum-squared residue function introduced in [13] and we consider two different definitions for residues. For computational feasibility reasons, co-clustering algorithms always relax these optimization constraints, e.g., using local optimization heuristics.

One can be interested in other kinds of constraints which are now defined.

Definition 3.2. (must-link/cannot-link) *If rows i_a and i_b (resp. columns j_a and j_b) are involved in a **must-link** constraint, denoted $c_=(i_a, i_b)$ (resp. $c_=(j_a, j_b)$), they must be in the same cluster of $C^r = r_1, \ldots, r_k$ (resp $C^c = c_1, \ldots, c_k$). If rows i_a, i_b (resp. columns j_a and j_b) are involved in a **cannot-link** constraint, denoted $c_{\neq}(i_a, i_b)$ (resp. $c_{\neq}(j_a, j_b)$), they cannot be in the same cluster of $C^r = r_1, \ldots, r_k$ (resp $C^c = c_1, \ldots, c_k$).*

Such types of constraints have been studied in the context of semi-supervised clustering [10]. It is here generalized in order to apply them on both row and column sets. In a gene expression matrix, it is then possible to exploit the knowledge about genes and/or experimental conditions. For example, if we know that gene i_a and gene i_b have the same function (say F) in the biological process, we can enforce a must-link constraint between these two genes to focus the search for co-clusters associating genes having such a function F. We could also add some cannot-link constraints to avoid associations between experimental conditions which we want to separate, e.g., avoiding to mix conditions that are related to different stages of a disease.

Let us now assume that a real value $s_c(j)$ (resp. $s_r(i)$) is associated to each column j (resp. row i). Then we have $s_r : \{1, 2, \ldots, m\} \to \mathbb{R}$ and $s_c : \{1, 2, \ldots, n\} \to \mathbb{R}$. For example, $s_c(j)$ (resp. $s_r(i)$) could be a temporal or spatial measure related to j (resp. i). In microarray data, $s_c(j)$ might be the sampling time related to the DNA chip (say experiment) j. Another example would be to consider $s_r(i)$ as a measure of the absolute spatial position of a gene i in the DNA sequence of the studied organism. The two functions s_r and s_c enable to define an order $\preceq$ over the set of columns and/or rows. Indeed, we say that $j_a \preceq j_b$ iff $s_c(j_a) \leq s_c(j_b)$. In the rest of the paper, we say that, if a function s_c exists, then all the elements j are ordered, i.e., $\forall j_a, j_b$ s.t. $j_a < j_b$, $s_c(j_a) \leq s_c(j_b)$ (the same property holds for rows).

It appears interesting to search for co-clusters which are coherent with the order defined by functions s_r and s_c. For instance, let us assume that we are interested in the different development stages of a given living organism, and that we want to discover those genes that are mainly involved in each stage. Therefore, we can look for clusters whose elements are contiguous w.r.t. time, i.e., enforcing an interval constraint.

```
Algorithm 1: GenCoClust($X$,$k$,$l$)
  Input: Data matrix $X$, $k$, $l$
  Output: Partitions $C_c$, $C_r$
  Initialize $C_c^0$ and $C_r^0$;
  $t = 0$;
  repeat
      With respect to co-clustering $C_c^t$ and $C_r^t$, compute the matrix approximation;
      Hold the column clustering $C_c^t$ fixed, and find a better row co-clustering, say, $C_r^{t+1}$;
      Hold the row clustering $C_r^{t+1}$ fixed, and find a better column co-clustering, say, $C_c^{t+1}$;
      $t = t + 1$;
  until convergence ;
```

Definition 3.3. (interval constraint) *If an order ($\preceq$) is defined over the column set (resp. row set), an* ***interval*** *constraint over this set, denoted $c_{int}(C^c)$, specifies that each cluster in C^c has to be an interval: $\forall c \in C^c$, if $j_a, j_b \in c$ then $\forall j_c$ such that $j_a \preceq j_c \preceq j_b$, $j_c \in c$.*

In general, the satisfaction of the must-link, cannot-link and interval constraints decreases the theoretical optimum of the objective function. We want a co-clustering algorithm which is able to take into account such constraints while trying to optimize the retained objective function. Notice that the satisfaction of a conjunction of constraints $c_=$, $c_{\neq}$ and c_{int} is not always feasible. For instance, for three objects i_1, i_2, i_3 such that $s(i_1) < s(i_2) < s(i_3)$, the conjunction $c_=(i_1, i_3) \wedge c_{\neq}(i_1, i_2) \wedge c_{int}(C^r)$ can never be satisfied, even though the sub-constraints of this conjunction do not cause any problem. In this paper, we assume that the processed conjunction of constraints is feasible. We refer to [17, 16] for studies on constraint feasibility for both partitioning and hierarchical clustering methods.

4 A generic constrained co-clustering scheme

A large class of objective functions, the so-called Bregman divergences, can be optimized via a meta-algorithm called Bregman Co-clustering [4] (see Algorithm 1). Its principle is quite simple: it alternatively refines row and column clusters, while optimizing an objective function which takes into account both partitions. In the first step, an initialization (e.g., a random initialization) is given for column clustering C_c^0 and row clustering C_r^0. In the next step, a matrix approximation (e.g., by means of clustering centroids) is computed for this pair of partitions. Then, a new column partition is computed while the row partition is kept fixed. Finally, the algorithm holds the column partition fixed, and it updates the column clustering. The last three steps are repeated until a convergence criterium is satisfied. Clearly, our goal is to extend such a general approach to a constrained co-clustering setting.

4.1 Satisfying must-link and cannot-link constraints The transitivity of must-link constraints is a well known property. We can transform a set of must-link constraints over rows into a collection $\mathcal{M}_r = M_1, \dots, M_N$, where each M_i is a set of rows involved by the same transitive closure of must-link constraints. Let us denote $\mathcal{M}_c$ the same set built for columns and let $\mathcal{C}_r$ and $\mathcal{C}_c$ be the sets of cannot-link constraints for rows and columns respectively.

Algorithm 2 enables to co-cluster data when conjunctions of must-link et cannot-link constraints are given. It starts with some initialization (e.g., a random initialization) of partitions $\mathcal{C}_r$ and $\mathcal{C}_c$. During each iteration, the algorithm associates each column (resp. row) to the nearest column (resp. row) cluster which does not introduce any cannot-link violation. If a column (resp. row) is involved in a must-link constraint, the algorithm associates the whole set of columns (resp. rows) involved in the

```
Algorithm 2: GenCoCoClust1($X$,$k$,$l$,$\mathcal{M}_r$,$\mathcal{M}_c$,$\mathcal{C}_r$,$\mathcal{C}_c$)
  Input: Data matrix $X$, $k$, $l$, cannot-link sets $\mathcal{C}_r$ and $\mathcal{C}_c$, collections $\mathcal{M}_r$ et $\mathcal{M}_c$
  Output: Partitions $C_c$, $C_r$
  Initialize $C_c^0$ and $C_r^0$, possibly considering $\mathcal{M}_r$, $\mathcal{M}_c$, $\mathcal{C}_r$ and $\mathcal{C}_c$;
  $t = 0$;
  repeat
      With respect to co-clustering $C_c^t$ and $C_r^t$, compute the matrix approximation;
      foreach column $j$ do
          if $\exists M_v \in \mathcal{M}_c$ s.t. $j \in M_v$ then
              assign all columns in $M_v$ to the closest column cluster s.t. no cannot-link constraint is
              violated;
          else
              assign $j$ to the closest column cluster s.t. no cannot-link constraint is violated;
          end
      end
      Define $C_c^{t+1}$ following the previous assignment step;
      foreach row $i$ do
          if $\exists M_u \in \mathcal{M}_r$ s.t. $i \in M_u$ then
              assign all rows in $M_u$ to the closest row cluster s.t. no cannot-link constraint is
              violated;
          else
              assign $i$ to the closest row cluster s.t. no cannot-link constraint is violated;
          end
      end
      Define $C_r^{t+1}$ following the previous assignment step;
      $t = t + 1$;
  until convergence ;
```

transitive closure of this constraint to the closest column (resp. row) cluster controlling that there is no cannot-link constraint which is violated by this operation. Then the algorithm updates the column clustering $\mathcal{C}_c$ (resp. row clustering $\mathcal{C}_r$) following the assignment schema resulting from the previously described operations. This process is iterated until a convergence criterium is satisfied.

Notice that we do not need to consider constraint satisfiability at the initialization step: it is tackled by the first iteration of the algorithm. A possible improvement would be to enhance the assignment criterion for objects involved in cannot-link constraints. Moreover, we know that the satisfaction of a set of cannot-link constraints for a given number of clusters is an **NP**-complete task [17].

4.2 Satisfying the interval constraint Algorithm 3 enables to solve the satisfaction problem for the interval constraint. For such a constraint, the initialization ($\star$) of partitions has to produce a number l (resp. k) of intervals over columns (resp. rows). Then, the assignment process only considers the frontiers between intervals (where needed). Notice that, contrary to [32, 33], the satisfaction of the interval constraint on the computed bi-partition is here ensured.

In this paper, we do not consider the combination of these two algorithms to process a conjunction of must-link, cannot-link and interval constraints. However, let us sketch research guidelines for this purpose. We first have to ensure that each set $M_r \in \mathcal{M}_r$ (or $M_c \in \mathcal{M}_c$) is an interval. For instance, for a set of objects $\{i_1, i_2, i_3, i_4, i_5\}$, and a set $M_r = \{i_2, i_4\}$, we should include object i_3 into M_r because of the

```
Algorithm 3: GenCoCoClust2($X$,$k$,$l$)
  Input: Data matrix $X$, $k$, $l$
  Output: Partitions $C_c$, $C_r$
  Initialize $C_c^0$ and $C_r^0$ ($\star$);
  $t = 0$;
  repeat
      With respect to co-clustering $C_c^t$ and $C_r^t$, compute the matrix approximation;
      if Interval constraint on rows then
          Hold the column clustering $C_c^t$ fixed, and find a better row co-clustering, say, $C_r^{t+1}$ by
          redefining cluster frontiers;
      else
          Hold the column clustering $C_c^t$ fixed, and find a better row co-clustering, say, $C_r^{t+1}$;
      end
      if Interval constraint on columns then
          Hold the row clustering $C_r^{t+1}$ fixed, and find a better column co-clustering, say, $C_c^{t+1}$;
          by redefining cluster frontiers;
      else
          Hold the row clustering $C_r^{t+1}$ fixed, and find a better column co-clustering, say, $C_c^{t+1}$;
      end
      $t = t + 1$;
  until convergence ;
```

definition of intervals. Then, we need the initialization to produce a partition which takes into account the whole set of constraints (notice again that the satisfaction of a conjunction of cannot-link constraints is a **NP**-complete problem). Finally, it is possible to reuse the strategy described by Algorithm 2 only on the frontiers, following the schema presented in Algorithm 3. Finally, let us emphasize that this approach is intrinsically different than applying existing mono-dimensional constraint-based clustering algorithms alternatively on row and column vectors. In fact, each column (row) reassignment step of the algorithm takes into account the previous row (column) reassignment step through the common objective function. As a consequence, constraints on one dimension can influence the partition on the other dimension.

5 Using sum-squared residues

Our framework for constrained co-clustering can be adapted to many different co-clustering objective functions (see [4] for an in-depth study of such functions). We consider instances that iteratively minimize the sum of squared residues. The loss in mutual information, i.e., the objective function minimized by the co-clustering approach in [19], would be another possibility. This is out of the scope of this paper to discuss this issue further. We decide however to focus on two distinct definitions of residues such that we provide two instances of our generic constrained co-clustering algorithm. These objective functions have been introduced in [13] for gene expression data unconstrained co-clustering.

We look for a partition of a data matrix $X \in \mathbb{R}^{m \times n}$ into k row clusters, and l column clusters. Let I be the set of indices of the rows belonging to a row cluster, and J the set of indices of the columns belonging to a column cluster. The sub-matrix of X determined by I and J is called a *co-cluster*.

Definition 5.1. (residue) *Given an element x_{ij} of X, the residue of x_{ij} in the co-cluster defined by the sets of indices I and J, and whose respective cardinalities are $|I|$ and $|J|$, is given by*

$$h_{ij} \quad = \quad x_{ij} - x_{Ij} - x_{iJ} + x_{IJ} \tag{5.1}$$

$$h_{ij} \quad = \quad x_{ij} - x_{IJ} \tag{5.2}$$

where $x_{IJ} = \frac{\sum_{i \in I, j \in J} x_{ij}}{|I| \cdot |J|}$, $x_{Ij} = \frac{\sum_{i \in I} x_{ij}}{|I|}$, $x_{iJ} = \frac{\sum_{j \in J} x_{ij}}{|J|}$.

The first formulation (Equation 5.1) is the measure designed Cheng and Church [12] for local pattern discovery in gene expression data. The second (Equation 5.2) corresponds to the measure used by Hartigan [22] in one of the earliest co-clustering setting.

Let $H = [h_{ij}] \in \mathbb{R}^{m \times n}$ denote the matrix of residues computed using the previous definitions. The objective function to be minimized is the sum of squared residues [13] computed as follows:

$$||H||^2 = \sum_{I,J} ||h_{IJ}||^2 = \sum_{I,J} \sum_{i \in I, j \in J} h_{ij}^2 \tag{5.3}$$

We can rewrite the residue matrix in a more compact form. Let us introduce the matrices $R \in \mathbb{R}^{m \times k}$ and $C \in \mathbb{R}^{n \times l}$ which are defined as follows: each element (i, r) $(1 \leq r \leq k)$ of R is equal to ${m_r}^{-1/2}$ if i is in co-cluster r (m_r is the number of rows in r), 0 otherwise. Each element (j, c) $(1 \leq c \leq l)$ of the matrix C is equal to ${n_c}^{-1/2}$ if j is in c (n_c being the number of columns in c), 0 otherwise. The residue matrix becomes:

$$H = (I - RR^T)X(I - CC^T) \tag{5.4}$$

for the Cheng and Church's residue, and

$$H = RR^T XCC^T \tag{5.5}$$

for the Hartigan's residue.

The proof of validity of these equations is given in [13]. The authors first demonstrate that $(RR^T X)_{ij} = x_{Ij}$, $(XCC^T)_{ij} = x_{iJ}$ and $(RR^T XCC^T)_{ij} = x_{IJ}$, before showing that Equations 5.4 and 5.5 are correct. They conclude that, if we consider the projections $(I - RR^T)X$ and $RR^T X$ of the matrix X, then $||H||^2$ gives the objective function of k-means for this modified matrix.

Let us now consider our algorithmic contribution. Our approach uses the introduced "ping-pong" technique that processes alternatively by means of a k-means method columns and rows. It means that matrix C is updated only after determining the nearest column cluster for each column (and similarly for rows). Therefore, we propose to decompose the objective function captured by Equation 5.4 in terms of columns. Given $X^P = (I - RR^T)X$, $X^C = (I - RR^T)XC$, and $\hat{X}^P = (I - RR^T)XCC^T = X^C C^T$, we can rewrite the specified objective function as follows:

$$\begin{aligned} ||X^P - \hat{X}^P||^2 &= \sum_{c=1}^{l} \sum_{j \in J_c} ||X^P_{.j} - \hat{X}^P_{.j}||^2 \\ &= \sum_{c=1}^{l} \sum_{j \in J_c} ||X^P_{.j} - (X^C C^T)_{.j}||^2 \\ &= \sum_{c=1}^{l} \sum_{j \in J_c} ||X^P_{.j} - n_c^{1/2} X^C_{.c}||^2. \end{aligned}$$

In the same way, setting $X^P = X(I - CC^T)$, $X^R = R^T X(I - CC^T)$, and $\hat{X}^P = RR^T X(I - CC^T) = RX^R$, we obtain the following decomposition in terms of rows:

$$||X^P - \hat{X}^P||^2 = \sum_{r=1}^{k} \sum_{j \in I_r} ||X^P_{i.} - m_r^{1/2} X^R_{r.}||^2.$$

Then, matrices X^C and X^R correspond to the cluster centroids for columns and rows respectively.

If we now consider the objective function specified by Equation 5.5, given $X^C = RR^TXC$, $X^R = R^TXCC^T$, and $\hat{X} = RR^TXCC^T = X^CC^T$, we obtain the following decompositions, respectively in terms of columns and rows:

$$\begin{aligned} ||X - \hat{X}||^2 &= \sum_{c=1}^{l} \sum_{j \in J_c} ||X_{.j} - n_c^{1/2} X_{.c}^C||^2 \\ ||X - \hat{X}||^2 &= \sum_{r=1}^{k} \sum_{j \in I_r} ||X_{i.} - m_r^{1/2} X_{r.}^R||^2 \end{aligned}$$

We can now provide our constrained co-clustering algorithmic instances. First, we give a version to solve the satisfaction problem for a conjunction of must-link and cannot-link constraints. Then, we introduce a version which processes the interval constraint.

Algorithm 4 instantiates the generic algorithm for conjunctions of must-link and cannot-link constraints (see Algorithm 2). The lines labeled with **A** refer to the objective function specified by Equation 5.4. The lines labeled with **B** refer to the version which optimizes the objective function specified by Equation 5.5. First, the algorithm initializes matrices C and R. Then, during each iteration, the algorithm associates each column (resp. row) to the nearest column (resp. row) cluster which does not introduce any cannot-link violation. If a column j (resp. row i) is involved in a must-link constraint (see Algorithms 5 and 6), the algorithm associates the whole set of columns M_v (resp. set of rows M_u) involved in the transitive closure of this constraint to the column (resp. row) cluster such that the average distance is minimum, and controlling that there is no cannot-link constraint which is violated by this operation. Since the assignment step is order-dependent, rows and columns are randomly ordered at each iteration. Then the algorithm updates the matrix C (resp. R) following the assignment schema resulting from the previously described operations. This process is iterated until the diminution of the objective function value turns to be smaller than a user-defined threshold τ.

Algorithm 7 instantiates the generic algorithm for exploiting an interval constraint (see Algorithm 3). Again, we have in Algorithm 7 the two instances that correspond to the two different types of residue (**A** for Cheng and Church's residue, and **B** for Hartigan's residue). The initialization of partitions ($\star$) concerned by this interval constraint should produce a number l (resp. k) of intervals over columns (resp. rows). Then, the assignment process only considers the frontiers between intervals. More precisely, it first processes the left frontier, then the right frontier iteratively. A column (resp. row) can be assigned to the adjacent interval if the distance is smaller than the distance computed over its original interval. In this case, we continue processing the remaining columns (resp. rows). When the left frontier and the right frontier of an interval correspond to the same column (resp. row), the algorithm starts to process the next frontier. If there is no necessity to reassign the column (resp. row), the algorithm stops the current frontier processing and it skips to the following one. The row clustering computation step is straightforward and it is omitted here ($\star\star$) for the sake of brevity.

According to the objective function defined by Equation 5.4, an optimal co-clustering result for X_r (i.e., the toy example from Fig. 1), is given by matrix X_r^1 in Fig. 2. If we enforce Object 1 and Object 2 to be in two different clusters by setting a cannot-link constraint $c_{\neq}(1,2)$, then we obtain the results shown by matrix X_r^2. Notice that, even if Object 5 is not concerned by this constraint, in the final bi-partition it is clustered together with Object 2. In fact, Object 2 and Object 5 share the same attribute values. If we set an interval constraint on the set of columns (namely $c_{int}(C^c)$), we obtain the co-cluster structure shown by X_r^3 in Fig. 2. Notice that, though only the column set is constrained, the resulting

Algorithm 4: CoCoClust1(X,k,l,$\mathcal{M}_r$,$\mathcal{M}_c$,$\mathcal{C}_r$,$\mathcal{C}_c$)

```
Input: Data matrix $X$, $k$, $l$, cannot-link sets $\mathcal{C}_r$ and $\mathcal{C}_c$, collections $\mathcal{M}_r$ and $\mathcal{M}_c$
Output: Matrices $R$ and $C$
Initialize $R$ and $C$;
$\Delta = ||X||^2$; $\tau = 10^{-5}||X||^2$;
$t = 0$;
A. $obj^t = ||(I - RR^T)X(I - CC^T)||^2$;
B. $obj^t = ||RR^T XCC^T||^2$;
while $\Delta > \tau$ do
    $t = t + 1$;
    A. $X^C = (I - RR^T)XC$; $X^P = (I - RR^T)X$;
    B. $X^C = RR^T XC$; $X^P = X$;
    foreach $1 \leq j \leq n$ do
        $L = \emptyset$;
        if $\exists M_v \in \mathcal{M}_c$ s.t. $j \in M_v$ then
            MLColumnAssign($X$,$l$,$L$,$M_v$,$\mathcal{C}_c$);
        else
            $L = \{1 \leq c \leq l \mid \nexists j_c \mid \gamma^t[j_c] = c \wedge c_{\neq}(j, j_c) \in \mathcal{C}_c\}$;
            $\gamma^t[j] = \text{argmin}_{c\in L} ||X^P_{.j} - n_c^{-1/2} X^C_{.c}||^2$;
        end
    end
    Update $C$ using $\gamma$;
    A. $X^R = R^T X(I - CC^T)$; $X^P = X(I - CC^T)$;
    B. $X^R = R^T XCC^T$; $X^P = X$;
    foreach $1 \leq i \leq m$ do
        $L = \emptyset$;
        if $\exists M_u \in \mathcal{M}_r$ s.t. $i \in M_u$ then
            MLRowAssign($X$,$k$,$L$,$M_u$,$\mathcal{C}_r$);
        else
            $L = \{1 \leq r \leq k \mid \nexists i_r \mid \rho^t[i_r] = r \wedge c_{\neq}(i, i_r) \in \mathcal{C}_r\}$;
            $\rho^t[i] = \text{argmin}_{r\in L} ||X^P_{i.} - m_r^{-1/2} X^R_{r.}||^2$;
        end
    end
    A. $obj^t = ||(I - RR^T)X(I - CC^T)||^2$;
    B. $obj^t = ||RR^T XCC^T||^2$;
    $\Delta = |obj^t - obj^{t-1}|$;
end
```

Algorithm 5: MLColumnAssign(X,l,L,M_v,$\mathcal{C}_c$)

```
foreach $j_v \in M_v$ do
    $L = L \cup \{1 \leq c \leq l \mid \nexists j_c \mid \gamma^t[j_c] = c \wedge c_{\neq}(j_v, j_c) \in \mathcal{C}_c\}$;
end
$\gamma^t[M_v] = \text{argmin}_{c\in L} \frac{\sum_{j_v \in M_v} ||X^P_{.j} - n_c^{-1/2} X^C_{.c}||^2}{|M_v|}$;
```

```
Algorithm 6: MLRowAssign($X$,$k$,$L$,$M_u$,$\mathcal{C}_r$)
  foreach $i_u \in M_u$ do
    $L = L \cup \{1 \leq r \leq k \mid \nexists i_r \mid \rho^t[i_r] = c \wedge c_{\neq}(i_u, i_r) \in \mathcal{C}_r\}$;
  end
  $\rho^t[M_u] = \mathrm{argmin}_{r \in L} \frac{\sum_{r_u \in M_u} ||X^P_{i.} - m_r^{-1/2} X^C_{r.}||^2}{|M_u|}$;
```

$$X_r^1=\begin{array}{ccc|cc|c} 1&4&5&2&3& \\ \hline 3&2&4&0&0&1 \\ 1&1&2&4&5&2 \\ 4&4&5&1&0&3 \\ 2&3&4&0&1&4 \\ 1&1&2&4&5&5 \\ \hline 1&0&0&4&6&6 \\ 0&0&0&5&6&7 \end{array} \quad X_r^2=\begin{array}{ccc|cc|c} 1&4&5&2&3& \\ \hline 3&2&4&0&0&1 \\ 4&4&5&1&0&3 \\ 2&3&4&0&1&4 \\ \hline 1&1&2&4&5&2 \\ 1&1&2&4&5&5 \\ 1&0&0&4&6&6 \\ 0&0&0&5&6&7 \end{array} \quad X_r^3=\begin{array}{ccc|cc|c} 1&2&3&4&5& \\ \hline 3&0&0&2&4&1 \\ 4&1&0&4&5&3 \\ 2&0&1&3&4&4 \\ \hline 1&4&5&1&2&2 \\ 1&4&5&1&2&5 \\ 1&4&6&0&0&6 \\ 0&5&6&0&0&7 \end{array}$$

Figure 2: Three co-clustering results on X_r (with permutations to emphasize obtained co-clusters)

object partition is also different from the one obtained when no constraint is given. Indeed, our objective function takes into account both object and attribute partitions.

Let us briefly comment the complexity of our algorithms. Considering Algorithm 4, computing $(I - RR^T)X(I - CC^T)$ (or RR^TXCC^T) only requires the number of operations needed to compute R^TXC, i.e., $kn(m+l)$. It gives $O(N)$ time complexity (when $N = mn$) under the reasonable hypothesis that $k \simeq l << m \simeq n$. Assigning columns and rows to the new clusters can be performed in $O(N(k+l))$ time at each iteration. The overall complexity of the algorithm is then $O(N(k+l)t)$, where t is the number of iterations to achieve co-clustering. Complexity of Algorithm 7 is trivially the same. In general, the fact that we only process the interval frontiers increase the performances during the assignment step.

6 Experimental validation

We now provide a comprehensive set of experimental results to illustrate the behavior of our algorithms. We study both the quality of the results and time performances w.r.t. various combinations of mining parameters. Experiments have been performed on artificially generated datasets for Section 6.1. Several useful applications on real datasets are reported in Section 6.2 for gene expression data analysis and in Section 6.3 for protein motif extraction.

6.1 Experimental setting and results on synthetic data To show the impact of constraints on co-clustering, we need to compare the results w.r.t. a target partition. Consequently, given four parameters m, n, k and l, we first generate a partition of m rows into k clusters and a partition of n columns into l clusters. Then, we choose a random value x for each of the kl co-clusters, and fill each co-cluster by generating random values following a gaussian law having a mean of x and a standard deviation of δx ($0 \leq \delta \leq 1$). For these experiments, we generated three datasets: the first one, called data3x3 with an embedded 3×3 bi-partition over a set of 1000 rows and 100 columns; the second one, called data10x5, generated from a 10×5 bi-partition over a set of 1000 rows and 100 columns; the third one, called datax20x10, with a built-in 20×10 bi-partition over the same sets of rows and columns. Constraints are generated by considering the three embedded bi-partitions. We randomly pick two

```
Algorithm 7: CoCoClust2($X$,$k$,$l$,$introw$,$intcol$)
  Input: Data matrix $X$, $k$ and $l$, two boolean values $introw$ and $intcol$
  Output: Matrices $R$ and $C$
  Initialize $R$, $C$, $left$, $right$; $(\star)$
  $\Delta = ||X||^2$; $\tau = 10^{-5}||X||^2$;
  $t = 0$;
  A. $obj^t = ||(I - RR^T)X(I - CC^T)||^2$;
  B. $obj^t = ||RR^TXCC^T||^2$;
  while $\Delta > \tau$ do
      $t = t + 1$;
      A. $X^C = (I - RR^T)XC$; $X^P = (I - RR^T)X$;
      B. $X^C = RR^TXC$; $X^P = X$;
      if $intcol = true$ then
          foreach $1 \le c \le l$ do
              $stop = false$;
              while $c > 1 \wedge stop = false \wedge right[c] > left[c]$ do
                  $j = left[c]$;
                  if $||X^P_{.j} - n_{c-1}^{-1/2}X^C_{.c-1}||^2 < ||X^P_{.j} - n_c^{-1/2}X^C_{.c}||^2$ then
                      $\gamma^t[j] = c - 1$; $left[c] = left[c] + 1$; $right[c-1] = right[c-1] + 1$;
                  else
                      $stop = true$;
                  end
              end
              $stop = false$;
              while $c < l \wedge stop = false \wedge right[c] > left[c]$ do
                  $j = right[c]$;
                  if $||X^P_{.j} - n_{c+1}^{-1/2}X^C_{.c+1}||^2 < ||X^P_{.j} - n_c^{-1/2}X^C_{.c}||^2$ then
                      $\gamma^t[j] = c + 1$; $left[c+1] = left[c+1] - 1$; $right[c] = right[c] - 1$;
                  else
                      $stop = true$;
                  end
              end
          end
      else
          foreach $1 \le j \le n$ do
              $\gamma^t[j] = \text{argmin}_{c \in L} ||X^P_{.j} - n_c^{-1/2}X^C_{.c}||^2$;
          end
      end
      Update $C$ using $\gamma$;
      A. $X^R = R^TX(I - CC^T)$; $X^P = X(I - CC^T)$;
      B. $X^R = R^TXCC^T$; $X^P = X$;
      {Reassign rows}$(\star\star)$;
      A. $obj^t = ||(I - RR^T)X(I - CC^T)||^2$;
      B. $obj^t = ||RR^TXCC^T||^2$;
      $\Delta = |obj^t - obj^{t-1}|$;
  end
```

objects and generate a must-link constraint if those two objects share the same cluster label, otherwise we generate a cannot-link constraint.

Our algorithms are implemented in C, and all the experiments have been performed on a PC (Windows, Intel Core 2 Duo 2.00GHz, 2GB RAM). In all our experiments the value of the stopping parameter τ was set to $10^{-5}||X||^2$. The experiments described in this section have been performed using the Hartigan's definition of residue.

To evaluate the agreement between the embedded bi-partition and the ones discovered by our constrained algorithm, we use the adjusted Rand index [24]. If $\mathbf{C} = \{C_1 \dots C_z\}$ is the partition built by the clustering algorithm and $\mathbf{P} = \{P_1 \dots P_z\}$ is a predefined partition, each pair of points can be assigned to the same cluster or to two different clusters in each partition. Let a be the number of pairs belonging to the same cluster of $\mathbf{C}$ and to the same cluster of $\mathbf{P}$. The expected value of a denoted $exp(a)$ (p being the number of points) is computed as follows:

$$exp(a) = \frac{|\pi(C)| \cdot |\pi(P)|}{p(p-1)/2}$$

where

$$|\pi(C)| = \frac{\sum_{k=1}^{z} |C_k|(|C_k|-1)}{2}$$

$$|\pi(P)| = \frac{\sum_{k=1}^{z} |P_k|(|P_k|-1)}{2}.$$

Then, the maximum value for a is:

$$max(a) = \frac{1}{2}\left(|\pi(C)| + |\pi(P)|\right)$$

The agreement between $\mathbf{C}$ and $\mathbf{P}$ is estimated as follows:

$$AR(\mathbf{C}, \mathbf{P}) = \frac{a - exp(a)}{max(a) - exp(a)}$$

Notice that when $AR(\mathbf{C}, \mathbf{P}) = 1$, we have identical partitions.

We also evaluate the final objective function value, as well as the average number of iterations performed by the algorithm to achieve the convergence criterion. Finally, we measure how many times the algorithm fails because of the impossibility to satisfy a cannot-link constraint. Notice that even if this situation is possible, we expect the failure rate to be quite small in practice.

We are interested in the behavior of our algorithm w.r.t. the number of constraints. Therefore, we generated various sets of constraints with an increasing number of pair constraints for rows and columns. The size of the constraint set for rows varies between 0 and 50 (the incrementing step is 10). The size of the column constraint set varies between 0 and 30 (with an incrementing step of 5). We consider all the possible combinations of constraint sets. Moreover, for each combination, we generated 5 different constraint sets, in order to reduce the bias introduced by the particular choice of pairs. As initial co-clustering is randomly initialized, we run our algorithm 10 times on each constraint set. All measures are averaged over these 50 trials.

Figures 3 and 4 show the results for this set of experiments. The average values for the adjusted Rand index (see Fig. 3) increase with the number of constraints on rows and columns. In general, constraints on one dimension influence the partition on the other dimension as well. This behavior is however more obvious on column partitions (See Figures 3a, c and e). The results show that our approach is robust w.r.t. the number of clusters. In the first dataset (Figures 3a and b), a few number of constraints enable to obtain perfect bi-partitions. When the number of clusters is higher (Figures 3c to f), the gain in terms of cluster agreement is even more convincing. The analysis of the average final objective values (Figure 4a, c and e), confirms the effectiveness of using constraint in a co-clustering framework: the more we introduce constraints, the more the objective function is optimized. In particular, most of the results show that using constraints on both sides leads to better results than using constraints on only one side. Notice that we omitted the standard deviations of the computed parameters in these figures. In all these experiments, the stability of the co-clustering algorithm increases with the number of used constraints on both sets (i.e., the standard deviations decreases). In particular, for data3x3, when the number of constraints is high, the standard deviations are equal to zero, i.e., each randomly initialized instance of our algorithm finds exactly the desired result. As a counterpart, the average number of iterations before convergence is slightly higher (Figure 4b, d and f), but it seems that it is not influenced by the size of the constraint set. In this sets of experiments, the average failure number is always 0, except for the data3x3 matrix when the column constraint set size is 30. In this case, it is about 2%. We estimate that this rate is acceptable and it could be improved by using more sophisticated techniques to process cannot-link constraints.

6.2 Application to gene expression data analysis We studied the impact of our constraint-based co-clustering approach on the well-documented microarray dataset drosophila [2]. It concerns the gene expression of the Drosophila melanogaster during its life cycle. The expression levels of 3 944 genes are evaluated for 57 sequential time periods divided into embryonic, larval and pupal stages. It has been retrieved from `http://smd.stanford.edu` (the preprocessed and cleaned version of this dataset is available on `http://kdd.di.unito.it/~pensa/site/datasets/`). Missing values (less than 1%) were replaced by zeros. In the first bunch of experiments (must-link and cannot-link constraints), the value of the stopping parameter τ was set to $10^{-5}||X||^2$. In the second one (interval constraint), τ was set to $10^{-4}||X||^2$. Running time is about 30 seconds.

6.2.1 Results for must-link et cannot-link constraints Let us first try to discover one fixed partition by using the co-cluster label to define sets of pairwise constraints, both on gene and biological sample sets. We measured the impact of combining constraints over row sets and column sets on the drosophila data using the Hartigan's definition of residue. For this purpose, we selected a bi-partition among the unconstrained co-clustering results. In particular, we chose the co-clustering results with the minimum objective function value obtained at the end of the iterative process. This value is 1.36935×10^5. We use the resulting bi-partition to generate various sets of constraints. The size of the constraint set for rows varies between 0 and 75 (the incrementing step is 15). The size of the column constraint set varies between 0 and 15 (with an incrementing step of 3). For each combination, we generated 5 different constraint sets, and we have been using our algorithm 10 times on each constraint set. The results are given in Fig. 5.

The gain in using constraints on both dimensions is clear from Figures 5a and b, which plot the average Rand index values computed over columns and rows respectively. The objective function value almost reaches the target value when the number of constraints on both sides increases. Moreover, the average number of iterations is always between 12 and 26. It seems to be independent from the size of the constraint sets on columns. The constraints on rows seems to influence this parameter more significantly (though not drastically). Finally, using constraints on columns improves the stability of the co-clustering algorithm, as shown in Figures 5e and f. Constraints on rows seem to negatively influence the stability for the Rand index computed on columns (the standard deviation on columns is omitted, since there are no significant variations), while the do not influence significantly the stability of the final objective function value. This is also due to the fact that the percentage of row constraints on the number of

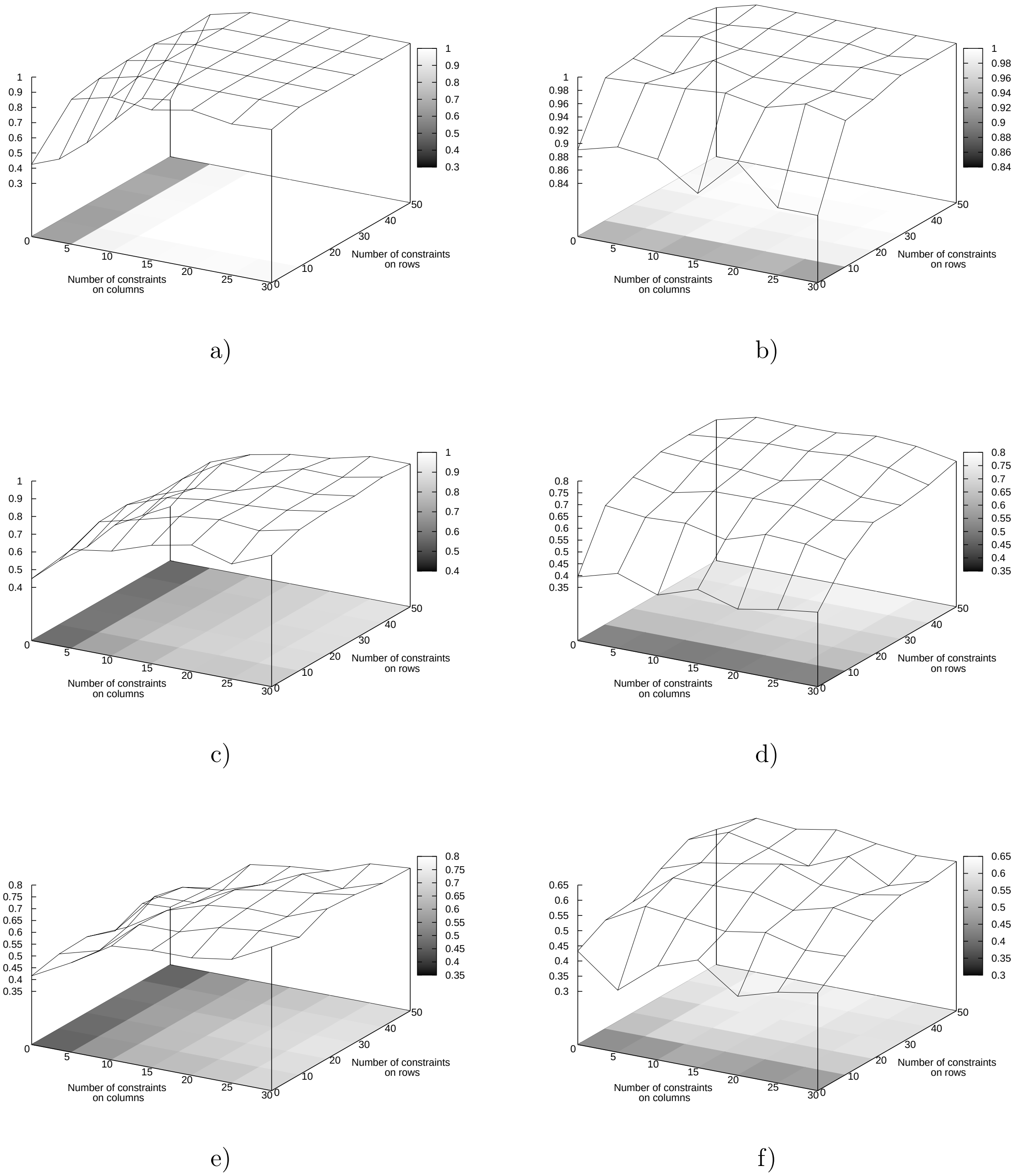
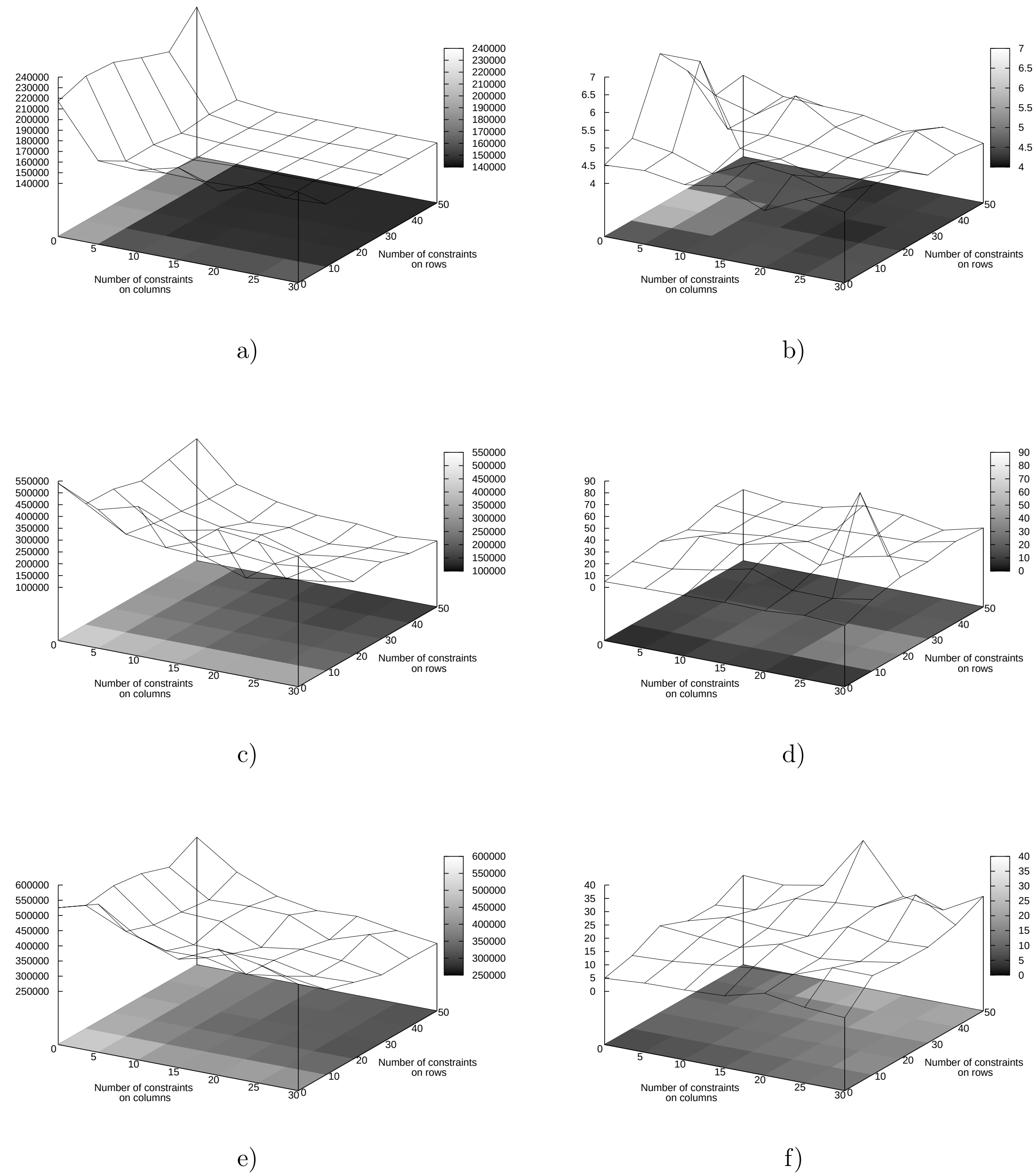


Figure 3: Average adjusted Rand indexes computer over columns (left) and rows (right) for data3x3 (a and b), data10x5 (c and d), and data20x10 (e and f).

rows is small. In conclusion, this study confirms that the overall stability is similar or better than the stability obtained by the unconstrained algorithm. Last but not least, the failure rate is not significant, since our algorithm failed in finding a solution only in one performed trial over a total number of 1250 trials.



a)

b)

c)

d)

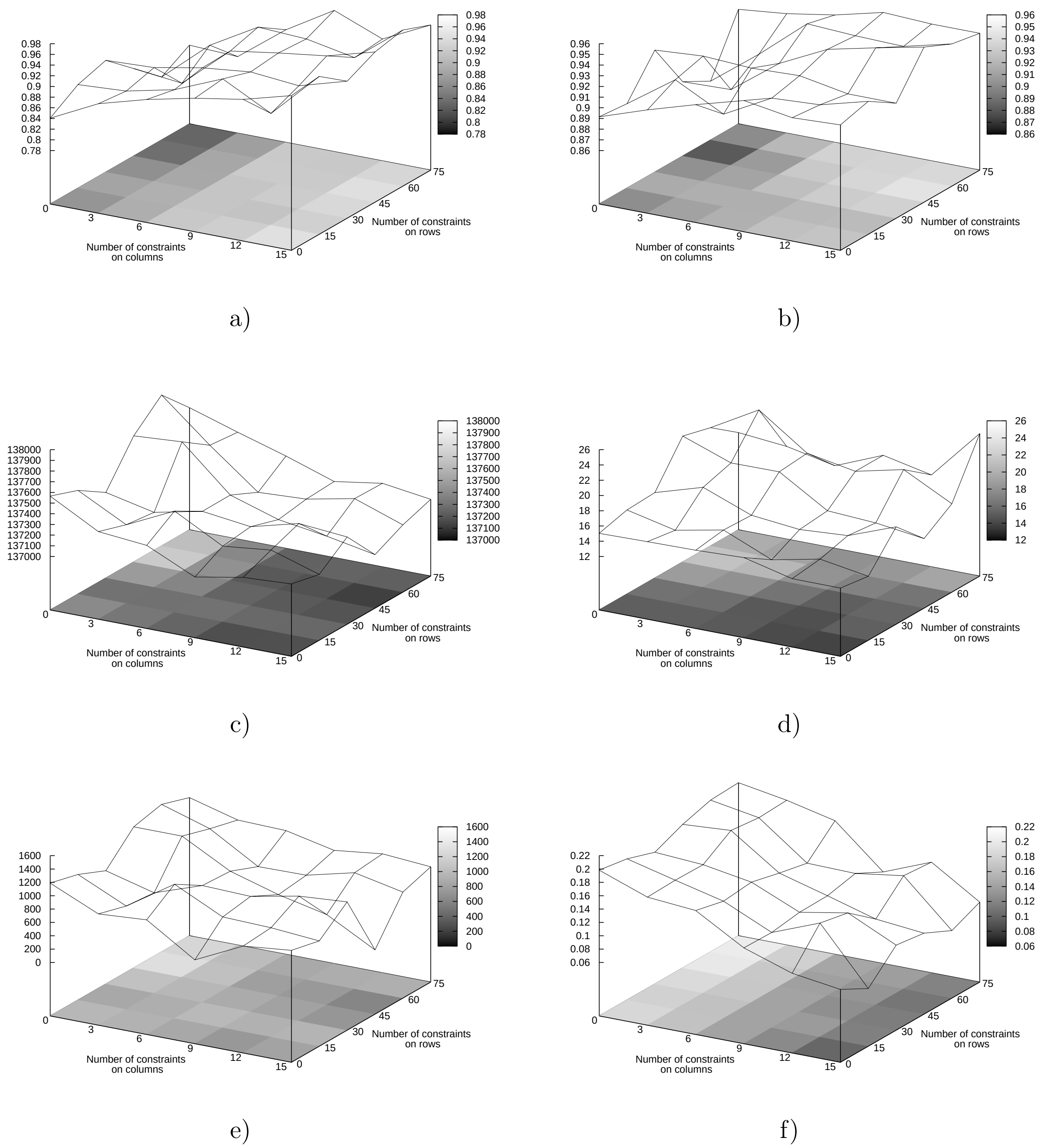
e)

f)

Figure 4: Average final objective function values (left) and number of iterations (right) for data3x3 (a and b), data10x5 (c and d), and data20x10 (e and f).

6.2.2 Results for the interval constraint We evaluated the added value of the interval constraint by applying our algorithm to the drosophila dataset. Here, our goal is to rediscover the three phases of the drosophila life cycle using, as unique information, the number of clusters ($k = l = 3$).

We compared the adjusted Rand index for the constrained and unconstrained versions of our



a)

b)

c)

d)

e)

f)

Figure 5: Average adjusted Rand indexes computer over columns (a) and rows (b), final objective function values (c) number of iterations (d), and standard deviations of the objective function value (e) and adjusted Rand index on columns (f) for the drosophila dataset.

algorithm (both optimizing the Cheng and Church's objective function) and for a collection of 20 randomly initialized runs. The results (see Tab. 1), show that using an interval constraint enables to find more accurately the three stages of the drosophila life cycle (the measured improvement for

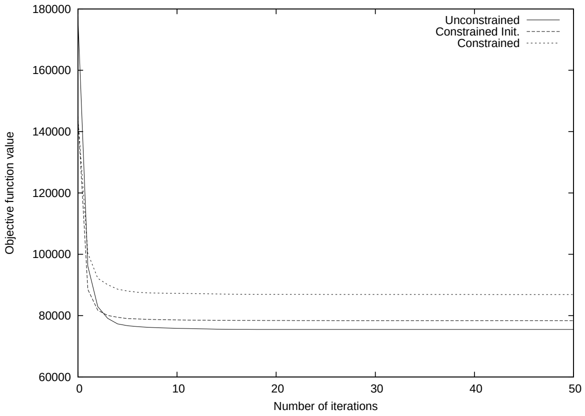


Figure 6: Object value vs. iterations

	AR	$\|\|H\|\|^2$	Nb.Iter.
Const.	**0.76±0.23**	$8.23 \pm 0.40 \times 10^4$	11.60±2.96
Unconst.	0.41±0.17	$7.73 \pm 0.48 \times 10^4$	14.60±7.60
Const.Init.	0.54±0.17	$7.86 \pm 0.44 \times 10^4$	**11.05±3.50**

Table 1: Adjusted Rand index, final objective function value and number of iterations.

the adjusted Rand index is about 85%). Moreover, the number of iterations needed to complete the co-clustering process is considerably smaller when enforcing such a constraint. Notice that the final value of the objective function for the unconstrained version of the algorithm is better than for the constrained version. It means that the structure which our algorithm is able to discover is unlikely to be the global optimum for this dataset. Despite of this, the unconstrained algorithm has never managed to find intervals. Notice that, in this case, using the interval constraint improve the stability of the co-clustering algorithm for all the measured parameters.

In Fig. 6, we plot the average behavior of the objective function value w.r.t. iterations. For this type of dataset, an interval-based initialization step gives rise to a smaller initial objective function. Interestingly, the objective function value for the unconstrained co-clustering algorithm starts to be better after the first iteration. Afterwards, the difference between the two curves is just an offset, while the two convergence speeds are quite similar. This example illustrates the tradeoff between the objective function optimization and the constraint satisfaction processes. We can comment such a process considering the metaphor of the “tug of war” game. When the only competitor is the objective function, the only limit is its global optimum. When a second competitor (say constraints) plays, it reduces the objective function strength.

In order to measure the impact of the initialization method on the iterative behavior of the algorithm, we measured all the performances parameters already used for the previous comparison (see Table 1). Initializing the column partitions with a set of intervals improve the average adjusted Rand index value over 20 runs, but this value is still far from the one achieved by the constrained version. The number of iterations is the only performance index which improves w.r.t. both unconstrained and constrained algorithms. Notice that, even if a constraint-oriented initialization has been used, none of the 20 executions has been able to find intervals.

Finally, Fig. 6 also illustrates that, after an improvement in the initial objective function, there are no significant difference between the unconstrained algorithm and the interval-initialized one. It means that the active constraint process introduced within Algorithm 7 is critical: a constraint-oriented initialization is not sufficient to enforce the interval constraint.

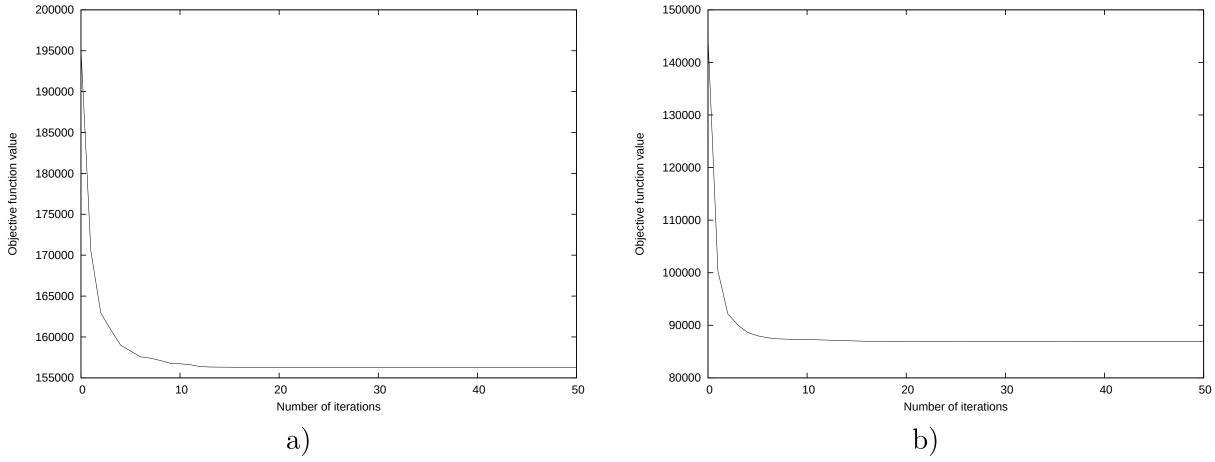


Figure 7: Object value vs. iterations with the Hartigan (a) and Cheng & Church (b) objective functions.

6.2.3 Convergence analysis We now provide an experimental evidence of the convergence of our algorithm under the interval constraint. We apply our algorithm to the drosophila dataset using both the Hartigan's residue and the Cheng & Church's residue. We performed 20 trials and stopped the execution after 50 iterations. The averaged objective function values at each iteration are plotted in Figure 7. Clearly, the two objective functions decrease monotonically. Moreover, the convergence speed is quite high in both experiments: the algorithm needs less than 10 iterations to achieve a good objective function optimization level.

6.3 Application: Protein motif extraction Let us now propose the use of constrained co-clustering as a technique to discover protein motifs. The novelty of our approach relies on two main ideas: exhaustive search and automatic association of motifs with protein sub-families.

We extracted all possible motifs of a given length from protein sequences and stored them in a prefix tree. *All* patterns are stored along with the protein name and frequency. Then, the constrained co-clustering algorithm is used to find both protein motif classes and protein groups: in such a way, it is possible to correlate every protein group with one or more motif classes.

Formally, let Σ be a finite alphabet. Let $S = \{\sigma_1, ..., \sigma_m\}$ be a set of protein sequences, such that $\sigma_i \in \Sigma^*$ and let $\pi_1, ..., \pi_n$ be the motifs stored in the prefix tree. We build a matrix X of $n \times m$ such that:

$$X_{ij} = \text{frequency of pattern } \pi_i \text{ in the protein sequence } \sigma_j$$

We create a set of must-link constraints and a set of cannot-link constraints only on rows. We set a must-link constraint on every pair of motifs having an edit distance less than $\epsilon_{min} = 2$, and a cannot-link constraint on every pair of motifs π_i, π_j having edit distance greater than $\epsilon_{max} = \max(|\pi_i|, |\pi_j|) - 1$, where $|\pi|$ is the length of motif π. We run the co-clustering algorithm on the matrix X and using the aforementioned constrains. The result of the algorithm is a partition $\mathcal{P}(\sigma)$ of the protein sequences as well as a partition $\mathcal{P}(\pi)$ of the motifs. Each element of $\mathcal{P}(\sigma)$ is then associated with an element of $\mathcal{P}(\pi)$ by means of a statistical measure based on cluster cardinality [14].

The empirical assessment of the system exploited a protein dataset created by mixing sequences from different protein families (Brazma et al. [11]). We retrieved the sequences from four protein families

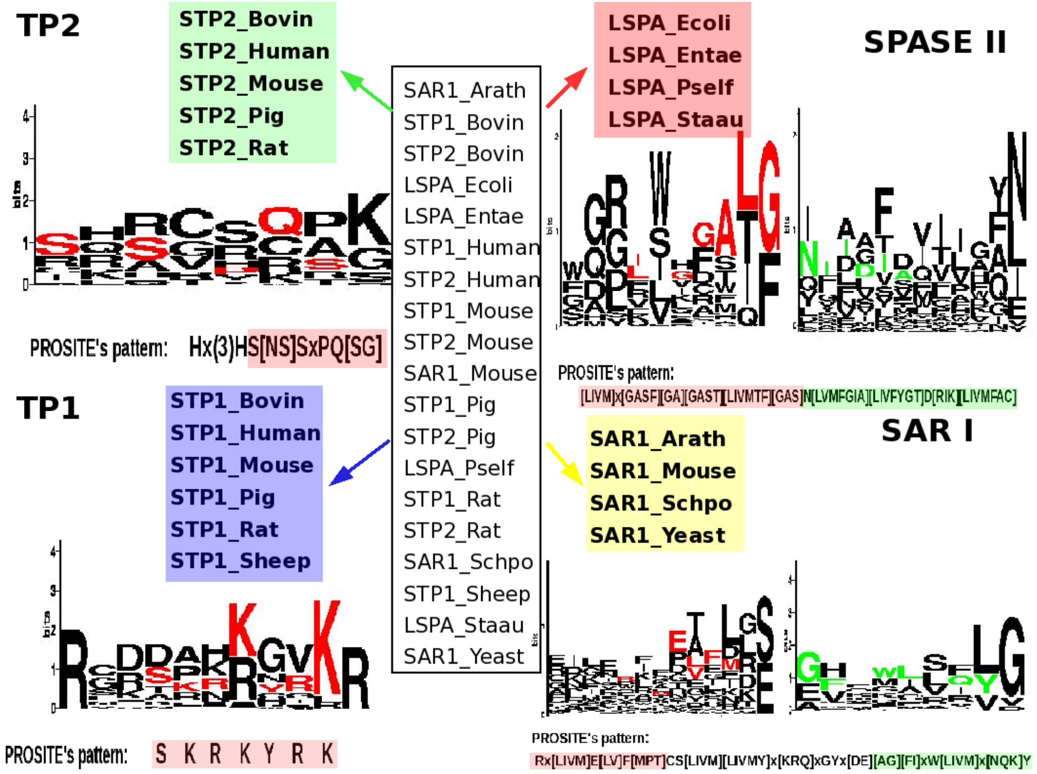


Figure 8: The pairs of protein clusters and motif clusters is shown.

stored in PROSITE [3] (release 13.0, January 2008). The families are: TP1 (prositeID: PS00541) containing 6 sequences, TP2_1 (prositeID: PS00970) containing 5 sequences, SAR_1 (prositeID: PS01020) containing 4 sequences and SPASE_II (prositeID: PS00855) containing 4 sequences. By construction, each protein family in the dataset is characterized by one specific motif.

In the experimentation we used all 630 constraints. Our frequency matrix has 7491 rows and 19 columns, we set the program parameters as follows: the number of *row clusters* varied from 50 to 120; the number of *column clusters* varied from 2 to 7; residues have been computed using the Hartigan's definition; the value of the stopping parameter τ was set to 10^{-3}.

Results corresponding to the combination of parameters that minimizes the objective function are summarized in Figure 8. In this particular experiments the number of row clusters and column clusters have been set to **50** and **4** respectively.

The whole set of protein sequences are reported in the central (white) box, whereas the four colored boxes corresponds to the column clusters found by the algorithm. It is worthwhile to notice that each column cluster contains only protein sequences belonging to the same protein family. This is particularly interesting since those clusters are found by the co-clustering algorithm on the basis of the sole information provided by motifs/proteins co-occurring frequencies.

Nearby each column cluster we report the associated motif clusters. Each motif cluster has been represented in a graphical way, after the necessary multi-alignment step, by means of sequence logos generated with WebLogo [15]. A sequence logo is a graphical representation of an amino acid or nucleic acid multiple sequence alignment developed by Schneider and Stephens [35]. Each logo consists of stacks of symbols, one stack for each position in the sequence. The overall height of the stack indicates the sequence conservation at that position, while the height of symbols within the stack indicates the relative frequency of each amino or nucleic acid at that position. In general, a sequence logo provides a richer and more precise description of, for example, a binding site, than would a consensus sequence.

Below each motif, we report the PROSITE pattern characterizing the unique protein family contained in the corresponding column cluster. Amino acids that appear both in the PROSITE pattern and in the motif logos have been highlighted in red and green. The common portion of the sequences is large. Moreover, it should be pointed out that our results are equivalent to those found by Brazma et al. on the same dataset. In both approaches, all four protein families are recovered and the most frequent motifs are identified. In our case we are also able to automatically find the association of motif and protein patterns.

7 Conclusion

Co-clustering is an interesting conceptual clustering approach. Improving co-cluster relevancy remains a difficult task in real-life exploratory data analysis processes. First, it is hard to capture subjective interestingness aspects, i.e., the analyst's expectation given her/his domain knowledge. Next, when these expectations can be declaratively specified, using them during the computational process of bi-partitions is challenging. In [32, 33], a simple approach was suggested that was dedicated to 0/1 data analysis and incomplete w.r.t. constraint processing. In this paper, we have proposed a new constrained co-clustering algorithm that is related to the the generic co-clustering setting from [4]. We explained how to exploit user-defined constraints like must-link, cannot-link, and interval constraints when co-clustering numerical matrices. Applications on kinetic gene expression data analysis and protein motif discovery tasks have been considered. Many other applications rely on ordered data analysis and may benefit from such constrained co-clustering approaches.

A short-term perspective is to combine properly the strategies for exploiting must-link and cannot-link constraints one one hand, and interval constraints on another hand. So far, our approach does not look for overlapping co-clusters while this may be interesting for many applications. We may study the possibility to discover overlapping co-clusters, like many local bi-clustering approaches already do (see, e.g., [12, 26]). Also, we are convinced that our approach can be easily extended towards other kinds of objective functions (e.g., mutual information [19]) since we already demonstrated that it works for different objective functions. It is also appealing to study other types of user-defined constraints (e.g., balancing constraints [6]).

Acknowledgements. This research is partially funded by EU contract GeoPKDD IST-FP6-014915 and the French ANR MDCO Project Bingo2. Ruggero G. Pensa and Francesca Cordero are co-funded by Regione Piemonte.

References

[1] A. Anagnostopoulos, A. Dasgupta, and R. Kumar. Approximation algorithms for co-clustering. In *Proceedings PODS 2008*, pages 201–210, Vancouver, BC, Canada, 2008.

[2] M.N. Arbeitman, E.E. Furlong, F. Imam, E. Johnson, B.H. Null, B.S. Baker, M.A. Krasnow, M.P. Scott, R.W. Davis, and K.P. White. Gene expression during the life cycle of drosophila melanogaster. *Science*, 297:2270–2275, september 2002.

[3] A. Bairoch. Prosite: a dictionary of sites and patterns in proteins. *Nucleic Acids Res*, 19:2241–2245, 1991.

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

[5] A. Banerjee and J. Ghosh. On scaling up balanced clustering algorithms. In *Proceedings SIAM SDM 2002*, Arlington, VA, USA, 2002.

[6] A. Banerjee and J. Ghosh. Scalable clustering algorithms with balancing constraints. *Data Min. Knowl. Discov.*, 13(3):365–395, 2006.

[7] S. Basu, A. Banerjee, and R. J. Mooney. Semi-supervised clustering by seeding. In *Proceedings ICML 2002*, pages 27–34, Sydney, Australia, 2002.

[8] S. Basu, M. Bilenko, and R. J. Mooney. A probabilistic framework for semi-supervised clustering. In *Proceedings ACM SIGKDD 2004*, pages 59–68, Seattle, WA, USA, 2004.

[9] S. Basu, I. Davidson, and K. Wagstaff (Editors). *Constrained Clustering: Advances in Algorithms, Theory and Applications.* Chapman & Hall/CRC Press, Data Mining and Knowledge Discovery Series, 2008.

[10] M. Bilenko, S. Basu, and R. J. Mooney. Integrating constraints and metric learning in semi-supervised clustering. In *Proceedings ICML 2004*, pages 81–88, Banff, Canada, 2004.

[11] A. Brazma, I. Jonassen, E. Ukkonen, and J. Vilo. Discovering patterns and subfamilies in biosequences. In *Proceedings ISMB 2009*, pages 34–43, St. Louis, MO, USA, 1996. AAAI press.

[12] Y. Cheng and G. M. Church. Biclustering of expression data. In *Proceedings ISMB 2000*, pages 93–103, San Diego, CA, USA, 2000. AAAI Press.

[13] H. Cho, I. S. Dhillon, Y. Guan, and S. Sra. Minimum sum-squared residue co-clustering of gene expression data. In *Proceedings SIAM SDM 2004*, Lake Buena Vista, FL, USA, 2004.

[14] F. Cordero, A. Visconti, and M. Botta. A new protein motif extraction framework based on constrained co-clustering. In *Proceedings SAC 2009*, pages 776–781, Honolulu, Hawaii, USA, 2009. ACM.

[15] G. E. Crooks, G. Hon, J-M. Chandonia, and S. E. Brenner. Weblogo: A sequence logo generator. *Genome Research*, 14:1188–1190, 2004.

[16] I. Davidson and S. S. Ravi. Agglomerative hierarchical clustering with constraints: Theoretical and empirical results. In *Proceedings PKDD 2005*, volume 3721 of *LNCS*, pages 59–70, Porto, Portugal, 2005. Springer.

[17] I. Davidson and S. S. Ravi. Clustering with constraints: Feasibility issues and the k-means algorithm. In *Proceedings SIAM SDM 2005*, Newport Beach, CA, USA, 2005.

[18] M. Deodhar, G. Gupta, J. Ghosh, H. Cho, and I. S. Dhillon. A scalable framework for discovering coherent co-clusters in noisy data. In *Proceedings ICML 2009*, pages 241–248, Montreal, Quebec, Canada, 2009.

[19] I. S. Dhillon, S. Mallela, and D. S. Modha. Information-theoretic co-clustering. In *Proceedings ACM SIGKDD 2003*, pages 89–98, Washington, DC, USA, 2003.

[20] E.E. Furlong, E.C. Andersen, B. Null, K.P. White, and M.P. Scott. Patterns of gene expression during drosophila mesoderm development. *Science*, 293:1629–1633, 2001.

[21] R. Ge, M. Ester, W. Jin, and I. Davidson. Constraint-driven clustering. In *Proceedings of ACM SIGKDD 2007*, pages 320–329, San Jose, CA, USA, 2007.

[22] J. A. Hartigan. Direct clustering of a data matrix. *Journal of the American Statistical Association*, 67(337):123–129, March 1972.

[23] B. Hendrickson and T. G. Kolda. Partitioning rectangular and structurally unsymmetric sparse matrices for parallel processing. *SIAM J. Sci. Comput*, 21:2048–2072, 1999.

[24] L. Hubert and P. Arabie. Comparing partitions. *Journal of Classification*, 2(1):193–218, 1985.

[25] D. Ienco, R. G. Pensa, and R. Meo. Parameter-free hierarchical co-clustering by n-ary splits. In *Proceedings ECML PKDD 2009*, volume 5781 of *LNCS*, pages 580–595, Bled, Slovenia, 2009. Springer.

[26] J. Ihmels, G. Friedlander, S. Bergmann, O. Sarig, Y. Ziv, and N. Barkai. Revealing modular organization in the yeast transcriptional network. *Nature Genetics*, 31:370–377, august 2002.

[27] D. Klein, S. D. Kamvar, and C. D. Manning. From instance-level constraints to space-level constraints: Making the most of prior knowledge in data clustering. In *Proceedings ICML 2002*, pages 307–314, Sydney, Australia, 2002.

[28] Y. Kluger, R. Basri, J. T. Chang, and M. Gerstein. Spectral biclustering of microarray data: coclustering genes and conditions. *Genome Research*, 13:703–716, 2003.

[29] L. Lazzeroni and A. Owen. Plaid models for gene expression data. Technical Report Stanford University, 2000.

[30] S. C. Madeira and A. L. Oliveira. Biclustering algorithms for biological data analysis: A survey. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 1(1):24–45, 2004.

[31] R. G. Pensa and J-F. Boulicaut. Constrained co-clustering of gene expression data. In *Proceedings SIAM SDM 2008*, pages 25–36, Atlanta, GA, USA, 2008.

[32] R. G. Pensa, C. Robardet, and J-F. Boulicaut. Towards constrained co-clustering in ordered 0/1 data sets.

In *Proceedings ISMIS 2006*, volume 4203 of *LNCS*, pages 425–434, Bari, Italy, 2006. Springer.

[33] R. G. Pensa, C. Robardet, and J-F. Boulicaut. Constraint-driven co-clustering of 0/1 data. In *Constrained Clustering: Advances in Algorithms, Theory and Applications*, pages 123–148. Chapman & Hall/CRC Press, Data Mining and Knowledge Discovery Series, 2008.

[34] C. Robardet and F. Feschet. Efficient local search in conceptual clustering. In *Proceedings DS 2001*, volume 2226 of *LNCS*, pages 323–335, Washington, DC, USA, 2001. Springer.

[35] T. D. Schneider and R. M. Stephens. Sequence logos: A new way to display consensus sequences. *Nucleic Acids Res.*, 18:6097–6100, 1990.

[36] E. Segal, M. Shapira, A. Regev, D. Pe'er, D. Botstein, D. Koller, and N. Friedman. Module networks: identifying regulatory modules and their condition-specific regulators from gene expression data. *Nat Genet.*, 34:166–176, 2003.

[37] H. Shan and A. Banerjee. Bayesian co-clustering. In *Proceedings ICDM 2008*, pages 530–539, Pisa, Italy, 2008. IEEE Computer Society.

[38] K. Wagstaff, C. Cardie, S. Rogers, and S. Schrödl. Constrained k-means clustering with background knowledge. In *Proceedings ICML 2001*, pages 577–584, Williamstown, MA, USA, 2001.