# Graphical Model approaches for Biclustering

by

Matteo Denitto

Submitted to the Department of Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Computer Science
S.S.D. ING-INF05
Cycle XXIX/2014

at the

Università degli Studi di Verona

February 2017



Author . . . . . .
Department of Computer Science
Feb. 20, 2017

Certified by . . . . . .
dr. Manuele Bicego
Assistant Professor
Thesis Tutor

Accepted by . . . . . .
Prof. Massimo Merro
Chairman of the PhD School Council

*Graphical Model approaches for Biclustering* – Matteo Denitto
PhD Thesis, Verona, Feb. 20, 2017.
ISBN: XXXXXXXXXXXXX

**Graphical Model approaches for Biclustering**

by

Matteo Denitto

Submitted to the Department of Computer Science
on Feb. 20, 2017, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Computer Science
S.S.D. ING-INF05
Cycle XXIX/2014

## Abstract

In many scientific areas, it is crucial to group (cluster) a set of objects, based on a set of observed features. Such operation is widely known as *Clustering* and it has been exploited in the most different scenarios ranging from Economics to Biology passing through Psychology. Making a step forward, there exist contexts where it is crucial to group objects and simultaneously identify the features that allow to recognize such objects from the others. In gene expression analysis, for instance, the identification of subsets of genes showing a coherent pattern of expression in subsets of objects/samples can provide crucial information about active biological processes. Such information, which cannot be retrieved by classical clustering approaches, can be extracted with the so called *Biclustering*, a class of approaches which aim at simultaneously clustering both rows and columns of a given data matrix (where each row corresponds to a different object/sample and each column to a different feature). The problem of biclustering, also known as co-clustering, has been recently exploited in a wide range of scenarios such as Bioinformatics, market segmentation, data mining, text analysis and recommender systems.

Many approaches have been proposed to address the biclustering problem, each one characterized by different properties such as interpretability, effectiveness or computational complexity. A recent trend involves the exploitation of sophisticated computational models (*Graphical Models*) to face the intrinsic complexity of biclustering, and to retrieve very accurate solutions. Graphical Models represent the decomposition of a global objective function to analyse in a set of smaller/local functions defined over a subset of variables. The advantages in using Graphical Models relies in the fact that the graphical representation can highlight useful hidden properties of the considered objective function, plus, the analysis of smaller local problems can be dealt with less computational effort. Due to the difficulties in obtaining a representative and solvable model, and since biclustering is a complex and challenging problem, there exist few promising approaches in literature based on Graphical models facing biclustering.

This thesis is inserted in the above mentioned scenario and it investigates the exploitation of Graphical Models to face the biclustering problem. We explored different type of Graphical Models, in particular: *Factor Graphs* and *Bayesian Networks*. We present three novel algorithms (with extensions) and evaluate such techniques using available benchmark datasets. All the models have been compared with the state-of-the-art competitors and the results show that Factor Graph approaches lead to solid and efficient solutions for dataset of contained dimensions, whereas Bayesian Networks can manage huge datasets, with the overcome that setting the parameters can be not trivial. As another contribution of the thesis, we widen the range of biclustering applications by studying the suitability of these approaches in some Computer Vision problems where biclustering has been never adopted before.

Summarizing, with this thesis we provide evidence that Graphical Model techniques can have a significant impact in the biclustering scenario. Moreover, we demonstrate that biclustering techniques are ductile and can produce effective solutions in the most different fields of applications.

Thesis Advisor: dr. Manuele Bicego
Title: Assistant Professor

Chairman of the PhD School Council: Prof. Massimo Merro

# Contents

# Chapter 1

# Introduction

Every living being relies on senses to face the daily difficulties that arise in nature; like all the other animals we, humans, also developed the use of the senses to interact with the environment. Using a series of general concepts (or *patterns*) learned from the experience and the cognitive ability of *recognition* we can, for examples, recognize each character of the alphabet, distinguish between male and female faces or identify a known person when hearing a voice on the phone. These tasks, that are guided by elaborate biomolecular processes and experience, are included in the vast field of *pattern recognition* [12, 112, 132, 139].

*Automatic* systems for pattern recognition represent a very important research field that gained a great explosion of interest in the last century. Such systems aim at developing automatic techniques allowing a calculator to imitate the human sensorial abilities to retrieve the same type of information. Once introduced, this family of approaches instantly had an enormous impact, so much that now it is used to support the analysis of the scientists in many fields (such as Signal Processing, [9, 17], analysis of biomedical images [36] and so on). Pattern recognition methodologies can be divided in two big classes: supervised schemes and unsupervised schemes.

Given a new observation, and a set of possible categories, the supervised learning – better known as *classification* – represents the problem of recognizing to which category the new observation belongs. This choice is made by devising a model built on a set of observations where category memberships (labels) are known, such set of observations is commonly named "training set". This represents the well known paradigm of "learning by examples": use a set of patterns from the problem, together with the correct categories, to learn the models so that to learn how the different classes of the problem can be distinguished. Hence the classification process must be preceded by a training phase where the approach "learns" the characteristics of each category.

On the other hand, unsupervised learning - or *clustering* - differs from classification principally because true labels are not known. In [64] clustering is described as the process of grouping together points/objects, on the basis of a similarity criteria,

*i.e.* points belonging to the same group (cluster) are similar, and points belonging to different groups (clusters) are dissimilar. Clustering is an intrinsic ill-posed problem, since the definition of cluster is vague[1]: there exist different interpretations of the concept *similarity*. In fact, given a set of points/object, these could be grouped differently according to different similarity criteria (*e.g.,* colour, shape). In principle, the number of groups to retrieve can vary, together with the population of each group. Let us clarifying the concept with a macroscopic example. Consider a basket of fruits containing different types of apples and pears. One reasonable criteria would be to consider as similar fruits belonging to the same category, thus separating apples from pears. However, let us assume that we are investigating substances causing the different fruits pigmentations. Now a more reasonable choice would be to divide fruits on the basis of their colours, hence obtaining red fruits on one side and green fruits on the other side (mixing both apples and pairs).

Clustering analysis allows to distil useful information from the data, and for this reason it finds large consensus in various scientific areas such as Biology, Economics, Psychology and so forth [25, 40, 116, 142]. However, there are situations where classical clustering approaches do not represent the best solution: (for example) data contaminated with noise, or when misleading/confusing features are present. One scenario where the limits of classical clustering approaches arise is represented by the analysis of Gene Expression matrices derived from DNA microarray experiments. Briefly, DNA microarray is a process used to measure the expression level of a large number of genes in different conditions (or experiments). The result of a DNA microarray analysis is a matrix where every row represents one of the genes analysed and every column represents one of the experimental conditions, so a spot in the matrix indicates the expression level of a certain gene in a certain experiment. This matrix is known as *gene expression matrix* [13]. A possible application of clustering in this context is to determine which genes show similar behaviours over all the experiments, possibly leading to the discovery of co-regulated mechanism. However, an interesting question in this context may be the following: are there genes that share similar attitude only in a certain subset of experiments? If so, can we determine those genes and experiments? Clearly, this issue could not be solved by using a standard clustering approach, because the task requires to find a sub-matrix of the gene expression matrix (i.e. subset of both rows and columns). However this task can be solved by *biclustering* techniques which have been introduced in microarray scenario by Cheng et al in [19].

A general definition of biclustering can be the one given by Madeira and Oliveira in their seminal paper [83]. They defined biclustering as a "distinct class of clustering algorithms that perform simultaneous row-column clustering". It is easy to see that biclustering is more complex than clustering since it adds another degree of freedom with respect to the classical clustering analysis. Even if biclustering was

[1]However various researchers put a great deal of effort in providing a common and general *cluster* definition [1, 2, 102].

born in bioinformatics, it has been recently applied in various scenarios such as text analysis, recommender systems and information security [3, 16, 138].

In recent years, different approaches to biclustering expression microarray data have been presented, each one characterized by different features, such as computational complexity, effectiveness, interpretability, optimization function and others [83,107]. Inspired by their performances in the Clustering context, a recent research trend in biclustering regards the exploitation of *Graphical Models* [8, 72, 131]. Often, Computer Science problems are formalized adopting a set of variables and a global objective function representing the relationships between variables. Such function, associated with an inference/optimization problem, can be exploited to obtain information on the data of interest. However dealing with the complete function can be complicate, whereas it is easier to deal with its decomposition into locally interacting factors. A graphical representation for such decomposition is called Graphical Model [67]. Graphical models have been already adopted in many scientific scenarios, such as Computer Vision [41], Biology [72], Speech Recognition [11], Tracking [119], and so forth. In the biclustering context, however, there exist only a few approaches investigating the potential of Graphical Models. Arguably, this is mainly due to the difficulty of designing Graphical Models with an effective trade-off between representation power and computability. On the one hand, we have to derive a decomposition of the function for the problem at hand, which should be descriptive enough to capture its nature (in this sense, the more complex the model, the better). On the other hand, we have to consider the computational feasibility of the resulting optimization/inference task, which highly depends on the structure of the model (in this case, the simpler, the better).

This thesis is inserted in this context, investigating the potential of Graphical Models based approaches in the biclustering scenario. In particular we derive three different class of approaches focusing on the class of *Factor Graphs* and *Bayesian Networks* [43]. We adopt two different resolution techniques to obtain information from the devised model, namely the *Max-Sum* algorithm (for Factor Graphs approaches) [45] and the *Expectation-Maximization* algorithm for the Bayesian Networks [42]. Both classes of algorithms involve the derivation and computation of rules to solve the model, and obtaining such rules represents a significant part of the effort of the thesis. In general, when dealing with Graphical Models, we interchange two distinct phases: i) the *design* phase where the model is devised and ii) the *resolution* phase where the chosen algorithm is applied to obtain information. Since the two stages are intrinsically connected, this results in different consecutive passages between the two phases before obtaining a effective Graphical Model.

We test the devised Factor Graphs and Bayesian Networks approaches on both synthetic benchmarks and real world datasets, proving that Graphical Model based techniques can significantly improve the state-of-the-art. Moreover, we exploited the Bayesian Network approach to face two Computer Vision scenarios (Multiple Structure Recovery and Stable Region Correspondences); remarkably, in these sce-

narios biclustering techniques were never applied before. In this case, the results obtained show that biclustering approaches are versatile and they can be promisingly adopted in various scenarios other than Bioinformatics.

In the following Sections we present the main contributions of the thesis, its organization and the related publications.

## 1.1 Contributions

The thesis contributions can be sketched in the following points:

- **Derivation of Factor Graph based approaches for biclustering:** we devise two novel algorithms exploiting Factor Graphs for the biclustering problem. Once designed, retrieving information from a Factor Graph is not trivial; particularly the algorithm we adopt leads to the analytical derivation of rules which must be computed iteratively. Current Factor Graphs approaches for biclustering have a common drawback, which is represented by the scalability issue [131]. In this thesis we make an important step toward the reduction of this limit devising Factor Graph based approaches which scale in a more reasonable way. We then test the performances of the proposed methods in comparison with the current state of the art involving both Factor Graph based approaches and not, demonstrating the proposed method potentials in both synthetic and real datasets in terms of solutions quality (and scalability). As a further contribution, following a recent trend involving time series datasets and biclustering, we provide a novel extension to the devised model allowing to analyse time series datasets; note that this represents the first Factor Graph based approach to the Time-Series biclustering problem.

- **Derivation of Bayesian Network approaches for Biclustering:** the second class of approaches faces biclustering exploiting a particular class of Bayesian Networks, and adopting a Probabilistic Low-Rank Matrix Factorization perspective [14, 60, 151]. Most of the techniques described in literature involving matrix factorization propose models including *sparsity*: the number of interesting points is limited compared to the dataset size. In fact, due to the enormous dataset size on which it is usually applied, sparsity plays a fundamental role in devising probabilistic biclustering algorithms. Differently from what proposed in literature, we introduce a novel probabilistic model which key ingredient is represented by a prior, called Spike and Slab (and mainly applied in linear regression), which usefulness in the biclustering problem was never investigated. As previously mentioned, Graphical Models resolution techniques are very sensitive to the model design, in a sense that their efficiency is influenced by the model topology. The introduction of Spike and Slab prior allow us to analytically derive efficient iterative updates

for the Expectation Maximization algorithm, leading to a highly scalable and effective technique. This is not the case of other competitors, such as [59], where the model design did not lead to an analytically derivable resolution algorithm. Also in this case we compared the performances with the-state-of-the-art, proving the reliability of our approach in different biclustering scenarios. As a further contribution, we also provide an extension of the model which allows the user to introduce a-priori information about rows/columns that should belong to the same bicluster, this information being crucial in many different scenarios.

- **Widening the range of applications for biclustering:** there exist a wide range of applications, where clustering techniques are commonly adopted, that could benefit from the biclustering analysis. In particular, in this thesis we investigate the usefulness of our biclustering solutions on two interesting problems of the Computer Vision scenario, namely the Multiple Structure Recovery and the Stable Regions Correspondences. We compare the obtained results with both classical clustering techniques (which represent the standard choices in these contexts) and with other biclustering competitors. We show that in both scenarios biclustering techniques favourably compare with the current clustering state-of-the-art. Moreover, we show that our approaches can improve the results obtained by other biclustering competitors.

## 1.2 Organization of the Thesis

The remainder of the thesis is organized as follows: Chap. 2 provides a detailed description of the biclustering problem together with the relative state-of-the-art. Chap. 3 describes the basics of the methods employed through the thesis. The techniques devised during the PhD, and their experimental evaluation, are described in Chap. 4 and Chap. 5. Chap. 6 presents the scenarios where we introduced biclustering. Finally, we draw the conclusions and analyse possible future directions in Chap. 7.

## 1.3 Publications

Some parts of this thesis have been published in conference proceedings or in international journals. The Factor Graph based techniques presented in Chap. 4 has been published on Pattern Recognition journal [28] and three conference proceedings (S+SSPR14 [31], IJCAI 15 [30] and ACM-SAC 17 whose proceedings are not available yet). The Bayesian Network approaches, described in Chap. 5, are currently under consideration for the publication in the Pattern Recognition journal

(second round of reviews). Concerning applications, the results on Multiple Structure Recovery described in Sec. 6.1 appeared in the S+SSPR16 proceedings [29]. Finally, material included in Sec. 6.2 is part of a paper under preparation to be submitted as soon as possible.

Summarizing, here is the list of publication obtained from the work done in this thesis:

1. M. Denitto, A. Farinelli, G. Franco and M. Bicego: "A binary Factor Graph Model to biclustering". Proceedings of IAPR Joint International Workshops on Statistical techniques in Pattern Recognition/Structural and Syntactic Pattern Recognition 2014 (S+SSPR 2014). LNCS 8621, P. Frnti, G. Brown, M. Loog, F. Escolano, M. Pelillo, Springer, pp 394-403 (2014)

2. M. Denitto, A. Farinelli, M. Bicego: "Biclustering gene expressions using factor graphs and the max-sum algorithm". Proceedings of the 24th International Conference on Artificial Intelligence 2015 (IJCAI). AAAI Press, pp 925-931 (2015)

3. M. Denitto, L. Magri, A. Farinelli, A. Fusiello, M. Bicego: "Multiple Structure Recovery via Probabilistic Biclustering". Proceedings of IAPR Joint International Workshops on Statistical techniques in Pattern Recognition/Structural and Syntactic Pattern Recognition 2016 (S+SSPR 2016).

4. M. Denitto, A. Farinelli, M.A.T. Figueiredo, M. Bicego: "A biclustering approach based on factor graphs and the max-sum algorithm", Pattern Recognition, Volume 62, February 2017, pp 114-124

5. M. Denitto, A. Farinelli, M. Bicego: "Biclustering of Time Series Data using Factor Graphs". Proceedings of ACM SIGAPP Symposium On Applied Computing (SAC BIO 2017).

6. M. Denitto, A. Farinelli, M.A.T. Figueiredo, M. Bicego: "Spike and Slab Biclustering", Pattern Recognition, *Under consideration*

During the Ph.D., I also participated to a project aimed at the investigation of quantum annealing solutions for the biclustering problem; which leaded to the following pubblication:

7. L. Bottarelli, M. Bicego, M. Denitto, A. Di Pierro, A. Farinelli: "A Quantum Annealing Approach to Biclustering". Proceedings of Theory and Practice of Natural Computing 2016 (TPNC 2016).

# Chapter 2

# The Problem: Biclustering

Pattern Recognition is the class of Computer Science approaches focusing on the ability of taking decisions by analysing characteristics describing the available data [12,129]. Specifically, Pattern Recognition aims at recovering models (the so called "patterns") underlying in a set of objects/points; this process is performed by reasoning on some measurable/detectable characteristics (called *features*) on the available data. As mentioned in the previous Chapter, Pattern Recognition approaches could be roughly divided in two main categories: i) *Classification* (or supervised learning) where the true nature (*label*) of each object is known and exploited to build algorithms capable to distinguish between objects of different classes; ii) *Clustering* (or unsupervised learning) where the labels are not known and the goal is to group objects in different *clusters* on the basis of a similarity criteria.

In the next section we focus on the latter category of approaches, providing the details about the class of clustering techniques known as *biclustering*.

## 2.1 From Clustering to Biclustering

In their seminal paper, Jain et al. define clustering as "the unsupervised classification of patterns (observations, data items, or feature vectors) into groups (clusters)" such that patterns in the same group are "similar" and patterns of different groups are "dissimlar" [64]. Due to the missing labels, this problem in intrinsically more complex than classification because validation is very difficult to be carried out. Furthermore the concept of "group" is often difficult to formalize. For instance given a set of objects to group, these could be clustered in different manners depending on the chosen *similarity criterion*, and the definition of *correct grouping* could be not so clear. Generally, clustering problems are described by a data matrix where each row represents a certain patterns to analyse and each column refers to a particular feature.

As pointed out by Madeira et al. [83], clustering algorithms run into a significant difficulty: underlying patterns could involve groups of rows only in *few specific*

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| *user-1* | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 |
| *user-2* | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 |
| *user-3* | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |

Figure 2-1: Biclustering on user-item matrices can retrieve information about users having common preferences only in small category of items (sharing common hobbies). In this case *user-1* and *user-2* are both into carpentry, and have different interests concerning music and movies. Due to the global lack of correlation, this information cannot be retrieved by classical clustering approaches.

*features*. For instance, in the gene expression scenario, the general understanding of cellular processes leads to expect subsets of genes (rows of the data matrix) to be co-regulated and co-expressed only under certain experimental conditions (columns of the data matrix), but to behave almost independently under other conditions. However the same behaviour can be found in many other contexts: in the field of recommender systems for example, let us imagine a binary matrix where an entry $(i, j) = 1$ if the user $i$ bought the object $j$ (Figure 2-1 sketches the example). The goal here is to find users having similar tastes, hence buying the same objects. However it is hard to expect that users act coherently among the whole list of objects provided by a certain shop. Whereas it is reasonable to assume that users sharing a common hobby (*e.g.*, carpentry) could have bought same objects in that category, and similarly, it is also reasonable to expect that the users would act independently for items not related to that category. Since classical clustering approaches exploit clues of *whole rows*, the lack of correlation in this case prevents clustering techniques to retrieve such *local* information (users sharing carpentry as hobby).

Hence, analysing *local patterns* allows to retrieve relationships that could not be apparent otherwise, introducing the concept of *biclustering* [19]. The difference between clustering and biclustering is elucidated in figure 2-2. In a bicluster, each row is selected exploiting the information contained only in a subset of the columns, and vice versa. The goal of biclustering techniques is thus to identify subgroups of rows and subgroups of columns, namely to reveal groups of rows that show similar behavioural patterns under a specific subset of the columns. Biclustering techniques are crucial for the following reasons:

- only small sets of rows can participate in behavioural patterns;
- similarly, behavioural patterns can involve only few conditions;
- a single row could participate in multiple behavioural patterns that may or not

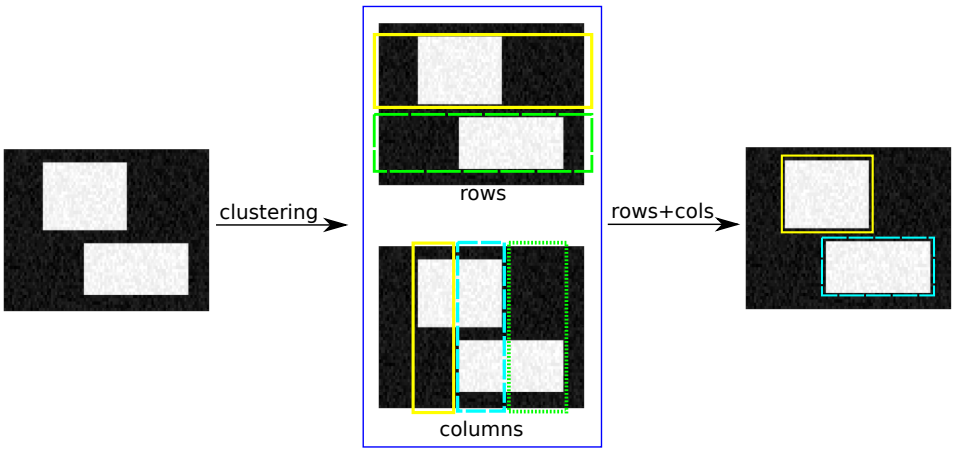



Figure 2-2: Clustering versus Biclustering. Given a general data matrix (left): on the one hand classical clustering approaches retrieve submatrices where a subset of rows behave coherently in all the columns (middle top), or vice versa (middle bottom); on the other hand biclustering techniques recover submatrices where a particular subset of rows behave coherently in a certain subset of columns, and vice versa.

be co-occur in all columns;

- similarly, a single column could be involved in different behavioural patterns that may occur or not occur in all rows.

- differently from clustering approaches, biclustering techniques are not exhaustive: there are rows/columns not assigned to a bicluster.

Even if biclustering was born and mainly applied in Bioinformatics scenario, it has been recently adopted in a wider range of applications that span from market segmentation (*e.g.,* to identify market segments among tourists who behave similarly) to text mining (where one example is provided by the identification of words and documents belonging to the same topics) [16, 33, 73, 91]. Biclustering is also widely known in literature as co-clustering or bi-dimensional clustering [32, 94, 108].

In the following Sections we formally define the biclustering problem and we illustrate the related work present in literature, explaining the differences between the principal class of approaches.

## 2.2 Biclustering definition

Formally, biclustering can be formulated as follows. Given a data matrix $A \in \mathbb{R}^{n \times m}$, let $N = \{1, \ldots, n\}$ denote the set of row indices and $M = \{1, \ldots, m\}$ the

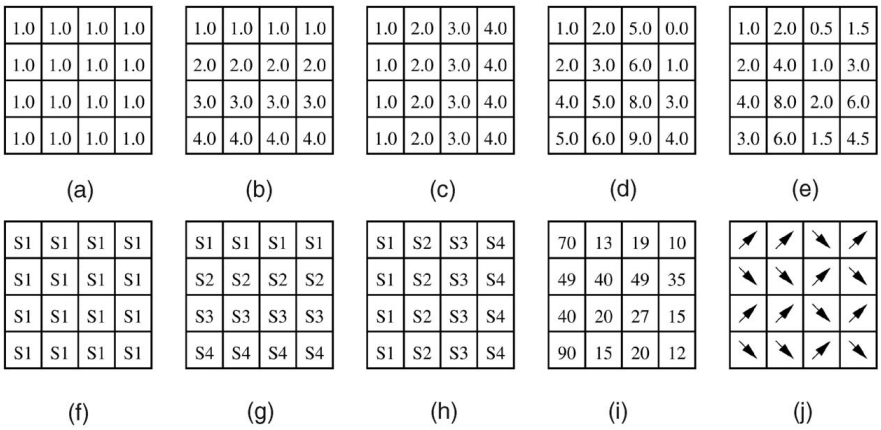



Figure 2-3: Examples of different types of biclusters. (a) Constant bicluster, (b) constant rows, (c) constant columns, (d) coherent values (addictive model), (e) coherent values (multiplicative model), (f) overall coherent evolution, (g) coherent evolution on the rows, (h) coherent evolution on the columns, (i) coherent evolution on the columns, and (j) coherent sign changes on rows and columns [83].

set of column indices. We denote as $A_{TK}$ the sub-matrix that includes the rows in $T \subseteq N$ and the columns in $K \subseteq M$. Using this notation, we can introduce the concepts of clusters and biclusters as:

- a *cluster of rows:* sub-matrix $A_{TM}$ where the rows $T = \{t_1, \ldots, t_p\}$ present coherent behaviour across the set of all columns; $T \subseteq I$ and $p \leq n$.

- a *cluster of columns* is defined as the sub-matrix $A_{NK}$ where the columns $K = \{k_1, \ldots, k_r\}$ present coherent behaviour across the set of all rows; $K \subseteq J$ and $r \leq m$.

- a *bicluster* is then defined as the sub-matrix $A_{TK}$ where the rows $T = \{t_1, \ldots, t_p\}$ present coherent behaviour across the set columns $K = \{k_1, \ldots, k_r\}$, and vice versa; $T \subseteq I$ , $p \leq n$, K$\subseteq J$ and $r \leq m$.

The available literature offers a wide range of coherence criteria, the choice of which critically influences the nature of the biclusters obtained [53, 57, 83, 98]. Broadly, biclusters can be divided in four classes [83]:

- *Bicluster with constant values*: is defined as a sub-matrix $A_{TK}$ where

$$a_{tk} = \alpha \qquad \forall t \in T, \forall k \in K$$

  $\alpha$ is the value present the entries of the sub-matrix, as represented in the figure by 2-3(a);

- *Bicluster with constant values on rows or columns*: is defined as a sub-matrix $A_{TK}$ where every rows (or columns) is one of the following

$$a_{tk} = \alpha + \beta_t \tag{2.1a}$$
$$a_{tk} = \alpha \times \beta_t \tag{2.1b}$$
$$a_{tk} = \alpha + \delta_k \tag{2.1c}$$
$$a_{tk} = \alpha \times \delta_k \tag{2.1d}$$

  where $\alpha$ is the value of the bicluster and $\beta$ is the row ($\delta$ for columns) adjustment; it could be an *additive* adjustment (Eq. 2.1a and Eq. 2.1c) or a *multiplicative* adjustment (Eq. 2.1b and Eq. 2.1d), as described by Fig. 2-3(b,c);

- *Bicluster with consistent values*: is based on an additive model and is defined as a sub-matrix $A_{TK}$ where the value $a_{tk}$ can be predicted by

$$a_{tk} = \alpha + \beta_t + \gamma_k$$

  where $\alpha$ is the value of the bicluster, $\beta_t$ is the row adjustment coefficient for $t \in T$ and $\gamma_k$ is the adjustment coefficient for $k \in K$, as shown in 2-3(d,e);

- *Bicluster with coherent evolutions*: a mathematical definition does not exist but can be roughly defined as a set of rows (or columns) that preserve some sort of order or relationship across the columns (or rows). An example is a bicluster in which there exists a permutation of columns that perform a strictly increasing trend in rows values:

$$a_{tp} \leq a_{tp-1} \leq \dots a_{t1}$$

  with $p$ number of columns in the bicluster, as shown by 2-3(f,g,h,i,j).

Please note that the first three classes involve data matrix numeric values and can be retrieved by analysing points sharing similar behaviors according to their values. The last class aims at finding coherent behaviors considering the points trend which can be expressed by symbols (as the increasing/decreasing order or positive/negative changes relatively to a normal value).

Concerning biclusters size and shape, similar to clustering, it is really difficult to assume fixed size and shapes for biclusters, as these can significantly vary depending on the scenario. Plus, an important characteristic of the biclusters shape concerns the overlap: overlapping biclusters are biclusters that share some data matrix entries. Figure 2-4, taken from the survey [83], reports typical examples on the bicluster structure and shape.

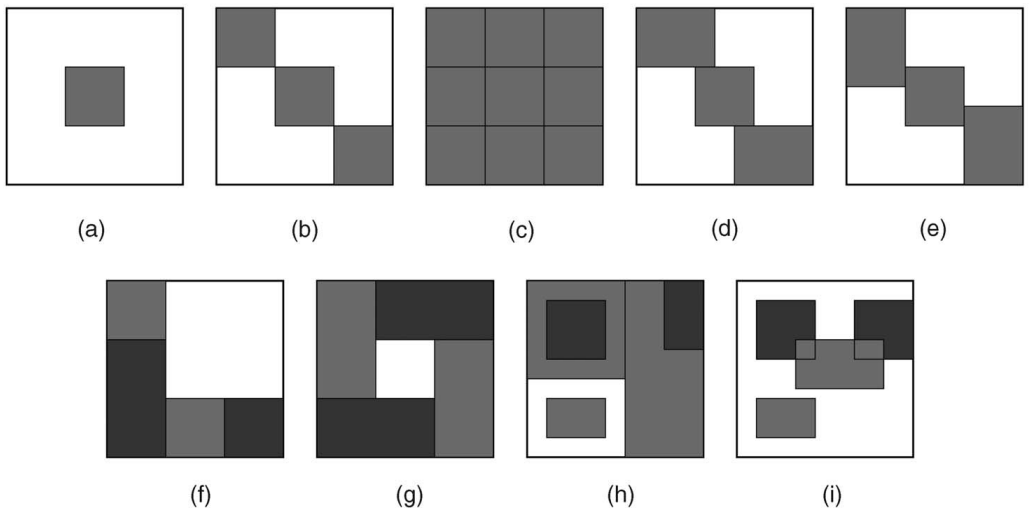



Figure 2-4: Biclusters structure. (a) Single bicluster, (b) exclusive row and column biclusters, (c) checkerboard structure, (d) exclusive rows biclusters, (e) exclusive columns biclusters, (f) nonoverlapping biclusters with tree structure, (g) nonoverlapping nonexclusive biclusters, (h) overlapping biclusters with hierarchical structure, and (i) arbitrarily positioned overlapping biclusters.

## 2.3 State of the Art

Due to its importance in many scientific domains, different approaches facing biclustering have been proposed in literature, each one characterized by different features, such as computational complexity, effectiveness, interpretability, optimization criterion and others ( [39, 83, 91, 107]). Due to the wide variance of techniques that have been developed, many taxonomies have been introduced [83, 91, 98]. In what follows we report the one proposed in [98] where biclustering methods are divided into four main classes which are described in the following paragraphs.

**Correlation maximization methods** This class of approaches seeks for sub-matrices where the rows (or columns) correlate highly among the columns (or rows). This requires the definition of a correlation criterion with which validate the results. Particularly, the algorithm proposed by Cheng and Church [19] iteratively searches for this type of biclusters by imposing the condition that the mean square residue is below some parameter $d$. In more detail, authors in [19] propose different algorithms all relying on the same basic idea: starting from the whole data matrix they iteratively delete/add (the first step is forced to be a deletion) one column (or row) at a time, keeping as current solution the operation that minimizes the mean square residue. In each iteration the previous solution is refined and the algorithms stop once the current solution has a mean squared residue below a certain threshold

taken set as parameter. The FLexible Overlapped biClustering (FLOC) technique, proposed by Yang et al. [145], is another example of correlation maximization technique. The FLOC biclustering algorithm starts from a set of seeds (initial biclusters) and carries out an iterative process to improve the overall quality of the biclustering. At each iteration, each row and column is moved among biclusters to produce a better biclustering in terms of lower mean squared residues. The best biclustering obtained during each iteration will serve as the initial biclustering for the next iteration. The algorithm terminates when the current iteration fails to improve the overall biclustering quality.

**Variance minimization methods** Biclusters retrieved by these techniques are obtained by minimizing the row variance along the columns, or vice-versa [98]. A typical example of this class of approaches is represented by the xMOTIF algorithm presented in [93]. Briefly, authors in [93] propose to randomly select one row and a random subset of columns where that rows contain nearly the same values; they thus look for others rows having a nearly equal behaviour in that particular subset of columns. The procedure is then repeated to retrieve different biclusters and these are kept/discarded if the bicluster cardinality is above/under a certain threshold. Another algorithm belonging to this class is the one proposed by Hartigan in [56] and then re-implemented in several versions (such as [136, 148]). The original algorithm consists in the iterative splitting of the data matrix minimizing the partitions variances on both rows and columns. The splitting stops once the variance reduction due to further splitting is under a certain threshold by chance.

**Two-way clustering methods** To retrieve biclusters, this category of techniques combines the results obtained by performing clustering on rows and columns separately. For instance, the algorithm proposed by Getz et al. in [48] repeatedly performs clustering on the rows and samples whilst the stable clusters of rows are adopted as the seeds for the columns clustering, and vice versa. We also proposed a Coupled Two-Way Clustering [34] where we perform clustering iteratively on rows and columns separately (inspired by [48]), then combining the results to obtain relevant biclusters. Another example is an algorithm proposed in [122], which initiates the analysis by clustering the rows to a predefined number of groups, and then clusters the columns by featuring each group of rows. Next, the algorithm selects the heterogeneous groups of rows and columns which best represent the distribution of the data, and the whole process is repeated on the selected rows and columns, until the predefined termination condition is satisfied.

**Probabilistic/generative methods** As suggested by the name, such techniques exploit probabilistic models and inference tools to retrieve biclusters from a given data matrix. Due to huge number of tools provided by probabilistic studies, a wide

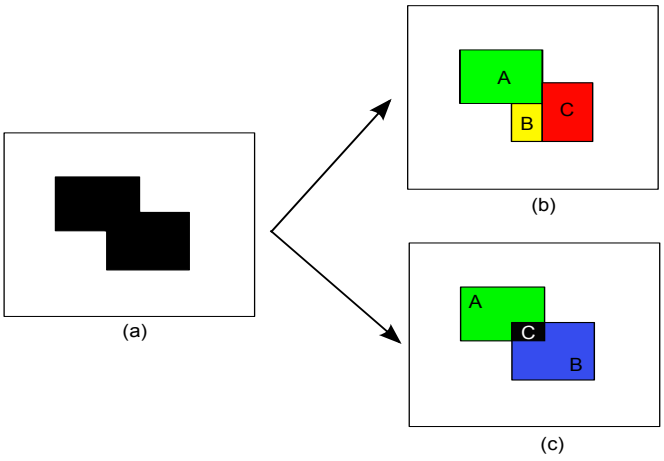



Figure 2-5: A demonstration of how the biclustering analysis on the same data matrix (a) can give two different and valid sets of biclusters according to the algorithm used. The solution shown in (b) can be obtained using an algorithm that recognize only not-overlapped biclusters; solution shown in (c) can be provided by an algorithm that recognizes overlapped biclusters, in this case the solution set is made by $\{A,B,C\}$ with $C = A \cap B$.

range of different probabilistic biclustering approaches have been proposed in literature. In fact, there are algorithms exploiting graph-based models (such as [121]), approaches adopting Gibbs sampling (such as [115]) or Markov chains ( [110]). Moreover, a recent trend of techniques exploits the results obtained in the field of Matrix Factorization and investigates the usage of such methods in the biclustering scenario. Among these we find the recent Factor Analysis for BIclustering Acquisition [59]. This category of approaches is analysed in details in Chap. 5.

As in clustering, due to the vague definition of the problem and due to the different facets relying behind the concept of similarity, two different biclustering approaches may provide two different solutions of the same data matrix. An example is shown in figure 2-5.

# Chapter 3

# Background: Graphical Models

In this Chapter we present the background knowledge behind *Graphical Models*; which are the basis of the algorithms devised during the Ph.D..

The idea of modelling systems using graphic representations is common to several scientific areas (e.g. circuit diagrams, signal flow diagrams and biological process diagrams). According to Lauritzen, models which can be described by a graphical representation take the name of *graphical model* [77]. In more detail, citing Jensen: "Graphical models are communication languages. They consist of a qualitative part, where features from graph theory are used, and a quantitative part consisting of potentials, which are real-valued functions over sets of nodes from the graph" [67]. Different taxonomies and formalisms have been proposed in literature concerning the different types of Graphical Models [12, 43, 71]; each one slightly differs from the others on the representation schemes or in the categories subdivision. Among these, due to the suitability with the proposed approaches, in what follows we present the taxonomy introduced by Brendan J. Frey in the book "Graphical Models for Machine Learning and Digital Communication" [43]. To better introduce the differences among Graphical Models it is convenient to describe them from a probabilistic perspective, thus in what follows we present a brief overview on probability and graph theory.

## 3.1 A probabilistic perspective

Uncertainty is a key concept in the field of Pattern Recognition deriving from noise on measurements, as well as the finite size of data sets. Probability theory provides a consistent framework for the quantification and manipulation of uncertainty and forms one of the central foundations for Pattern Recognition. To describe real scenarios in probability terms we adopt what are known as *random variables*: variables whose possible values depend on a set of uncertain outcome events (*i.e.,* a variable describing the possible outcomes of a coin toss, is a random variable). Considering two discrete random variables $x$ and $y$, even the most complex probabilistic manip-

ulations can be expressed in terms of the two elementary rules of probability, know as the *sum rule* and *product rule*:

$$\textbf{sum rule} \qquad p(x) = \sum_{y} p(x, y) \tag{3.1}$$

$$\textbf{product rule} \qquad p(x, y) = p(x|y)p(y) \tag{3.2}$$

where the *conditional probability* $p(x|y)$ reflects the probability for the event x to occur, given the event y. The sum rule is also widely known as *marginalization* and it is the sum over all possible values of $y$ (an integral is needed if $y$ is continue instead of discrete). As an example of application, we can derive the *Bayes' rule* by applying two times the product rule obtaining:

$$\textbf{Bayes' rule} \qquad p(x|y) = \frac{p(y|x)p(x)}{p(y)}. \tag{3.3}$$

These simple equations provide all the ingredients of *probabilistic generative models* [68]. Generative modelling aims at the statistical formalization of models explaining the observed data, as a combination of hidden variables (representing the causes) coupled with conditional interdependencies [68]. Many different tasks (such as inference and learning, which are presented in the following sections) can be performed by the algebraic manipulation of the previously presented equations (examples and details of such tasks are presented in the following sections). However, it is highly advantageous to augment the analysis using graphical representations of probability distributions, leading to the so called *probabilistic graphical models*. Borrowing elements of *graph theory*, a *graph* comprises *nodes* connected by *edges* [43]. In a probabilistic graphical model, each node represents a random variable, and the links express probabilistic relationships between these variables. Graphical models offer several useful properties:

1. they clearly highlight the features of a generative model structure, and this can be exploited to design and motivate new models;

2. some properties of the model, such as conditional independence properties, can be obtained by inspecting the graph;

3. graphical manipulations on the structure of the model, in which underlying mathematical expressions are carried along implicitly, can be lead to complex computations, such as inference and learning.

As proposed by Brendan J. Frey in [43], probabilistic Graphical models can be divided in three major classes: *Markov Random Field* (or *undirected graphical models*), *Bayesian Networks* (or *directed graphical models*), and *Factor Graphs*. The techniques devised and adopted in these thesis belong to the latter two categories, which are presented in the following Sections.

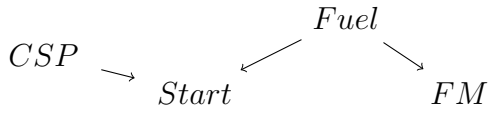



Figure 3-1: Causes representation for the car-start problem.

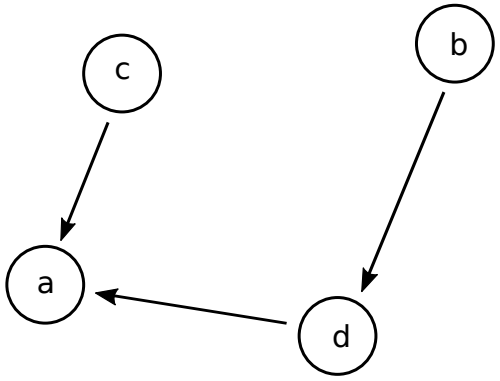
## 3.2 Bayesian Networks

To introduce Bayesian Networks, it is useful to start with a toy-example (taken from [67]).

Suppose we want to represent the car start problem with a graph, and for simplicity, suppose there exist only the following variables: *Fuel, Start, Clean Spark Plugs (CSP) and Fuel Meter Standing (FMS)*. We also assume the variables domain to be $\{yes, no\}$ for the first three and $\{full, half, empty\}$ for the last one. In this context we can introduce the concept of causality as the agent or efficacy that connects one process (the cause) with another process or state (the effect), where the first is partly responsible for the second, and the second is dependent on the first [27]. For instance, it is easy to see that the *state* (value assumed by the variables) of *Fuel* and *CSP* have a causal impact on the variable *Start*, (*i.e.,* if the state of *Fuel* is *no*, thus it is reasonable to expect that also the state of *Start* would be *no*). Similarly, the state of *Fuel* has an impact on the state of *FMS*. In Fig. 3-1 we report what is known as causal network [68] and it sketches the causal relations between the defined variables. Such network allows us to think about situations causing a car-start problem. For example, if we realize to have a start problem, looking at the diagram we can see that possible causes rely on *Fuel* and *CSP*. Hence, our certainty of *Fuel* being *no* is increased, thus we expect *FMS* to be in state *empty*. However, by checking the fuel meter, we read *half* and by reasoning backward, we must reconsider our certainty about *Fuel* being *no*. The obvious conclusion of this reasoning is: "Lack of fuel does not seem to be the reason for my start problem, so most probably spark plugs are not clean". This *qualitative* process is called *Causal Reasoning* and it is based on the concept of certainty/uncertainty and causality. However, causal relations also have a *quantitative* side, or *strength*, which is expressed by attaching numbers to the links. Given the notions introduced in the previous section, (conditional) probabilities represent a reasonable choice for the strength. A Directed Acyclic Graph where the variables are connected if, and only if, there exist a relationship of conditional probability between them takes the name of *Bayesian Network*.

A Bayesian Networks graphically represents how the global joint distribution decomposes as a product between the conditional probabilities, this is also known as the *Chain Rule* for Bayesian Networks. More formally, a Bayesian Network has:

- a set of *variables* and a set of *directed edges* between variables




$$P(a, b, c, d) = P(a|c, d)P(d|b)P(b)P(c)$$

Figure 3-2: An example of Bayesian Network.

- each variable has a finite set of mutually exclusive states (discrete variables); or, in the of continuous variables, they have infinite states whose uncertainty is described by a probability density function
- variables and edges must compose a DAG
- for each variable $x$ with parents $y_1, \ldots, y_n$ we specify the conditional probability $p(x|y_1, \ldots, y_n)$.

Hence, given a set of random variables $\mathbf{x} = \{x_1, \ldots, x_n\}$, a Bayesian Network on $\mathbf{x}$ describes:

$$p(\mathbf{x}) = \prod_{i=1}^{n} p(x_i|x_{pa_i})$$

where $x_{pa_i}$ is the set containing the parents of $x_i$. Please note that the definition of Bayesian Networks does not rely on the concept of causality, and there is no requirement that the links represent causal impact [67]. An example of Bayesian Network is provided in Fig. 3-2.

Turning back to the car-start example, the relative Bayesian Network is depicted in Fig. 3-3, and it expresses the following joint probability:

$$\begin{aligned} &P(Fuel,FM, CSP, Start) = \\ &\qquad P(Start|Fuel, CSP)P(FM|Fuel)P(Fuel)P(CSP). \end{aligned} \tag{3.4}$$

Introducing the probabilities needed to analyse the network, we assume the prior probabilities for ***Fuel*** and ***CSP*** to be:

$$P(Fuel) = (0.98, 0.02), \qquad P(CSP) = (0.96, 0.04);$$

whereas the conditional probabilities for $P(FM|Fuel)$ and $P(Start|Fuel, CSP)$

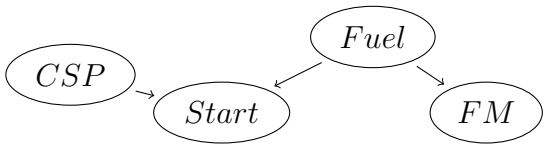



Figure 3-3: Bayesian Network for the car-start problem.

| | $Fuel = yes$ | $Fuel = no$ |
|---|---|---|
| $FM = full$ | 0.39 | 0.001 |
| $FM = half$ | 0.6 | 0.001 |
| $FM = empty$ | 0.01 | 0.998 |

Table 3.1: Probabilities for $P(FM|Fuel)$.

are reported in Tab. 3.1 and Tab. 3.2. Generally such probabilities are not known and we need to estimate them. In this scenario, typical pattern recognition strategies exploit the so-called "learning from examples" paradigm: a large set of objects or instances of the problem is acquired and used to learn the model parameters.

The joint probabilities described by Eq. 3.4 can be exploited to retrace the previously performed reasoning, supported by some quantitative information. For simplicity in Tab. 3.3 and 3.4 we report the results obtained by computing Eq. 3.4. Hence, realizing to have a car-start problem means that we observe the state of $Start$ being $no$. To understand the reason we thus need to compute the probabilities of the possible causes: $P(CSP, Start = no)$ and $P(Fuel|Start = no)$. Thus, by marginalizing $FM$ and $Fuel$ in Tab. 3.4 we obtain $P(CSP, Start = no) = (0.02864, 0.03965)$ and, by dividing with $P(Start = no)$ (which is the sum of the two numbers in $P(CSP, Start = no)$) we obtain the conditional probability $P(CSP|Start = no) = (0.42, 0.58)$. This last operation is not other than a *normalization* to obtain values summing up to one. Similarly, we can easily calculate $P(Fuel|Start = no) = (0.71, 0.29)$. Then, by reading the Fuel meter we obtain the information that $FM = half$, hence we should limit the calculation context to $FM = half$ and $Start = no$. The related probabilities are the one presented in the second column of in Tab.3.4, hence by marginalizing and normalizing we get $P(Fuel|Start = no, FM = half) = (0.999, 0.001)$ and $P(CSP|Start = no, FM = half) = (0.196, 0.804)$. The calculus provides an accurate estimate and we can state that the conclusion drawn previously is reasonable and reliable.

Commonly, Bayesian Networks are exploited in much more complex scenarios than the simple car-start problem just presented. In particular, concerning real case situations it is useful to introduce *hidden/latent* variables: these variables are meant to represent latent causes that influence the *visible* variables (the available data) [43]. Graphically, to distinguish between them we adopt filled circles to represent visible variables and empty circles to represent hidden variables. Furthermore,

| | $Fuel = yes$ | $Fuel = no$ |
|---|---|---|
| $CSP = yes$ | (0.99,0.01) | (0,1) |
| $CS = no$ | (0.01,0.99) | (0,1) |

Table 3.2: Probabilities for $P(Start|Fuel, CSP)$.

| | $FM = full$ | $FM = half$ | $FM = empty$ |
|---|---|---|---|
| $CSP = yes$ | (0.363,0) | (0.559,0) | (0.0093,0) |
| $CSP = no$ | (0.00015,0) | (0.00024,0) | $(3.9 * 10^{-}6,0)$ |

Table 3.3: Joint probabilities for $P(Fuel, FM, CSP, Start = yes)$. The numbers represent (Fuel = yes, Fuel = no).

we assume that a particular visible variable is continue and it is generated by a parametric distribution, the information about parameters can be inserted in the model through shapeless nodes containing the parameter name. In Fig. 3-4 we provide an example of a simple Bayesian Network where the observable variable $x$ is generated by choosing between a couple of Gaussian distributions having respectively $\mu_1, \sigma_1$ and $\mu_2, \sigma_2$ as parameters. The distribution choice is guided by the hidden variable $h$ which derives from a *Bernoulli* distribution having $\alpha$ as parameter.

Different tasks have been introduced in analysing Bayesian Networks, all these can be divided in two main categories: *inference* and *learning*.

**Inference on Bayesian Networks** Considering a set of random variables $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ that covary according to a joint distribution $P(x_1, x_2, \ldots, x_n)$. For any two subsets of variables $\mathbf{x}^1 \subseteq \mathbf{x}$ and $\mathbf{x}^2 \subseteq \mathbf{x}$, we refer to a decision based on $P(\mathbf{x}^1|\mathbf{x}^2)$, as *probabilistic inference*. We can think of inference as the action of querying a trained Bayesian Network: a Bayesian Network where all the probabilities and parameters have been estimated. The quantitative reasoning proposed for the car-start problem is a clear example of inference on a trained Bayesian Network. Many different approaches for inference have been proposed in literature. *Belief Propagation* (also known as *sum-product*) [77, 100] is one of the first and most famous techniques proposing to propagate information between connected nodes to perform inference. *Monte Carlo* inference [54] (comprising the notorious *Gibbs Sampling*) makes use of pseudo-random numbers to perform inference on a Bayesian Networks [43]. Another category of inference approaches is represented by the *Variational inference* [70, 135] which adopts a nonstochastic technique to directly address the inference quality.

| | $FM = full$ | $FM = half$ | $FM = empty$ |
|---|---|---|---|
| $CSP = yes$ | (0.00367,$1.9 * 10^{-}5$) | (0.00564,$1.9 * 10^{-}5$) | ($9.4 * 10^{-}5$,0.0192) |
| $CSP = no$ | (0.01514,$8 * 10^{-}7$) | (0.0233,$8 * 10^{-}7$) | (0.000388,0.00798) |

Table 3.4: Joint probabilities for $P(Fuel, FM, CSP, Start = no)$. The numbers represent (Fuel = yes, Fuel = no).

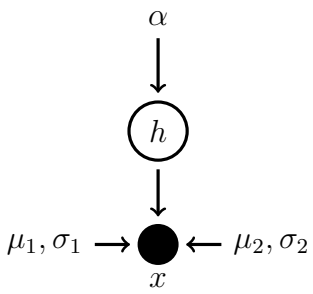



Figure 3-4: Example of Bayesian Network where an observed variable $x$ is generated from one of two Gaussians with different parameters chosen through a latent variable $h$, and $\alpha$ indicates the probability to peek from one of the two Gaussians.

### 3.2.1 Learning on Bayesian Networks: The Expectation- Maximization Algorithm

Complementary to probabilistic inference, we refer to *learning* as the procedure aiming at the estimation of the model probabilities and parameters. Resembling the car-start problem, the procedures allowing us to retrieve the probabilities in Tab. 3.2 and 3.1 belong to the class of learning approaches. In the learning phase, we want to estimate plausible configurations of the model parameters. Learning is possible provided that we are given several examples: the is that there exists a certain setting of the parameters that produced the observed training data.

In what follows we present the Expectation-Maximization (EM) algorithm, which is the learning algorithm we adopted to estimate the parameters of the devised models.

#### Expectation - Maximization

Given a Bayesian Network where where $\mathbf{z}$ represents the set of the missing/hidden variables, $\mathbf{x}$ describes the set of visible variables and $\theta$ is the set of parameters. Expectation Maximization is a class of iterative algorithms designed to obtain a marginal maximum likelihood estimate $\hat{\theta} = \arg\max_\theta P(x|\theta)$, where the marginal likelihood results from marginalizing out a set of missing/hidden/latent variables $\mathbf{z}$, *i.e.*, $P(\mathbf{x}|\theta) = \int P(x, \mathbf{z}|\theta)\, d\mathbf{z}$ (with summation rather than integration, if $\mathbf{z}$ is discrete). The algorithm alternates between two steps:

**E-step:** computes the conditional expectation of the complete log-likelihood, given the current parameter estimate $\hat{\theta}^{(t)}$ and the observed data $\mathbf{x}$, the so-called Q-function:

$$Q(\theta, \hat{\theta}^{(t)}) = \mathbb{E}_{\mathbf{z}} \left[ \log P(\mathbf{x}, \mathbf{z}|\theta) | \mathbf{x}, \hat{\theta}^{(t)} \right].$$

**M-step:** updates the parameter estimate by maximizing the Q-function:

$$\hat{\theta}^{(t+1)} = \arg\max_{\theta} Q(\theta, \hat{\theta}^{(t)}).$$

Briefly, this function represents the expected value of the complete log-likelihood, expectation taken with respect to the missing variables; and maximizing it provides a new parameters estimation.

The algorithm is initialized by choosing some starting value $\theta^0$ for the parameters, then at each iteration the current estimate $\theta^{old}$ is updated by a pair of successive E and M steps obtaining $\theta^{new}$. In the E-step we calculate the posterior distribution of the latent variable $P(\mathbf{z}|\mathbf{x}, \hat{\theta}^{old})$, given the current parameters estimate; this is called $Q(\theta, \theta^{old})$. Inside the M-step we maximize the previously obtained $Q$ to retrieve the revised parameter estimate $\theta^{new}$:

$$\theta^{new} = \arg\max_{\theta} Q(\theta, \theta^{old}).$$

Shortly, the EM algorithm is composed by four parts:

1. Initialize $\theta^{old}$ to $\theta^0$
2. **E-Step**: calculate $Q(\theta, \theta^{old}) = \mathbb{E}_{\mathbf{z}}[\log(P(\mathbf{x}, \mathbf{z}|\mathbf{x}, \theta^{old}))]$
3. **M-Step**: obtain $\theta^{new} = \arg\max_{\theta} Q(\theta, \theta^{old})$
4. Check for convergence, if not $\theta^{old} \leftarrow \theta^{new}$ and repeat from Item 2.

One of the main advantages in adopting EM algorithms to estimate the parameters is that, although it leads to an iterative algorithm, each iteration is guaranteed to increase the log-likelihood and the algorithm is guaranteed to converge to a local maximum of the likelihood function.

However this is not always the case. In dealing with complex models, there are situations where computing the expectation yielding the Q-function may not be trivial, as it may involve intractable integration. To face this instances it is common to exploit approximations of the intractable integrals, leading to rough estimation of the parameter set. This class of algorithms where the E step is not derived analytically takes the name of *Variational EM* algorithms [44]. Another obstacle that can arise in dealing with EM algorithms concern the maximization in the M-Step. In fact it is not obvious that the resolution of the integral in the E-step provides to

an exactly computable maximization step. In contrast, it is common to resort to an approximate parameter maximization. These approaches are known as *Generalized EM*. Specifically, in Chap 5 we exploit this latter category of EM.

In what follows we introduce the other class of approaches belonging to Graphical Models that we adopted in the biclustering context: *Factor Graphs*.

## 3.3 Factor Graphs

As previously presented, Bayesian Networks allow to express the joint distribution of several random variables as a factorization of functions over a subset of variables. This decomposition leads to more efficient algorithms in retrieving information from the devised model in both inference and learning tasks; contributing to a wide spread of these techniques in the most different scenarios [12, 66, 95].

To extend this idea Brendan J. Frey introduced a novel formalism defining the so called *Factor Graphs* [45]. Factor graphs represent a generalisation of Bayesian Networks in a sense that they can encode every Bayesian Network model, but they also adopt the same formalism to describe non-probabilistic scenarios. In fact, Factor Graphs can be used to describe the decomposition of any function that evaluates to a semi-ring [1] [43]. This is possible because the definition of local functions, called *factors*, is not constrained to be probabilistic.

Let $x$ be an $l$-dimensional vector of variables, and let $g(\mathbf{x})$ denote the value of some global function for a given configuration $(\mathbf{x})$. Factor Graphs are represented by a *bipartite graph* with variables and functions nodes. Formally, a factor graph includes two types of nodes: a collection of *variable nodes*, one for each of the $l$ variables ($x_1, ..., x_l$, usually drawn as circles) and a collection of *factor nodes* ($f_1, ..., f_d$, usually drawn as squares). The graph is bipartite in the sense that there are only edges/connections between factor and variable nodes (not between two variable or two factor nodes). The function $g$ can be represented by a Factor Graph if it can be factored as

$$g(\mathbf{x}) = \prod_{t=1}^{d} f_t(x_{S_t}), \tag{3.5}$$

where $S_t \subseteq \{1, ..., n\}$ is the subset of variable nodes indices to which factor node $f_t$ is connected, and $x_{S_t}$ denotes the corresponding sub-vector of $x$ (*i.e.,* the scope of $f_t$). The Fig. 3-5 represents an example of how the Bayesian Network for the car start problem depicted in 3-3 can be expressed as a Factor Graph.

As in the general case of Graphical Models, one of the operations for what Factor Graphs have been widely exploited is *inference*. As presented in the previous Section, we refer to inference as the procedure in which some of the variable nodes

[1] A semiring is a set together with two binary operators satisfying the following conditions: i) Additive associativity, ii) Additive commutativity, iii) Multiplicative associativity and iv) Left and right distributivity. For a detailed definition of semi-ring we refer interested readers to [75].

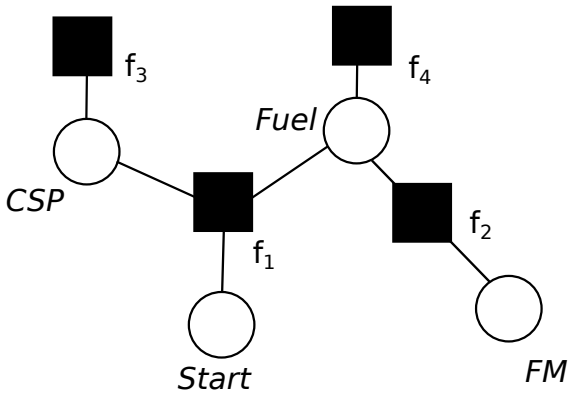



Figure 3-5: A possible Factor Graph for the car start problem. This is obtained by defining $f_1 = P(Start|Fuel, CSP)$, $f_2 = P(FM|Fuel)$, $f_3 = P(Fuel)$ and $f_4 = P(CSP)$. Thus obtaining the global objective function $g(Fuel, FM, CSP, Start) = f_1(Start|Fuel, CSP)f_2(FM|Fuel)f_3(Fuel)f_4(CSP)$.

are observed and we want to retrieve other variables distributions (*i.e.* posterior distributions). Another task where Factor Graph have been adopted, deeply connected to inference, concerns *optimization*: given a global objective function $g(\mathbf{x})$ we refer to optimization as the task allowing to retrieve the variable configuration(s) $\mathbf{x}^*$ such that the function $g$ is maximized (or minimized).

In general, there are two main classes of resolution algorithms for Factor Graphs: message passing algorithms and search based algorithms [5]. The main difference between these two classes is the way in which they tackle the complexity of the optimization: message passing algorithms aim at approximating the global objective function described with Factor Graphs by performing local optimization and propagating information throughout the graph; in contrast, search-based algorithms aim at efficiently exploring the possible variable assignments (*i.e.*, the search space) [5]. Thanks to their success in coding theory [5], message-passing approaches received significant attention [141] and this led the research community to develop a general message-passing procedure, known as *Generalized Distributive Law* (GDL). GDL can greatly reduce the number of operations required in a certain class of computational problem (for a detailed discussion of the GDL derivation and their applicability please refer to [4]). The GDL message passing scheme provides algorithms for both optimization and inference: two of the most famous algorithms belonging to GDL are the *sum-product* algorithm (used for inference problems)[2], and the *max-sum* algorithm (used for optimization problems). A crucial role for all GDL algorithms (i.e., max-sum, sum-product, max-product) is played by messages, which essentially contain all the relevant information for the receiving node. Particularly message-passing algorithms quickly became the standard choice for

[2]Shown to be equivalent to belief propagation [133] for probabilistic graphical models.

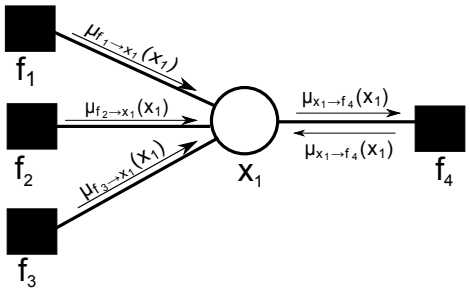



Figure 3-6: Example of computation of messages in a factor graph. In the computation of the $\mu_{x_1 \to f_4}(x_1)$ message we consider the value of all the messages entering in $x_1$ except $\mu_{f_4 \to x_1}(x_1)$

most applications, with the *max-sum* algorithm being one of the most famous instances in this class, having been used in many applications of Factor Graphs models [4, 12, 35, 41, 146][3]. Max-sum is a message passing scheme that exploits the structure of the graph, using the distributive law that holds between the max and sum operations (hence its name) [4]. It can be shown that if the Factor Graphs has no cycles (*e.g.*, if it is a tree), then max-sum algorithm is guaranteed to converge to the maximum of the objective function [140].

Specifically, depending whether a message is sent from a variable to a function, or vice versa, we can define two kinds of messages:

1. messages going from variable nodes to function nodes:

$$\mu_{x_m \to f_t}(x_m) \tag{3.6}$$

   where $x_m$ is the variable node which sends the message and $f_t$ is the function node that receives the message;

2. messages going from function nodes $f_t$ to variable nodes $x_m$:

$$\mu_{f_t \to x_m}(x_m) \tag{3.7}$$

   where $f_t$ is the node which sends the message and $x_m$ is the node that receive the message.

Typically, every message $\mu_{x_m \to f_t}(x_m)$ is calculated on the basis of the value of all messages entering in $x_m$, excluding the one sent by $f_s$ ($\mu_{f_t \to x_m}(x_m)$), as described in figure 3-6 (messages that start from functions are computed in a similar manner). The detailed operations for messages computation will be provided in the following section.

[3]This algorithm has been exploited also for biclustering, without using factor graphs [97].

| | $x = 0$ | x = 1 |
|---|---|---|
| $y = 0$ | 0.3 | 0.4 |
| $y = 1$ | 0.3 | 0.0 |

Table 3.5: Example of a joint distribution over two binary variables

### 3.3.1 Max-Sum algorithm

In this section we provide the details to implement the Max-sum algorithm, following the probabilistic presentation in [12]. Thus we suppose to deal with a probabilistic Factor Graph representing the joint distribution over a set of random variables, hence $g(\mathbf{x}) = P(\mathbf{x})$.

As previously mentioned, GDL algorithms exploit the distributive law that holds between a certain couple of operations (such as max and sum, sum and product or max and product) to retrieve information from a given Factor Graph. For example, Sum-Product (also known as *belief propagation* for the Bayesian Networks) provides the marginals over the variables. Specifically, it analyses the graph to obtain the marginals for every variable, this can be exploited to find the $x_i^*$ values maximizing the marginal. This would give the set of value that are *individually* the most probable. Sum product is not the focus of this thesis, and we refer interested readers to [12, 75, 80].

Given a factor graph, the max-sum algorithm aims at: 1) retrieving the configuration that maximizes the distribution $p(\mathbf{x})$, and 2) calculating the value of $p(\mathbf{x})$. A typical task when dealing with Factor Graphs is represented by optimization. Its goal is to find the set of values that *jointly* have the largest probability; that is, the vector $x^{\max}$ that maximizes the joint distribution, as in:

$$x^{\max} = \arg\max_{x} p(\mathbf{x}) \tag{3.8}$$

Then, the value of the maximum joint probability is given by:

$$p(x^{\max}) = \max_{x} p(\mathbf{x}) \tag{3.9}$$

Generally, the solution $x^{\max}$ is different from the set of variables configurations $x_i^*$ provided by sum-product. For instance, consider the distribution presented in the Table 3.5. The global maximum is obtained by setting $x = 1$ and $y = 0$, which provides the value $0.4$. However, the max marginal (the individual max) for $x$, which is obtained by summing over the possible values of $y$, is equal to $0.6$ (corresponding to $x = 0$). Similarly the marginal for $y$ is $0.7$ (corresponding to $y = 0$). Hence the marginals are maximized for $x = 0$ and $y = 0$, leading to $0.3$, which is different from the joint maximum value $0.4$.

To better understand the max-sum operations we can write the maximization in

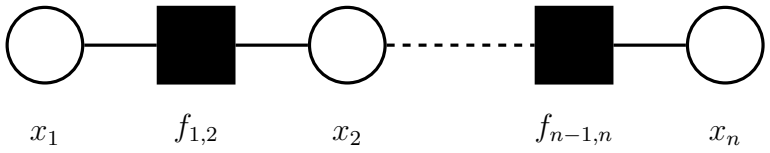



Figure 3-7: Example of factor graph constituted by a chain of variable nodes

terms of its components:

$$\max_{\mathbf{x}} = \max_{x_1} \dots \max_{x_n} p((x))$$

where $n$ is the number of variables, and then substitute for $p(\mathbf{x})$ using Eq. (3.5) as in

$$p(\mathbf{x}) = f_1(\dots) \cdots f_m(\dots).$$

We can now exploit the rule

$$\max(ab, ac) = a \max(b, c)$$

to exchange products and maximizations. Considering the example of a node chain where functions are linked only to consecutive variables (as described in figure 3-7), we obtain:

$$\max_{\mathbf{x}} p(\mathbf{x}) = \max_{x_1} \cdots \max_{x_n} [f_{1,2}(\dots) \dots f_{n-1,n}(\dots)] \tag{3.10}$$

$$= \max_{x_1} f_{1,2} \dots [\cdots \max_{x_n} f_{n-1,n}(\dots)]. \tag{3.11}$$

Note that exchanging maximum with products leads to more efficient computations.

In the case of a more complex factor graph, a tree for instance, we need to designate a particular node to be the root. Starting from the leaves we can propagate messages to the root, with each node sending its message once it has received all incoming messages from its neighbors. In this case the message sent by a node is the max of the product of all incoming messages. Once arrived to the root the final maximization is performed over the product of all messages arrived, thus obtaining the maximum for $p(\mathbf{x})$. This process is called *max-product*, note that at this stage messages have been sent only from leaves to the root.

In practice, products of many small probabilities can lead to numerical instability, so in many cases it is convenient to work with the logarithm of the joint distribution. Plus, since logarithm is a monotonic function max operator and logarithm operator can be interchanged:

$$\ln \max_{\mathbf{x}} p(\mathbf{x}) = \max_{\mathbf{x}} \ln p(\mathbf{x}) \tag{3.12}$$

Working with logarithms has the effect of replacing the products in the max-product algorithm with sums and so we obtain the *max-sum algorithm* which propagates the following messages:

$$\mu_{f\to x}(x) = \max_{x_1 \ldots x_n} \left[ \ln f(x, x_1, \ldots, x_n) + \sum_{m \in ne(f)\setminus x} \mu_{x_m \to f(x_m)}(x_m) \right] \tag{3.13}$$

$$\mu_{x\to f}(x) = \sum_{l \in \text{ne}(x)\setminus f} \mu_{f_l \to x}(x) \tag{3.14}$$

With these equations we can infer the value of the joint distribution propagating the messages from leaves to root. Concerning the practical usage of Max-Sum, where Factor Graphs present cycles, such messages are interchanged until a convergence criteria is met. Such criteria usually depends on the problem under analysis; common choices concern the variable configurations or the quantity of novel information introduced in consecutive messages. Once converged to find the configuration of variables that jointly maximizes $p(\mathbf{x})$, we assign to each variable the value maximizing the sum of all the incoming messages.

$$x^{max} = \arg\max_{x} \left[ \sum_{s \in \text{ne}(x)} \mu_{f_s \to x}(x) \right]. \tag{3.15}$$

Although the derivation of Max-Sum from a probabilistic perspective, differently from other Graphical Models, the probabilistic interpretation is not mandatory. In this thesis, we adopt this non-probabilistic view of Factor Graphs. As previously mentioned, Factor Graphs provides very flexible representation tools and resolution techniques, since there are no limitations on the form of the local functions or concerning the variable domains. However, the choices made when designing a Factor Graphs have a drastic effect on the difficulty of the resulting optimization problem. In particular, it is important to note that the derivation and the efficacy of these messages highly depend on the structure of the factors and the connections in the graph, see the maximization in Eq. 3.13.

A notorious example of a non-probabilistic Factor Graph, solved with Max-Sum, is the Affinity Propagation algorithm [41] which we present in the following Section. We present Affinity Propagation because it has been developed to solve the clustering problem. Further, it provides some interesting directions to follow in order to build an effective model.

## 3.4 An Example: Affinity Propagation

Affinity Propagation (AP) is a well known clustering technique recently proposed by Frey and Dueck [41]. The efficacy of this algorithm (in terms of clustering accuracy) and efficiency (due to the fast resolution) have been shown in many different clustering contexts [41].

The main idea behind AP is to perform clustering by finding a set of *exemplar points* that best represent the whole data set. Thus each point choses an exemplar on the basis of a similarity criteria, and points choosing the same exemplar belong to the same cluster. This is obtained by representing the input data as a factor graph where the objective function is then optimized by running the max-sum algorithm previously presented.

In particular, in Affinity Propagation the factor graph is composed by two parts: the first encodes the choice of the points and their exemplars via a binary matrix $C$, where an entry $C(i,j) = c_{i,j}$ is set to one if the point $i$ chooses $j$ as exemplar. This choice is ruled by the pairwise similarity values $s_{i,j}$, which define the similarity between each pair of points $i$ and $j$. The values $s_{i,i}$, given as an input, represent the *preference* for point $i$ of being itself an exemplar: such choice influences the final number of clusters, which is automatically found by the algorithm. The second part of the factor graph defines two constraints, which ensure to retrieve only valid solutions:

1. *1-of-N constraint*: every point has to chose one, and only one, exemplar. This can be represented by a function $I$ over $n$ nodes:

$$I_i = \begin{cases} 0, & \text{if } \sum_{i=1}^{n} c_{i,j} = 1 \\ -\infty, & otherwise \end{cases} \tag{3.16}$$

   where $n$ is the number of the points;

2. *Exemplar consistency constraint*: if a point is chosen as an exemplar by some other data point, it must choose itself as an exemplar. This constraint avoids circular choices ("a" chooses "b", "b" chooses "c", "c" chooses "a") and can be represented by a function $E$ over $n$ nodes:

$$E_j = \begin{cases} -\infty, & \text{if} \quad c_{jj} = 0 \quad \text{and} \quad \sum_{i=1}^{n} c_{i,j} \geq 1 \\ 0, & otherwise \end{cases} \tag{3.17}$$

   where $n$ is the number of data points.

Note that we have as many $I$ and $E$ functions as the number of data points in input. Figure 3-8 reports the factor graph used in AP. The objective function expressed by the AP factor graph is the sum of all the factors, i.e., the constraints expressed

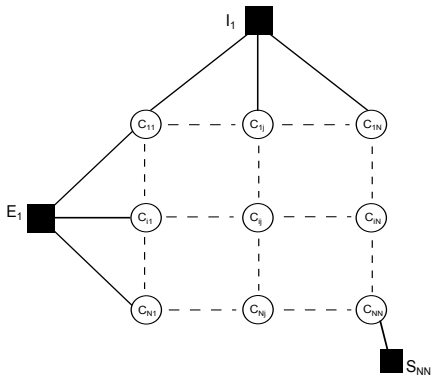



Figure 3-8: Factor Graph for Affinity Propagation

in Equations (3.16) and (3.17) and the sum of all similarity functions which are defined as the similarity value $s_{i,j}$ multiplied by the variables $c_{i,j}$.

$$F = \sum_{i=1}^{n} \sum_{j=1}^{n} s_{ij} \cdot c_{ij} + \sum_{i=1}^{n} I_i + \sum_{i=1}^{n} E_j \tag{3.18}$$

# Chapter 4

# Factor Graph based approaches for Biclustering

In this Chapter we present the relevant state-of-the-art concerning Factor Graphs techniques for the biclustering problem, focusing on the promising approach described by Tu et al. in [131]. Then, in Sec. 4.2 we describe our Factor Graph model solved with a Linear Programming technique. In Sec. 4.3 we present our most important contribution concerning Factor Graphs approaches for biclustering. Finally, we provides two variants extending this latter approach to other specific scenarios.

## 4.1 Related Work

The algorithm presented in [131] represents an exemplar based biclustering approach, hence it is herein referred to as *Exemplar-Based biclustering* (EB). As in clustering, exemplar-based techniques aim at assigning to each point an *exemplar* on the basis of a similarity criteria. Points choosing the same exemplar belong to the same cluster. However, differently from clustering scenario, authors in [131] considers each data matrix entry as a point to analyse. This is a reasonable choice since we are looking for coherent *sub-matrices* (instead of subsets of whole rows or columns as in clustering).

To cast this idea in a Factor Graph, authors couple to each point/entry an integer variable $c_{ij}$ indicating the index of its exemplar. As in Affinity Propagation the exemplar choices are guided by a similarity criterion which is encoded with factor $CP_{ij}$ taking different coherent values depending on the choice of the entry $(i, j)$. Nonetheless authors in [131] include an hard constraint to ensure that a group of entries choosing the same exemplar compose a bicluster; in fact, grouping similar entries could not lead to obtain sub-matrices as solutions.

Given a data matrix $A$ with $n$ rows and $m$ columns, this model leads to a compact Factor Graph having $n \times m$ integer variables (with values in $\{1, \ldots, nm\}$) and three different factors: one guiding the exemplar choices, one ensuring the ex-

emplar consistency choices (as in AP) and one ensuring the bicluster integrity. In detail, the proposed Factor Graph is sketched in Fig. 4-1 and it is composed by:

1. one *variable node* for each entry: it indicates the *index* of the chosen exemplar

2. one *Coherence and Prior* factor (**CP** factors): it encodes coherence information in the model. Depending on the coherence adopted we can vary the type of biclusters retrieved by this model.

3. one *exemplar consistency constraint* (**g** factors) for each couple of variables: it prevents circular choices and ensure choices consistency. Given two variables $c_{ij}$ and $c_{kl}$ such constraint is defined as:

$$g_{ijkl} = \begin{cases} -\infty & \text{if } c_{ij} = idx(k,l) \text{but} c_{kl} \neq= idx(k,l) \\ & \text{or } c_{kl} = idx(i,j) \text{ but } c_{ij} \neq= idx(i,j) \\ 0 & \text{otherwise} \end{cases} \tag{4.1}$$

   where $idx : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ is a function assigning to each possible couple $(i, j)$ a unique index $z$. With $1 \leq i \leq n$, $1 \leq j \leq n$ and $1 \leq z \leq nm$.

4. one *bicluster integrity constraint* (**f** factors) for each couple of variables: it ensures the results to have a full rectangular shape and hence to be biclusters. Given two variables $c_{ij}$ and $c_{kl}$ such constraint is defined as:

$$f_{ijkl} = \begin{cases} -\infty & \text{if } c_{ij} = c_{kl} \neq c_{kj} \text{ or } \neq c_{il} \\ & \text{or if } c_{il} = c_{kj} \neq c_{kl} \text{ or } \neq c_{ij} \\ 0 & \text{otherwise} \end{cases} \tag{4.2}$$

Given a variable, note that each message sent/received by that variable should contain a value for every possible assignment. In this case variables are integers going from one to the number of data matrix entries. It is reasonable to expect that this can lead to a scalability issue with the increasing of the data matrix dimensions. In fact, although the presented model is compact, max-sum messages update rules cannot be performed efficiently in this case. This is due to the design choice of adopting integer variables (instead of binaries as in the case of Affinity Propagation). Thus, authors exploited exact max-sum messages to analyse up to $10 \times 10$ matrices and, for larger matrices, they had to resort to an approximate max-sum algorithm and a greedy approach.

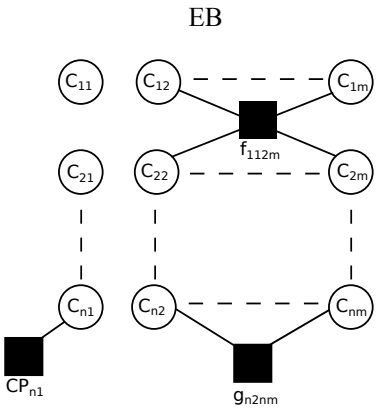



Figure 4-1: Factor graph for the Exemplar Based biclustering approach.

## 4.2 Biclustering Affinity Propagation (BAP)

In this section we propose a *binary* exemplar-based factor graph which consider as points the data matrix entries, as done in EB. We thus look for biclusters as sets of "coherent" entries of the matrix respecting the specific spatial constraint (*i.e.,* full rectangular shape) introduced in EB. We want to investigate the exploitation of binary variables because their binary nature was crucial in Affinity Propagation to derive efficient max-sum messages update rules. To obtain this, we re-define the factor graph of Affinity Propagation (described in Sec.3.4) as follows: we introduce one variable for each *pair of entries* of the data matrix $A$ encoding the exemplar choice; further, we introduce a constraint ensuring points belonging to the same cluster to be a bicluster.

In more detail, since our goal is to cluster the single entries of the data matrix; we encode the exemplar chosen by each entry of the data matrix $A$ with a four-dimensional Boolean matrix $C$, where an entry $C(i,j,t,k) = c_{ijtk}$ is 1 if the entry $a_{ij}$ chooses $a_{tk}$ as its exemplar. For interpretability reasons we index the second point position with a single value ($z = 1, 2, 3, \cdots, n \cdot m$) obtaining a three-dimensional structure $C(i,j,z)$. Now,

$$C(i,j,z) = \begin{cases} 1, & \text{if } (i,j) \text{ chose } z \text{ as exemplar} \\ 0, & \text{otherwise} \end{cases} \tag{4.3}$$

where: $1 \leq i \leq n$ , $1 \leq j \leq m$ , $1 \leq z \leq p$ , $p = n \cdot m$.

As in Affinity Propagation, the choice is guided by a similarity matrix $S$. Given a pair of entries $a_{ij}$ and $a_{tk}$, $S(a_{ij}, a_{tk})$ encodes the similarity between them. As for $C$, we rearrange this four-dimensional matrix in a three dimensional one $S(i,j,z)$, obtaining:

$$S(i,j,z) = Sim(a_{ij}, a_z) \tag{4.4}$$

where $Sim(\cdot,\cdot)$ represents the similarity criterion adopted to choose the exemplars. Following Affinity Propagation, we then add factors ensuring choices consistency, particularly:

- the constraint $I_{ij}$ (similar to (Eq. 3.16) forces one entry to choose only one exemplar). It is defined as

$$I_{ij} = \begin{cases} 0, & \text{if } \sum_{z=1}^{p} c_{ijz} = 1 \\ -\infty, & otherwise \end{cases} \tag{4.5}$$

- the constraint $E_z$ (similar to (Eq. 3.17)) prevents circular choices. It is defined as

$$E_z = \begin{cases} -\infty, & \text{if } \quad c_{idx^{-1}(z)z} = 0 \quad \text{and} \quad \sum_{i=1}^{n}\sum_{j=1}^{m} c_{ijz} \geq 1 \\ 0, & otherwise \end{cases} \tag{4.6}$$

  where $idx^{-1} : \mathbb{N} \to \mathbb{N} \times \mathbb{N}$ is the inverse function of $idx$ (defined in previous Section). It assigns to each value $z$ a couple of values $(i, j)$ such that $idx(i, j) = z$ and $idx^{-1}(z) = (i, j)$.

Next, we introduce an extra constraint, ensuring entries belonging to the same cluster to represent a bicluster. In this perspective, we observe that, given a certain value $z$, the bidimensional matrix

$$C(:,:,z) = \begin{bmatrix} c_{11z} & c_{12z} & \dots & c_{1mz} \\ c_{21z} & c_{22z} & \dots & c_{2mz} \\ \vdots & \vdots & \ddots & \vdots \\ c_{n1z} & c_{n2z} & \dots & c_{nmz} \end{bmatrix} \quad \text{with } 1 \leq z \leq n \cdot m \tag{4.7}$$

immediately summarizes the relationships between all the entries and $z$: in particular, $c_{ijz} = 1$ indicates that $a_{i,j}$ has chosen $a_z$ as its exemplar. This means that every matrix $C(:,:,z)$ represents a potential bicluster. However to be considered bicluster, such matrix should fulfil one of the two following conditions:

1. (trivial constraint) it should contain all zeros: there are no points choosing as exemplar the point $z$;

2. (bicluster integrity constraint) the coordinates of the entries with 1 (namely the coordinates of the entries in the bicluster) should represent *all* the points of a given subset of rows and columns: in simple words, after rows-columns re-arrangements, the ones in the $C(:,:,z)$ matrix should form a full rectangle (a rectangle with no zero elements).

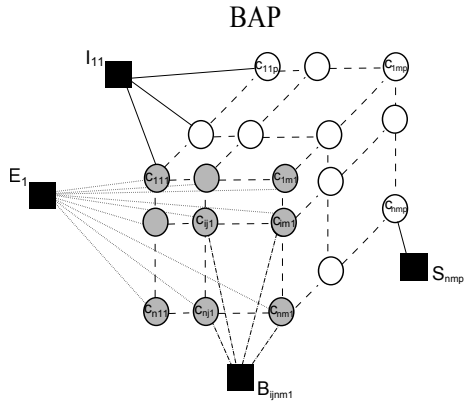



Figure 4-2: Factor graph for the Biclustering Affinity Propagation approach.

This can be ensured by defining a constraint for every 4 points of the matrix $C(:,:,z)$: if $c_{ijz}$ and $c_{tkz}$ are set to 1, then also $c_{ikz}$ and $c_{tjz}$ should be set to 1. More formally, the *bicluster integrity constraint* is defined as:

$$B_{ijtkz} = \begin{cases} -\infty, & \text{if} \quad c_{ijz} = 1, c_{tkz} = 1 \quad \text{and} \quad c_{ikz} \cdot c_{tjz} = 0 \\ 0, & otherwise \end{cases} \tag{4.8}$$

Notice that the function B is defined, for every sheet $z$, on all the possible pairs of points $c_{ij}$ and $c_{tk}$. Given these ingredients, which represent the Factor Graph in Fig. 4-2 ,the objective function is defined by the sum of the intra-biclusters similarity (via the matrix $C$ and $S$) and the constraints ($I$, $E$, and $B$):

$$F = \sum_{i,j,z} c_{ijz} \cdot s_{ijz} + \sum_{i,j} I_{ij} + \sum_{z} E_z + \sum_{z,i,j,t,k} B_{ijtkz} \tag{4.9}$$

where: $1 \leq i \leq n, 1 \leq j \leq m, 1 \leq z \leq n \cdot m, 1 \leq t \leq n$ and $1 \leq k \leq m$.

### 4.2.1 Model Optimization

In AP the binary nature of the nodes in the graph was crucial to calculate an approximation of the maximum through the max-sum algorithm. However, the biclustering integrity constraint (defined over every pair of entries of the matrix) induces a high number of cycles in the graph, and it is well known that the performances of the max-sum algorithm degrade in such conditions [140]. Therefore we follow an alternative route, giving a linear formulation of the objective function, and we adopt linear programming (LP) techniques [26] to find the optimal variable assignment. In general, LP approaches maximize/minimize an objective function where the con-

straints defined on the data points are all linear [26][1]. In the objective function (Eq. 4.9), the first addend is already written in a linear form.

Concerning the first two constraints:

- Since the set of $I_{ij}$ constraints are defined as a sum of variables, they can be directly encoded in an LP solver.
- Dealing with the set of $E_z$ constraint is slightly more complex. To represent it in a linear form it is convenient to redefine these constraints as: if $z$ does not choose itself as exemplar, then nobody else can choose it as an exemplar. In other words, if the entry in the $C$ matrix that represents the exemplar choice of $z$ for itself is zero, the $z$-sheet must be completely zero. Considering our model we can write it in linear form as:

$$\text{if} \quad C(i_z, j_z, z) = 0 \quad \text{then} \quad \sum_{i=1}^{n}\sum_{j=1}^{m} C(i, j, z) \leq 0.$$

Regarding the linear representation of the biclustering integrity constraint (Eq. 4.8): the idea is that, when considering the matrix $C(:, :, z)$, the biclustering integrity constraint is satisfied if, and only if, all rows (or columns) of this matrix are either zero or equal to each other. By exploiting the Boolean nature of the variables, this can be enforced by checking if, for every pair of rows (or columns) $U = (u_1, \ldots, u_m)$ and $X = (x_1, \ldots, x_m)$, one of the following conditions is true: i) $U = X$, ii) U = 0, iii) X = 0. This can be expressed through Boolean algebra as: i) NOT $(\sum_i (u_i \oplus x_i))$, ii) NOT $(\sum_i u_i)$, NOT $(\sum_i x_i)$, where "sums" (*i.e.,* both "$\sum$" an "+") denote the OR operator and "$\oplus$" is the XOR operator. By using De Morgan laws and some proprieties of the Boolean algebra we can derive the set of linear constraints representing the OR operation between the previous i), ii) and iii) constraints as:

$$\begin{array}{llll} -u_1 + x_1 + u_2 < 2 & -u_1 + x_1 + u_3 < 2 & \cdots & -u_1 + x_1 + u_n < 2 \\ u_1 - x_1 + x_3 < 2 & -u_2 + x_2 + u_1 < 2 & \cdots & u_1 - x_1 + x_n < 2 \end{array}$$

this has to be done for all pairs of rows (or columns) of every matrix $C(:, :, z)$.

Now, all the elements of the model (objective function and constraints) are linear, and the model can be solved by using LP approaches.

[1] An equation involving $n$ variables is linear if it can be written in the form $ax_1 + b_x 2 + \cdots + cx_n + d = 0$

### 4.2.2 Complexity and Applicability

Regarding time complexity, an Integer Programming problem is exponential in the number of constraints (in the worst case). However, there are many well established methods which provide, on average, fast solutions.

Concerning space complexity, given an input matrix formed by $n$ rows and $m$ columns, the model contains $O(n^2m^2)$ variables and $O(nm)$ functions for the constraints $I$ and $E$. Unfortunately, when considering the *biclustering integrity constraint*, the number of functions to completely describe all possibilities raises to $O(m^3n^3)$. While being still polynomial (and not exponential) in the number of rows and columns of the data matrix, the number of functions to store in memory can be very large. In particular, for typical biclustering problems (e.g., microarray analysis), the data matrix can contain hundreds of rows and columns, hence our approach might require a prohibitive amount of memory to store the model.

To overcome this scalability issue we provide an approximate version which is able to work adequately on large matrices. The algorithm is defined as follows: we run our algorithm on smaller matrices, extract biclusters and devise an aggregation algorithm to find biclusters in the original data matrix.

The steps of such algorithm can be described as:

1. Analyse the data matrix by means of smaller, fixed dimension sub-matrices, with no overlap;
   (a) in every sub-matrix retrieve the optimal solution exploiting our model and the LP approach.
2. Aggregation algorithm: we process the set of obtained biclusters in three steps
   (a) apply Affinity Propagation on the obtained bicluster exemplars to partition the biclusters in groups of biclusters with coherent values;
   (b) for every cluster of biclusters: perform a classical agglomerative clustering of biclusters by using as similarity the degree of overlapping columns/rows. The final partition represents a set of biclusters with no row/column overlap with the other groups.
   (c) Post-process the final groups in order to be sure that they represent an actual bicluster: this is done by removing rows (or columns) which violate the bicluster definition (*i.e.,* to be a sub-matrix).

Notice that this last step is necessary because merging biclusters may not produce a bicluster as result.

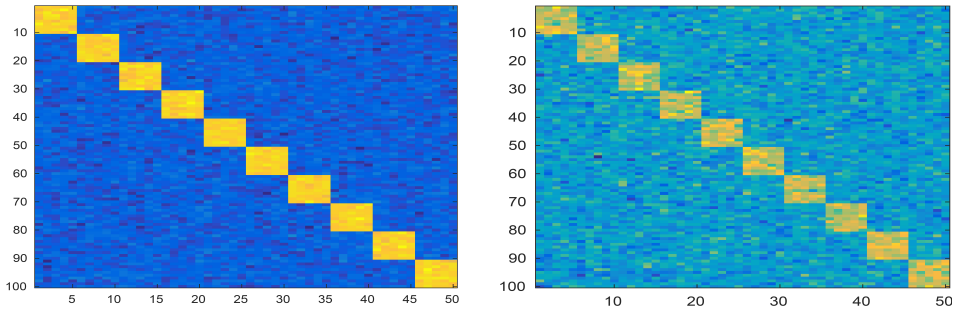



Figure 4-3: Examples of data matrices with low noise level (a) and high noise level (b).

.

### 4.2.3 Results

The described methodology has been tested on a set of synthetic matrices which represent a classical benchmark in the microarray scenario [107]: such set comprises synthetic expression matrices, perturbed with different schemes[2]; examples of such matrices are sketched in Fig. 4-3. In the experiments, we have 10 non-overlapping biclusters, each extending over 10 rows and 5 columns. Such datasets have been widely used to investigate the effects of noise on the performance of various biclustering approaches.

The accuracy of the obtained biclusters set has been assessed with the so-called *Gene Match Score* [107]: given two set of biclusters $B_1$ and $B_2$, it reflects the average of the maximum match scores for all biclusters in $B_1$ with respect to the biclusters in $B_2$. The Gene Match Score is computed as:

$$GMS(B_1, B_2) = \frac{1}{|B_1|} \sum_{(R_1,C_1)\in B_1} \max_{(R_2,C_2)\in B_2} \frac{|R_1 \cap R_2|}{|R_1 \cup R_2|}. \tag{4.10}$$

Now, let $B_r$ being the set of retrieved biclusters and let $B_{gt}$ being the set of optimal biclusters (ground truth): the average bicluster *relevance* reflects to what extent the generated biclusters represent a true bicluster in gene dimensions and it is computed as $GMS(B_r, B_{gt})$; whereas the average bicluster *recovery* quanties how well each of the true biclusters is recovered by the biclustering algorithm and it is computed as $GMS(B_{gt}, B_r)$ (such scores vary between 0 and 1, where the higher the better the accuracy).

In our model we used as similarity the negative of the Euclidean distance (as in [34]), which allows to retrieve only constant value biclusters. As in the original Affinity Propagation model, a proper setting of the preferences (namely the

[2]All datasets may be downloaded from: www.tik.ee.ethz.ch/sop/bimax.

self similarities) is crucial: in our experiments we found that a good choice is represented by the first integer number below the median. The Linear Programming model was implemented and resolved using CPLEX (version 12.4). CPLEX represents the standard choice in the optimization community and it is widely adopted in many different contexts (as in [24, 88]); it implements the *simplex* method which examines the corners of the feasible region (region in the variable space where the constraints are respected) and stops when the optimal solution has been found (for further details on the CPLEX solver, we refer interested readers to [26]).

Figure 4-4(a) reports the mean recovery of the retrieved biclusters by varying the sub-matrices dimensions with respect to the noise level; similarly Fig. 4-4(b) reports the mean relevance values for different levels of noise and for different dimensions of the sub-matrices, averaged over the different repetitions (also standard deviations are displayed). In Fig. 4-4(c) and 4-4(d) we also reported the results obtained by state-of-the-art methods on the same dataset. As expected the LP approach provides better solutions as the kernel dimension increases. Please note that when using the [1x1] sub-matices only the aggregation algorithm described in the previous section is employed (every data point is in its own bicluster). As we can see in Fig.4-4, increasing the noise completely corrupts the performances of the aggregation algorithm. Crucially, obtained results are competitive with other state of the art approaches (as we can see by comparing the obtained results with 4-4(c) and 4-4(d)), confirming the potentialities of the proposed approach.

## 4.3 One Bicluster solution

The techniques previously introduced represent a valuable example of what discussed in Chap. 1 and Chap. 3: dealing with Graphical Models requires carefulness in designing the model, otherwise the solution procedures could struggle in solving the model. In fact, both EB (described in Sec. 4.1) and BAP (described in Sec. 4.2) show a limited scalability that hinders their practical usage ($10 \times 10$ matrices analysed with non-heuristics techniques). Specifically: in EB the exploitation of integer variables (instead of binaries) leads to inefficient messages; whereas in BAP the biclustering integrity constraint, coupled with the enormous number of variables, introduces too many cycles for Max-Sum to be effective. However, we think that there is much room for improvement concerning scalability; and in this Section we present what we believe is an important step towards this direction.

We specifically reformulate biclustering as a sequential search problem, discovering one bicluster at time (an approach employed by other authors in literature [7, 20, 52]). Particularly, we aim at retrieving at each iteration the largest bicluster possible. This is because in different application scenarios (*e.g.,* gene expression) larger biclusters are more informative. We thus derive a novel binary FG (called *OOB*) that retrieves the biggest bicluster possible on the basis of a given

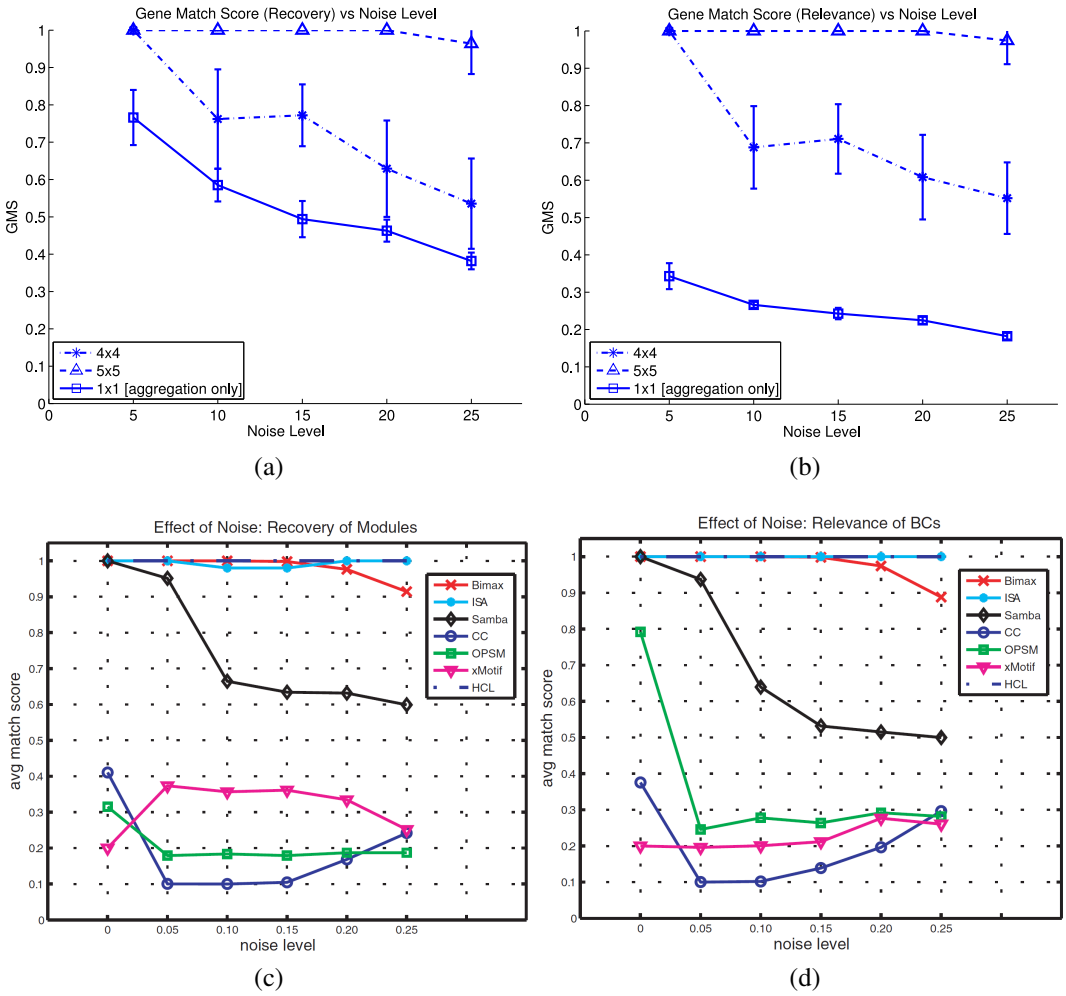



Figure 4-4: Results for the proposed approach: (a) recovery and (b) relevance. And for state-of-the-art techniques: (c) recovery and (d) relevance.

coherence criterion. Crucially, the proposed model remains *compact* and includes only *binary* variables, allowing efficient optimization via the max-sum algorithm, which drastically alleviates the scaling issues described above.

### 4.3.1 The model

The proposed model takes inspiration for what proposed in EB and BAP, and it is composed by four main ingredients.

1. *Formulation of bisclutering as an incremental search for the largest bicluster*. Several approaches in the literature propose techniques that sequentially identify the bicluster, one at a time [7,20,59]. Once a bicluster is identified, it is masked in the data matrix (*e.g.*, by replacing it with background noise [20]) and the next bicluster is sought. In this class of methods, we use a model with binary variables, one for each entry of the matrix, indicating whether that entry belongs, or not, to the solution. In our model, a solution is represented by a binary matrix $C \in \{0,1\}^{n \times m}$, where each entry $c_{ij}$ indicates whether the entry $(i,j)$ belongs to the bicluster ($c_{i,j} = 1$) or not ($c_{i,j} = 0$). We thus use binary variables (as in BAP), but a compact set thereof (as in EB). Since we are looking for the largest possible bicluster, solutions where many $c_{i,j}$ are set to 1 should be preferred.

2. *Bicluster coherence criterion*. The model should prefer solutions containing coherent entries, or, likewise, it should penalize incoherent solutions. In our model, the incoherence of a bicluster is measured as the sum of the pairwise incoherences between all the points belonging to the bicluster. This differs from BAP and EB, where only the coherence with the exemplar of the bicluster is considered. We will see that our choice can lead to more robust solutions in the presence of noise (see results in Sec. 4.3.3). There are several possibilities to define the incoherence $I(a_{ij}, a_{tk})$ between two entries, depending on which kind of bicluster we are looking for (constant-valued, additive, multiplicative, or others [83]). The most straightforward option is to simply use a constant-type incoherence (as in BAP):

$$I(a_{ij}, a_{tk}) = (a_{ij} - a_{tk})^2. \tag{4.11}$$

   By using this incoherence measure we direct the search towards constant biclusters (i.e. biclusters with the smallest possible variance). If we are aiming at additively coherent biclusters, the incoherence can be defined as in EB:

$$I(a_{ij}, a_{tk}) = (a_{ij} - a_{tj} + a_{tk} - a_{ik})^2. \tag{4.12}$$

   In fact, this choice directs the search towards biclusters where entry activation

levels are given by the sum of some constants, one for each row and column; this is useful if we are looking for patterns among the columns [83].

3. *A bicluster is a sub-matrix.* The model should consider only valid assignments, that is, corresponding to a bicluster: this requirement can be expressed by enforcing that the points belonging to a bicluster have to constitute a sub-matrix (*i.e.*, a subset of rows and columns). In particular, considering every pair of rows (or columns) in matrix $C$, the constraint is satisfied if any of the two following conditions are met: i) the rows (or columns) share the same pattern or ii) one of the rows (or columns) is completely zero. Due to the simplicity of this conditions, the number of constraints drastically reduces: the EB and BAP models require one constraint for every couple of entries, whereas this model only involves a constraint for every pair of rows (or every pair of columns). This reduction is even more significant when the number of rows is much smaller than the number of columns (or viceversa), such as in relevant biology applications [83, 98]. For instance, concerning real microarray gene expression datasets, we commonly analyse matrices having thousands of rows and hundreds columns.

4. *The entry level "counts".* The fourth ingredient is based on the observation that in many biclustering applications (such as preference matrices, count matrices, gene expression datasets) the most important biclusters are those containing high-valued entries. This can be directly encoded in the model by rewarding solutions that contain entries with high values (without loss of generality, we assume that all entries $a_{ij}$ have positive values). This aspect was neglected in the EB and BAP models, where the assignment to a bicluster was made only on the basis of the coherence.

The proposed model, graphically sketched in Figure 4-5, is fully defined by the variables and factors that we describe next.

Given a data matrix $A$ as defined in Sec. 2.2, the variables are organized in a binary matrix $C \in \{0,1\}^{n \times m}$, where each entry $c_{ij}$ (for $i \in N$, $j \in M$) indicates if the corresponding entry belongs ($c_{i,j} = 1$) or not ($c_{i,j} = 0$) to the bicluster. The objective function (to be maximized with respect to $C$) includes three types of factors:

- Unary factors (one per entry, function only of the corresponding entry), given by

$$A_{ij}(c_{ij}) = a_{ij}\, c_{ij} = \begin{cases} a_{ij}, & \text{if } c_{ij} = 1 \\ \\ 0, & \text{otherwise,} \end{cases} \tag{4.13}$$

  encouraging the bicluster to contain entries of $A$ with high activation level.

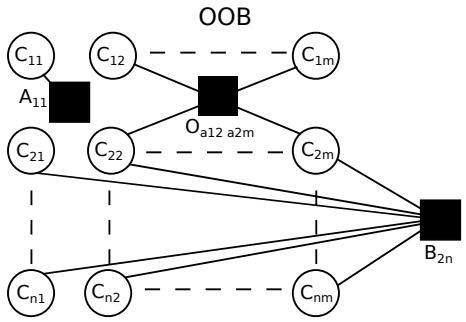



Figure 4-5: Factor graph for the Repeatedly One Bicluster approach.

- Pairwise factors (one for each pair of entries) given by

$$O_{ij,tk}(c_{ij}, c_{tk}) = -\, I(a_{ij}, a_{tk})\, c_{ij}\, c_{tk}, \tag{4.14}$$

  encouraging the bicluster to minimize its internal incoherence, as assessed by the incoherence measure $I$.

- Column/row pair factors (one per pair of columns or pair of rows), forcing the maximizer of the objective function to correspond to a bicluster:

$$B_{jk}(c_{:j}, c_{:k}) = \begin{cases} 0, & \text{if } \left(\sum_i c_{ij}\right)\left(\sum_i c_{ik}\right)\left(\sum_i |c_{ij} - c_{ik}|\right) = 0 \\ -\infty, & \text{otherwise} \end{cases} \tag{4.15}$$

  where $c_{:j}$ denotes the $j$-th column of $C$.

Summarizing, given the variables and factors defined above, the proposed FG encodes the following function (to be maximized):

$$F(C) = \sum_{i=1}^{n}\sum_{j=1}^{m} A_{ij}(c_{ij}) + w \sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{t=1}^{n}\sum_{k=1}^{m} O_{ij,tk}(c_{ij}, c_{tk}) + \sum_{j=1}^{m}\sum_{k=1}^{m} B_{jk}(c_{:j}, c_{:k}). \tag{4.16}$$

Apart from the $B_{jk}$ factors (which ensure bicluster solutions), we have two competing driving forces: on the one hand, if two entries are in the solution, say $c_{ij} = 1$ and $c_{tk} = 1$, their activation levels ($a_{ij}$ and $a_{tk}$) are added to the objective function, which encourages large biclusters; on the other hand, such inclusion decreases the objective function by $w\, I(a_{ij}, a_{tk})$ (note the minus sign in Equation (4.14)), discouraging incoherent points from being included in the solution. The relative strength of these two forces is controled by parameter $w$, whose setting is crucial to regulate the bicluster dimensions.

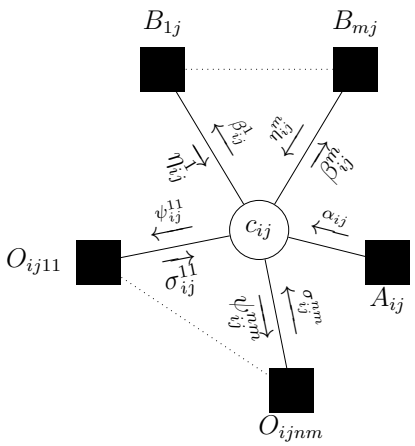



Figure 4-6: Sketch of the factor graph showing the connections between one variable and all its factors, comprehending the interchange messages.

### 4.3.2 Messages

To maximize the objective function in Equation 4.16 we adopt the max-sum algorithm (described in Section 3.3.1), in this section we present the derivations of each message update rule.

These derivations represent the bottleneck of many approaches. In fact the maximization in Eq. 3.13 is intractable since, for a given function and considering integer variables, we should analyse $d^k$ configurations (with $d$ variables domain size and $k$ number of neighbours). Furthermore, note that a general message $\mu(x)$ should be a vector storing the message value for each possible configuration of $x$. For instance, if $x$ can take $k$ possible values $\mu$ should be a vector $\mu = [\mu(x = 1), \cdots, \mu(x = k)]$. Once Max-Sum converges the value of a variable is assigned as the value for whom the sum of the incoming messages is maximum. Specifically, in the case of binary variables we should store the value of $\mu(x = 0)$ and $\mu(x = 1)$. However, without loss of information, we can store the difference between this values (i.e. $\mu = \mu(x = 1) - \mu(x = 0)$ ) and assign 1 to a variable if the message is positive and 0 otherwise. The following messages are derived on the basis of this consideration. Figure 4-6 shows a sketch of the FG connections for a given variable, and the messages exchanged.

#### Messages from Variables to Factors

Following Eq. 3.14 the value of the message going from a variable $x$ to factor $f$ is the sum of the incoming messages without considering the one arriving from $f$.

Given the Factor Graph in Figure 4-6 we obtain:

- $\psi_{ij}^{tk}(c_{ij}) = \mu_{c_{ij} \to O_{ijtk}}(c_{ij})$

- if $c_{ij} = 1$

$$\psi_{ij}^{tk}(1) = \sum_{\hat{tk}} \sigma_{ij}^{\hat{tk}}(1) + \sum_{k} \eta_{ij}^{k}(1) + \alpha_{ij}(1)$$

- if $c_{ij} = 0$

$$\psi_{ij}^{tk}(0) = \sum_{\hat{tk}} \sigma_{ij}^{\hat{tk}}(0) + \sum_{k} \eta_{ij}^{k}(0) + \alpha_{ij}(0)$$

- then

$$\begin{aligned}\psi_{ij}^{tk} &= \psi_{ij}^{tk}(c_{ij})(1) - \psi_{ij}^{tk}(c_{ij})(0) \\ &= \sum_{\hat{tk}} \sigma_{ij}^{\hat{tk}} + \sum_{k} \eta_{ij}^{k} + \alpha_{ij}\end{aligned}$$

- $\beta_{ij}^{k}(c_{ij}) = \mu_{c_{ij} \to B_{jk}}(c_{ij})$

  - if $c_{ij} = 1$

$$\beta_{ij}^{k}(1) = \sum_{\hat{k}} \eta_{ij}^{\hat{k}}(1) + \sum_{tk} \sigma_{ij}^{tk}(1) + \alpha_{ij}(1)$$

  - if $c_{ij} = 0$

$$\beta_{ij}^{k}(0) = \sum_{\hat{k}} \eta_{ij}^{\hat{k}}(1) + \sum_{tk} \sigma_{ij}^{tk}(0) + \alpha_{ij}(0)$$

  - then

$$\begin{aligned}\beta_{ij}^{k} &= \beta_{ij}^{k}(1) - \beta_{ij}^{k}(0) \\ &= \sum_{\hat{k}} \eta_{ij}^{\hat{k}} + \sum_{tk} \sigma_{ij}^{tk} + \alpha_{ij}\end{aligned}$$

- The message going from a variable $c_{ij}$ to its function $A_{ij}$ is not shown in Figure 4-6 because it is commonly set to 0, hence this is not considered [41]. The intuition behind this choice is that the information provided by $A_{ij}$ (i.e. the entry value) must not change across the Max-Sum iterations.

**Messages from Factors to Variables**

Reminding that the maximization in Eq. 3.13 is intractable and represents the bottle-neck of many approaches; in this section we present how we can exploit the binary nature of the variables and the hard constraints to reduce the possible configurations in such maximization.

**Note:** we exploit the following properties in the derivation:

Rule 1) $a - \max(a, b) = \min(0, a - b)$;
Rule 2) $\max(a - b) - b = \max(a - b, 0)$;
Rule 3) $\max(a, b) - \max(c, d) = \max[\min(a - c, a - d), \min(b - c, b - d)]$;
Rule 4) $a - \max(b, c) = \min(a - b, a - c)$

Given the Factor Graph in Figure 4-6 we obtain:

- $\sigma_{ij}^{tk}(c_{ij}) = \mu_{O_{ijtk} \to c_{ij}}(c_{ij})$

$$\sigma_{ij}^{tk}(c_{ij}) = \max_{c_{tk}} \left[ O_{ijtk}(c_{ij}, c_{tk}) + \psi_{tk}^{ij}(c_{tk}) \right]$$

  - if $c_{ij} = 1$

$$\sigma_{ij}^{tk}(1) = \max_{c_{tk}} \left[ O_{ijtk}(1, c_{tk}) + \psi_{tk}^{ij}(c_{tk}) \right]$$

    we must consider two cases:

    1. $c_{tk} = 1$

$$\sigma_{ij}^{tk}(1) = w * I(a_{ij}, a_{tk}) + \psi_{tk}^{ij}(1)$$

    2. $c_{tk} = 0$

$$\sigma_{ij}^{tk}(1) = \psi_{tk}^{ij}(0)$$

  - if $c_{ij} = 0$

$$\begin{aligned} \sigma_{ij}^{tk}(0) &= \max_{c_{tk}} \left[ O_{ijtk}(0, c_{tk}) + \psi_{ij}^{tk}(c_{tk}) \right] \\ &= \max_{c_{tk}} \left[ \psi_{tk}^{ij}(c_{tk}) \right] \end{aligned}$$

  - then considering the equations just retrieved (i.e. the ones for $c_{ij} = 1$ and the one for $c_{ij} = 0$) we need to compute

$$\begin{aligned} \sigma_{ij}^{tk} &= \max \sigma_{ij}^{tk}(1) - \max \sigma_{ij}^{tk}(0) \\ &= \max \left[ w * I(a_{ij}, a_{tk}) + \psi_{ij}^{tk}(1), \psi_{ij}^{tk}(0) \right] - \max \left[ \psi_{tk}^{ij}(1), \psi_{tk}^{ij}(c_{tk}0) \right] \\ &= \max \left[ \min \left( w * I(a_{ij}, a_{tk}), w * I(a_{ij}, a_{tk}) + \psi_{tk}^{ij} \right), \min \left( \psi_{tk}^{ij}, 0 \right) \right] \end{aligned}$$

- $\eta_{ij}^{k}(c_{ij}) = \mu_{B_{jk} \to c_{ij}}(c_{ij})$

$$\eta_{ij}^{k}(c_{ij}) = \max_{c_{\hat{ij}}} \left[ B_{jk}(c_{1j}, \ldots, c_{ij}, \ldots, c_{nk}) + \sum_{\hat{i}} \beta_{\hat{ij}}^{k}(c_{\hat{ij}}) + \sum_{t} \beta_{tk}^{j}(c_{tk}) \right] \tag{4.17}$$

– if $c_{ij} = 1$

$$\eta_{ij}^k(c_{ij}) = \max_{c_{\hat{i}j}} \left[ B_{jk}(c_{1j}, \ldots, c_{ij} = 1, \ldots, c_{nk}) + \sum_{\hat{i}} \beta_{\hat{i}j}^k(c_{\hat{i}j}) + \sum_{t}^{n} \beta_{tk}^j(c_{tk}) \right]$$

Surely the maximizer of such function is a configuration respecting the biclustering constraint (otherwise the $B$ factor provides a minus infinity in the objective function). Hence, considering the constraint in Equation (4.15) if $c_{ij} = 1$ there are only two possible situations where the constraint is satisfied: i) the $k^{th}$ column is completely equal to the $j^{th}$ or ii) the $k^{th}$ column is completely zero. In order to distinguish these quantities we introduce different notations obtaining:

1. the two columns are equal:

$$\eta_{ij}^k(1) = \max_{c_{\hat{i}j}} \left[ \beta_{ik}^j(1) + \sum_{\hat{i}} \beta_{\hat{i}j}^k(c_{\hat{i}j}) + \beta_{\hat{i}k}^j(c_{\hat{i}j}) \right] = \eta_{ij}^k(1)^\dagger$$

   note that in the second part of this equation $c_{\hat{i}j}$ is used in both betas (instead of $c_{\hat{i}k}$), this is to enforce the fact that we are considering columns with the same configuration.

2. the $k^{th}$ column is completely zero

$$\eta_{ij}^k(1) = \max_{c_{\hat{i}j}} \left[ \sum_{\hat{i}} \beta_{\hat{i}j}^k(c_{\hat{i}j}) + \sum_{t} \beta_{tk}^j(0) \right] = \eta_{ij}^k(1)^\ddagger$$

– if $c_{ij} = 0$

$$\eta_{ij}^k(c_{ij}) = \max_{c_{\hat{i}j}} \left[ B_{jk}(c_{1j}, \ldots, c_{ij} = 0, \ldots, c_{nk}) + \sum_{\hat{i}} \beta_{\hat{i}j}^k(c_{\hat{i}j}) + \sum_{t}^{n} \beta_{tk}^j(c_{tk}) \right].$$

Similarly, if $c_{ij} = 0$ there are only two possible situations where the constraint is satisfied: i) the $k^{th}$ column is completely equal to the $j^{th}$ or ii) the $j^{th}$ column is completely zero. Hence we obtain

1. the $j^{th}$ column is completely zero:

$$\eta_{ij}^k(0) = \max_{c_{\hat{i}j}} \left[ \sum_{\hat{i}} \beta_{\hat{i}j}^k(0) + \sum_{t} \beta_{tj}^k(c_{tk}) \right] = \eta_{ij}^k(0)^\ddagger$$

2. the two columns are equal:

$$\eta_{ij}^k(0) = \max_{c_{\hat{i}j}} \left[ \beta_{ik}^j(0) + \beta_{tj}^k(1) + \beta_{tk}^j(1) + \sum_{l \in N \setminus \{i,t\}} \left( \beta_{lj}^k(c_{lj}) + \beta_{lk}^j(c_{lj}) \right) \right] = \eta_{ij}^k(0)^\dagger$$

note that if the column $j$ is not completely zero (previous case) hence there is at least another $c_{tj} = 1$ (with $t \neq i$).

– then considering the four equations just retrieved (i.e the two for $c_{ij} = 1$ and the two for $c_{ij} = 0$) we must calculate

$$\begin{aligned}
\eta_{ij}^k &= \max \eta_{ij}^k(1) - \max \eta_{ij}^k(0) \\
&= \max(\eta_{ij}^k(1)^\dagger, \eta_{ij}^k(1)^\ddagger) - max(\eta_{ij}^k(0)^\ddagger, \eta_{ij}^k(0)^\dagger) \\
&= \max \Big[ \min \left( \eta_{ij}^k(1)^\dagger - \eta_{ij}^k(0)^\ddagger, \eta_{ij}^k(1)^\dagger - \eta_{ij}^k(0)^\dagger \right), \\
&\qquad \min \left( \eta_{ij}^k(1)^\ddagger - \eta_{ij}^k(0)^\ddagger, \eta_{ij}^k(1)^\ddagger - \eta_{ij}^k(0)^\dagger \right) \Big] \qquad (4.18)
\end{aligned}$$

In what follows we analyse each of the terms involved in (4.18) separately. Referring to Section 3.3 we obtain:
$\Theta = \eta_{ij}^k(1)^\dagger - \eta_{ij}^k(0)^\ddagger$=

$$= \max_{c_{\hat{i}j}} \left[ \beta_{ik}^j(1) + \sum_{\hat{i}} \beta_{\hat{i}j}^k(c_{\hat{i}j}) + \beta_{\hat{i}k}^j(c_{\hat{i}j}) \right] - \max_{c_{\hat{i}j}} \left[ \sum_{\hat{i}} \beta_{\hat{i}j}^k(0) + \sum_t \beta_{tj}^k(c_{tk}) \right] =$$

$$= \beta_{ik}^j(1) - \max \beta_{ik}^j(c_{ik}) + \sum_{\hat{i}} \left[ \max \left( \beta_{\hat{i}j}^k(c_{\hat{i}j}) + \beta_{\hat{i}k}^j(c_{\hat{i}j}) \right) - \max \left( \beta_{\hat{i}j}^k(0) + \beta_{ik}^j(c_{\hat{i}j}) \right) \right] =$$

$$\Theta = \min(0, \beta_{ik}^j) + \sum_{\hat{i}} \max \left[ \min(\beta_{\hat{i}j}^k, \beta_{ij}^k + \beta_{\hat{i}k}^j), \min(-\beta_{\hat{i}k}^j, 0) \right]$$

$I = \eta_{ij}^k(1)^\dagger - \eta_{ij}^k(0)^\dagger$ =

$$= \max_{c_{\hat{i}j}} \left[ \beta_{ik}^j(1) + \sum_{\hat{i}} \beta_{\hat{i}j}^k(c_{\hat{i}j}) + \beta_{\hat{i}k}^j(c_{\hat{i}j}) \right] - + \max_{c_{\hat{i}j}} \left[ \beta_{ik}^j(0) + \beta_{tj}^k(1) + \beta_{tk}^j(1) + \sum_{l \in N \setminus \{i,t\}} \left( \beta_{lj}^k(c_{lj}) + \beta_{lk}^j(c_{lj}) \right) \right] =$$

Supposing that the optimal $t$ is given, we can write

$$= \beta_{ik}^{j}(1) + \beta_{tj}^{k}(c_{tj}) + \beta_{tk}^{j}(c_{tj}) - \beta_{tj}^{k}(1) - \beta_{tk}^{j}(1) + \\ + \sum_{l \in N \setminus \{i,t\}} \left[ \max\left(\beta_{lj}^{k}(c_{lj}) + \beta_{lk}^{j}(c_{lj})\right) - \max\left(\beta_{lj}^{k}(c_{lj}) + \beta_{lk}^{j}(c_{lj})\right) \right]$$

$$= \beta_{ik}^{j} + \max(0, -\beta_{tj}^{k} - \beta_{tk}^{j}).$$

Since this is the result between the difference of two maxima, and the first maximum is fixed, we can now obtain the general value for the best $t$ by minimizing the second maximum and hence

$$I = \beta_{ik}^{j} + \min_{\hat{i}} \left[ \max(0, -\beta_{\hat{i}j}^{k} - \beta_{\hat{i}k}^{j}) \right].$$

$K = \eta_{ij}^{k}(1)^{\ddagger} - \eta_{ij}^{k}(0)^{\ddagger}$ =

$$= \max_{c_{\hat{i}j}} \left[ \sum_{\hat{i}} \beta_{\hat{i}j}^{k}(c_{\hat{i}j}) + \sum_{t} \beta_{tk}^{j}(0) \right] - \max_{c_{\hat{i}j}} \left[ \sum_{\hat{i}} \beta_{\hat{i}j}^{k}(0) + \sum_{t} \beta_{tj}^{k}(c_{tk}) \right] =$$

$$= \sum_{\hat{i}} \left[ \max\left(\beta_{\hat{i}j}^{k}(c_{\hat{i}j}) + \beta_{hatik}^{j}(0)\right) - \max\left(\beta_{\hat{i}j}^{k}(0) + \beta_{hatik}^{j}(c_{\hat{i}k})\right) \right] + \\ + \beta_{ik}^{j}(0) - max\left(\beta_{ik}^{j}(0)\right)$$

the first part of this equation (the one included in the sum) can be solved adopting the third rule previously described and the final result is:

$$K = \min(0, -\beta_{ik}^{j}) + \sum_{\hat{i}} \max\left[ \min\left(0, -\beta_{\hat{i}k}^{j}\right), \min\left(\beta_{\hat{i}j}^{k}, \beta_{\hat{i}j}^{k} - \beta_{\hat{i}k}^{j}\right) \right]$$

$\Lambda = \eta_{ij}^{k}(1)^{\ddagger} - \eta_{ij}^{k}(0)^{\dagger}$ =

$$= \max_{c_{\hat{i}j}} \left[ \sum_{\hat{i}} \beta_{\hat{i}j}^{k}(c_{\hat{i}j}) + \sum_{t} \beta_{tk}^{j}(0) \right] - \\ + \max_{c_{\hat{i}j}} \left[ \beta_{ik}^{j}(0) + \beta_{tj}^{k}(1) + \beta_{tk}^{j}(1) + \sum_{l \in N \setminus \{i,t\}} \left(\beta_{lj}^{k}(c_{lj}) + \beta_{lk}^{j}(c_{lj})\right) \right]$$

As previously mentioned, also in this case we assume that the best $t$ is

given obtaining

$$= \sum_{\hat{i}} \max\left[\beta^k_{\hat{i}j}(c_{\hat{i}j}) + \beta^j_{\hat{i}k}(0)\right] - \beta^k_{tj}(1) - \beta^j_{tk}(1) + \\ - \sum_{l \in N \setminus \{i,t\}} \left[\beta^k_{\hat{i}j}(c_{\hat{i}j}) + \beta^j_{\hat{i}k}(c_{\hat{i}j})\right] =$$

$$= -\beta^k_{tj}(1) - \beta^j_{tk}(1) + \max\left(\beta^j_{tk}(0) + \beta^k_{tj}(c_{tj})\right) + \\ + \sum_{l \in N \setminus \{i,t\}} \left[\max\left(\beta^k_{\hat{i}j}(c_{\hat{i}j}) + \beta^j_{\hat{i}k}(0)\right) - \max\left(\beta^k_{\hat{i}j}(c_{\hat{i}j}) + \beta^j_{\hat{i}k}(c_{\hat{i}j})\right)\right] =$$

Again, the part included in the sum can be solved adopting the third rule previously described and obtaining:

$$= \max\left(-\beta^j_{tk}, -\beta^j_{tk} - \beta^k_{tj}\right) + \\ + \sum_{\hat{i} \neq t,i} \left[\max\left(\min(0, -\beta^k_{\hat{i}j} - \beta^j_{\hat{i}k}), \min(\beta^k_{\hat{i}j}, -\beta^j_{\hat{i}k})\right)\right]$$

This is the result for a fixed $t$, to obtain the general update rule we need to minimize with respect to $t$ resulting in:

$$\Lambda = \min_{t \neq i} \Big[ \max\left(-\beta^j_{tk}, -\beta^j_{tk} - \beta^k_{tj}\right) + \\ + \sum_{l \neq t,i} \left(\max\left(\min(0, -\beta^k_{lj} - \beta^j_{lk}), \min(\beta^k_{lj}, -\beta^j_{lk})\right)\right) \Big].$$

Hence the message update rule for $\eta^k_{ij}$ is

$$\eta^k_{ij} = \max\left[\min(\Theta, I), \min(K, \Lambda)\right]$$

These messages, starting from an assignment where are all $c_{ij} = 0$, are exchanged and updated until a convergence criterion is met. In our experiments, the algorithm stops when the objective function does not change for a fixed number of consecutive iterations.

### Computational Complexity and Scalability

Given an $n \times m$ data matrix, the model has $O(n^2m^2)$ space complexity, due to the $n^2m^2$ factors $O_{ij,tk}$. Concerning time, the message update procedure is quadratic in

the number of columns (or rows), thus being reasonably fast. In particular, the time complexity is dominated by $\eta_{ij}^k$ messages, which are $O(n^2)$. The other messages are less demanding, each having the following computational costs: $\psi_{ij}^{tk}$ is $O(n)$; $\beta_{ij}^k$ is $O(n)$; $\sigma_{ij}^{tk}$ is $O(1)$.

The derivation of such messages allow us to analyse $50 \times 50$ matrices with exact max-sum update rules. We think this is an important step toward efficient Factor Graph based biclustering, since previous methods solved up to $10 \times 10$ matrices. Although this promising result, the proposed approach cannot handle real matrices directly; in particular, the memory required by the $\sigma$ messages for $n$ genes and $m$ conditions involves storing $n^2m^2$ values. To circumvent this difficulty, in analysing real data we adopted a scheme similar to that employed for BAP (described in Sec. 4.2.2): the algorithm is executed on different portions of the matrices; successively, we employ an aggregation method to merge the obtained biclusters. The idea is that by aggregating accurate results obtained on portions of the data matrix can provide solutions of superior quality for the overall biclustering problem. This is largely confirmed by our experimental analysis. More in detail, we employ the following procedure:

1. the whole matrix is randomly partitioned into non-overlapping submatrices (a reasonable partition would not divide the data matrix on both rows and columns, to reduce the complexity of the re-aggregation step); on each of these submatrices, we run the proposed algorithm;

2. the obtained biclusters are grouped using AP; since every sub-matrix involves the whole set of columns, the similarity criterion between two biclusters is defined as the number of columns they share;

3. finally, we validate the groups of biclusters by looking at the FG objective function in Eq. (4.16): in particular, we compute its value for each bicluster group considering the matrix composed by the rows and the columns of the biclusters belonging to that group; then, if the objective function is positive (meaning that the max-sum algorithm could have put them together), the group is kept and the final bicluster is obtained by merging rows and columns of all its biclusters.[3]

### 4.3.3 Experimental evaluation

In this section we evaluate the OOB approach using both synthetic and real datasets derived from gene expressions. As a preliminary step, we provide some empirical evidence on the effectiveness and efficiency of the max-sum algorithm we derived,

[3]To retrieve different biclusters the methodology is repeated varying the set of sub-matrices.

also comparing it with the very recent AD$^3$ method [87], which in principle can provide better solutions but at a cost of slower convergence. AD$^3$ is based on the alternating directions method of multipliers. Like other dual decomposition algorithms, AD$^3$ has a modular architecture, where local subproblems are solved independently, and their solutions are gathered to compute a global update. The key characteristic of AD3 is that each local subproblem has a quadratic regularizer, leading to faster convergence, both theoretically and in practice.

Next, we evaluated the quality of the solutions provided by our approach in comparison with the other FG-based approaches (EB and BAP), as well as other state-of-the-art techniques [20, 32, 59, 61, 107, 117, 121, 131, 145]. The goal of these experiments is twofold: (i) to empirically show (on synthetic datasets) that our model provides more robust solutions than previous FG-based biclustering approaches; (ii) to assess the significance of our method on real-world gene expression datesets, also in comparison with different non-FG-based approaches.

**Analysis of the optimization algorithm**

We begin with some comments on the convergence of the objective function along the max-sum iterations, then we further motivate its adoption through a comparison with the recent AD$^3$ algorithm [87].

Concerning the first aspect, it is known that max-sum is only guaranteed to converge in the absence of cycles [12, 140], a condition that is clearly not satisfied by the proposed FG. However, in all the experiments herein reported, the algorithm always converged, usually within the first 250 iterations, with the messages reaching stable values. An example of the objective function evolution is shown in Figure 4-7, where it can be seen how it rapidly reaches its final value.

Concerning the second aspect, note that, as reported in Sec. 3.3, there are different classes of methods for optimizing FG models. When the probabilistic interpretation is not relevant, the max-sum algorithm is arguably the most common choice, due to its effectiveness and efficiency in many different scenarios [4,12,35,41,146]. However, it is important to assess how max-sum compares to other techniques, such as the recent ones based on dual decomposition [5, 49, 69, 74, 75, 87]. In particular, here we compare max-sum with the recent AD$^3$ (*alternating directions dual decomposition* [87][4]). In AD$^3$, each factor is represented as a constrained optimization (sub)problem, solved via the *augmented Lagrangian method*, which works by introducing a quadratic penalization on the constraint violation [87]. The authors of [87] provide guidelines on how to solve exactly and efficiently some particular types of sub-problems, namely those described with binary pairwise factors imposing first-order logic constraints. However, AD$^3$ can be adopted for arbitrary factors, given access to a "black box MAP solver" for the sub-problems generated by that factor. To instantiate AD$^3$ for our FG, we re-formulate it so that it contains only

[4]The code is available at *https://github.com/andre-martins/AD3*.

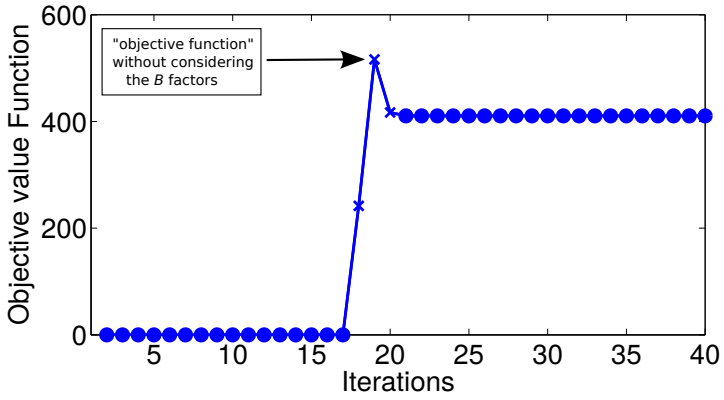



Figure 4-7: An example of objective function evolution along the max-sum iterations. Each circle indicate a valid configurations (assignment respecting the constraints) and each cross indicates a non-valid configuration (the corresponding value of the objective function is $-\infty$ in these cases, thus we show the value of $F$ without the $B_{jk}$ factors).

the type of constraints for which closed-form derivations are available (i.e. one-hot XOR, OR, OR-with-OUTPUT, negations, De Morgans laws, and AND-With-Output [87]). Hence, to efficiently enforce the biclustering constraints, we added indicators selecting the "active" rows or columns and forcing a variable $c_{ij}$ to be one if and only if row $i$ and column $j$ are active. Since we are keeping the intersection between selected rows and column, the result is always a sub-matrix and hence a bicluster.

In order to compare the two optimization techniques, we used the following synthetic dataset: (i) matrix $A$ contains $20 \times 20$ random values uniformly distributed in $[0,\ 1]$; (ii) an additively coherent bicluster, with 25% of the size of $A$, is placed at a random position; (iii) finally, the entire matrix is perturbed with Gaussian noise, with standard deviation equal to a percentage of the difference between the mean of the bicluster and the mean of the background. Concerning this noise percentage, we considered 5 values ranging from 0 (no noise) to 20% (high noise); for each of these noise percentage values, 15 matrices were generated, yielding a total of 75 data matrices. Our max-sum algorithm and AD$^3$ were compared in terms of both computational time and quality of the solution. This last aspect has been assessed using two standard indices: *purity* (*i.e.*, the percentage of points retrieved that actually belong to the real bicluster); *inverse purity* (*i.e.*, percentage of points belonging to the true bicluster which have been retrieved by the algorithms). Notice that these quantities are commonly known as *precision* and *recall*, in pattern recognition and information retrieval; concerning the Gene Match Score index presented for BAP in Sec. 4.2: purity and inverse purity respectively reflect the score provided by relevance and recovery.

Concerning execution time, $AD^3$ turned out to be very slow in this problem, with a computational time $3$ orders of magnitude slower than that of our algorithm (as shown in Figure 4-8b): this was somehow expected, namely due to the number of constraints introduced in the adaptation of the FG to the structure of $AD^3$. Concerns quality, Figure 4-8a shows the averaged product of purity and inverse purity as a function of the noise level, showing that max-sum seems to be slightly more robust to noise in this experiment. These results provides further evidence that, if the FG is well designed, the max-sum algorithm can provide very fast and robust optimization.

**Experiments on Synthetic datasets**

We compared the OOB with the two FG-based methods described in this Chapter (BAP and EB), using the same synthetic data generation method described in the previous section. In this case, we used $30$ matrices for each noise level (for a total of 150 matrices), and each matrix has dimensions $50 \times 50$. In this experiment, we considered two types of biclusters: constant-valued biclusters (we call this "constant bicluster benchmark") and additively coherent biclusters (we call this "evolutionary bicluster benchmark"). For both BAP and EB, we employed the approximate versions, because the dimension of the analyzed matrices is far beyond the computational capabilities of their exact versions. In particular, for BAP we used the heuristic aggregation methodology described in Sec. 4.2, which groups results obtained on smaller matrices (here we used 5x5 matrices, with no overlap). Regarding similarity, we employed the negative of the Euclidean distance for the constant bicluster benchmark, whereas for the evolutionary bicluster benchmark, we adopted the negative of Eq. (4.12).

Concerning EB, we used the authors' implementation[5] of the greedy algorithm given in [131] . Since this algorithm provides a pool of biclusters as solution, for a fair comparison all parameters were varied inside the suggested range[6] using ten equally spaced values and only the bicluster which maximizes the product of purity and inverse purity was considered.

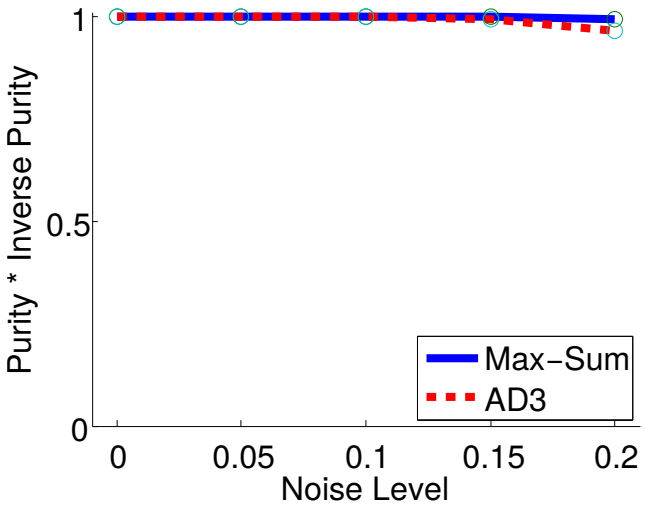
For the proposed method, the experimental details are the following:

- *Message scheduling*: even if different schedule are available [4], we adopt the most common approach of updating the messages in parallel, based on those from the previous iteration.

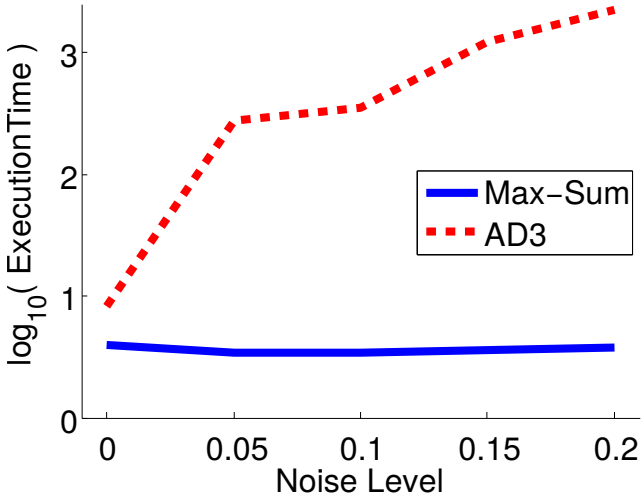
- *Convergence criteria*: the algorithm is stopped if the variable configurations do not change for 100 consecutive iterations.

[5] Available at *http://sist.shanghaitech.edu.cn/faculty/tukw/sdm11code.zip*

[6] Parameter ranges are described in the code documentation.




(a)




(b)

Figure 4-8: The plot 4-8a shows a comparison of the Purity and Inverse Purity obtained by the two optimization algorithms on a dataset with increasing noise level; while plot 4-8b shows a comparison of the computational time occurred to obtained such results.

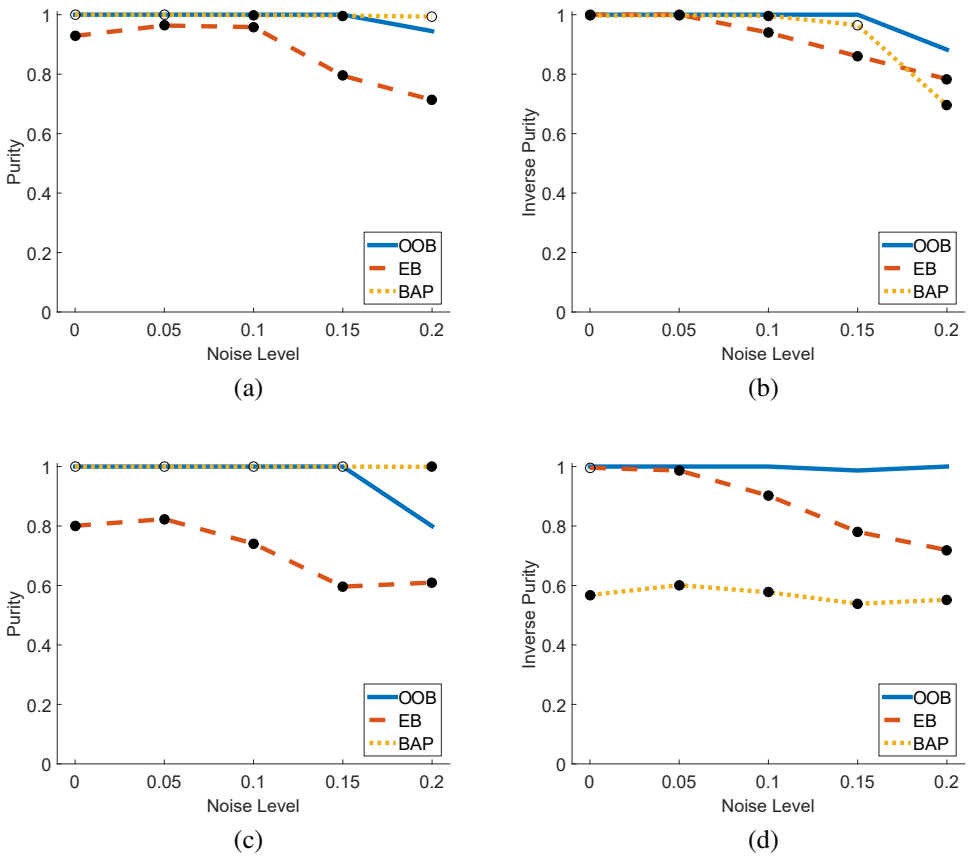



Figure 4-9: Purity and inverse purity for matrices with constant and additively coherent (evolutive) biclusters.

- *Parameter* $w$: we tune the coherence factors weight $w$ using the same strategy adopted for EB: in particular, we test 8 different values ranging from $2^{-8}$ to $2^{-1}$, and we use the best result.

The results are shown in Figure 4-9, where purity and inverse purity are plotted as a function of the noise level. Each point represents the average over the 30 runs at the corresponding noise level.

Further, we exploit a statistical test to better understand the differences between the results obtained by the different approaches. We thus performed a paired T-test comparing other FG based algorithms against the one described in this section. Specifically, in the plots, a full marker on the EB (or BAP) curve indicates that the difference between such method and the approach just described is statistically significant with significance level of 5%.

These results show that the approach we just present significantly outperforms the two approximate BAP and EB methods, especially for higher levels of noise,

| Significance Level (%) | 5 | 1 | 0.5 | 0.1 | 0.01 |
|---|---|---|---|---|---|
| Cheng&Church | ∼80 | ∼70 | ∼70 | ∼55 | ∼45 |
| ROCC | ∼98 | ∼98 | ∼98 | ∼98 | ∼98 |
| Bimax | 100 | 100 | 100 | 100 | 100 |
| EB | 100 | 100 | 100 | 100 | 100 |
| OOB | 100 | 100 | 100 | 100 | 100 |

Table 4.1: Results on Yeast dataset. Results for other approaches have been taken from Figure 3 in [131]. Algorithms reference: Cheng&Church [20], ROCC [32], Bimax [107], EB [131]

thus confirming the need for more robust optimization strategies in noisy scenarios. Concerning the two competing FG-based methods, it is important to note that BAP only performs well on the constant bicluster benchmark. Arguably, using Eq. (4.12) as similarity is not enough to appropriately identify evolutionary biclusters, which may be due to the fact that if two points are on the same row (or columns) the function (4.12) returns zero as coherence value, increasing the chance that those two points are included in the solution. This leads to larger biclusters including also background entries. The EB algorithm seems to be more robust, even if suffering more in case of additive coherent biclusters. Maybe, in this case, assuming that a bicluster is fully described by a single entry (the exemplar) is too strict, especially when the complexity increases due to the presence of strong noise.

### Gene expression data

The OOB algorithm was tested on two real gene expression datasets: the *Yeast* dataset [47] and the *Breast Tumor* dataset [98]. This type of matrices contains the expression levels of genes (rows) in a set of experimental conditions (columns): the goal is to retrieve biclusters where a subset of genes presents a coherent behaviour in a subset of experiments. These experiments have been performed adopting the heuristic described in Section 4.3.2. We assess the performance on both datasets following the experimental protocol proposed by EB authors in [131]: the obtained biclusters were evaluated by analyzing the Gene Ontology terms of the genes belonging to the same bicluster via the FuncAssociate[7] web-service, using five significance levels. To avoid implementation mistakes, we decided to collect other approaches performance from the state-of-the-art without re-executing the experiments. Unfortunately, different methods have been tested on the two different datasets and for these reasons, even if the evaluation protocol adopted was the same in the two datasets, we used two different tables to show the results.

For the Yeast dataset, in order to be comparable with the results in [131], only the 100 largest biclusters (with a maximum overlap of 25%) were considered as

[7] http://llama.mshri.on.ca/funcassociate/

| FABIA | ISA | Hiearc. | SAMBA | FLOC | OOB |
|---|---|---|---|---|---|
| 55% | 63% | 70% | 73% | 85% | **87.5%** |

Table 4.2: Results on Breast tumor dataset. Results for other approaches have been taken from [98]. Algorithms reference: FABIA [59], ISA [61], Hierarchical [117], SAMBA [121], FLOC [145]

part of the solution. Table 4.1 reports percentage accuracies obtained with our algorithm, for different significance levels, together with other state-of-the-art results from [131]. The table shows that the proposed approach compares very well with other methods on this dataset.

For Breast Tumor dataset, as it is commonly done in the literature [10, 113], we began by performing variance-based gene selection, in order to reduce the dimensionality of the data matrix. Then, we applied our method, using the validation protocol adopted in [98]: only the 40 largest biclusters were considered as part of the solution, for which the Gene Ontology index was evaluated at a 5% significance level. The results are reported in Table 4.2, in comparison with all the state-of-the-art methods studied in [98]; we can conclude that our method sets a new state-of-the-art for this dataset.

## 4.4 Other FG-based Approaches

One of the main advantages in using Factor Graphs is represented by the possibility of easily modifying the model by introducing or removing novel factors, given that Max-Sum messages can be computed efficiently. This Section provides some guidelines on how OOB can be extended to obtain more specific solutions.

First, we start by considering that setting the parameter $w$ in Equation 4.16 may be not trivial in some cases. In more details, this parameter "guides" the bicluster dimension by managing the importance of the similarity criteria (encouraging smaller solutions) with respect to the activation level (encouraging bigger solutions). As most parameters, its actual setting is crucial to obtain high quality solutions. To overcome the tuning of this parameter in scenarios where the bicluster size can be estimated, we introduce in the model the possibility to define, following some a-priori knowledge, a *favoured/fixed* size for the biclusters.

Second, inspired by a recent trend (*e.g.*, [81]), we propose an extension of OOB algorithm allowing the analysis of time series data. In these cases, samples (columns) refer to experimental conditions or features which are temporally related (e.g. growth stages of a plant, disease evolution, conditional measurements, video scenes). Biclustering approaches in this scenario focus on the identification of genes that act similarly in consecutive subset of samples [82], hence introducing a spatial constraint that cannot be satisfied by classical biclustering approaches, see Figure 4-10.

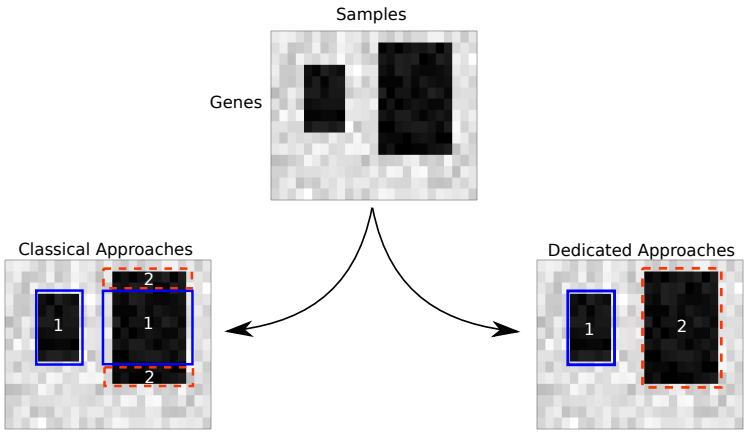



Figure 4-10: Biclustering of Time Series Data. The picture shows how the data matrix (on top) would be analysed by classical biclustering approaches (left), and how dedicated Time-Series biclustering approaches retrieve the desired information (right).

Although OOB algorithm was not designed to retrieve biclusters of a preferred size or to analyse temporal series, by introducing a novel set of factors, such tasks can be performed by OOB (provided that the update rules of the max-sum messages can be computed efficiently). To devise this advanced models we resort to a recent class of factors widely known in literature as Tractable Higher Order Potentials (THOP) [123]. With the term THOP we refer to a group of factors/constraints adoptable in binary models and for whom the Max-Sum update rules can be efficiently derived [41, 123]. Given a set of binary variables $\mathbf{x}$, there exist two big classes of THOPs: *cardinality potential* and *order based potentials*. Briefly, cardinality potentials concern functions assuming different values on the basis of the number of the active variables (variables in $\mathbf{x}$ which are set to 1). Whereas order based potentials involve functions assuming different values based on the position of the active variables [123].

### Preferred Size Model

In this case we resort to a class of THOP known as *cardinality potentials*. Given a set of binary variables $\mathbf{x} = (x_1, x_2, \ldots, x_l)$, cardinality potentials allow the user to guide the solution to be of a preferred size (i.e. the number of ones in $x$). Such feature can be exploited in various contexts (*e.g.*, market segmentation where market budget could be fixed [33]). These potentials can be specified in different manners depending on the problem to solve:

$$\sum_i x_i = k, \sum_i x_i \neq k, \sum_i x_i > k, \sum_i x_i < k$$

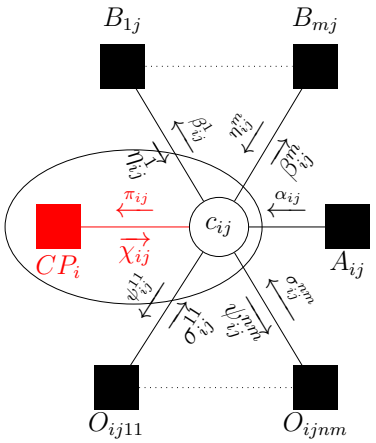



Figure 4-11: OOB model with the introduction of the *preferred-size factor*

with $k \in \mathbb{N}$. Here we describe the general representation from which the more specific ones can be derived. The cardinality potential we present is a function $Q : \mathbb{N} \rightarrow \mathbb{R}$, assigning a certain score to each possible cardinality value in $\{0, 1, 2, \ldots, l\}$, in this context we refer to cardinality as the sum of the active variables.

We thus insert a novel type of factors: $CP$. Such factors contribute to the global objective function described in Eq. 4.16, by increasing its value for solution having preferred size dimensions. Hence a $CP$ factor on $\mathbf{x}$ is defined as

$$CP = Q(\sum_i x_i) \tag{4.19}$$

where $Q$ is the above defined function. By setting the scores of $Q$ in correspondence of the interested size, we can encourage solutions of that dimension. For example, if we want to encourage the solution to have exactly $k$ columns, we just need to set an high score for $Q(\sum_i x_i) = k$ and apply the $CP$ factor to each row of the model. The introduction of such factors leads to the model described in Fig. 4-11.

Regarding messages updates for this novel factor, as illustrated in Sec. 4.3.2, messages going from variables to functions are easy to handle since they involve only summations. Specifically, referring to Equation 3.14, a particular message $\pi_{ij}$ (going from variable $c_{ij}$ to factor $CP_i$) can be computed as the summation of all the incoming messages in $c_{ij}$ except for $\chi_{ij}$ obtaining:

$$\pi_{ij} = \sum_{k=1}^{m} \eta_{ij}^{k} + \sum_{tk=(1,1)}^{(n,m)} \sigma_{ij}^{tk} + \alpha_{ij} \tag{4.20}$$

For the messages going from factors to variables the computation is slightly more complex since different steps are needed. Exploiting the cardinality potentials described in [123], we obtain the procedure shown in Algorithm 1.

**Algorithm 1** Computation of the $\chi_{ij} messages$

**Require:** Incoming messages $\pi_{i:}$.

Sort $\pi_{i:}$ in descending order obtaining $\pi_{i_b^*}$ where $i_b^*$ is the index of the incoming message with $b^{th}$ largest value
**for** $z \in \{0, \ldots, N\}$ **do**
  $u_{-1}(z) = \sum_{z'=1}^{z} \left[\pi_{i_{z'}^*} + Q(z'-1)\right]$
  $u_0(z) = \sum_{z'=0}^{z} \left[\pi_{i_{z'}^*} + Q(z')\right]$
  $u_1(z) = \sum_{z'=0}^{z-1} \left[\pi_{i_{z'}^*} + Q(z'+1)\right]$
**end for**
**for** $z \in \{0, \ldots, N\}$ **do**
  $s_1^L(z) = \max_{z' \in \{0,\ldots,z\}} c_1(z')$
  $s_{-1}^R(z) = \max_{z' \in \{z,\ldots,N\}} c_{-1}(z')$
  $s_0^L(z) = \max_{z' \in \{0,\ldots,z\}} c_0(z')$
  $s_0^R(z) = \max_{z' \in \{z,\ldots,N\}} c_0(z')$
**end for**
$\chi_{ij}(0) = \max\left[s_0^L(r(j)-1), s_{-1}^R(r(j)+1)\right]$
$\chi_{ij}(1) = \max\left[s_1^L(r(j)-1), s_0^R(r(j)+1)\right]$
$\chi_{ij} = \chi_{ij}(1) - \chi_{ij}(0)$

To actually plug in the novel factors in the OOB model we need to introduce some slight modifications in the messages previously derived. By observing Equations 3.13 and 3.14, it is straightforward that only messages going from variables to factors are affected by these modifications; which simply involve the introduction of $\chi$ messages in the respective summation.

This variant of the OOB algorithm has been preliminary tested on synthetic examples. However we are planning to assess its performances on real scenarios in the real near future.

### Time Series Model

As depicted in Fig. 4-10, the analysis of time-series cannot be performed with classical biclustering approaches. This is due to temporal relationship holding between consecutive columns (or rows).

To introduce such information in the model we resort to another THOP called *convex set potential* [123]. This potential, belonging to the group of *order based potentials*, defines an hard constraints enforcing the set of active variables to be a convex set (*i.e., a set of ones without zeros in between*). This potential reflects our need in obtaining solution involving continuous columns (or rows). Given a set of variable binary variables $x$, we define the *Time-Series* constraint $TS$ as a convex-set potential. Assuming that the columns of data matrix $A$ represent the experimental

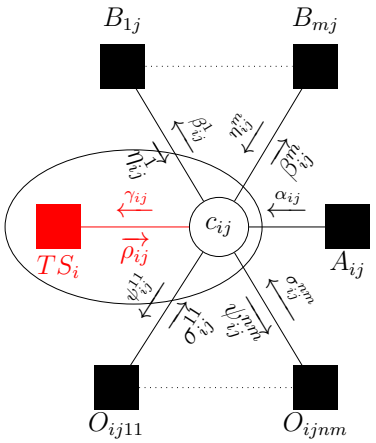



Figure 4-12: Biclustering Factor Graph for Time-Series dataset

conditions, if we apply such constraint on each row of $C$ the solution is enforced to have contiguous columns and hence to contain a time-series bicluster. Thus, we define the Time Series constraint as:

$$TS_i(c_{i1}, \ldots, c_{im}) = \begin{cases} 0 & \text{if ones in } [c_{i1}, \ldots, c_{im}] \\ & \text{form a contiguous subset} \\ -\infty & \text{otherwise;} \end{cases}$$

and by applying one of this for each row, we force the solution to have contiguous columns. Figure 4-12 represents the time-series model.

Concerning messages updates, as for the preferred-size model, the messages from variables to functions involve the summation of all the messages incoming in $c_{ij}$ except for $\rho_{ij}$; whereas basic messages will include $\rho_{ij}$, as follow:

$$\psi_{ij}^{tk} = \sum_{\hat{tk}} \sigma_{ij}^{\hat{tk}} + \sum_{k} \eta_{ij}^{k} + \alpha_{ij} + \rho_{ij}.$$
$$\beta_{ij}^{k} = \sum_{\hat{k}} \eta_{ij}^{\hat{k}} + \sum_{tk} \sigma_{ij}^{tk} + \alpha_{ij} + \rho_{ij}.$$

Whereas the message going from a factor node $TS_i$ to a variable node $c_{ij}$ can be computed as follows:

$$\begin{aligned} \rho_{ij} =& MS(c_{1:j-1}, c_{j-1} = 1) + MS(c_{j+1:N}, c_{j+1} = 1) + \\ & - \max\left(MS(c_{1:j-1}), MS(c_{j+1:N})\right). \end{aligned}$$

Considering $\rho$ messages provided by each variable as weights, $MS(c_{1:j-1}, c_{j-1} =$

1) is a function retrieving the maximum weighted contiguous subsequence in the subset $\{c_1, \ldots, c_{j-1}\}$ and forcing $c_{j-1}$ to be equal to 1 (if the second part is missing, no variables are constrained).

As for the preferred-size model, to actually plug in the Time-Series factors, we need to introduce $\rho$ messages in the summations concerning messages going from variables to previously defined factors.

Note that the usage of such factors is not exclusive, hence we can obtain other models where both of them are included allowing us to obtain more specific solutions which are contiguous and, preferably, of a given size.

The Time series Factor Graph has been preliminary tested on a real gene expression dataset to assess its performances. The results are presented in what follows.

**Experiments** In this case we adopt the Cho *et al* yeast cell-cycle dataset [23]. The dataset is composed by a matrix where 6457 genes have been sampled in 17 consecutive time steps with an interval of 10 minutes.

Also in this situation the bottleneck in the practical usage is represented by the space complexity required by the model. Hence, as previously proposed, we run the algorithm on randomly extracted sub-matrices where about 120 rows have been selected. First, to reduce the matrix dimensionality we applied a variance-based gene selection, as already adopted in relevant literature [10, 113]. Each row of the obtained matrix have been then rescaled such that the 2-sigma interval lies in $[0, 1]$, and missing values have been recovered using the method proposed in [130]. We retrieve one temporal bicluster from each sub-matrix. As suggested by results on synthetic datasets, the approach has been executed with parameter $w$ equal to 0.0625. The introduction of Time-Series factors does not affect the Max-Sum convergence nor the execution time required. We keep all the obtained temporal bicluster and we assess their quality with the following evaluation criteria:

1. *Mean Square Residue (MSR)* - introduced in [20] it assess the fluctuation of expression level for all rows in the bicluster. The smaller the MSR, the higher the correlation. Given a bicluster $A_{TK}$, the MSR is computed as

$$MSR(TK) = \frac{1}{|T||K|} \sum_{t \in T} \sum_{k \in K} (a_{tk} - a_{Tk} - a_{tK} + a_{TK})^2$$

   where $a_{Tk}$ is the mean of the $k^th$ column in the bicluster, $a_{tK}$ is the mean of the $t^th$ row in the bicluster, and $a_{TK}$ is the mean of the whole bicluster.

2. *Gene Ontology (GO) terms* - as expressed in previous sections, GO terms are a fundamental qualitative information that highlights whether the genes contained in a certain bicluster are biologically related or not. The GO enrichment analysis have been performed via the *GOstat* online application[8] [6]

[8] http://gostat.wehi.edu.au/

| ID | #Rows | #Cols (first-last) | MSR | GO-Terms |
|---|---|---|---|---|
| 1 | 20 | 3 (14 - 16) | 0.0017 | GO:0065008<br>GO:0022890 |
| 2 | 18 | 4 (12 - 15) | 0.0011 | GO:0042254<br>GO:0022613 |
| 3 | 18 | 3 (14 - 16) | 0.0007 | GO:0009063<br>GO:0044270 |
| 4 | 17 | 4 (6 - 9) | 0.0015 | GO:0005515<br>GO:0051649 |
| 5 | 17 | 4 (6 - 9) | 0.0015 | GO:0008202<br>GO:0016125 |
| 6 | 17 | 3 (11 - 13) | 0.0008 | GO:0065008<br>GO:0022890 |
| 7 | 17 | 3 (6 - 8) | 0.0013 | GO:0031974<br>GO:0043233 |
| 8 | 16 | 3 (3 - 5) | 0.0005 | GO:0031980<br>GO:0005759 |
| 9 | 16 | 3 (7 - 9) | 0.0016 | GO:0016020<br>GO:0031090 |
| 10 | 16 | 3 (4 - 6) | 0.0017 | GO:0005730<br>GO:0015078 |

Table 4.3: Top 10 largest Biclusters obtained by the proposed approach on the yeast dataset.

setting as p-value threshold 0.05.

Table 4.3 reports the results concerning the top 10 largest biclusters, showing for each bicluster: the number of rows composing the bicluster, the columns interval selected, the MSR and the top 2 GO terms according to the p-value and number of genes involved. It can be seen by the MSR that the retrieved biclusters present high coherence (in [19] authors adopt 300 as threshold for the MSR). Furthermore, it can be seen by the GO terms column that every bicluster contains genes involved in the same biological processes; in fact for every bicluster we can retrieve at least 2 GO terms with p-value below 0.05. Surely it would be interesting to compare this approach with other state-of-the-art techniques such as [82, 144, 150], however with these preliminary results we want to demonstrate that the extension of FG based approaches to time-series dataset is sound and, moreover, it provides accurate and biologically meaningful solutions. To make results comparable with current state-of-the-art a first step could be trying to merge the results thus increasing the biclusters size. A possible solution can be similar to the one exploited in Sec. 4.2: i) given the set of biclusters previously obtained, merge all the rows of the biclusters

| ID | #Rows | #Cols (first-last) | MSR | GO-Terms |
|---|---|---|---|---|
| 11 | 38 | 3 (14 - 16) | 0.0013 | **GO:0005737**<br>GO:0022890 |
| 12 | 57 | 4 (6 - 9) | 0.0015 | GO:0044425<br>GO:0005515 |

Table 4.4: Result of merged biclusters. Highlighted in bold a GO term that was not enriched in basic results, this GO term involve 31 of the 38 genes in the bicluster.

sharing the same column subset; ii) then evaluate the FG objective function for each novel bicluster and, if the value is bigger then the sum of the previous ones, replace them.

Table 4.4 presents the characteristics of two biclusters obtained with the heuristic just described. Although the biclusters size are two/three times bigger than the one presented in 4.3, the MSR of the retrieved biclusters is similar. This suggests that the proposed approach would provide accurate solution also on bigger data matrices. Moreover grouping the results allows to retrieve different and more informative GO terms: specifically in the GO term highlighted in bold 31 of the 38 selected genes are involved in the same biological processes.

## 4.5 Conclusions and Future Works

In this chapter we presented the Factor Graph based approaches we devised to face the biclustering problem. In this context we produced the following contributions:

1. we devised a preliminar algorithm trying to extend what proposed in AP to the biclustering context. We thus designed the *BAP* model where we propose an exemplar based grouping of the entries; this is possible by introducing a novel constraint ensuring groups of entries to be biclusters (*i.e.,* having rectangular shapes). We reformulated the model linearly and we solved it optimally exploiting Linear Programming techniques. Results suggested that investigating the usage of binary variables in FG approaches could lead to more efficient and effective models.

2. We thus reformulate the biclustering problem as the iterative search of one bicluster at a time. This is the basis concept of the OOB approach. We devised this model as the iterative search of the biggest and most coherent biclusters. Thus there are two opposite forces (sum of the entries and incoherence) regulating the bicluster composition. We also reformulate the biclustering constraint. The devised model allowed us to analytically derive efficient Max-Sum update rules. We thus obtained a compact and efficient model. Results

show the OOB favourably compares with the relevant state-of-the-art on both synthetic and real datasets.

3. The designing of the OOB approach led to an important step forward concerning the scalability issue highlighted in previous FG based techniques for biclustering; passing from $10 \times 10$ analysed matrices up to $50 \times 50$ matrices.

4. We derived other two OOB variants by exploiting THOPs: a particular set of factors whose message update rules can be computed efficiently. The novel factors allowed us to retrieve preferred size constraint and to analyse Time Series dataset. These extensions do not affect the complexity (in both time and space) of the original OOB model. The principal objective of such extensions was to provide guidelines on how a Factor Graph model can be specialized/generalized by introducing/removing novel factors.

5. Finally, we experimentally tested the Time-Series approach on a real gene expression time series dataset. In this context, preliminary results suggest that FG based approaches are persistent to the introduction of novel factors; and in this specific case, the model can provide coherent and biologically significant solutions.

Although OOB provides a more scalable algorithm, the scalability problem is still far from being solved. An interesting study in this direction concerns the investigation of integer variables (instead of binary), even though is much more difficult to obtain efficient Max-Sum messages update rules. For this reason, it could be also interesting to investigate the existence of particular factors (based on integer variables) whose Max-Sum update rules could be computed efficiently (similarly to what done with THOPS in the case of binary variables). Other research directions may concern the investigation of the devised approaches (OOB and its extensions) in other scenarios (biological or not); with particular dedication to the differences between the preferred size model and the original OOB:

# Chapter 5

# Bayesian Network approaches for Biclustering

Previous chapter presented the Factor Graph based approaches for the biclustering problem. In this Chapter we introduce the devised Bayesian Networks approaches relying on Matrix Factorization concepts. Thus, we first present the relevant state-of-the-art concerning probabilistic matrix factorization and biclustering. Then we describe our novel model exploiting Bayesian Network and the EM algorithm to solve the biclustering problem. Finally, we introduce an extension allowing to exploit prior-knowledge to improve the results.

## 5.1 Biclustering and Sparse Low-Rank Factorization

As widely discussed in Sec. 2.2, several different approaches have been proposed for biclustering, and most of them can be divided into four main classes [98]: *correlation maximization methods*, *variance minimization methods*, *two-way clustering methods* and *probabilistic/generative methods*.

Among these techniques a recent trend of approaches exploits tools deriving from the *Matrix Factorization* scenario [51, 59, 79, 94, 105, 108, 134, 137, 147]. Typically, a matrix factorization aims at approximating a given data-matrix through the multiplications of other matrices. The obtained matrices can provide novel information concerning the values distribution inside the original data matrix. A notorious example of a matrix factorization technique is represented by the Singular Value Decomposition (SVD) algorithm [50] where the data matrix $D$ is decomposed by the product of three matrices $D = USV^T$ [50]: in this case, considering the matrices $DD^T$ and $D^TD$, the matrix $S$ provides their eigenvalues (which are the same), whereas the eigenvectors are stored in the rows of matrix $U$ (in one case) and in the columns of matrix $V^T$ (in the other case). Specifically for the biclustering context, recently non-negative tri-factorization techniques have been presented [79, 137, 147]; however, in what follows we do not discuss such algorithms,

since they do not involve a probabilistic representation of the problem.

Although probabilistic approaches tend to be computational heavy, they offer several advantages. The underlying *generative model*, as discussed in Sec. 3.2, can be used to generate synthetic data, which when compared with real data, allows assessing the validity of the model. Moreover, probabilistic inference provides estimates of the confidence/uncertainty level of the obtained estimates. Different probabilistic approaches to biclustering have been proposed, ranging from methods that use graph-based models [121] to Gibbs sampling [115] and Markov chains [110]. Concerning probabilistic approaches for biclustering adopting matrix factorization, most of them rely on what are known as *latent block models* [51, 94, 105, 108, 134]. Latent block models approaches are probabilistic techniques, often solved through the EM algorithm, where the goal is to simultaneously rearrange rows and columns of a data matrix into groups of similar response patterns. The common assumption made by these models is that each row or column belongs exclusively to only one row or column group. Thus the data matrix, in this case, is divided into exhaustive and non overlapping biclusters containing similar patterns.

In contrast to what presented by previously mentioned methods, we present a probabilistic model where matrix factorization concepts are exploited to obtain (possibly) overlapping biclusters that do not represent an exhaustive partition of the original data-matrix.

### 5.1.1 Biclustering via Sparse Low-Rank Matrix Factorization (SLRMF)

In this section we present the connections between biclustering and sparse low-rank matrix factorization. Firstly, for clarity purposes, we need to introduce some notation that could slightly differ from the one presented in Sec. 2.2.

**Notation:** in what follows we refer to matrices using capital letters (*e.g.*, $D, V, Z$), to vectors with lower-case letters (*e.g.*, $d, v, z$), and to matrix/vector elements using subscripts (*e.g.*, the entry $(i, j)$ of matrix $D$ is $d_{ij}$ and the component $p$ of vector $d$ is $d_p$). The so-called "vec" operator (vectorization) takes a matrix argument and returns a vector with the matrix elements stacked column by column. The reverse operation is denoted $\text{vec}^{-1}$ (*i.e.*, such that $\text{vec}^{-1}\big(\text{vec}(D)\big) = D$). A pair of useful equalities concerning the vec operator are

$$\text{vec}(AB) = (I \otimes A)\text{vec}(B) = (B^T \otimes I)\text{vec}(A), \tag{5.1}$$

where $I$ is an identity matrix of adequate dimensions and $\otimes$ is the Kronecker matrix product [84]. Finally, given some matrix $A$, $\|A\|_F$ denotes its Frobenius norm, which is the Euclidean norm of its vectorization: $\|A\|_F = \|\text{vec}(A)\|_2$.

We have described in Sec.2.2 that there exist different adoptable coherence cri-

teria usable to solve the biclustering problem; among them one possible choice is that entries belonging to the same bicluster should have a similar value, significantly different from the other entries of the matrix. For example, a data matrix containing one bicluster with rows $T = \{1, 2\}$ and columns $K = \{1, 2\}$ may look like

$$D = \begin{bmatrix} \mathbf{10} & \mathbf{10} & 0 & 0 \\ \mathbf{10} & \mathbf{10} & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

From an algebraic point of view, this matrix can be represented by the outer product $D = vz^T$ of the vectors

$$v = \begin{bmatrix} \mathbf{5} \\ \mathbf{5} \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad \text{and} \quad z = \begin{bmatrix} \mathbf{2} \\ \mathbf{2} \\ 0 \\ 0 \end{bmatrix}.$$

Whereas in the case of a matrix with two biclusters with $T_1 = \{1, 2\}$, $T_2 = \{5, 6\}$, $K_1 = \{1, 2\}$ and $K_2 = \{3, 4\}$, as in

$$D = \begin{bmatrix} \mathbf{10} & \mathbf{10} & 0 & 0 \\ \mathbf{10} & \mathbf{10} & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & \mathbf{7} & \mathbf{7} \\ 0 & 0 & \mathbf{7} & \mathbf{7} \end{bmatrix}.$$

such matrix can be represented by the outer product $D = VZ^T$ where

$$v_1 = \begin{bmatrix} \mathbf{5} \\ \mathbf{5} \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad z_1 = \begin{bmatrix} \mathbf{2} \\ \mathbf{2} \\ 0 \\ 0 \end{bmatrix}, \quad v_2 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \mathbf{1} \\ \mathbf{1} \end{bmatrix}, \quad z_2 = \begin{bmatrix} 0 \\ 0 \\ \mathbf{7} \\ \mathbf{7} \end{bmatrix},$$

$$V = [v_1, v_2] \text{ and } Z = [z_1, z_2]^T.$$

In what follows we refer to $v$ vectors as *prototypes* and to $z$ vectors as *factors*. Note that by observing the prototypes in $V$ and the factors in $Z$ we can obtain informa-

tion about the biclusters positions and their values. Furthermore, the connection between biclustering and sparse low-rank matrix factorization can be evidenced by observing that the factorization of the original data matrix shows that it has rank no larger than the number of biclusters (usually much lower than the number of rows or columns). Therefore the matrix is *low-rank*. Moreover, if the size of the matrix $D$ is much bigger than the bicluster size (as it is typically the case in many applications), the resulting prototype and factor vectors should be composed mostly by zeros (*i.e.*, the prototypes and factors should be sparse).

Generalizing to $p$ biclusters, we can formulate the biclustering problem as the decomposition of a given data matrix $A$ as the sum of $k$ outer products,

$$D = \sum_{i=1}^{k} v_i z_i^T = VZ, \tag{5.2}$$

where $V = [v_1, \ldots, v_k] \in \mathbb{R}^{n \times k}$ and $Z = [z_1, \ldots, z_k]^T \in \mathbb{R}^{k \times m}$.

### 5.1.2 Biclustering via *Probabilistic SLRMF*

As mentioned in the previous Section, probabilistic approaches provide several advantages: for instance the possibility to couple a confidence level to the obtained estimates. Other probabilistic approaches have been proposed exploiting matrix factorization, among these the *Factor Analysis for BIclustering Acquisition* (FABIA) algorithm [59] represents some interesting features, and it is discussed in what follows.

FABIA is a generative model for biclustering based on factor analysis [59]. The model proposed to decompose the data matrix is obtained by adding noise to the strict low rank decomposition in (5.2),

$$D = \sum_{i=1}^{k} v_i z_i^T + Y = VZ + Y, \tag{5.3}$$

where matrix $Y \in \mathbb{R}^{n \times m}$ accounts for random noise or perturbations, assumed to be zero-mean Gaussian with a diagonal covariance matrix. As mentioned above the *protoypes* $v_i$ in $V$ and *factors* $z_i^T$ in $Z$ should be sparse. To induce sparsity, FABIA uses two type of priors: (i) an independent Laplacian prior, and (ii) a prior distribution that is non-zero only in region where prototypes are sparse. In this latter case a parameter $spL$ is introduced, and the points whose prior distribution is below $spL$ are set to zero. However, this model formulation leads to an analytically intractable likelihood, whose integral cannot be derived. This prevents the derivation of exact forms for the steps of the EM algorithm. In particular, as presented in Chap. 3, FABIA authors estimate the parameters resorting to a Variational EM [59], thus approximating the integral in the derivation of the $Q$ function. Another important

drawback of FABIA is the fact that there is not an explicit information about biclusters membership. In fact, to obtain such information, authors of FABIA adopt a downstream procedure involving two thresholds (one for the prototypes and one for the factors) [59]. Although authors propose an automatic setting of such thresholds, this suppose to know the percentage of rows/columns that will (on average) belong to a bicluster.

In the next section we present the approach we devised to overcome the FABIA drawbacks.

## 5.2 Spike and Slab Biclustering

As previously discussed, FABIA presents two main drawbacks: i) the prior distribution adopted led to the usage of Variational EM; ii) the bicluster memberships are recovered exploiting thresholds. In what follows we present our method which overcomes both.

Similarly to FABIA, our method involves two main ingredients:

1. the data matrix is approximated by a *low rank* matrix; in particular, each bicluster has rank 1 and corresponds to the outer multiplication of two vectors, with the data matrix being modeled as the sum of a collection of such rank-1 products (*i.e.*, biclusters).

2. the vectors that correspond to each bicluster are expected to be *sparse*; in fact, most data matrices adopted in the biclustering scenario have a large number of rows/columns (*i.e.*, thousands by hundreds, in gene expression data) and the biclusters typically involve only small portions thereof.

Since, in practice, no matrix of real data is exactly low rank, as in FABIA we model deviations from the low rank assumption as a Gaussian perturbation added to the underlying low-rank matrix. Furthermore, in order to enforce sparsity on the factors, we propose to use a *Spike and Slab* prior.

In contrast with FABIA, the proposed formulation leads to a computationally tractable likelihood, allowing us to estimate the proposed generative model parameters through an instance of the *expectation-maximization* (EM) algorithm. Before describing the approach, we introduce the prior distribution adopted: the so called Spike and Slab prior.

### 5.2.1 Spike and Slab prior

The original Spike and Slab model was proposed by Mitchell and Beauchamp [90] for variable selection in linear regression. It was later generalized and adopted by many authors as a general purpose sparsity-inducing prior [63]. In its basic form, the spike and slab is a univariate prior composed by the mixture of two zero-mean

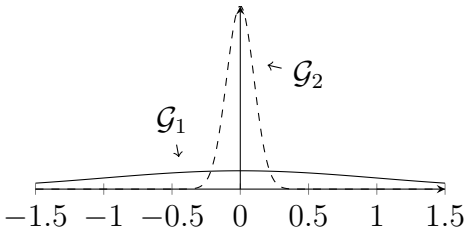



Figure 5-1: Example of spike and slab prior distribution. The figure shows how the two Gaussian distributions describe a sparse set of samples. The low variance Gaussian $\mathcal{G}_2 = \mathcal{N}(0, 0.1)$ describes nearly zero samples, while the large variance Gaussian $\mathcal{G}_1 = \mathcal{N}(0, 1)$ models large magnitude values.

Gaussian distributions: one with very small variance, modeling a high probability of nearly zero values, and another one with large variance, which models the presence of large values. Under this density, both very large and very small (nearly zero) samples have high likelihood, something that is not possible under a single Gaussian. An illustration of a Spike and Slab prior is shown in Figure5-1. To generate a point from such model we randomly select (with a certain probability) one of the two Gaussians, and then obtain a sample from the chosen distribution. The idea is that: sampling from the Gaussian with small variance we obtain background values; whereas sampling from the Gaussian with large variance we obtain foreground values.

Formally, for a given one dimensional point $x$, the spike and slab prior has the form

$$\mathcal{P}(x|\alpha, \tau_1, \tau_2) = \alpha\,\mathcal{N}(x|0, \tau_1^2) + (1-\alpha)\mathcal{N}(x|0, \tau_2^2) \tag{5.4}$$

with $\tau_2 \lll \tau_1$, parameter $0 \leq \alpha \leq 1$ regulates the sparsity degree, and $\mathcal{N}(x|\mu, \sigma^2)$ denotes a Gaussian density mean $\mu$ and variance $\sigma^2$, computed at $x$.

Note that (5.4) is equivalent to the following two-stage model

$$\mathcal{P}(x|h, \tau_1, \tau_2) = \mathcal{N}(x|0, \tau_1^2)^h \mathcal{N}(x|0, \tau_2^2)^{(1-h)}, \tag{5.5}$$

$$\mathcal{P}(h|\alpha) = \alpha^h (1-\alpha)^{1-h}, \tag{5.6}$$

where $h$ is a (not observed, or latent) binary variable following a Bernoulli distribution of parameter $\alpha$. The mixture in Eq. 5.4 results from marginalizing this model with respect to $h$.

### 5.2.2 The SSBi Model

Formalising the above-mentioned ingredients, we can define the *Spike and Slab BIclustering* (SSBi) approach as follows:

1. the data matrix $D \in \mathbb{R}^{n \times m}$ is modeled as in FABIA, *i.e.,* with a Gaussian

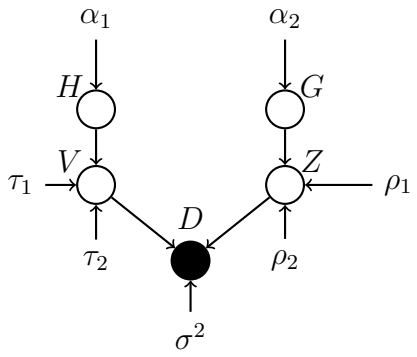



Figure 5-2: The Bayesian Network of the proposed spike and slab biclustering. The corresponding conditionals are given by Equations (5.7), (5.8), (5.9), (5.10), and (5.11).

distribution having the product $VZ$ as mean (with $V \in \mathbb{R}^{n\times k}$, $Z \in \mathbb{R}^{k\times m}$, and $k$ represents the number of biclusters to retrieve) and $\sigma$ as standard deviation (representing the approximation noise). This part provides the "low-rank" assumption, since the approximation matrix has rank $k$, at maximum.

2. the prototype and factor matrices (*i.e.*, $V$ and $Z$) are *sparse*, and we adopt a spike and slab prior to induce that feature.

The Bayesian Network describing the proposed approach (which is sketched in Figure 5-2) is formally defined as follows:

- Given the product $VZ$, the entries of the data matrix $D$ are i.i.d. Gaussian with variance $\sigma^2$:

$$\begin{aligned} \mathcal{P}(D|V,Z,\sigma^2) &= \mathcal{N}(D|VZ,\sigma^2 I) \\ &= \prod_{i=1}^{n}\prod_{j=1}^{m}\mathcal{N}(d_{ij}|(VZ)_{ij},\sigma^2). \end{aligned} \tag{5.7}$$

- The entries of $V$ follow a spike and slab prior with variances $\tau_1^2$ and $\tau_2^2$ (such that $\tau_1^2 \gg \tau_2^2$),

$$\mathcal{P}(V|H,\tau_1,\tau_2) = \prod_{i=1}^{n}\prod_{j=1}^{k}\mathcal{N}(v_{ij}|0,\tau_1^2)^{h_{ij}}\mathcal{N}(v_{ij}|0,\tau_2^2)^{1-h_{ij}} \tag{5.8}$$

where the binary latent variables in matrix $H$ follow a Bernoulli distribution of parameter $\alpha_1$,

$$\mathcal{P}(H|\alpha_1) = \prod_{i=1}^{n}\prod_{j=1}^{k}\alpha_1^{h_{ij}}(1-\alpha_1)^{1-h_{ij}}. \tag{5.9}$$

- The entries of $Z$ also follow a spike and slab prior, with variances $\rho_1^2$ and $\rho_2^2$ (such that $\rho_1^2 \gg \rho_2^2$),

$$\mathcal{P}(Z|G,\rho_1,\rho_2) = \prod_{i=1}^{k}\prod_{j=1}^{m}\mathcal{N}(z_{ij}|0,\rho_1^2)^{g_{ij}}\mathcal{N}(z_{ij}|0,\rho_2^2)^{1-g_{ij}} \tag{5.10}$$

  where the binary latent variables in matrix $G$ follow a Bernoulli distribution of parameter $\alpha_2$,

$$\mathcal{P}(G|\alpha_2) = \prod_{i=1}^{k}\prod_{j=1}^{m}\alpha_2^{g_{ij}}(1-\alpha_1)^{1-g_{ij}}. \tag{5.11}$$

Intuitively $\alpha_1$ and $\alpha_2$ regulate the sparsity degree in each prototype and factor vector or, equivalently, the biclusters dimensions on rows and columns respectively. The standard deviations $\tau_1, \tau_2, \rho_1$ and $\rho_2$ regulate the value ranges.

The joint distribution of all the variables and parameters involved in this model can now be written as

$$\begin{aligned}
&\mathcal{P}(D,V,Z,H,G,\sigma,\tau_1,\tau_2,\rho_1,\rho_2,\alpha_1,\alpha_2)\\
=\;&\mathcal{P}(D|V,Z,\sigma^2)\mathcal{P}(V|H,\tau_1,\tau_2)\mathcal{P}(Z|G,\rho_1,\rho_2)\mathcal{P}(H|\alpha_1)\mathcal{P}(G|\alpha_2)\\
&\mathcal{P}(\sigma,\tau_1,\tau_2,\rho_1,\rho_2,\alpha_1,\alpha_2),
\end{aligned} \tag{5.12}$$

where $\mathcal{P}(\tau_1,\tau_2,\rho_1,\rho_2,\alpha_1,\alpha_2)$ is a prior on the model parameters. In this paper, we consider this prior to be flat, that is, we seek maximum likelihood (ML) estimates thereof.

Finally, notice that this model may be easily extended to the case where each bicluster has its own parameter set (the spike and slab variances and mixing probability), rather than being assumed the same for all the biclusters. However, to keep the notation simpler, we will proceed with the simpler version that we have just introduced.

### 5.2.3 Parameter Estimation

With the proposed approach we aim at retrieving $k$ biclusters from a given data matrix $D$ exploiting the devised Bayesian Network in Fig. 5-2. Hence we estimate the model parameters $\tau_1, \tau_2, \rho_1, \rho_2, \alpha_1, \alpha_2$, and $\sigma$ resorting to the EM algorithm presented in Sec. 3.2.1. We want to remark that computing the expectation yielding the Q-function may not be trivial in general, as it may involve intractable integration, as in the case of the FABIA model [59].

Concerning the unobserved $V, Z, H$, and $G$, we have the choice of marginalizing them out, which can be done via the EM algorithm by treating them as latent

variables, or maximizing with respect to them, which corresponds to seeing them as parameters rather than latent variables. Inspired by [37], and in order to obtain a simpler E-Step, we treat $H$ and $G$ as hidden variables, but $V$ and $Z$ as unknown parameters to be estimated along with $\tau_1, \tau_2, \rho_1, \rho_2, \alpha_1, \alpha_2$, and $\sigma$. We could also treat $H$ and $G$ as parameters; however, since these are matrices of binary variables, maximize with respect to them would correspond to taking hard decisions, which may have a strong influence in the whole optimization procedure. On the other hand, $V$ and $Z$ are matrices of real-valued entries, thus estimating them has a smoother/weaker influence in the estimates of the other quantities. For these reasons, we define $V$ and $Z$ as parameters, and $H$ and $G$ as hidden variables.

In what follows, we present the form that the E-step and the M-step take in the proposed SSBi model.

### The E-Step

To keep the notation more compact, we denote the complete set of parameters as $\theta = \{V, Z, \sigma^2, \alpha_1, \alpha_2, \tau_1, \tau_2, \rho_1, \rho_2\}$. Recall that the joint distribution of all the variables and parameters is given in Eq. 5.12. With $D$ observed and $H$ and $G$ as latent, the $\mathcal{Q}$ function (needed for the EM) is obtained by computing

$$\mathcal{Q}(\theta, \hat{\theta}^{(t)}) = \mathbb{E}_{H,G}\Big[\log \mathcal{P}(D, H, G, \theta)\big|\hat{\theta}^{(t)}, D\Big].$$

After some simple but long and tedious analytic manipulations, and dropping any terms that do not depend on $\theta$, we obtain the following closed-form expression:

$$\begin{aligned}
&\mathcal{Q}(\theta, \hat{\theta}^{(t)}) = \qquad (5.13)\\
&\quad - \frac{nm}{2}\log(\sigma^2) - \frac{||D - VZ||^2}{2\sigma^2}\\
&\quad - \frac{||\overline{H}^{(t)}||_F}{2}\log(\tau_1^2) - \frac{||1 - \overline{H}^{(t)}||_F}{2}\log(\tau_2^2)\\
&\quad - \frac{||\overline{G}^{(t)}||_F}{2}\log(\rho_1^2) - \frac{||1 - \overline{G}^{(t)}||_F}{2}\log(\rho_2^2)\\
&\quad - \frac{1}{2}v^T\overline{H}^{(t)}v - \frac{1}{2}z^T\overline{G}^{(t)}z\\
&\quad + \Big(\sum_{p=1}^{nk}\overline{h}_p^{(t)}\Big)\log\Big(\frac{\alpha_1}{1-\alpha_1}\Big) + nk\log(1-\alpha_1)\\
&\quad + \Big(\sum_{j=1}^{km}\overline{g}_j^{(t)}\Big)\log\Big(\frac{\alpha_2}{1-\alpha_2}\Big) + km\log(1-\alpha_2)
\end{aligned}$$

where $v = \text{vec}(V)$, $z = \text{vec}(Z)$,

$$\overline{H}^{(t)} = \text{diag}\Big(\frac{\overline{h}_1^{(t)}}{\tau_1^2} + \frac{1-\overline{h}_1^{(t)}}{\tau_2^2}, \ldots, \frac{\overline{h}_{nk}^{(t)}}{\tau_1^2} + \frac{1-\overline{h}_{nk}^{(t)}}{\tau_2^2}\Big), \tag{5.14}$$

$$\overline{G}^{(t)} = \text{diag}\Big(\frac{\overline{g}_1^{(t)}}{\rho_1^2} + \frac{1-\overline{g}_1^{(t)}}{\rho_2^2}, \ldots, \frac{\overline{g}_{km}^{(t)}}{\rho_1^2} + \frac{1-\overline{h}_{km}^{(t)}}{\rho_2^2}\Big), \tag{5.15}$$

and, for $p = 1, ..., nk$, and $j = 1, ..., km$,

$$\overline{h}_p^{(t)} = \frac{\alpha_1 \mathcal{N}(v_p|0, \tau_1^2)}{\alpha_1 \mathcal{N}(v_p|0, \tau_1^2) + (1-\alpha_1)\mathcal{N}(v_p|0, \tau_2^2)} \tag{5.16}$$

$$\overline{g}_j^{(t)} = \frac{\alpha_2 \mathcal{N}(z_j|0, \rho_1^2)}{\alpha_2 \mathcal{N}(z_j|0, \rho_1^2) + (1-\alpha_2)\mathcal{N}(z_j|0, \rho_2^2)}. \tag{5.17}$$

### The M-Step

In the M-step, the parameter estimates are updated by maximizing $\mathcal{Q}(\theta, \hat{\theta}^{(t)})$ with respect to $\theta$. Examining the several terms in Eq. 5.13 reveals that there are two types of problems: with respect to $V$ and $Z$, we face a classic low-rank matrix factorization problem, in the form proposed in [15]; for the other parameters, closed-form updates can be obtained by equating the corresponding derivatives to zero.

**Prototypes and Factors** Considering only the terms in $\mathcal{Q}(\theta, \hat{\theta}^{(t)})$ that depend on $V$ and $Z$, we have the following low-rank factorization problem,

$$\underset{V,Z}{\arg\min} \Big[\frac{||D - VZ||_F^2}{2\sigma^2} + \frac{1}{2} v^T \overline{H}^{(t)} v + \frac{1}{2} z^T \overline{G}^{(t)} z\Big]. \tag{5.18}$$

which is a generalization of the recently proposed unified model proposed in [15]. The generalization in Eq. 5.18 consists in taking weighted Frobenius norms, instead of the plain Frobenius norm used in [15]; in fact, notice that $v^T S v$ (where $v = \text{vec}(V)$ and $S$ is some diagonal matrix) is simply the square of a weighted version of the Frobenius norm: $v^T S v = \sum_i S_{ii} v_i^2$.

Inspired by the optimization method in [15], we tackle problem in Eq. 5.18 via the *augmented Lagrangian method* (ALM) [96], also known as the *method of multipliers* (MM) [58, 106]. The first step is to re-write Eq. 5.18 as an equivalent constrained problem, via a procedure known as variable splitting (*i.e.*, introducing

a new variable $C$ to take the place of the low rank product $VZ$):

$$\begin{aligned} &\arg\min_{V,Z,C} \Big[\frac{||D-C||_F^2}{2\sigma^2} + \frac{1}{2}v^T\overline{H}^{(t)}v + \frac{1}{2}z^T\overline{G}^{(t)}z\Big] \\ &\text{s.t.} \quad C = VZ. \end{aligned} \tag{5.19}$$

For computational purposes, it is more convenient to write a fully vectorized version of this problem; to that end (and as for $v = \text{vec}(V)$ and $z = \text{vec}(Z)$), we define $c = \text{vec}(C)$ and $d = \text{vec}(D)$, leading to

$$\begin{aligned} &\arg\min_{v,z,c} \Big[\frac{||d-c||_2^2}{2\sigma^2} + \frac{1}{2}v^T\overline{H}^{(t)}v + \frac{1}{2}z^T\overline{G}^{(t)}z\Big] \\ &\text{s.t.} \quad c = (I \otimes V)z, \end{aligned} \tag{5.20}$$

where the constraint $c = (I \otimes V)z$ is equivalent to $C = VZ$ (as is clear from (5.1)). Notice that the constraint can also be written as $c = (Z^T \otimes I)v$ (as is also clear from (5.1)). For later use, we define the two following matrices:

$$A(z) = (Z^T \otimes I) \qquad \text{and} \qquad B(v) = (I \otimes V). \tag{5.21}$$

The augmented Lagrangian for problem Eq. 5.19 is obtained by adding a quadratic penalty to the Lagrange function of problem Eq. 5.20,

$$\begin{aligned} \mathcal{L}_\rho(v,z,c,y) \;=\; & \frac{||d-c||^2}{2\sigma^2} + \frac{1}{2}v^T\overline{H}v + \frac{1}{2}z^T\overline{G}z + \\ & \frac{\rho}{2}||B(v)z - c||^2 + y^T(c - B(v)z), \end{aligned} \tag{5.22}$$

where $y$ is the vector of Lagrange multipliers, $\rho \geq 0$ is a parameter, and we have written $\overline{H} = \overline{H}^{(t)}$ and $\overline{G} = \overline{G}^{(t)}$ to keep the notation lighter. Notice that the vector $B(v)z$ can also be equivalently written as $A(z)v$. The ALM proceeds by alternating between minimizing $\mathcal{L}_\rho(v,z,c,y)$ with respect to the variables $v, z, c$ and updating the Lagrange multipliers.

Unfortunately, $\mathcal{L}_\rho(v,z,c,y)$ cannot be minimized in closed-form simultaneously with respect to $v, z, c$, thus we follow the approach in [15] and solve it by a non-linear block Gauss-Seidel (NLBGS) method, *i.e.*, we cycle through minimizations with respect to $v, z$, and $c$, until some convergence criterion is satisfied, taking advantage of the fact that each of these minimizations can be written in closed form, simply by equating the corresponding gradients to zero. Letting the iteration counter of NLBGS be $s$ and denoting $A^{(s)} = A(z^{(s)})$ and $B^{(s)} = B(v^{(s)})$, the resulting update expressions are (for $s = 1, 2, ...$)

$$v^{(s+1)} = \Big(\overline{H} + \rho\big(A^{(s)}\big)^T A^{(s)}\Big)^{-1}\Big(\big(A^{(s)}\big)^T y + \rho\big(A^{(s)}\big)^T c^{(s)}\Big) \tag{5.23}$$

$$z^{(s+1)} = \Big(\overline{G} + \rho\big(B^{(s+1)}\big)^T B^{(s+1)}\Big)^{-1}\Big(\big(B^{(s+1)}\big)^T y + \rho\big(B^{(s+1)}\big)^T c^{(s)}\Big) \tag{5.24}$$

$$c^{(s+1)} = \frac{d - \sigma^2 y + \rho B^{(s+1)} z^{(s+1)}}{1 + \sigma^2 \rho}. \tag{5.25}$$

In summary, the updated $V^{(t+1)}$ and $Z^{(t+1)}$, which are the solutions of problem Eq. 5.18, are obtained by cycling through Eq. 5.23, Eq. 5.24, and Eq. 5.25, until some convergence criterion is satisfied.

**Other parameters** The update of other parameters $(\tau_1^2, \tau_2^2, \rho_1^2, \rho_2^2, \sigma, \alpha_1, \alpha_2)$ are obtained by setting the corresponding partial derivatives of $\mathcal{Q}(\theta, \hat{\theta}^{(t)})$ to zero, yielding the following expressions:

$$\tau_1^2 = \big(v^T \overline{H} v\big)/\|\overline{H}\|_F \tag{5.26}$$

$$\tau_2^2 = v^T(1 - \overline{H})v/\|1 - \overline{H}\|_F \tag{5.27}$$

$$\rho_1^2 = z^T \overline{G} z/\|\overline{G}\|_F \tag{5.28}$$

$$\rho_2^2 = z^T(1 - \overline{G})z/\|1 - \overline{G}\|_F \tag{5.29}$$

$$\alpha_1 = \left(\sum_{p=1}^{nk} \overline{h}_p\right)/(nk) \tag{5.30}$$

$$\alpha_2 = \left(\sum_{p=1}^{nk} \overline{g}_p\right)/(mk) \tag{5.31}$$

$$\sigma^2 = \|D - VZ\|_F^2/(nm), \tag{5.32}$$

where we have omitted the iteration counter superscript $(\cdot)^{(t)}$, to keep the notation lighter.

### The Complete Algorithm

The final complete algorithm obtained by putting together the E-step and M-step derived in the previous subsections is presented in Algorithm 2. Some comments and explanations about the algorithm are in order, and are presented in the next few paragraphs.

Initialization is carried out in lines 1 and 2, where TSVD$(k)$ stands for the $k$-*truncated singular value decomposition*, which corresponds to computing the SVD of $D$ and keeping only the left and right singular vectors corresponding to the $k$ largest singular values (this is known to correspond to the best rank $k$ approxima-

tion of $D$ in the Frobenious norm sense). The other parameters are initialized as follows: $\sigma^2$ is initialized according to Eq. 5.32, using the initial $V$ and $Z$; $\alpha_1$ and $\alpha_2$ are initialized to $1/2$; finally, the spike and slab variances $\tau_1^2$ and $\rho_1^2$ are initialized as the standard deviation of $V$ and $Z$ respectively, and $\tau_2^2$ and $\rho_2^2$ are set to one tenth of $\tau_1^2$ and $\rho_1^2$. Line 5 corresponds to the E-step of the EM algorithm, as explained in Subsection 5.2.3. Lines 6, 7, and 8 are the initialization of the ALM method described in Subsection 5.2.3. The inner loop of the NLBGS algorithm that implements the update step (with respect to $v$, $z$, and $c$) of ALM is implemented in lines 10–16; the update of the Lagrange multipliers $y$ is implemented in line 17. As in [15], the ALM parameter $\rho$ is increased at each interation, by multiplying it by $\mu = 1.05$ in line 18. Finally the remaining model parameter estimates are updated according to Eq. 5.26–Eq. 5.32, in line 20, completing the M-step.

It is important to stress that the SSBi model and the SSBiEM algorithm herein presented can be trivially generalized to the case where each bicluster has its own spike and slab parameters; instead of a common set $\tau_1^2, \tau_2^2, \rho_1^2, \rho_2^2, \alpha_1, \alpha_2$, each bicluster (*i.e.*, each of the $k$ rows of $V$ and columns of $Z$) will have its own set of parameters, resulting in a more complicated (but essentially equivalent) set of update equations. To keep the notation simpler, we abstained from presenting that more general version of the model and algorithm, but it was used in the experiments reported below.

Since the complete algorithm includes 3 nested loops (EM, ALM, NLBGS), it involves three stopping criteria (lines 4, 9, and 10). The EM stopping criterion is based on the relative change of the log-likelihood function falling below some threshold. The ALM iterations stop when the relative change in the Lagrange multiplier vector $y$ is less than a threshold. Finally, the inner NLBGS loop is stopped when the maximum of relative changes in the involved variables is below a threshold.

Of course, the EM algorithm obtained involves an approximate M-Step (hence providing a Generalized EM algorithm) where an iterative procedure minimizes a highly non-convex function, thus there are no formal guarantees of convergence and the results may depend on the initialization. However, in all the experiments discussed in the following section, SSBiEM converged to an effective solution.

### 5.2.4 Experimental Evaluation

In this Section, the SSBiEM algorithm[1] is experimentally evaluated on both synthetic and real datasets.

[1] SSBiEM is available as Matlab function (.m) at *https://github.com/emme-di/SSBiEM/*

**Algorithm 2** SSBiEM

**Require:** Data matrix $D$, number of biclusters $k$.

Initialize $V, Z$ using TSVD$(k)$
Initialize $\tau_1^2, \tau_2^2, \rho_1^2, \rho_2^2, \alpha_1, \alpha_2, \sigma^2$ (see text)
Initialize $v \leftarrow \text{vec}(V)$ and $z \leftarrow \text{vec}(Z)$
**while** EM not converged **do**

**E-Step**:

compute $\overline{H}$ and $\overline{G}$ using (5.14), (5.15), (5.16), (5.17)
$B \leftarrow I \otimes V$, where $V \leftarrow \text{vec}^{-1}(v)$
$A \leftarrow Z^T \otimes I$, where $Z \leftarrow \text{vec}^{-1}(z)$
$c \leftarrow Bz$

**M-Step**:

**while** ALM not converged **do**
  **while** NLBGS not converged **do**
    Update $v$ according to (5.23)
    $B \leftarrow I \otimes V$, where $V \leftarrow \text{vec}^{-1}(v)$
    Update $z$ according to (5.24)
    $A \leftarrow Z^T \otimes I$, where $Z \leftarrow \text{vec}^{-1}(z)$
    Update $c$ according to (5.25)
  **end while**
  $y \leftarrow y + \rho(c - Bz)$
  $\rho \leftarrow \min(\rho\mu, 10^{20})$
**end while**

update parameters according to (5.26)–(5.32)
**end while**
**return** $V, Z, \overline{H}, \overline{G}$

### Synthetic Benchmark

To obtain a fair comparison and a clear perspective on the performance of SSBi with respect to the FABIA approach, we carry out experiments on the synthetic benchmark datasets proposed by authors of FABIA in [59], and adopt their evaluation criteria. The dataset is composed by 100 matrices of dimension $1000 \times 100$, simulating real world gene expression datasets. Each matrix contains 10 implanted biclusters, generated with a multiplicative structure, where the positions and dimensions were randomly chosen, thus we run SSBiEM with $k = 10$. For a given set of true biclusters $T$ and a set of retrieved biclusters $B$, the accuracy of $B$ with the following three steps [59]:

1. compute the *Jaccard similarity coefficient* $J(t, b)$ of all the pairs $(t, b) \in T \times B$; notice that $J(t, b) \in [0,\ 1]$, with $J(t, b) = 0$, if $t \cap b = \emptyset$, and $J(t, b) = 1$, if $t = b$;

2. via the *Kuhn–Munkres algorithm* (a.k.a. the *Hungarian algorithm* [92]), assign each bicluster in $T$ to one in $B$, by maximizing the sum of the Jaccard similarities of the assigned pairs;

3. divide the resulting assignment value by $\max\{|T|, |B|\}$; the final result is a quantity in $[0,\ 1]$, which is equal to 1 if and only if $B = T$.

The results are shown in Table 5.1, where we compare SSBiEM with two versions of FABIA and with other state-of-the-art methods (the results of FABIA and of the other methods are those reported in [59]). SSBiEM outperforms all the other methods on this dataset, proving its effectiveness.

### Real Dataset

The SSBiEM algorithm was also compared with other state-of-the-art methods on real microarray gene expression data: therefore we chose the Breast Tumor dataset where FABIA was among the worst according to results reported in [91]. Since FABIA is the closest formulation to SSBi, we decided to test SSBi on a dataset where FABIA does not perform well, to understand if the novelties introduced in SSBi with respect to FABIA provide a significant improvement on the results.

As discussed in Chap. 4, the performances on gene expression matrices is assessed by analyzing *Gene Onthology* (GO) terms [83, 91]. The analysis of these terms is performed automatically using the FuncAssociate web-server.

As also commonly proposed in the literature [10, 114], we applied a variance-based gene selection procedure to reduce the dataset dimensionality, keeping 2500 genes. For a fair comparison, in [91] the authors selected the same number of biclusters for each method (40). For our algorithm, since the background of the data matrix is not zero, we run the SSBiEM algorithm with the number of biclusters set

| Method | Score | References |
|---|---|---|
| SSBi | **0.606** | |
| FABIAS | 0.564 | [59] |
| FABIA | 0.478 | [59] |
| MFSC | 0.057 | [60] |
| plaid_ss | 0.045 | [78] |
| plaid_ms | 0.072 | [78] |
| plaid_ms_5 | 0.083 | [78] |
| ISA_1 | 0.333 | [61] |
| ISA_2 | 0.299 | [61] |
| ISA_3 | 0.188 | [61] |
| OPSM | 0.012 | [7] |
| SAMBA | 0.006 | [121] |
| xMOTIF | 0.002 | [93] |
| Bimax | 0.004 | [107] |

Table 5.1: Synthetic Dataset Results. The table compares state-of-the-art approaches on the synthetic benchmark dataset proposed in [59]. Other approaches results have been taken from [59].

to 41. This provides a bicluster that accounts for the background noise of the data matrix, containing all the rows and columns thereof. In the end, that background bicluster is discarded to obtain the pool of 40 biclusters needed to assess the method.

The results are shown in Table 5.2 (the scores of the other methods are those reported in [91]). As can be seen by the table, SSBiEM reaches the results obtained by the OOB method (described in the previous Chapter). This confirms both the quality of the OOB approach, which we run on sub-matrices and then re-aggregate the results; and the efficacy of SSBiEM in isolating significant biclusters. Compared to FABIA, the proposed SSBiEM outperforms the results reported in [91]. Arguably, this is probably due the automatic estimation of the biclusters memberships. In fact, in FABIA to obtain high quality solutions it is crucial to accurately set the thresholds (which decide the biclusters memberships); and this, starting from the same factorization, can obviously lead to significantly different results.

## 5.3 Prior Knowledge Spike and Slab Biclustering

In this section we present an extension to the previously described SSBiEM algorithm, allowing to include some prior knowledge in the model. In this context the prior knowledge could derive from prior notions about relationships between rows or between columns. In the clustering scenario there are situations – known

| Method | Score (%) | References |
|---|---|---|
| FABIA | 55 | [59] |
| ISA | 63 | [61] |
| Hierarc. | 70 | [117] |
| SAMBA | 73 | [121] |
| FLOC | 85 | [145] |
| OOB | 87.5 | |
| SSBiEM | **87.5** | |

Table 5.2: Breast Tumor Dataset Results. The table shows the results on the real Breast Tumor gene expression dataset taken from [91]. The GO results for the other approaches have been taken from [91].

as *Semi Supervised Clustering* – where labels are completely absent, but there are (usually pair-wise) relations that one wishes to enforce or encourage [37]. Such *semi-supervision* information may be *hard* or *soft*; this later is clearly the most natural formulation for cases where one wishes to *encourage*, *not enforce*, certain relations. An obvious example is image segmentation, seen as clustering under a spatial prior, where neighboring sites should be encouraged, but not constrained, to belong to the same cluster/segment.

A similar reasoning can also be performed for biclustering. In microarray analysis, for instance, prior biological information could be exploited to improve the results obtained by classical biclustering algorithm (e.g., genes belonging to the same biological process should be encouraged to be together). Thus, with this extension, we present a way to encourage certain pair of rows and/or columns to belong to the same bicluster.

One example where prior information has been exploited in a biclustering scenario is provided by [103]. In this case authors assume that some a priori knowledge is available regarding genes in a gene expression matrix. Such information is thus exploited by introducing hard constraints between interested genes, thus forcing certain genes to be in the same bicluster.

Differently from what proposed in [103], we aim at introducing *soft pairwise preferences* in the model involving rows and columns separately. In more details, in this extension we modify the probability definition of both prototypes and factors. Considering the set of $n$ rows of a given data matrix $D \in \mathbb{R}^{n \times m}$, our goal is to modify the probabilistic assumption made for $V$ by casting a novel matrix $S^v \in \mathbb{R}^{n \times n}$ indicating the preferences for each couple of rows to belong to the same bicluster. Similarly, the same procedure is adopted for the columns of the given data matrix, defining $S^z \in \mathbb{R}^{m \times m}$ which is cast in the probability definition of $Z$.

More formally, given the model described in the previous Section, one way to achieve this is to multiply the corresponding probability function by another one

that encourages this pair-wise grouping. Inspired by [37], we hence modify the definition of $V$ matrix in Sec. 5.2 by introducing a *pairwise regularized* Spike and Slab prior obtaining:

$$\begin{aligned}
\mathcal{P}(V|H,\tau_1,\tau_2) &= \frac{1}{\Xi}\left[\prod_{j=1}^{k}\exp\left(-\frac{\beta}{4}\sum_{i=1}^{n}\sum_{l=1}^{n}S_{il}^{v}(v_{ij}-v_{lj})^2\right)\right] \\
&\quad \left(\prod_{i=1}^{n}\prod_{j=1}^{k}\mathcal{N}(v_{ij}|0,\tau_1^2)^{h_{ij}}\mathcal{N}(v_{ij}|0,\tau_2^2)^{1-h_{ij}}\right) \\
&= \frac{1}{\Xi}\left[\prod_{j=1}^{k}\exp\left(-\frac{1}{2}v_j^T\Delta_v v_j\right)\right] \\
&\quad \left(\prod_{i=1}^{n}\prod_{j=1}^{k}\mathcal{N}(v_{ij}|0,\tau_1^2)^{h_{ij}}\mathcal{N}(v_{ij}|0,\tau_2^2)^{1-h_{ij}}\right) \qquad (5.33)
\end{aligned}$$

where vectors $v_j$ denotes the $j$-th column of matrix $V$ and $S_{il}^{v} = S_{li}^{v} \geq 0$ is the intensity with which we want to encourage $v_{ij}$ and $v_{lj}$ to be similar (thus encouraging them to be in the same bicluster), $\Delta_v$ is the $n \times n$ Laplacian matrix of a graph with edge weights given by $S_{il}^{v}$,

$$\Delta_v = \beta\left(\text{diag}\left(\sum_{i=1}^{n}S_{1i}^{v},\ldots,\sum_{i=1}^{n}S_{ni}^{v}\right) - S^{v}\right), \qquad (5.34)$$

and $S^v$ is the matrix with elements $S_{il}^{v}$. The parameter $\beta$ is adopted to tune the importance of the prior knowledge in the model. Of course, multiplying two probability density functions does not yield a valid probability density function, unless one carries out normalization $\Xi$, but this normalization is irrelevant for our purpose. Similarly for $Z$ we obtain:

$$\begin{aligned}
\mathcal{P}(Z|G,\rho_1,\rho_2) &= \frac{1}{\Xi}\left[\prod_{j=1}^{k}\exp\left(-\frac{\beta}{4}\sum_{i=1}^{m}\sum_{l=1}^{m}S_{il}^{z}(z_{ij}-z_{lj})^2\right)\right] \\
&\quad \left(\prod_{i=1}^{k}\prod_{j=1}^{m}\mathcal{N}(z_{ij}|0,\rho_1^2)^{g_{ij}}\mathcal{N}(z_{ij}|0,\rho_2^2)^{1-g_{ij}}\right) \\
&= \frac{1}{\Xi}\left[\prod_{j=1}^{k}\exp\left(-\frac{1}{2}z_j^T\Delta_z z_j\right)\right] \\
&\quad \left(\prod_{i=1}^{k}\prod_{j=1}^{m}\mathcal{N}(z_{ij}|0,\rho_1^2)^{g_{ij}}\mathcal{N}(z_{ij}|0,\rho_2^2)^{1-g_{ij}}\right) \qquad (5.35)
\end{aligned}$$

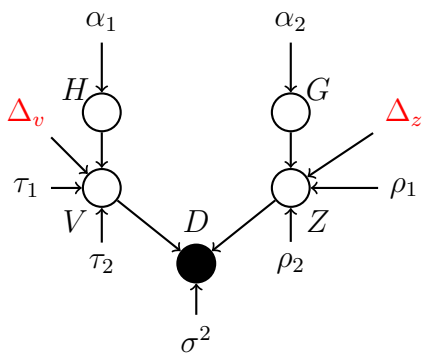



Figure 5-3: The Bayesian Network of the Prior Knowledge Spike and Slab Biclustering.

where vectors $z_j$ denotes the $j$-th column of matrix $Z$ and $S^z_{il} = S^z_{li} \geq 0$ is the intensity with which we want to encourage $z_{ij}$ and $z_{lj}$ to be similar (thus encouraging them to be in the same bicluster), $\Delta_z$ is the $m \times m$ Laplacian matrix of a graph with edge weights given by $S^v_{il}$,

$$\Delta_z = \beta \left( \text{diag} \left( \sum_{i=1}^{m} S^z_{1i}, \ldots, \sum_{i=1}^{m} S^z_{mi} \right) - S^z \right). \tag{5.36}$$

and $S^z$ is the matrix with elements $S^z_{il}$.

Graphically, the previously defined Bayesian networks presents the introduction of two other parameters $\Delta_v$ and $\Delta_z$ as sketched in Fig. 5-3. The above mentioned modifications slightly affect the parameter estimation process. Specifically, concerning prototype and factors, the low rank factorization problem described by Eq. 5.18 is enriched by a sum between two matrices becoming:

$$\underset{V,Z}{\arg\min} \left[ \frac{||D - VZ||_F^2}{2\sigma^2} + \frac{1}{2} v^T (\overline{H}^{(t)} + \overline{\Delta_v}) v + \frac{1}{2} z^T (\overline{G}^{(t)} + \overline{\Delta_z}) z \right]. \tag{5.37}$$

where $\overline{\Delta_v} = \text{block-diag}(\Delta_v, \ldots, \Delta_v)$ is an $(nk) \times (nk)$ block diagonal matrix with $k$ copies of the $\Delta_v$ (the same holds for $\Delta_z$).

Compared to the problem presented in Eq. 5.18, if we call $\overline{H} = \overline{H}^{(t)} + \overline{\Delta_v}$ and $\overline{G} = \overline{G}^{(t)} + \overline{\Delta_z}$ we obtain exactly the same problem:

$$\underset{V,Z}{\arg\min} \left[ \frac{||D - VZ||_F^2}{2\sigma^2} + \frac{1}{2} v^T \overline{H} v + \frac{1}{2} z^T \overline{G} z \right]. \tag{5.38}$$

Thus, the passages to derive the iterative rules for the M-Step are exactly the same. Hence, we just need to replace $\overline{H}$ in Eq. 5.23 with $\overline{H}^{(t)} + \overline{\Delta_v}$; and $\overline{G}$ in Eq. 5.24

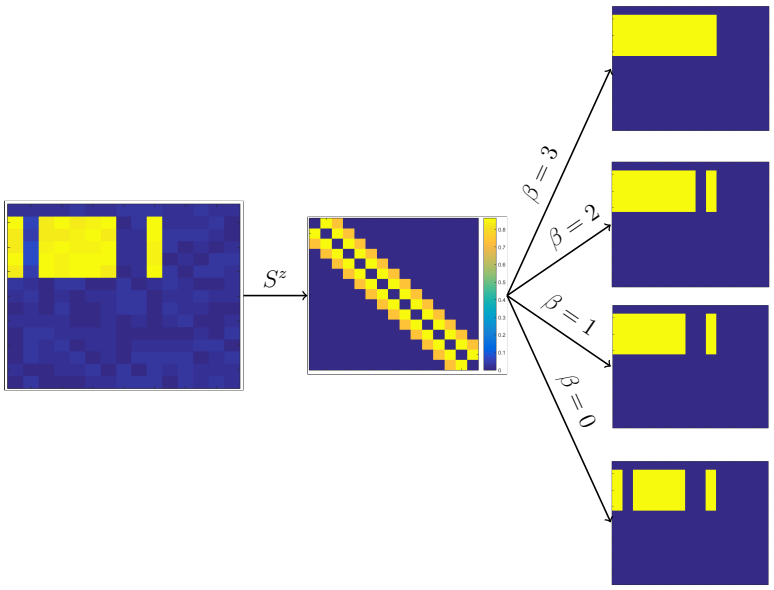



Figure 5-4: Example of the PKSSBi in action. By introducing the similarity $S^z$ on the columns, which contains high preferences only for nearby columns, and by changing its importance through $\beta$ we can obtain whatever desired solutions.

with $\overline{G}^{(t)} + \overline{\Delta_z}$, to obtain:

$$v^{(s+1)} = \left((\overline{H}^{(t)} + \overline{\Delta_v}) + \rho\left(A^{(s)}\right)^T A^{(s)}\right)^{-1}\left(\left(A^{(s)}\right)^T y + \rho\left(A^{(s)}\right)^T c^{(s)}\right) \tag{5.39}$$

$$z^{(s+1)} = \left((\overline{G}^{(t)} + \overline{\Delta_z}) + \rho\left(B^{(s+1)}\right)^T B^{(s+1)}\right)^{-1}\left(\left(B^{(s+1)}\right)^T y + \rho\left(B^{(s+1)}\right)^T c^{(s)}\right). \tag{5.40}$$

Figure 5-4 presents an intuitive example about the method dynamics: increasing $\beta$ raises the importance of the preferences stored in matrices $S^v$ and $S^z$. An application for this novel extension is presented in Sec. 6.2.

## 5.4 Conclusion and Future Works

In this Chapter we presented Bayesian Network models tackling the biclustering problem from a Probabilistic Sparse Low-Rank Matrix Factorization (PSLRMF) perspective. As previously stressed, sparsity plays a central role in these approaches; due to the enormous dimension of dataset and the (possibly) reduced dimension of biclusters. We thus proposed to induce sparsity in the devised Bayesian Network by exploiting the so called Spike and Slab prior. The presented formulation allowed us to retrieve analytically closed form rules for the EM algorithm, which we adopted to estimate the model parameters. As expected, this technique is not affected by

scalability problems of Factor Graph approaches. In more details, with respect to FABIA, we showed that the introduction of the Spike and Slab prior allowed us to automatically estimate the bicluster memberships. This information is crucial to extract high quality solutions from the obtained matrix factorisation. We tested the approach on both synthetic and real datasets demonstrating its effectiveness when compared to FABIA or other state-of-the-art approaches. Furthermore we propose an extension where prior knowledge can be encoded in the model to encourage particular rows/columns to belong to the same bicluster. In the short-term scenario, future works in this context may concern the exploitation of model selection techniques to automatically retrieve the natural number of biclusters. Particularly we can exploit the probabilistic nature of the model; or we can adopt existing indexes (such as BIC [109] or MDL [111]). Other interesting research directions may also concern the modification of the PKSSBi algorithm. In fact, there are other scenarios where prior information *prevents* certain rows/columns to belong to the same biclusters. The first idea would be to set a negative preference for rows/columns in the $S^v/S^z$ matrices. However, due to the formulation of $\Delta_v$ (Eq. 5.34) and $\Delta_z$ (Eq. 5.36) this is not doable.

# Chapter 6

# Applications

In this Chapter we show that biclustering can be exploited in a wide range of other scenarios. We thus identified two suitable Computer Vision contexts where biclustering have never been applied before: *Multiple Structure Recognition* problem and *Stable Region Correspondences*problem.

Briefly, Multiple Structure Recognition aims at the aggregation of 3D points in geometric structures which can help the "scene understanding". In this context the goal of biclustering concerns the identification of points in a given scene that rely on the same, predefined, *structures* (*i.e.*, points belonging to the same plane, as represented in Fig. 6-1). On the other hand, given a couple of shapes, the problem of Stable Region Correspondences aims at the individuation of regions in the two shape that behave coherently (*i.e.*, face portions in two different human shapes, as depicted in Fig. 6-2). Next sections provide the details of how these problems represent clear instances of the biclustering problem, and how we can exploit the devised techniques to solve them.

## 6.1 Multiple Structure Recovery via Biclustering

The extraction of multiple models from noisy or outlier-contaminated data – a.k.a. Multiple Structure Recovery (MSR) – is an important and challenging problem that emerges in many Computer Vision applications [38, 55, 99, 127]. In more details, MSR aims at retrieving parametric models from unstructured data in order to organize and aggregate visual content in significant higher-level geometric structures. This task is commonly found in many Computer Vision applications, a typical example being 3D reconstruction, where MSR is employed either to estimate multiple rigid moving objects (to initialize multibody Structure from Motion [38,99]), or to produce intermediate geometric interpretations of reconstructed 3D point cloud [55,127] (as depicted in Fig. 6-1).

Other instances include face clustering, body-pose estimation and video motion segmentation. In all these scenarios the information of interest can be extracted

Figure 6-1: Example of Multiple Structures Recovery problem. Different colors refer to different structures, red points represent outliers not belonging to any structure.

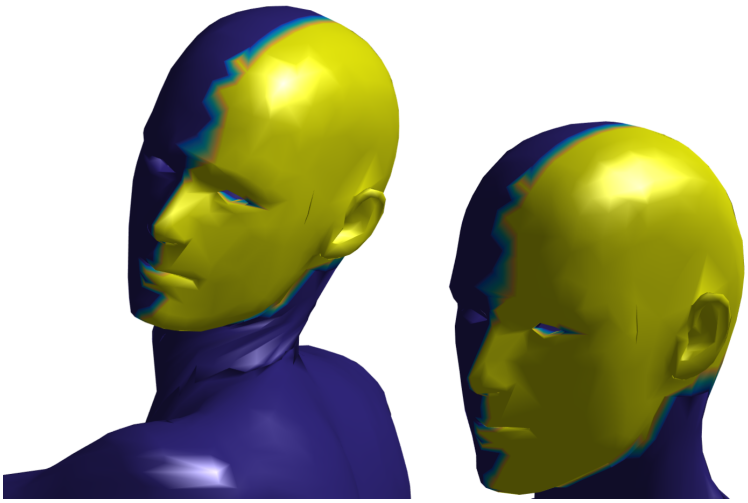


Figure 6-2: Example of Stable Region Correspondences problem. The yellow points behave similarly in the two different shapes.

from the observed data and aggregated in suitable structures by estimating some underlying geometric models, e.g. planar patches, homographies, fundamental matrices or linear subspaces. Since measurement errors in visual data are almost unavoidable in common vision applications, MSR has to be robust to outliers, and in addition, in order to handle with the presence of multiple structures in the the data it has also to cope with pseudo-outliers.

More formally, to set a general context, let $\mu$ be a model e.g. lines, subspace, homography, fundamental matrices or other geometric primitives and $X = \{x_1, \ldots, x_n\}$ be a finite set of $n$ points, possibly corrupted by noise and outlier. The problem of MSR consists in extracting $k$ instances of $\mu$ – termed structures – from the data, defining, at the same time, subsets $C_i \subset X, i = 1, ...,$ such that all points described by the i-th structures are aggregated in $C_i$ . Often the models considered are parametric, i.e. the structures can be represented as vectors in a proper parameter space.

### 6.1.1 Clustering Approaches for MSR

In this section we present the general taxonomy concerning clustering approaches for MSR, mainly focusing on the T-linkage and RPA algorithms.

The extensive landscape of approaches aimed at MSR can be broadly categorized along two mutually orthogonal strategies, namely *consensus analysis* and *preference analysis*. The consensus set of a model is defined as the set of data points that are close to the model within a certain threshold, whereas, in a dual fashion, the preference of a model is described by the set of structures the points belongs to.

Consensus analysis can be traced back to the time-honored RANSAC paradigm and its variants (e.g., [128]) and gave rise also to algorithms tailored for the case of multiple structures estimation (e.g., [152]). In short, consensus-oriented methods generate a pool of putative model hypotheses by random sampling, and retain the structures that explain more data by maximizing their consensus sets. This idea can also be found in the popular Hough transform and its generalization [143], where multiple models are revealed as peaks in a properly quantized hypothesis space. It is also the foundation of many optimization algorithms for geometric fitting, e.g., [62].

On the contrary, preference analysis reverses the role of data and models: rather than considering models and examining which points match them, we analyse how individual points are explained by models [21, 125, 149]. This information, is exploited to shift the MSR problem from the ambient space where data lives to a *conceptual* [101] one where it is addressed via cluster analysis techniques. Preference analysis methods can be regarded as processing the *Preference Matrix*, where each entry $(i, j)$ represents the vote/score granted by the $i$-th point to the $j$-th tentative structures (as described by Fig. 6-3). The rationale beyond this representation is that the agreement between the preferences of two points in this conceptual space reveals the multiple structures hidden in the data: points sharing the same prefer-

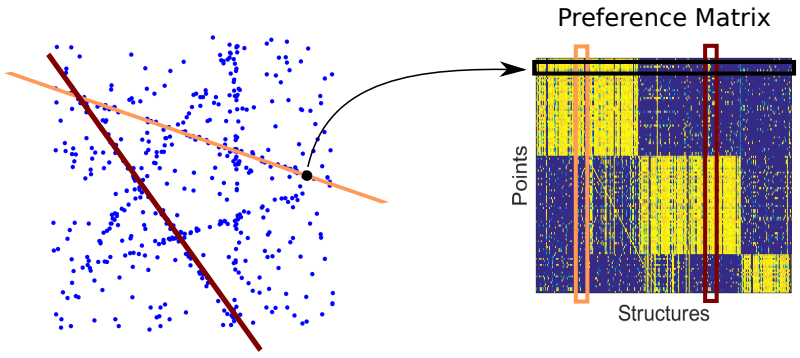



Figure 6-3: Example of preference matrix: the big dot row presents a high score in the column referring to the light structure, whereas it has a low one in correspondence of the darker structure column.

ences are likely to belong to the same structures.

Commonly, the problem of MSR has been successfully tackled by leveraging on clustering techniques [22, 65, 85, 86, 104, 126]. As in many other scenarios, the data matrix to analyse reports the points to cluster on one dimension and the features/descriptors on the other dimension [12]. In this context the feature vector used to represent data is derived from the preferences expressed by the data points for a pool of tentative structures (hypothesis) obtained by random sampling. Hence cluster analysis is performed via either agglomerative or partitional methods where distances measure the (dis)agreement between preferences.

J-Linkage [126], then improved to T-Linkage [85], and RPA [86] can be ascribed to these clustering-based methods as they share the same *first-represent-then-clusterize* approach. More in details, RPA and T-Linkage belong to the popular class of *spectral* clustering approaches. Such techniques are among the most adopted since they result in a standard and simple-to-solve linear algebra problem, avoiding local minima and initialization issues. Specifically, both T-linkage and RPA represent data points in a $m$-dimensional unitary cube as vectors whose components collect the preferences granted to a set of $m$ hypotheses structures instantiated by drawing at random $m$ minimal sample sets – the minimum-sized set of data points necessary to estimate a structure. Preferences are expressed with a soft vote in $[0, 1]$, according to the continuum of points distances from the interested structures, in two different fashions. As regards T-linkage, a voting function characterized by an hard cutoff is employed. RPA, instead, exploits the Cauchy weighting function (of the type employed in M-estimators) that has an infinite rejection point mitigating the sensitivity of the inlier threshold. T-Linkage captures this notion through the Tanimoto distance, which in turn is used to segment the data via a tailored version of average linkage that succeeds in detecting automatically the number of models. If rogue points contaminate the data, outlier structures need to be pruned via ad hoc-post processing techniques. RPA, on the contrary, requires the number of de-

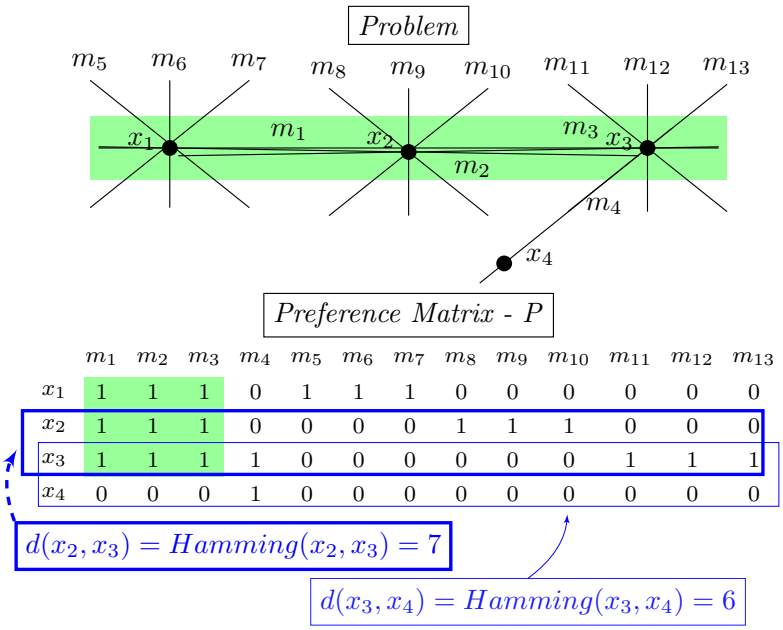
sired structures as input but inherently caters for gross contamination. At first a kernelized version of the Tanimoto distances is feed to Robust Principal Analysis to remove outlying preferences. Then Symmetric Non Negative Factorization [76] is performed on the low rank part of the kernel to segment the data. Hence, the attained partition is refined in a MSAC framework. More precisely, the consensus of the sampled hypotheses are scrutinized and the structures that, within each segment, support more points are retained as solutions.

While T-linkage can be considered as a pure preference method, RPA attempts to combine also the consensus-side of the MSR problem. However it does not fully reap the benefit of working with both the dimensions of the problem, as biclustering does, for preference and consensus are considered only sequentially.

### 6.1.2 Biclustering approaches for MSR

Although it has been shown that clustering provides good solution to the MSR problem, there are situations where the performances of clustering can be highly compromised by data matrix structure (e.g. noisy data matrices; or rows behaving similarly only in a small portion of the data matrix). For example let us consider the situation in Fig. 6-4 which describes a simple MSR problem: to group similar points on the basis of their behavior with respect to the proposed models we should perform clustering on the Preference Matrix $P$ which describes the relationship between the points $\{x_1, x_2, x_3, x_4\}$ and the models $\{m_1, \cdots, m_{13}\}$; specifically in this case $P$ is a binary matrix having ones in $(i, j)$ if, and only if, the *i-th* point is "well represented" by the *j-th* model. Assume we perform clustering adopting the Hamming distance (i.e. number of different bits): since the distance between the $x_3$ and the $x_4$ is smaller than the distance between $x_3$ and $x_2$, clustering would assign the third and the fourth point to the same group. However looking at the problem diagram it is clear that points $x_1, x_2$ and $x_3$ should belong to the same cluster. This information can be retrieved performing a simultaneous clustering of both rows and columns of the Preference Matrix, isolating a subset of models ($m_1, m_2$ and $m_3$) where the points $x_1, x_2, x_3$ share a similar behavior (shaded area in Fig. 6-4). This is exactly what biclustering techniques do.

More in general, the quality of the preference matrix constructed for the MSR problem is strictly connected to the quality of the considered structures. In cases where poor hyphotesis are considered clustering algorithms, which compare information on all models, may get confused due to the amount of noise introduced in the preference matrix. Furthermore, typical clustering approach (such as Kmeans, Affinity Propagation and Hierarchical clustering techniques) do not deal with overlapping clusters. And particularly in MSR scenario we have that points can be characterized by distinct subsets of models, see Fig. 6-5 for an example. Finally, typical MSR contexts are full of outliers and clustering techniques usually provide a partition of the data (every point under analysis is assigned to a cluster). This means




| | $m_1$ | $m_2$ | $m_3$ | $m_4$ | $m_5$ | $m_6$ | $m_7$ | $m_8$ | $m_9$ | $m_{10}$ | $m_{11}$ | $m_{12}$ | $m_{13}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $x_1$ | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| $x_2$ | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| $x_3$ | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| $x_4$ | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Figure 6-4: Shortfalls of clustering on MRS.

that to prune the outlier from the results we must devise an *ad hoc* technique.

Biclustering techniques overcome all these drawbacks since:

- bicluster algorithms focus on local correspondences (no influence from bad hyphotesis);
- biclustering algorithms manage rows and columns overlap (they can manage intersections);
- biclustering algorithms do not provide a partition of the data (the outliers are removed automatically).

To the best of our knowledge there exists only a preliminar work applying biclustering techniques to MSR [124]. In [124] authors show that the application of biclustering techniques to MSR is promising and provides superior solution when compared with clustering. While this provides a significant contribution to the state of the art, there is large room for improvements since the method adopted by the authors has some limitations (i.e. it works with sparse binary matrices and it needs some pre-processing/post-processing operations to retrieve the final solutions). The natural following step in this direction is then the use of more general biclustering approaches to MSR, such as probabilistic biclustering methods [59, 110, 115, 121]. We hence investigate the usage of both FABIA and SSBiEM for MSR.

Summarizing, we face MSR through the preference analysis paradigm. Given a set of points $\mathbf{x}$ and a set of models $\mathbf{h}$, this paradigm involves the computation of

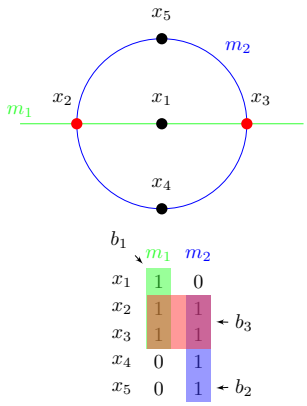



Figure 6-5: Example of overlapping biclusters on Multiple Structure Recovery. In this case the possible biclusters are the: $b_1$) the points lying on $m_1$, $b_2$) the points lying on $m_2$ and $b_3$) the points lying on the intersections.

a Preference matrix $P$ where an entry $p_{ij}$ provides a score indicating how "well" the point $x_i$ is represented by the hypothesis $h_j$ (the higher, the better). We thus propose to perform biclustering techniques on such matrix.

### Experimental Evaluation

This section provides the performances comparison between some clustering methods recently applied to MSR [85, 86, 104] and the biclustering methods described in the next Section: namely SSBiEM and FABIA. The comparison with [124] was not possible since the code is not available. The workflow of the overall procedure can be sketched as follows: starting from an image i) we generate the hypothesis and compute the Preference Matrix following the guidelines in [86]; ii) then the probabilistic biclustering technique have been applied. To assess the quality of the approaches we used the widely adopted *Adelaide* real benchmark dataset[1].

**Adelaide Dataset** We explored the performances of probabilistic biclustering on two type of experiments, namely motion and plane estimation. In motion segmentation experiments, we were provided with two different images of the same scene composed by several objects moving independently; the aim was to recover fundamental matrices to subsets of point matches that undergo the same motion. With respect to the plane segmentation scenario, given two uncalibrated views of a scene, the goal was to retrieve the multi-planar structures by fitting homographies to point correspondences. The AdelaideRMF dataset is composed of 38 image pairs (19 for motion segmentation and 19 for plane segmentation) with matching points contaminated by gross outliers. The ground-truth segmentations are also available. In

[1]The dataset can be downloaded from *https://cs.adelaide.edu.au/ hwong/doku.php?id=data*

order to assess the quality of the results, we adopted the misclassification errors, that counts the number of wrong point assignment according to the map between ground-truth labels and estimated ones that minimize the overall number of misclassified points (as in [118]). To better clarify, this is the opposite of what we called *purity* in previous sections. For fair comparison, the Preference Matrix fed to FABIA and SSBiEM was generated relying on the guided sampling scheme presented in [86].

FABIA parameters which regulate the factors/prototypes sparsity and the threshold to retrieve biclusters memberships have been varied in the range suggested by the authors in [59]. Specifically in Tab. 6.1 we report three different results for FABIA. "FABIA best" columns show the results where we consider, for each different matrix, the best performance with respect to the misclassification error. "FABIA best set" columns, which are slightly worse than the previous, are obtained by selecting the best set of parameters values minimizing the misclassification error (one for the motion segmentation and one for the plane estimation). The second-last columns ("FABIA automatic") show the misclassification performances where biclusters have been selected by adopting the quality measures provided by the FABIA algorithm. In fact, FABIA provides a score for each bicluster retrieved indicating its amount of information. To obtain the *FABIA automatic* column we kept, for each matrix, the set of biclusters with the highest average of such scores. Finally the last columns of the table represent the results obtained with the SSBiEM approach.

The performances of other methods are taken from [86]. Results show that biclustering techniques can provide higher quality solutions on the motion segmentation dataset, and on average it performs better on the planar segmentation. Furthermore, focusing on the motion segmentation dataset, there are only three situations where FABIA works worse than clustering approaches. A possible explanation on why FABIA struggles could be because general biclustering approaches are tested in scenarios where the number of biclusters is much higher than in MSR (i.e. $\sim$100 in Gene Expression analysis versus 3-7 in this dataset). To overcome this behavior we run FABIA increasing the number of biclusters to retrieve and aggregating the results on the basis of column overlap as done for OOB, this leads to an improvement of the solution quality; results are reported in Table 6.2.

Comparing SSBiEM with FABIA, we want to remark that SSBiEM learns all parameters (including memberships) directly from the data matrix without the need to set any. Table 6.1 shows that our approach outperforms "FABIA automatic" and favourably compares with "FABIA best set" columns, which we think represents a fair comparison (since we choose one parameter set for each datasets). However, although "FABIA best" retrieves better results, in this case we would set FABIA parameters manually for each different matrix; whereas SSBiEM computes them automatically.

| | k | %out | T-lnkg | RCMSA | RPA | FABIA best | FABIA best set | FABIA automatic | SSBiEM |
|---|---|---|---|---|---|---|---|---|---|
| biscuitbookbox | 3 | 37.21 | 3.10 | 16.92 | 3.88 | 3.86 | 4.17 | 62.55 | 8.65 |
| breadcartoychips | 4 | 35.20 | 14.29 | 25.69 | 7.50 | 4.2 | 7.76 | 65.40 | 17.89 |
| breadcubechips | 3 | 35.22 | 3.48 | 8.12 | 5.07 | 0.87 | 0.87 | 64.78 | 7.74 |
| breadtoycar | 3 | 34.15 | 9.15 | 18.29 | 7.52 | 0.60 | 0.72 | 66.27 | 5.90 |
| carchipscube | 3 | 36.59 | 4.27 | 18.90 | 6.50 | 1.52 | 1.70 | 63.64 | 8.36 |
| cubebreadtoychips | 4 | 28.03 | 9.24 | 13.27 | 4.99 | 1.07 | 9.79 | 73.09 | 11.07 |
| dinobooks | 3 | 44.54 | 20.94 | 23.50 | 15.14 | 9.72 | 10.44 | 56.94 | 20.83 |
| toycubecar | 3 | 36.36 | 15.66 | 13.81 | 9.43 | 9.50 | 25.70 | 64.00 | 10.80 |
| biscuit | 1 | 57.68 | 16.93 | 14.00 | 1.15 | 0 | 19.27 | 44.24 | 2.00 |
| biscuitbook | 2 | 47.51 | 3.23 | 8.41 | 3.23 | 1.32 | 1.58 | 52.49 | 3.93 |
| boardgame | 1 | 42.48 | 21.43 | 19.80 | 11.65 | 8.96 | 9.10 | 59.50 | 19.64 |
| book | 1 | 44.32 | 3.24 | 4.32 | 2.88 | 0 | 29.20 | 56.15 | 1.50 |
| breadcube | 2 | 32.19 | 19.31 | 9.87 | 4.58 | 19.42 | 20.66 | 68.18 | 5.12 |
| breadtoy | 2 | 37.41 | 5.40 | 3.96 | 2.76 | 19.62 | 19.65 | 63.19 | 0.97 |
| cube | 1 | 69.49 | 7.80 | 8.14 | 3.28 | 1.66 | 7.22 | 32.12 | 5.30 |
| cubetoy | 2 | 41.42 | 3.77 | 5.86 | 4.04 | 2.21 | 7.87 | 60.24 | 3.13 |
| game | 1 | 73.48 | 1.30 | 5.07 | 3.62 | 0 | 0.34 | 27.04 | 5.75 |
| gamebiscuit | 2 | 51.54 | 9.26 | 9.37 | 2.57 | 2.44 | 2.56 | 49.09 | 5.98 |
| cubechips | 2 | 51.62 | 6.14 | 7.70 | 4.57 | 0.53 | 0.85 | 49.65 | 4.30 |
| mean | | | 9.36 | 12.37 | 5.49 | 4.61 | 9.45 | 49.23 | 7.84 |
| median | | | 7.80 | 9.87 | 4.57 | 1.66 | 7.76 | 60.24 | 5.90 |

| | k | %out | T-lnkg | RCMSA | RPA | FABIA best | FABIA best set | FABIA automatic | SSBiEM |
|---|---|---|---|---|---|---|---|---|---|
| unionhouse | 5 | 18.78 | 48.99 | 2.64 | 10.87 | 21.54 | 38.01 | 23.49 | 23.49 |
| bonython | 1 | 75.13 | 11.92 | 17.79 | 15.89 | 6.82 | 8.69 | 26.26 | 17.47 |
| physics | 1 | 46.60 | 29.13 | 48.87 | 0.00 | 0.00 | 32.26 | 54.72 | 10.38 |
| elderhalla | 2 | 60.75 | 10.75 | 29.28 | 0.93 | 3.04 | 4.77 | 39.25 | 30.37 |
| ladysymon | 2 | 33.48 | 24.67 | 39.50 | 24.67 | 11.81 | 41.43 | 67.51 | 22.36 |
| library | 2 | 56.13 | 24.53 | 40.72 | 31.29 | 20.47 | 27.81 | 44.65 | 44.65 |
| nese | 2 | 30.29 | 7.05 | 46.34 | 0.83 | 4.92 | 14.80 | 66.54 | 2.91 |
| sene | 2 | 44.49 | 7.63 | 20.20 | 0.42 | 2.20 | 4.96 | 52.80 | 5.20 |
| napiera | 2 | 64.73 | 28.08 | 31.16 | 9.25 | 21.85 | 35.36 | 37.09 | 44.04 |
| hartley | 2 | 62.22 | 21.90 | 37.78 | 17.78 | 23.59 | 40.81 | 38.44 | 42.06 |
| oldclassicswing | 2 | 32.23 | 20.66 | 21.30 | 25.25 | 7.92 | 24.22 | 67.55 | 13.25 |
| barrsmith | 2 | 69.79 | 49.79 | 20.14 | 36.31 | 29.88 | 54.69 | 31.12 | 64.81 |
| neem | 3 | 37.83 | 25.65 | 41.45 | 19.86 | 11.20 | 23.49 | 63.49 | 38.42 |
| elderhallb | 3 | 49.80 | 31.02 | 35.78 | 17.82 | 18.63 | 34.27 | 52.16 | 38.67 |
| napierb | 3 | 37.13 | 13.50 | 29.40 | 31.22 | 36.68 | 39.54 | 60.62 | 40.62 |
| johnsona | 4 | 21.25 | 34.28 | 36.73 | 10.76 | 17.96 | 19.89 | 79.09 | 25.42 |
| johnsonb | 7 | 12.02 | 24.04 | 16.46 | 26.76 | 24.50 | 43.57 | 87.98 | 48.94 |
| unihouse | 5 | 18.78 | 33.13 | 2.56 | 5.21 | 15.76 | 26.07 | 83.45 | 29.02 |
| bonhall | 6 | 6.43 | 21.84 | 19.69 | 41.67 | 24.02 | 53.03 | 93.82 | 53.09 |
| mean | | | 24.66 | 28.30 | 17.20 | 15.94 | 29.88 | 50.93 | 31.33 |
| median | | | 23.38 | 29.40 | 17.53 | 17.96 | 32.26 | 54.72 | 30.37 |

Table 6.1: Misclassification error (ME %) for motion segmentation (left) and planar segmentation (right). $k$ is the number of models and % out is the percentage of outliers.

| | | |
|---|---|---|
| biscuitbookbox | k = 3 | 3.86 |
| | k = 4 | **1.35** |
| breadcube | k = 2 | 19.42 |
| | k = 4 | 11.36 |
| breadtoy | k = 2 | 19.62 |
| | k = 4 | **1.22** |

Table 6.2: Increasing the number of biclusters improve the results obtained by FABIA on the motion segmentation dataset.

## 6.2 Stable Region Correspondences

The other Computer Vision scenario where we introduce biclustering is the recent problem of *Stable Region Correspondences* [46]. Given a couple of shapes, SRC aims at the individuation of portions in the two shapes that behave *similarly*.

In the general scenario, the surface of a shape is commonly discretized by a polygonal mesh whose vertices can be represented by *descriptors*. Descriptors highlight local, or global, properties on the basis of each vertex neighbourhood. However descriptors representation is non homogeneous, since they can be scalar numbers or multidimensional arrays. Thus combining their information could be not trivial (as depicted in Fig 6-6). To pursue this task in [46] authors derive a similarity criteria allowing to compare two vertices on the basis of the computed descriptors.

This score states how similarly two vertices behave on the basis of the computed descriptors. Considering two shapes, we can now obtain this score for every couple of vertices belonging to different shapes. If we arrange such scores in a matrix such that the rows of this matrix represent the vertices of the first shape, and the columns represent the vertices of the second shape, we obtain the so called *Affinity Matrix*. Thus, the Affinity Matrix combines the information provided by the descriptors and, given a couple of vertices belonging to different shapes, it assigns a score according to the vertices behaviour: the more coherent, the higher. More formally: given a couple of shapes $S_1$ and $S_2$ having respectively $n$ and $m$ vertices, we adopt $s_i^1$ to indicate the $i$-th vertex belonging to the first shape (with $1 \leq \imath \leq n$) and $s_j^2$ for the $j$-th vertex of the second one (with $1 \leq \jmath \leq m$). Similarly we adopt $D_1(i)$ to indicate the set of descriptors evaluated on the $i$-th vertex of the first shape, and $D_2(j)$ to indicate the set of descriptors evaluated on the $j$-th vertex of the second shape. Now, the Affinity Matrix is a matrix $W \in \mathbb{R}_{n \times m}$ where $w_{ij} = Sim(D_1(i), D_2(j))$, where $Sim$ is the similarity criterion adopted to obtain the score[2]. This process results in a similarity matrix $W$ having as rows/columns the vertices of the first/second shape

[2]Further details on the coherence criteria is not argument of this thesis, we refer interested readers to [46].

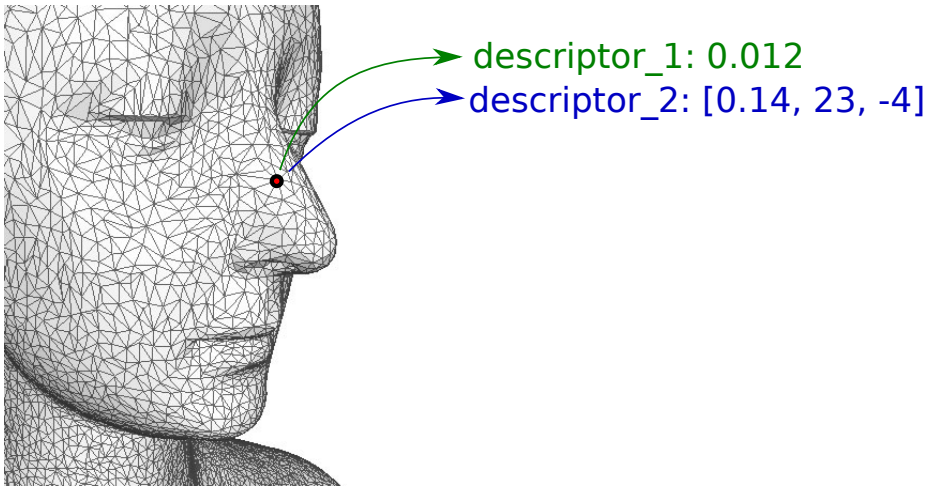



Figure 6-6: Shape vertices representation through non-homogeneous descriptors

indicating the similarity between a certain couple of vertices $i, j$.

In [46] authors tackle SRC exploiting the matrix $W$ to obtain a stable pair of matching regions. The general idea is exploit the Affinity Matrix to transport functions iteratively from one shape to the other, and then back to the original one. The intuition behind this reasoning is that in similar portions the transported functions should keep similar values; and analysing the functions evolution in consecutive iterations allows to retrieve a *stable* pair of regions. They then re-run this algorithm for the number of regions to retrieve, removing every time vertices already belonging to a stable region. Of course, the results obtained by the algorithm proposed in [46] are highly influenced by the definition of the functions adopted to perform the transportation.

Regarding the connections with biclustering, with respect to the Affinity Matrix, SRC aims at highlighting subsets of first shape vertices (subset of rows) acting coherently in subsets of second shape vertices (subset of columns). Thus, this formulation of SRC (as provided by authors in [46]) is a clear instance of the biclustering problem applied to the Affinity Matrix $W$. To clarify Fig. 6-7 shows an example of Affinity matrix, highlighting some of the relevant biclusters. We thus decide to assess the performances of our techniques on the SRC problem. Due to the high number of vertices contained by common shapes (about dozens of thousand vertices), we decide to investigate the usage of Spike and Slab Biclustering. Specifically, we exploit the Prior Knowledge extension proposed in in Sec. 5.3.

### 6.2.1 PKSSB for Stable Region Correspondences

The intuition behind the usage of PKSSB for this problem relies on the consideration that vertices in a shape carry on the intrinsic information of *spatiality*. Thus we want to investigate how the unexploited information of closeness can improve

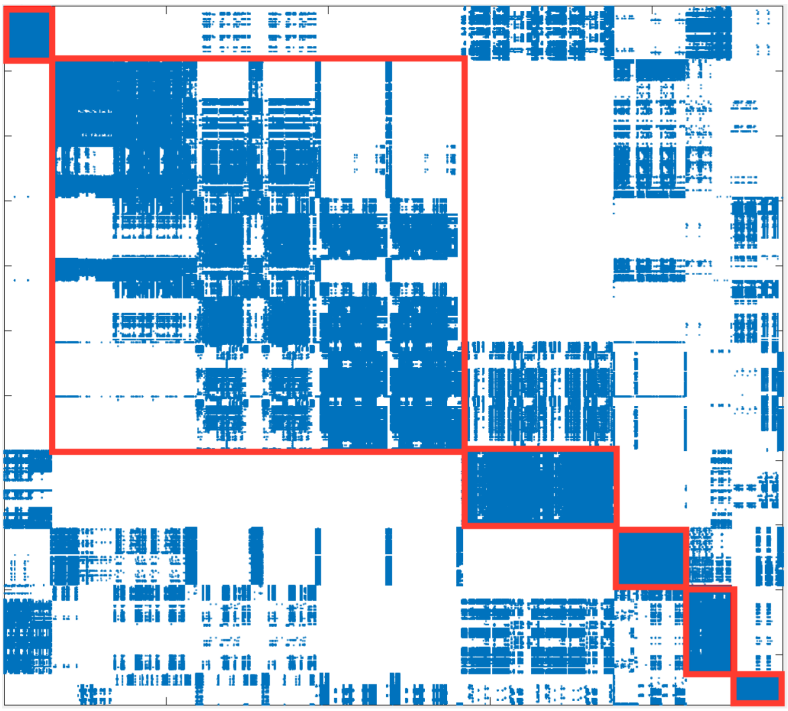


Figure 6-7: Example of Affinity Matrix: blocks represent relevant biclusters.

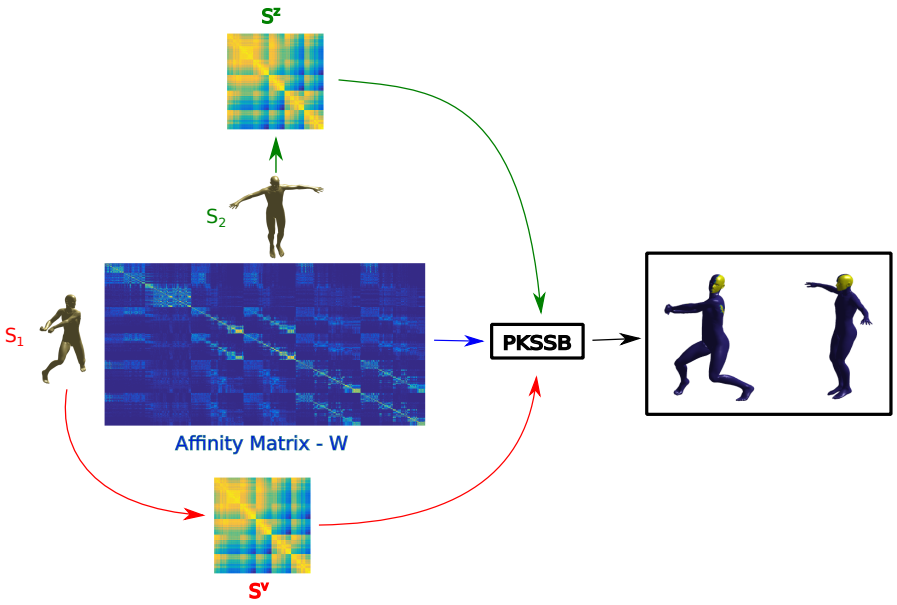



Figure 6-8: The framework of the the Prior Knowledge Spike and Slab Biclustering for the Stable Regions Correspondences

the results provided by the SSBiEM algorithm. In fact it is reasonable to expect that near vertices, hence belonging to the same region, should take part to the same biclusters.

Summarizing what presented in Sec.5.3, PKSSB allows the user to introduce some a-priori information about rows/columns that should take part to the same biclusters. Such information can be exploited through the definition of two square matrices: $S^v$ for the rows and $S^z$ for the columns. This matrices express the preferences with which two points should belong to the same bicluster. Specifically for the SRC scenario, each entry $S^{\bullet}_{ij} \geq 0$ (with $1 \leq i, j \leq n$ for the rows and $1 \leq i, j \leq m$ for the columns) indicates the preference magnitude with which two vertices $s^{\bullet}_i$ and $s^{\bullet}_j$ (of the same shape) should be in the same biclusters. Such information should help in obtaining well-connected regions, hence preventing the creation of holes. Thus, PKSSB takes as input three matrices: the Affinity Matrix $W$ and the two Preference Matrices $S^v$ and $S^z$. This information, combined with the number of biclusters $k$ to retrieve, allows the approach to isolate regions of the two shapes that behave coherently according to what represented by $W$. The framework is depicted in Fig. 6-8.

### Preliminary Results

We conduct a preliminary test taking into consideration one class of the Princeton Segmentation Benchmark Dataset [18]. This dataset contains 380 shapes divided in 19 object classes (20 shapes for each class). Each shape is represented by a triangu-

| Stable Regions method | PKSSB |
|---|---|
| 0.61 | 0.73 |

Table 6.3: Preliminary results of the PKSSB method compared with the algorithm presented in [46].

lar mesh and in each class there is a common meaningful segmentation. A manual segmentation of the surface meshes are given and shapes in each class are divided into semantic parts, yielding a ground truth segmentation. For this preliminar test we focused on the *glasses* class since most of the shapes have contained dimensions (about thousands vertices).

For each shape vertex we compute three different type of descriptors which are: *HKS* [120], *WKS* and *Multiscale mean curvature* [89]. HKS descriptors are based on *heat diffusion* and they provide information on how a certain vertex propagates heat in its neighbourhood, after a certain amount of time taken as parameter. Similarly to HKS WKS descriptors represents the average probability of measuring a quantum mechanical particle at a specic location, with different levels of energy. Finally, Multiscale mean curvature descriptors highlight different characteristics concerning the local geometry of a surface. In this case, as in [46], we take into account a set of hundred descriptors for each type obtaining 300 descriptors for each vertex.

Once all the descriptors have been computed, we randomly select 13 shapes couples and for each of this we retrieve the Affinity Matrix $W$ as described in [46]. As preference matrices, in order to exploit the spatial information, we decide to adopt the geodesic distances between vertices. More in details, given two vertices $v_1$ and $v_2$, the geodesic distance represents the length of the shortest path (*on the shape*) connecting $v_1$ to $v_2$. This can be computed as the integral of the surface curve evaluated from $v_1$ to $v_2$.

Hence we execute the PKSSB algorihm on the obtained matrices by varying the preference magnitude through a parameter $\beta$ multiplying the $S^v$ and $S^z$ matrices. Figure 6-9 shows some of the preliminar results. It describes how increasing $\beta$ actually forces closer points to belong the same biclusters, improving the results obtained by the basic SSBiEM. In fact it is clear how increasing the parameter leads to well-defined regions without imperfections or holes.

We can also quantitatively assess the performances of PKSSB. In fact on each shape there are some markers, positioned by hand, highlighting in the different shapes the points that should be grouped together. This ground truth can be adopted to compute the accuracy/purity on the retrieved stable regions. We compared the results of PKSSB with the method presented in [46]. Table 6.3 report the results, confirming that facing SRC problem with the PKSSB approach seems promising. Further experiments are ongoing during the writing of this thesis.

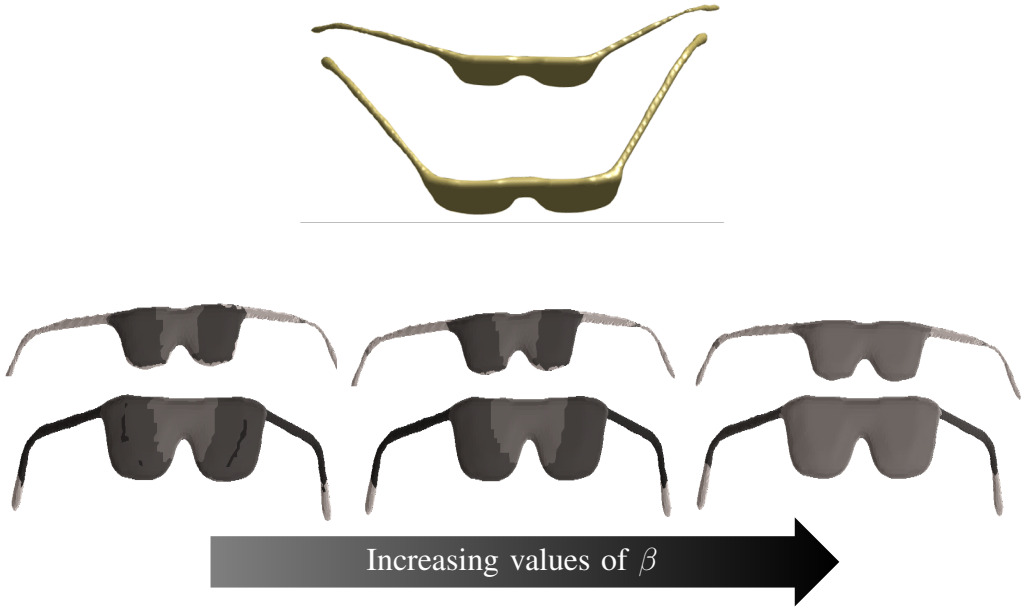



Figure 6-9: Results of PKSSB on the couple of shapes above. Results show that increasing $\beta$ (regulating the preferences magnitude) leads to less fragmented regions, as expected. In particular, the first result is the one obtained by the classic SSBiEM algorithm, hence $\beta = 0$.

# Chapter 7

# Conclusions and Future Works

Biclustering is a complex Pattern Recognition problem aiming at the simultaneous clustering of both rows and columns of a given data-matrix. It can extract from data hidden information that cannot be retrieved with classical clustering techniques: for this reason it has been exploited in many different scenarios (such as Biology, Recommender Systemsm or Text Analysis). A recent research trend regards the exploitation of Graphical Models to face the biclustering problem. Graphical models describe the decomposition of a global function in local sub-functions defined over a subset of the variables; the key idea behind this decomposition is that local sub-functions provide smaller problems easier to solve.

We inserted this thesis in the above-mentioned scenario, investigating the difficulties related to the exploitation of Graphical Models in the biclustering problem. In general, exploiting Graphical Models potential is not trivial since there exists a dualism between *representation* and *resolution*. In fact, both efficiency and efficacy of the resolution techniques depend on the design choices (*e.g.,* connections between variables, presence of cycles and so forth). Hence the challenge concerns the construction of a model which is as descriptive as possible, still remaining solvable. Furthermore, biclustering represents a complex and challenging problem, since it extends the combinatorial nature of clustering to both directions of a given data matrix. In this thesis we investigated the employment of Graphical Model based techniques in the biclustering scenario. First we analysed the relevant state-of-the-art approaches for biclustering highlighting the drawbacks of existing techniques based on graphical models: on one side existing approaches highly suffer the scalability problem, on the other hand other techniques resort to approximations to solve the proposed models. We thus pursued innovative research paths devising novel Graphical Models techniques to improve such weakness (*i.e.*, more scalable models solved with efficient analytical rules).

In more detail, this thesis provided the following contributions:

- **Factor Graph approaches for biclustering.** In this thesis we proposed different approaches based on Factor Graphs for the biclustering problem. Since

Max-Sum algorithm (which represents the standard choice in solving Factor Graphs) is highly influenced by the model design, and since biclustering is a difficult problem, few approaches for biclustering based on Factor Graphs have been proposed in literature. As a preliminar approach, we first proposed the Biclustering Affinity Propagation algorithm (BAP). BAP aimed at extending the good results obtained by the well known Affinity Propagation Factor Graph in the clustering scenario to biclustering. We thus derived a novel model performing an exemplar based clustering of the entries. We introduced a novel set of constraints which allow us to ensure that the obtained solutions are actually biclusters. However, these choices led to a model having too many cycles; thus preventing an effective resolution with the Max-Sum algorithm. We thus derived an equivalent linear formulation, and we solved it optimally with Linear Programming techniques. As a second relevant contribution, we reformulated the biclustering problem as the iterative search of one bicluster at a time. We designed a novel Factor Graph to solve this problem, and we derived a set of Max Sum messages update rules for the designed model, called OOB. The analytical derivation of such messages involved various meticulous and precise mathematical steps, thus involving a consistent amount of effort. With these messages derived we hence obtained an efficient biclustering technique solved with the Max-Sum algorithm. The methods performances have been assessed in both synthetic and real scenarios. For both BAP and OOB we empirically showed that Factor Graph based approaches provide accurate solutions and can improve the results obtained by current state-of-the-art techniques. Furthermore, OOB contributed in making a big step toward scalability (a key limitation of previous Factor Graph based approaches), allowing us to analyse up to $50 \times 50$ matrices (against the $10 \times 10$ provided by the state-of-the-art). As a third relevant contribution, we proposed two variants of OOB allowing the algorithm to retrieve more specific biclusters. In particular, we devised a novel set of factors specializing the model in two different ways: in one case we can exploit prior knowledge to set the preferred sizes of the retrieved biclusters; on the other hand we inserted a set of hard spatial constraint on one dimension of the model, allowing OOB to retrieve biclusters in Time-Series Datasets. Both these approaches (preferred size and time series) have been implemented exploiting the THOPs; a particular class of factors, defined on binary variables, for whom Max-Sum messages update rules can be computed efficiently. In this context we showed how it is possible to exploit Factor Graphs modularity to extend the devised model, given that Max-Sum messages for the novel add-ons can be performed efficiently. We further tested an extension of OOB on a real Gene Expression Time Series dataset. Also in this case the Factor Graph model provided promising results.

- **Bayesian Network approaches for Biclustering.** In this context we presented a probabilistic approach for biclustering based on Sparse Low-Rank Matrix Factorization. Due to the enormous size of common datasets for biclustering, sparsity plays a central role in probabilistic biclustering techniques. The state-of-the-art concerning probabilistic biclustering approaches focusing on Matrix Factorization techniques principally involves what are known as *latent-block models*. Briefly, the goal of such approaches is to rearrange both rows and columns of a given data matrix to obtain coherent pattern blocks. This idea intrinsically implies that biclusters do not overlap, and that the obtained biclusters represent a partition of the original data matrix. In contrast, only few approaches exploiting probabilistic matrix factorisation tools have been proposed to implement overlapping biclusters that do not form a partition, an example is FABIA [59]. In this thesis we devised a novel Bayesian Network where the main ingredient is represented by the introduction of a novel prior for the biclustering scenario: the so called Spike and Slab prior. More in detail, we proposed to approximate the original data matrix by a sum of rank-one matrices (plus noise). The structure and the values in the obtained matrices can provide information concerning the biclusters positions and values inside the original data matrix. Particularly, we exploited Spike and Slab prior to induce sparsity in the model. This, after tedious calculus manipulation, allowed us to derive efficient closed form updates for the EM algorithm, which we called SSBiEM. Differently from some state-of-the-art approaches, the choice of the Spike and Slab prior allowed us to directly estimate membership information (*i.e.,* which rows and columns belong to which bicluster) from the data. SSBiEM has been favourably compared with the state-of-the-art on both real and synthetic dataset. As a second relevant contribution, inspired by a recent work on clustering, we proposed a variant of the SSBiEM model able to incorporate some Prior Knowledge in the form of squared similarity matrices. In more detail, we defined two squared similarity matrices: one for the rows and one for the columns, containing the *preferences* for each couple of rows (or columns) to be in the same bicluster. The motivation behind this model is that, similarly to clustering, in biclustering scenario there exist various scenarios where we have high expectation for two points to be in the same group (i.e., two rows/columns to be in the same bicluster); and with this model we can now exploit such information to improve biclustering results.

- **Applications.** The third large class of contributions was aimed at enlarging the usage of biclustering techniques in alternative scenarios where it was never applied before. In particular, we investigated the suitability of the devised biclustering techniques in two Computer Vision scenarios, where biclustering has been never introduced: Multiple Structure Recovery and Sta-

ble Regions Correspondences. Briefly, Multiple Structure Recovery concerns the extraction of multiple models from noisy or outlier-contaminated data; whereas Stable Regions Correspondences aims at retrieving regions on different shapes that behave coherently (*i.e.,* fingers of a hand). Regarding Multiple Structure Recovery we showed that biclustering approaches provide superior quality results if compared with clustering techniques, which represent the classical solution in this context. Specifically, biclustering can retrieve accurate solutions also in highly noisy scenarios, which is the case of most Multiple Structure Recovery instances. On the other hand, we showed that the recent problem of Stable Regions Correspondences is an actual instance of biclustering. We also showed that SSBiEM algorithm, together with some Prior Knowledge information, can improve the results obtained by the current state-of-the-art. More in general, we showed that biclustering techniques provide efficient and accurate tools for many kind of problems, especially if these are represented by a preference/affinity matrix.

As a general final comment, we are convinced that we had provided large evidence that if the model is designed accurately, Graphical Model approaches provide efficient and effective solutions. Obviously, design a Graphical Model is not a trivial task. This arises mostly when dealing with Factor Graphs. In fact, due to the complete freedom concerning the design phase (local functions definition is unconstrained and the variable domain can be chosen freely), Factor Graphs allow to represent the problem in detail (the more complex the model, the better); on the other hand, the model complexity and its topology highly influence the resolution algorithm adopted to solve the model (the simpler the model, the better). Another problematic stems from the fact that both the resolution algorithms we adopted (Max-Sum and Expectation-Maximization) required the analytic derivations of closed form updates. These update rules must be computed iteratively to solve the model, thus their computation must be efficient to rend the devised approaches adoptable in real case scenarios. Further, we showed that Factor Graphs approaches provided good results on contained size dataset; in fact, although we improved the scalability of such approaches, the problem is not solved yet. On the other hand, Bayesian Networks allowed to analyse bigger matrices obtaining accurate results, with the drawbacks that tuning/learning the parameters represents a complex and crucial step.

The work done in this thesis pave the way for further studies, aimed at approaching novel challenges with Graphical Models and biclustering techniques. As presented in Chap. 4, there is still a lot of room for improvement concerning the scalability of Factor Graph based approaches. Similarly to what done with THOPs, a first interesting step would be to investigate the presence of particular factors, for integer variables, that could be solved efficiently by Max Sum messages. More in general, devising efficient resolution techniques for integer variables would open the door to a completely novel set of compact approaches. It would be also inter-

esting to investigate the performances of derived Factor Graphs techniques in other scenarios with limited size datasets. In biological or medical fields, for instance, there exist a lot of datasets that can be directly analysed by the proposed OOB, without the downstream heuristic aggregation. Finally, another aspect that could be faced concerns the exploration of the devised extensions, with particular dedication to the preferred size model.

Concerning the devised Bayesian Network techniques, a possible research direction in the short term may regard the automatic detection of the number of biclusters to retrieve. Particularly we can exploit the probabilistic nature of the model by adopting existing indexes (such as BIC or MDL). It would also be interesting to assess the performances of both SSBiEM and PK-SSBiEM in other Computer Vision scenarios. One example can be the Frequency Understanding problem where, given a set of vertices on a shape and a set of frequency functions, the goal is to retrieve a subset of vertices behaving similarly with respect to a subset of functions. This allows to retrieve, on a given shape, similar portions. Another problem that can be faced with biclustering in the Computer Vision scenario is represented by the Stable Region Correspondences applied to partial shapes. In fact, classical methods struggle in comparing two partial shapes, since the information is not complete. However biclustering approaches exploit the local information provided, and thus such problem can be faced with the same accuracy of the global ones. Moreover, in such scenarios, biclustering techniques can retrieve superior quality results if compared to classical clustering approaches. In fact, such application contexts are extremely noisy, and clustering approaches could fail (since they are much more sensitive to noise).

In conclusion, this thesis demonstrated the possibility of facing the biclustering problem adopting different types of Graphical Models. More than that, we provided evidence that the devised techniques can be successfully exported in many other scenarios than gene expression datasets, such as Multiple Structure Recovery and Stables Region Correspondences.

# Bibliography


[1] Ackerman, M., Ben-David, S.: Clusterability: A theoretical study. In: AISTATS. vol. 5, pp. 1–8 (2009)

[2] Ackerman, M., Ben-David, S., Loker, D.: Towards property-based classification of clustering paradigms. In: Advances in Neural Information Processing Systems. pp. 10–18 (2010)

[3] Ahmad, W., Khokhar, A.: Phoenix: Privacy preserving biclustering on horizontally partitioned data. Privacy, Security, and Trust in KDD pp. 14–32 (2008)

[4] Aji, S.M., McEliece, R.J.: The generalized distributive law. Information Theory, IEEE Transactions on 46(2), 325–343 (2000)

[5] Andres, B., Kappes, J.H., Köthe, U., Schnörr, C., Hamprecht, F.A.: An empirical comparison of inference algorithms for graphical models with higher order factors using opengm. In: Pattern Recognition, pp. 353–362. Springer (2010)

[6] Beibarth, T., Speed, T.P.: Gostat: find statistically overrepresented gene ontologies within a group of genes. Bioinformatics 20(9), 1464–1465 (2004), http://bioinformatics.oxfordjournals.org/content/20/9/1464.abstract

[7] Ben-Dor, A., Chor, B., Karp, R., Yakhini, Z.: Discovering local structure in gene expression data: the order-preserving submatrix problem. Journal of computational biology 10(3-4), 373–384 (2003)

[8] Bicego, M., Lovato, P., Ferrarini, A., Delledonne, M.: Biclustering of expression microarray data with topic models. In: Proceedings of the International Conference on Pattern Recognition. pp. 2728–2731 (2010)

[9] Bicego, M., Acosta-Muñoz, C., Orozco-Alzate, M.: Classification of seismic volcanic signals using hidden-markov-model-based generative embeddings. IEEE Transactions on Geoscience and Remote Sensing 51(6), 3400–3409 (2013)

[10] Bicego, M., Lovato, P., Perina, A., Fasoli, M., Delledonne, M., Pezzotti, M., Polverari, A., Murino, V.: Investigating topic models' capabilities in expression microarray data classification. IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB) 9(6), 1831–1836 (2012)

[11] Bilmes, J.A.: Graphical models and automatic speech recognition. In: Mathematical foundations of speech and language processing, pp. 191–245. Springer (2004)

[12] Bishop, C.: Pattern Recognition and Machine Learning. Springer-Verlag New York, Inc. (2006)

[13] Brown, P., Botstein, D.: Exploring the new world of the genome with dna microarrays. Nature Genetics 21, 33–37 (1999)

[14] Bunte, K., Leppaho, E., Saarinen, I., Kaski, S.: Sparse group factor analysis for biclustering of multiple data sources. Bioinformatics 32(16), 2457–2463 (2016)

[15] Cabral, R., De la Torre, F., Costeira, J.P., Bernardino, A.: Unifying nuclear norm and bilinear factorization approaches for low-rank matrix decomposition. In: Computer Vision (ICCV), 2013 IEEE International Conference on. pp. 2488–2495. IEEE (2013)

[16] de Castro, P., de França, F., Ferreira, H., Von Zuben, F.: Applying biclustering to text mining: an immune-inspired approach. Artificial Immune Systems pp. 83–94 (2007)

[17] Castro-Cabrera, P.A., Orozco-Alzate, M., Adami, A., Bicego, M., Londono-Bonilla, J.M., Castellanos-Domínguez, G.: A comparison between time-frequency and cepstral feature representations for the classification of seismic-volcanic signals. In: Iberoamerican Congress on Pattern Recognition. pp. 440–447. Springer (2014)

[18] Chen, X., Golovinskiy, A., Funkhouser, T.: A benchmark for 3d mesh segmentation. In: ACM Transactions on Graphics (TOG). vol. 28, p. 73. ACM (2009)

[19] Cheng, Y., Church, G.: Biclustering of expression data. In: Proc. Eighth Int. Conf. on Intelligent Systems for Molecular Biology (ISMB00). pp. 93–103 (2000)

[20] Cheng, Y., Church, G.M.: Biclustering of expression data. In: Ismb. vol. 8, pp. 93–103 (2000)

[21] Chin, T., Wang, H., Suter, D.: Robust fitting of multiple structures: The statistical learning approach. In: Int. Conf. on Computer Vision. pp. 413–420 (2009)

[22] Chin, T., Wang, H., Suter, D.: Robust fitting of multiple structures: The statistical learning approach. In: Int. Conf. on Computer Vision. pp. 413–420 (2009)

[23] Cho, R.J., Campbell, M.J., Winzeler, E.A., Steinmetz, L., Conway, A., Wodicka, L., Wolfsberg, T.G., Gabrielian, A.E., Landsman, D., Lockhart, D.J., et al.: A genome-wide transcriptional analysis of the mitotic cell cycle. Molecular cell 2(1), 65–73 (1998)

[24] Chu, P.C., Beasley, J.E.: A genetic algorithm for the multidimensional knapsack problem. Journal of heuristics 4(1), 63–86 (1998)

[25] Chuang, K.S., Tzeng, H.L., Chen, S., Wu, J., Chen, T.J.: Fuzzy c-means clustering with spatial information for image segmentation. Computerized Medical Imaging and Graphics 30(1), 9 – 15 (2006)

[26] Dantzig, G.: Linear Programming and Extensions. Princeton University Press (Aug 1963)

[27] Davidson, D.: Actions, reasons, and causes. The journal of philosophy 60(23), 685–700 (1963)

[28] Denitto, M., Farinelli, A., Figueiredo, M., Bicego, M.: A biclustering approach based on factor graphs and the max-sum algorithm. Pattern Recognition 62, 114–124 (2017)

[29] Denitto, M., Magri, L., Farinelli, A., Fusiello, A., Bicego, M.: Multiple structure recovery via probabilistic biclustering. In: Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR). pp. 274–284. Springer (2016)

[30] Denitto, M., Farinelli, A., Bicego, M.: Biclustering gene expressions using factor graphs and the max-sum algorithm. In: Proceedings of the 24th International Conference on Artificial Intelligence. pp. 925–931. AAAI Press (2015)

[31] Denitto, M., Farinelli, A., Franco, G., Bicego, M.: A binary factor graph model for biclustering. In: Structural, Syntactic, and Statistical Pattern Recognition, pp. 394–403. Springer (2014)

[32] Deodhar, M., Gupta, G., Ghosh, J., Cho, H., Dhillon, I.: A scalable framework for discovering coherent co-clusters in noisy data. In: Proceedings of the 26th Annual International Conference on Machine Learning. pp. 241–248. ACM (2009)

[33] Dolnicar, S., Kaiser, S., Lazarevski, K., Leisch, F.: Biclustering overcoming data dimensionality problems in market segmentation. Journal of Travel Research 51(1), 41–49 (2012)

[34] Farinelli, A., Denitto, M., Bicego, M.: Biclustering of expression microarray data using affinity propagation. In: Loog, M., Wessels, L., Reinders, M., Ridder, D. (eds.) Pattern Recognition in Bioinformatics, Lecture Notes in Computer Science, vol. 7036, pp. 13–24. Springer Berlin Heidelberg (2011)

[35] Farinelli, A., Rogers, A., Petcu, A., Jennings, N.R.: Decentralised coordination of low-power embedded devices using the max-sum algorithm. In: Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems-Volume 2. pp. 639–646. International Foundation for Autonomous Agents and Multiagent Systems (2008)

[36] Felipe, J.C., Ribeiro, M.X., Sousa, E.P., Traina, A.J., Traina Jr, C.: Effective shape-based retrieval and classification of mammograms. In: Proceedings of the 2006 ACM symposium on Applied computing. pp. 250–255. ACM (2006)

[37] Figueiredo, M.A., Cheng, D.S., Murino, V.: Clustering under prior knowledge with application to image segmentation. Advances in Neural Information Processing Systems 19, 401 (2007)

[38] Fitzgibbon, A.W., Zisserman, A.: Multibody structure and motion: 3-d reconstruction of independently moving objects. In: Computer Vision-ECCV 2000, pp. 891–906. Springer (2000)

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

[40] Frakes, W.B., Baeza-Yates, R.: Information retrieval: data structures and algorithms (1992)

[41] Frey, B., Dueck, D.: Clustering by passing messages between data points. Science 315, 972–976 (2007)

[42] Frey, B., Jojic, N.: A comparison of algorithms for inference and learning in probabilistic graphical models. IEEE Transactions on Pattern Analysis and Machine Intelligence 27, 1–25 (2005)

[43] Frey, B.J.: Graphical models for machine learning and digital communication. MIT press (1998)

[44] Frey, B.J., Jojic, N.: A comparison of algorithms for inference and learning in probabilistic graphical models. Pattern Analysis and Machine Intelligence, IEEE Transactions on 27(9), 1392–1416 (2005)

[45] Frey, B.J., Kschischang, F.R., Loeliger, H.A., Wiberg, N.: Factor graphs and algorithms. In: Proceedings of the Annual Allerton Conference on Communication Control and Computing. vol. 35, pp. 666–680. UNIVERSITY OF ILLINOIS (1997)

[46] Ganapathi-Subramanian, V., Thibert, B., Ovsjanikov, M., Guibas, L.: Stable region correspondences between non-isometric shapes. In: Computer Graphics Forum. vol. 35, pp. 121–133. Wiley Online Library (2016)

[47] Gasch, A.P., Spellman, P.T., Kao, C.M., Carmel-Harel, O., Eisen, M.B., Storz, G., Botstein, D., Brown, P.O.: Genomic expression programs in the response of yeast cells to environmental changes. Molecular biology of the cell 11(12), 4241–4257 (2000)

[48] Getz, G., Levine, E., Domany, E.: Coupled two-way clustering analysis of gene microarray data. Proc Natl Acad Sci U S A 97(22), 12079–12084 (2000)

[49] Globerson, A., Jaakkola, T.S.: Fixing max-product: Convergent message passing algorithms for map lp-relaxations. In: Advances in neural information processing systems. pp. 553–560 (2008)

[50] Golub, G.H., Reinsch, C.: Singular value decomposition and least squares solutions. Numerische mathematik 14(5), 403–420 (1970)

[51] Govaert, G., Nadif, M.: An em algorithm for the block mixture model. IEEE Transactions on Pattern Analysis and machine intelligence 27(4), 643–647 (2005)

[52] Gremalschi, S., Altun, G.: Mean squared residue based biclustering algorithms. In: Bioinformatics Research and Applications, pp. 232–243. Springer (2008)

[53] Gupta, N., Aggarwal, S.: Mib: Using mutual information for biclustering gene expression data. Pattern Recognition 43(8), 2692–2697 (2010)

[54] Hammersley, J.: Monte carlo methods. Springer Science & Business Media (2013)

[55] Häne, C., Zach, C., Zeisl, B., Pollefeys, M.: A patch prior for dense 3d reconstruction in man-made environments. In: 3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT), 2012 Second International Conference on. pp. 563–570. IEEE (2012)

[56] Hartigan, J.A.: Direct clustering of a data matrix. Journal of the american statistical association 67(337), 123–129 (1972)

[57] Henriques, R., Antunes, C., Madeira, S.C.: A structured view on pattern mining-based biclustering. Pattern Recognition 48(12), 3941–3958 (2015)

[58] Hestenes, M.: Multiplier and gradient methods. Journal of Optimization Theory and Applications 4, 303–320 (1969)

[59] Hochreiter, S., Bodenhofer, U., Heusel, M., Mayr, A., Mitterecker, A., Kasim, A., Khamiakova, T., Van Sanden, S., Lin, D., Talloen, W., et al.: Fabia: factor analysis for bicluster acquisition. Bioinformatics 26(12), 1520–1527 (2010)

[60] Hoyer, P.O.: Non-negative matrix factorization with sparseness constraints. The Journal of Machine Learning Research 5, 1457–1469 (2004)

[61] Ihmels, J., Bergmann, S., Barkai, N.: Defining transcription modules using large-scale gene expression data. Bioinformatics 20(13), 1993–2003 (2004)

[62] Isack, H., Boykov, Y.: Energy-based geometric multi-model fitting. International Journal of Computer Vision 97(2), 123–147 (2012)

[63] Ishwaran, H., Rao, J.S.: Spike and slab variable selection: frequentist and bayesian strategies. Annals of Statistics pp. 730–773 (2005)

[64] Jain, A., Murty, M., Flynn, P.: Data clustering: a review. ACM computing surveys 21, 264–323 (1999)

[65] Jain, S., Govindu, V.M.: Efficient higher-order clustering on the grassmann manifold. In: Int. Conf. on Computer Vision (2013)

[66] Jensen, F.V.: An introduction to Bayesian networks, vol. 210. UCL press London (1996)

[67] Jensen, F.V.: Bayesian Networks and Decision Graphs. Springer-Verlag New York, Inc., Secaucus, NJ, USA (2001)

[68] Jensen, F.V.: Bayesian networks and decision graphs. series for statistics for engineering and information science (2001)

[69] Jojic, V., Gould, S., Koller, D.: Accelerated dual decomposition for map inference. In: Proceedings of the 27th International Conference on Machine Learning (ICML-10). pp. 503–510 (2010)

[70] Jordan, M.I., Ghahramani, Z., Jaakkola, T.S., Saul, L.K.: An introduction to variational methods for graphical models. Machine learning 37(2), 183–233 (1999)

[71] Jordan, M.I.: Learning in graphical models, vol. 89. Springer Science & Business Media (1998)

[72] Joung, J.G., Fei, Z.: Identification of microrna regulatory modules in arabidopsis via a probabilistic graphical model. Bioinformatics 25(3), 387–393 (2009)

[73] Kaytoue, M., Codocedo, V., Buzmakov, A., Baixeries, J., Kuznetsov, S.O., Napoli, A.: Pattern structures and concept lattices for data mining and knowledge processing. In: Machine Learning and Knowledge Discovery in Databases, pp. 227–231. Springer (2015)

[74] Komodakis, N., Paragios, N., Tziritas, G.: Mrf optimization via dual decomposition: Message-passing revisited. In: Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on. pp. 1–8. IEEE (2007)

[75] Kschischang, F., Frey, B., Loeliger, H.A.: Factor graphs and the sum-product algorithm. Information Theory, IEEE Transactions on 47(2), 498 –519 (feb 2001)

[76] Kuang, D., Yun, S., Park, H.: Symnmf: nonnegative low-rank approximation of a similarity matrix for graph clustering. Journal of Global Optimization pp. 1–30 (2014)

[77] Lauritzen, S.L.: Graphical models. Oxford University Press (1996)

[78] Lazzeroni, L., Owen, A., et al.: Plaid models for gene expression data. Statistica sinica 12(1), 61–86 (2002)

[79] Li, T., Ding, C.: The relationships among various nonnegative matrix factorization methods for clustering. In: Data Mining, 2006. ICDM'06. Sixth International Conference on. pp. 362–371. IEEE (2006)

[80] Loeliger, H.A.: An introduction to factor graphs. Signal Processing Magazine, IEEE 21(1), 28–41 (2004)

[81] Madeira, S.C., Oliveira, A.L.: A linear time biclustering algorithm for time series gene expression data. In: Algorithms in Bioinformatics: 5th International Workshop, WABI 2005, Mallorca, Spain, October 3-6, 2005. Proceedings. pp. 39–52. Springer Berlin Heidelberg, Berlin, Heidelberg (2005)

[82] Madeira, S.C., Teixeira, M.C., Sa-Correia, I., Oliveira, A.L.: Identification of regulatory modules in time series gene expression data using a linear time biclustering algorithm. IEEE/ACM Transactions on Computational Biology and Bioinformatics 7(1), 153–165 (2010)

[83] Madeira, S., Oliveira, A.: Biclustering algorithms for biological data analysis: a survey. IEEE transactions on Computational Biology and Bioinformatics 1, 24–44 (2004)

[84] Magnus, J., Neudecker, H.: Matrix Differential Calculus with Applications in Statistics and Econometrics. Wiley (1999)

[85] Magri, L., Fusiello, A.: T-linkage: a continuous relaxation of j-linkage for multi-model fitting. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 3954–3961 (2014)

[86] Magri, L., Fusiello, A.: Robust multiple model fitting with preference analysis and low-rank approximation. In: Proceedings of the British Machine Vision Conference (BMVC). pp. 20.1–20.12. BMVA Press (September 2015)

[87] Martins, A.F., Figueiredo, M.A., Aguiar, P.M., Smith, N.A., Xing, E.P.: Ad3: Alternating directions dual decomposition for map inference in graphical models. Journal of Machine Learning Research 16, 495–545 (2015)

[88] Melouk, S., Damodaran, P., Chang, P.Y.: Minimizing makespan for single machine batch processing with non-identical job sizes using simulated annealing. International Journal of Production Economics 87(2), 141–147 (2004)

[89] Meyer, M., Desbrun, M., Schröder, P., Barr, A.H.: Discrete differential-geometry operators for triangulated 2-manifolds. In: Visualization and mathematics III, pp. 35–57. Springer (2003)

[90] Mitchell, T.J., Beauchamp, J.J.: Bayesian variable selection in linear regression. Journal of the American Statistical Association 83(404), 1023–1032 (1988)

[91] Mukhopadhyay, A., Maulik, U., Bandyopadhyay, S., Coello, C.A.C.: Survey of multiobjective evolutionary algorithms for data mining: Part ii. Evolutionary Computation, IEEE Transactions on 18(1), 20–35 (2014)

[92] Munkres, J.: Algorithms for the assignment and transportation problems. Journal of the Society for Industrial and Applied Mathematics 5(1), 32–38 (1957)

[93] Murali, T., Kasif, S.: Extracting conserved gene expression motifs from gene expression data. In: Pacific Symposium on Biocomputing. vol. 8, pp. 77–88. World Scientific (2003)

[94] Nadif, M., Govaert, G.: Model-based co-clustering for continuous data. In: Machine Learning and Applications (ICMLA), 2010 Ninth International Conference on. pp. 175–180. IEEE (2010)

[95] Nielsen, T.D., Jensen, F.V.: Bayesian networks and decision graphs. Springer Science & Business Media (2009)

[96] Nocedal, J., Wright, S.: Numerical Optimization. Springer (2006)

[97] O'Connor, L., Feizi, S.: Biclustering using message passing. In: Advances in Neural Information Processing Systems. pp. 3617–3625 (2014)

[98] Oghabian, A., Kilpinen, S., Hautaniemi, S., Czeizler, E.: Biclustering methods: Biological relevance and application in gene expression analysis. PloS one 9(3), e90801 (2014)

[99] Ozden, K.E., Schindler, K., Van Gool, L.: Multibody structure-from-motion in practice. Pattern Analysis and Machine Intelligence, IEEE Transactions on 32(6), 1134–1141 (2010)

[100] Pearl, J.: Fusion, propagation, and structuring in belief networks. Artificial intelligence 29(3), 241–288 (1986)

[101] Pekalska, E., Duin, R.P.: The Dissimilarity Representation for Pattern Recognition: Foundations And Applications (Machine Perception and Artificial Intelligence). World Scientific Publishing (2005)

[102] Pelillo, M.: What is a cluster? perspectives from game theory. In: Proc. of the NIPS Workshop on Clustering Theory (2009)

[103] Perina, A., Lovato, P., Murino, V., Bicego, M.: Biologically-aware latent dirichlet allocation (balda) for the classification of expression microarray. In: IAPR International Conference on Pattern Recognition in Bioinformatics. pp. 230–241. Springer (2010)

[104] Pham, T.T., Chin, T.J., Yu, J., Suter, D.: The random cluster model for robust geometric fitting. Pattern Analysis and Machine Intelligence 36(8), 1658–1671 (2014)

[105] Pledger, S., Arnold, R.: Multivariate methods using mixtures: Correspondence analysis, scaling and pattern-detection. Computational Statistics & Data Analysis 71, 241–261 (2014)

[106] Powell, M.: A method for nonlinear constraints in minimization problems. In: Fletcher, R. (ed.) Optimization, pp. 283–298. Academic Press (1969)

[107] Prelic, A., Bleuler, S., Zimmermann, P., Wille, A., Bhlmann, P., Gruissem, W., Hennig, L., Thiele, L., Zitzler, E.: Comparison of biclustering methods: A systematic comparison and evaluation of biclustering methods for gene expression data. Bioinformatics 22(9), 1122–1129 (May 2006)

[108] Priam, R., Nadif, M., Govaert, G.: Gaussian topographic co-clustering model. In: International Symposium on Intelligent Data Analysis. pp. 345–356. Springer (2013)

[109] Raftery, A.E.: A note on bayes factors for log-linear contingency table models with vague prior information. Journal of the Royal Statistical Society, Series B 48, 249–250 (1986)

[110] Reiss, D.J., Baliga, N.S., Bonneau, R.: Integrated biclustering of heterogeneous genome-wide datasets for the inference of global regulatory networks. BMC bioinformatics 7(1), 280 (2006)

[111] Rissanen, J.: A universal prior for integers and estimation by minimum description length. The Annals of statistics pp. 416–431 (1983)

[112] R.O. Duda, P.E. Hart, D.S.: Pattern Classification (2nd Edition). Wiley Interscience (2001)

[113] Rogers, S., Girolami, M., Campbell, C., Breitling, R.: The latent process decomposition of cdna microarray data sets. IEEE/ACM Transactions on Computational Biology and Bioinformatics 2(2), 143–156 (2005)

[114] Rogers, S., Girolami, M., Campbell, C., Breitling, R.: The latent process decomposition of cdna microarray data sets. IEEE/ACM Transactions on Computational Biology and Bioinformatics 2(2), 143–156 (2005)

[115] Sheng, Q., Moreau, Y., De Moor, B.: Biclustering microarray data by gibbs sampling. Bioinformatics 19(suppl 2), ii196–ii205 (2003)

[116] Shi, J., Malik, J.: Normalized cuts and image segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence 22(8), 888–905 (2000)

[117] Sokal, R.R.: A statistical method for evaluating systematic relationships. Univ Kans Sci Bull 38, 1409–1438 (1958)

[118] Soltanolkotabi, M., Elhamifar, E., Candès, E.J.: Robust subspace clustering. Ann. Statist. 42(2), 669–699 (04 2014)

[119] Sudderth, E.B.: Graphical models for visual object recognition and tracking. Ph.D. thesis, Massachusetts Institute of Technology (2006)

[120] Sun, J., Ovsjanikov, M., Guibas, L.: A concise and provably informative multi-scale signature based on heat diffusion. In: Computer graphics forum. vol. 28, pp. 1383–1392. Wiley Online Library (2009)

[121] Tanay, A., Sharan, R., Shamir, R.: Discovering statistically significant biclusters in gene expression data. Bioinformatics 18(suppl 1), S136–S144 (2002)

[122] Tang, C., Zhang, L., Zhang, A., Ramanathan, M.: Interrelated two-way clustering: an unsupervised approach for gene expression data analysis. In: Bioinformatics and Bioengineering Conference, 2001. Proceedings of the IEEE 2nd International Symposium on. pp. 41–48. IEEE (2001)

[123] Tarlow, D., Givoni, I.E., Zemel, R.S.: Hop-map: Efficient message passing with high order potentials. In: International Conference on Artificial Intelligence and Statistics. pp. 812–819 (2010)

[124] Tepper, M., Sapiro, G.: A biclustering framework for consensus problems. SIAM Journal on Imaging Sciences 7(4), 2488–2525 (2014)

[125] Toldo, R., Fusiello, A.: Robust multiple structures estimation with J-linkage. In: European Conf. on Computer Vision (2008)

[126] Toldo, R., Fusiello, A.: Robust multiple structures estimation with j-linkage. In: ECCV 2008, pp. 537–547. Springer (2008)

[127] Toldo, R., Fusiello, A.: Image-consistent patches from unstructured points with j-linkage. Image and Vision Computing 31(10), 756–770 (2013)

[128] Torr, P.H.S., Zisserman, A.: MLESAC: A new robust estimator with application to estimating image geometry. Comp. Vis. and Image Underst. (1), 138–156 (2000)

[129] Tou, J.T., Gonzalez, R.C.: Pattern recognition principles (1974)

[130] Troyanskaya, O., Cantor, M., Sherlock, G., Brown, P., Hastie, T., Tibshirani, R., Botstein, D., Altman, R.B.: Missing value estimation methods for dna microarrays. Bioinformatics 17(6), 520–525 (2001)

[131] Tu, K., Ouyang, X., Han, D., Honavar, V.: Exemplar-based robust coherent biclustering. In: SDM. pp. 884–895. SIAM (2011)

[132] Valafar, F.: Pattern recognition techniques in microarray data analysis: A survey. Annals of the New York Academy of Sciences 980, 41–64 (2002)

[133] Vámos, T.: Judea pearl: Probabilistic reasoning in intelligent systems. Decision Support Systems 8(1), 73–75 (1992)

[134] Vu, D., Aitkin, M.: Variational algorithms for biclustering models. Computational Statistics & Data Analysis 89, 12–24 (2015)

[135] Wainwright, M.J., Jordan, M.I., et al.: Graphical models, exponential families, and variational inference. Foundations and Trends® in Machine Learning 1(1–2), 1–305 (2008)

[136] Wang, H., Wang, W., Yang, J., Yu, P.S.: Clustering by pattern similarity in large data sets. In: Proceedings of the 2002 ACM SIGMOD international conference on Management of data. pp. 394–405. ACM (2002)

[137] Wang, H., Nie, F., Huang, H., Makedon, F.: Fast nonnegative matrix tri-factorization for large-scale data co-clustering. In: IJCAI Proceedings-International Joint Conference on Artificial Intelligence. vol. 22, p. 1553 (2011)

[138] Wang, P.S.P.: Pattern Recognition and Machine Vision, vol. 6. River Publishers (2010)

[139] Watanabe, S.: Pattern recognition: human and mechanical. John Wiley & Sons, Inc., New York, NY, USA (1985)

[140] Weiss, Y., Freeman, W.: Correctness of belief propagation in gaussian graphical models of arbitrary topology. Neural Computation 13(10), 2173–2200 (2001)

[141] Wiberg, N., Loeliger, H.A., Kotter, R.: Codes and iterative decoding on general graphs. European Transactions on telecommunications 6(5), 513–525 (1995)

[142] Wu, W., Xiong, H., Shekhar, S.: Clustering and information retrieval, vol. 11. Springer Science & Business Media (2013)

[143] Xu, L., Oja, E., Kultanen, P.: A new curve detection method: randomized Hough transform (RHT). Pattern Recognition Letters 11(5), 331–338 (1990)

[144] Xue, Y., Liao, Z., Li, M., Luo, J., Hu, X., Luo, G., Chen, W.S.: A new biclustering algorithm for time-series gene expression data analysis. In: Computational Intelligence and Security (CIS), 2014 Tenth International Conference on. pp. 268–272. IEEE (2014)

[145] Yang, J., Wang, H., Wang, W., Yu, P.S.: An improved biclustering method for analyzing gene expression profiles. International Journal on Artificial Intelligence Tools 14(05), 771–789 (2005)

[146] Yedidsion, H., Zivan, R., Farinelli, A.: Explorative max-sum for teams of mobile sensing agents. In: Proceedings of the 2014 International Conference on Autonomous Agents and Multi-agent Systems. pp. 549–556. AAMAS '14, International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC (2014)

[147] Yoo, J., Choi, S.: Orthogonal nonnegative matrix tri-factorization for co-clustering: Multiplicative updates on stiefel manifolds. Information processing & management 46(5), 559–570 (2010)

[148] Yoon, S., Nardini, C., Benini, L., De Micheli, G.: Discovering coherent biclusters from gene expression data using zero-suppressed binary decision diagrams. IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB) 2(4), 339–354 (2005)

[149] Zhang, W., Kosecká, J.: Nonparametric estimation of multiple structures with outliers. In: European Conf. on Computer Vision. vol. 4358, pp. 60–74 (2006)

[150] Zhang, Y., Zha, H., Chu, C.H.: A time-series biclustering algorithm for revealing co-regulated genes. In: International Conference on Information Technology: Coding and Computing (ITCC'05)-Volume II. vol. 1, pp. 32–37. IEEE (2005)

[151] Zhang, Z.Y., Li, T., Ding, C., Ren, X.W., Zhang, X.S.: Binary matrix factorization for analyzing gene expression data. Data Mining and Knowledge Discovery 20(1), 28–52 (2010)

[152] Zuliani, M., Kenney, C.S., Manjunath, B.S.: The multiRANSAC algorithm and its application to detect planar homographies. In: Int. Conf. on Image Processing (2005)