Pattern Recognition 62 (2017) 114–124

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/pr

A biclustering approach based on factor graphs and the max-sum algorithm

M. Denitto [a,*], A. Farinelli [a], M.A.T. Figueiredo [b], M. Bicego [a]

[a] University of Verona, Strada Le Grazie 15, Ca' Vignal 2, Verona, Italy
[b] Instituto de Telecomunicações, and Instituto Superior Técnico, Universidade de Lisboa, Av. Rovisco Pais 1, Lisboa, Portugal

ARTICLE INFO

Article history:
Received 25 February 2016
Received in revised form
8 July 2016
Accepted 30 August 2016
Available online 31 August 2016

Keywords:
Biclustering
Coclustering
Factor graphs
Max-sum
Expression data

ABSTRACT

Biclustering represents an intrinsically complex problem, where the aim is to perform a simultaneous row- and column-clustering of a given data matrix. Some recent approaches model this problem using factor graphs, so to exploit their ability to open the door to efficient optimization approaches for well designed function decompositions. However, while such models provide promising results, they do not scale to data matrices of reasonable size. In this paper, we take a step towards addressing this issue, by proposing a novel approach to biclustering based on factor graphs, which yields high quality solutions and scales more favorably than previous methods. Specifically, we cast biclustering as the sequential search for a single bicluster, and propose a *binary* and *compact* factor graph that can be solved efficiently using the max-sum algorithm. The proposed approach has been tested and compared with state-of-the-art methods on four datasets (two synthetic and two real world data), providing encouraging results with respect both to previous approaches based on factor graphs and to other state-of-the-art methods.

1. Introduction

In many scientific areas, it may be of interest to group/cluster a set of objects, based on a set of observed features. 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 biology processes. Such information, which cannot be retrieved by classical clustering, can be extracted by simultaneously clustering both the rows and the columns of a given data matrix (where each row corresponds to a different object/sample and each column to a different feature), a task that is known as *biclustering*. The problem of biclustering, also known as co-clustering, and closely related to subspace clustering, has been introduced in the bioinformatics field to analyse expression data [1–7]; recently, it has been adopted in a wider range of applications such as market segmentation and data mining [8–11].

Many approaches have been proposed to address the biclustering problem, ranging from probabilistic models (such as the plaid model [12] or FABIA [13]) to two-way clustering methods [14–16]. A recent comprehensive review was produced by Oghabian et al. [5].

In order to face the intrinsic complexity of biclustering, some recent approaches started to exploit sophisticated computational models, which are able to provide very accurate solutions; one example of this trend is the class of approaches based on *factor graphs* (FG) [15,17,18]. FG are a class of graphical models [19] which encode a *global* function as a collection of factors (*local* functions) defined over subsets of variables [20], with such decomposition being described via a bi-partite graph of variables and factors. The main advantage in FG consists in the exploitation of the global function decomposition to solve smaller and local subproblems, providing accurate approximation for the optimization of the objective function. In fact, such decomposition can lead to powerful algorithms, which have been used to effectively solve various computational tasks [19]). A notable example is the *affinity propagation* method [21], where the authors use a FG to encode an exemplar-based model for the well known clustering problem [21].

In the biclustering context, however, the potential of FG has not yet been fully investigated, with only a few approaches having been proposed [15,17,18]. Arguably, this is mainly due to the difficulty of designing FG with an effective trade-off between representation power and the 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 FG (in this case, the simpler, the better). For example, the max-sum algorithm – which is the most popular

* Corresponding author.
E-mail address: matteo.denitto@univr.it (M. Denitto).

http://dx.doi.org/10.1016/j.patcog.2016.08.033
0031-3203/

choice for optimizing FG [22,19,21][1] – provides solutions that approximate the global optimum, where the quality of such solution depends on the cycles present in the FG (the optimal solution is guaranteed in case of no cycles [24]); moreover, its computational effort strongly depends on the nature of the variables – e.g. binary variables lead to very fast optimization, such as in the affinity propagation method [21].

The crucial issues mentioned in the previous paragraph need to be considered when designing FG-based approaches and are far from being solved, particularly for biclustering [15,17,18]. Recent work has adapted the affinity propagation method [21] to biclustering, by performing iterative and sequential row-column clustering [15]. However, while that approach provides good results (on a synthetic benchmark), the function being optimized is still designed for clustering, not for biclustering; consequently, that approach fails in providing a descriptive function to be optimized. This aspect has been improved by clustering directly the entries of the data matrix, extending the FG of affinity propagation with a constraint that only allows clusters corresponding to biclusters (i.e., intersection of subsets of rows and columns) [18]; the objective function now better describes the problem, but the resulting model is too complex, with too many cycles to be effectively optimized by the max-sum algorithm (a fact that lead the authors of that work to propose a linear programming method, which, however, does not scale beyond 10×10 matrices). An alternative and interesting approach consists in making the model more compact (fewer variables) by using n-ary (rather than binary) variables [17]. However, as mentioned above, the binary nature of the variables is essential to obtain an efficient and fast optimization algorithm; in fact, the optimization technique therein proposed is only able to deal with matrices no larger than 10×10, while for larger matrices the authors introduced an approximate max-sum algorithm and a greedy approach.

In this paper, we take what we believe is an important step towards efficient FG-based biclustering, by proposing a novel approach able to better face the afore-mentioned issues. More specifically: we reformulate biclustering as a sequential search problem, discovering one bicluster at time (an approach employed by other authors [25–27]); we derive a novel binary FG that retrieves one bicluster on the basis of a given 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 of previous FG-based approaches.

We test our approach on two synthetic benchmark and two real-world problems involving gene expression data. On the synthetic benchmarks, which contain matrices of dimension 50×50, we compare our approach with those of [17,18]; however, since those methods cannot be applied in their original formulation for matrices of this size, we used the approximate versions provided by those authors. Our optimization procedure based on max-sum has been also favorably compared with an alternative optimization based on the recent AD^3 algorithm [28]. The real gene expression datasets (*yeast* and *breast tumor*) allow comparing our approach with that in [17], as well as with other state-of-the-art approaches [9]. The obtained results confirm the suitability of the proposed FG-based method.

2. Background and related works

This section provides an introduction to biclustering and FG, followed by a detailed discussion of the current FG-based approaches to biclustering.

[1] This algorithm has been exploited also for biclustering, without using factor graphs [23].

2.1. Biclustering

The goal of biclustering is the simultaneous clustering of the rows and columns of a given data matrix. Formally, it 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 set of column indices. We denote as A_{TK} the submatrix that includes the rows in $T \subseteq N$ and the columns in $K \subseteq M$. Using this notation, we can define a $p \times r$ (where $1 \leq p \leq n, 1 \leq r \leq m$) *bicluster* as submatrix A_{TK}, such that the rows in $T = \{t_1, \ldots, t_p\}$ present a "coherent behavior" (in some sense) across the columns in $K = \{k_1, \ldots, k_r\}$, and vice versa. The available literature offers a wide range of coherence criteria, the choice of which critically influences the nature of the biclusters obtained [7,5,1,29].

2.2. Factor graphs

Let x be an l-dimensional vector of variables, and let $g(x)$ denote the value of some objective function for a given configuration x. Formally, a factor graph includes two types of nodes: a collection of variable nodes, one of each of the l variables ($x_1, \ldots, x_l$, usually drawn as circles) and a collection of *factor nodes* ($f_1, \ldots, f_d$, 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). Function f is represented by this factor graph if can be factored as

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

where $S_t \subseteq \{1, \ldots, n\}$ is the subset of variable nodes to which factor node f_t is connected, and x_{S_t} denotes the corresponding sub-vector of x. Factor graphs were created and are most often used in a probabilistic inference context, where $g(x)$ is usually the (posterior) probability function in some inference problem: $p(x) = g(x)/Z$, where Z is a normalization constant [19]. One of the standard problems in that context, finding the *maximum a posteriori* (MAP) configuration $\hat{x} = \text{argmax}_x p(x)$, corresponds thus to finding the maximizer(s) of $g(x)$ [19,30]. Naturally, the probabilistic interpretation is not mandatory and there are FG-based approaches devoid of any probabilistic interpretation, as is the case of affinity propagation [21]; there, the variables simply represent decisions, and the FG expresses the decomposition of the global objective function and possible constraints. In this paper, we adopt this non-probabilistic view of FG.

As mentioned in Section 1, FG are very flexible models, since there are no limitations on the form of the local functions f_t; for example, they can be designed to impose hard or soft constraints, or have different types of variable domains (e.g., categorical, integer, binary, real numbers). However, the choices made when designing a FG have a drastic effect on the difficulty of the resulting optimization problem.

2.3. Optimization of factor graphs

The optimization algorithms typically used for FG belong to two classes: message passing and search-based [31]. 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 by the FG 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) [31]. Thanks to its remarkable success in coding theory [32], message-passing methods are the standard choice for most applications, with the *max-sum*

algorithm being one of the most famous instances in this class, having been used in many applications of FG models [19,21,22,33,34]. 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) [22]. It can be shown that if the FG has no cycles (e.g., if it is a tree), then max-sum can find the maximum of the objective function [24].

The max-sum algorithm is based on the definition of two functions, called *messages*, which exchange information between connected nodes:

1. From factor f_t to variable x_i (with $i \in S_t$):

$$\mu_{f_t \to x_i}(x_i) = \max_{x_{S_t \setminus \{i\}}} \left[\log f_t(x_{S_t}) + \sum_{m \in S_t \setminus i} \mu_{x_m \to f_t}(x_m) \right]; \tag{2}$$

2. From variable x_i to factor f_t (with $i \in S_t$):

$$\mu_{x_i \to f_t}(x_i) = \sum_{l \in \mathrm{ne}(i) \setminus \{t\}} \mu_{f_l \to x_i}(x_i) \tag{3}$$

where $\mathrm{ne}(i) \subseteq \{1, \ldots, d\}$ denotes the subset of factor nodes that are connected to variable node i. These messages are iteratively exchanged until a convergence criterion is met. A common choice is to stop if the variable configuration does not change for a given number of iterations (in the case of binary or categoral variables) or the objective function value changes less than some threshold. The final variable configuration is obtained by assigning to each variable the domain value that maximizes the summation of all the incoming messages for to the corresponding node [19,21]. An example of a FG and the max-sum algorithm are illustrated in Fig. 1.

A crucial issue in the practical use of max-sum is the derivation of efficient procedures to compute the messages. This is a non-trivial task, especially for the messages from functions to nodes, which involve a maximization that can be intractable. Devising efficient and compact messages is clearly linked to the structure of the FG, namely: the order of the factors, presence/absence of constraints, type of variables (binary/discrete/continuous).

2.4. Biclustering with factor graphs

This subsection briefly reviews the approaches in the literature that address biclustering by designing a FG [15,18,17]; this will help in understanding and putting in context the approach that we will propose below. We focus on [17,18], since [15] does not propose a novel FG for biclustering, but simply iteratively applies a FG-based approach to clustering rows or columns.

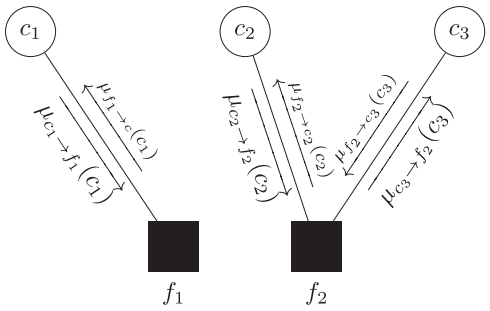

Fig. 1. A simple example of a factor graph and max-sum messages.

2.4.1. Biclustering affinity propagation

The approach introduced in Ref. [18], called *biclustering affinity propagation* (BAP), extends to the biclustering case the FG that underlies the *affinity propagation* (AP) clustering method proposed in Ref. [21].

In particular, authors of [18] proposes to directly group the *data matrix entries* (instead of whole rows or columns), ensuring that the obtained groups of entries correspond to biclusters: a subset of rows and a subset of columns. This intrinsically defines a topological constraint: the biclusters (after appropriate row and column permutations) must have a rectangular shape. Hence, the authors extend the FG underlying AP by introducing another constraint ensuring the result to be a sub-matrix. This is done by defining a 4-dimensional binary model with $n \times m \times n \times m$ variables, each encoding the exemplar choices between the points (variable $c_{i,j,t,k}$ is 1 if and only the entry (i,j) chooses (t,k) as exemplar), with factors representing the following three constraints:

(i) 1-of-N constraint (**I** factors): every entry must chose one, and only one, exemplar;
(ii) *exemplar consistency* (**E** factors): if an entry (i,j) chooses entry (t,k) as exemplar, (t,k) must choose itself;
(iii) *bicluster integrity* (**B** factors): if entries (i,j) and (t,k) choose (z,y) as exemplar, then (i,k) and (t,j) must also choose (z,y) as exemplar;

The resulting FG is very large: for a $n \times m$ data matrix, the model has n^2m^2 variables; moreover the model contains $n\,m$ **I** and **E** factors, plus n^3m^3 **B** factors. For example, for a 10×10 matrix, the model contains 10^4 variables and more than 10^6 factors. Moreover, the model contains many cycles, so that the max-sum algorithm is not effective. Because of that, the authors of [18] proposed a linear programming solution, which however did not allow them to analyse matrices larger than 10×10. The FG of this approach is illustrate in Fig. 2a (as in [18], the 4D model is encoded as a 3D model, where the third dimension encodes all the nm entries of the matrix).

2.4.2. Exemplar based biclustering

The second FG-based approach [17], herein referred to as *exemplar-based biclustering* (EB), shares many aspects with the above described BAP method, and it also yields an exemplar-based biclustering solution. The main difference concerns the description of the representative choices: while in BAP [18] the variables are binary, EB [17] uses integer-valued variables indexing the chosen exemplars. This leads to a much more compact FG: nm variables, instead of n^2m^2 (see Fig. 2b). The constraints encoded in the model are similar to those of BAP, but their encoding differs since the variable type is different. In detail:

(i) the 1-of-N constraint is intrinsically encoded in the variables, as their integer type forces each point to choose exactly one exemplar;
(ii) *exemplar consistency* is now imposed to every couple of variables (**g** factors);
(iii) *bicluster integrity* is imposed exactly as in the BAP model (**f** factors);

The resulting model is much more compact, with a drastically lower number of variables and factors. However, since the variables are (mn)-ary, the efficient messages update rules for the max-sum algorithm as proposed in AP [21] cannot be used here. In fact, the largest matrix analyzed in Ref. [17] with standard max-sum messages has size 10×10; for larger matrices, the authors had to resort to an approximate max-sum algorithm and a greedy approach.

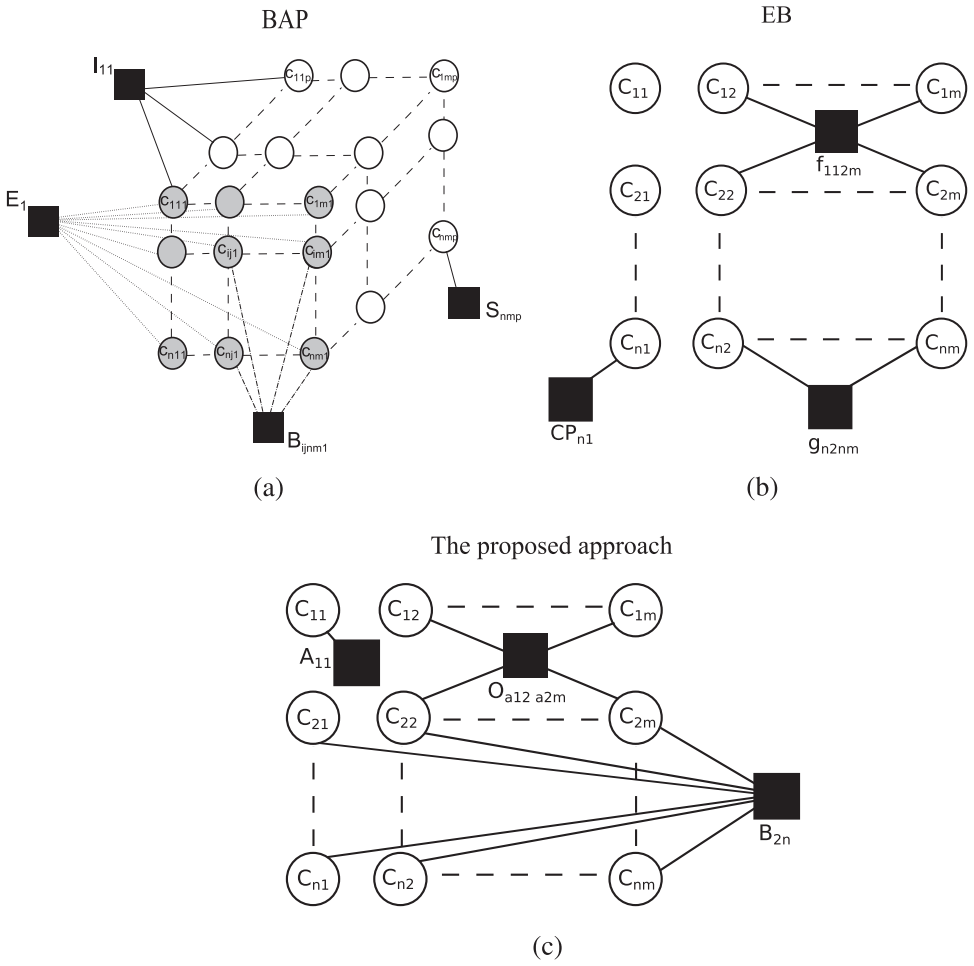

Fig. 2. Factor graphs for biclustering.

3. The proposed approach

Building upon the basic ideas proposed in Refs. [17,18], we design a *compact* and *binary* FG model for biclustering, for which we derive a fast and effective max-sum algorithm.

3.1. The ingredients

The proposed model uses 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 [25,26,13]. Once a bicluster in identified, it is masked in the data matrix (e.g., by replacing it with background noise [25] 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 if that entry belongs, or not, to the bicluster. We thus use binary variables (as in BAP), but a compact set thereof (as in EB). 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 ($c_{i,j} = 0$). 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, equivalently, 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. There are several possibilities to define the incoherence $I(a_{ij}, a_{tk})$ between two entries, depending on which kind of bicluster we are seeking (constant-valued, additive, multiplicative, or others [7]). The most straightforward option is to simply use a constant-type incoherence (as in BAP):

$$I(a_{ij}, a_{tk}) = (a_{ij} - a_{tk})^2. \tag{4}$$

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 Refs. [17,25]:

$$I(a_{ij}, a_{tk}) = (a_{ij} - a_{tj} + a_{tk} - a_{ik})^2. \tag{5}$$

In fact this choice directs the search towards additively coherent biclusters (i.e. 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 [7]).
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 and 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 some biology applications [5,7].

4. *The entry level "counts"*. For 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.

3.2. The formal model

Recall that the data matrix is denoted as $A = (a_{ij})$, $i \in N = \{1, \ldots, n\}$, for $j \in M = \{1, \ldots, m\}$, such that $a_{ij} \geq 0$. The goal is to find a bicluster A_{TK}, that is a sub-matrix of A containing the rows in $T \subseteq N$ and the columns in $K \subseteq M$. The proposed model, graphically sketched in Fig. 2c, is fully defined by the variables and factors next described.

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

$$A_{ij}(c_{ij}) = a_{ij}c_{ij} = \begin{cases} a_{ij}, & \text{if } c_{ij} = 1 \\ 0, & \text{otherwise,} \end{cases} \tag{6}$$

  encouraging the bicluster to contain entries of A with high activation level.
- Pairwise factors (one for each pair of entries) given by

$$O_{ij,tk}(c_{ij}, c_{tk}) = -I(a_{ij}, a_{tk})c_{ij}c_{tk}, \tag{7}$$

  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:

$$B_{jk}(c_{:j}, c_{:k}) = \begin{cases} 0, & \text{if} \left(\sum_i c_{ij} \right) \left(\sum_i c_{ik} \right) \left(\sum_i |c_{ij} - c_{ik}| \right) = 0 \\ -\infty, & \text{otherwise} \end{cases} \tag{8}$$

  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):

$$F(C) = \sum_{i=1}^{n}\sum_{j=1}^{m} A_{ij}(c_{ij}) + w \sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{t=1}^{n}\sum_{k=1}^{m} O_{ij,tk}(c_{ij}, c_{tk}) + \sum_{j=1}^{m}\sum_{k=1}^{m} B_{jk}(c_{:j}, c_{:k}). \tag{9}$$

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 Eq. (7)), discouraging incoherent points from being included in the solution. The relative strength of these two forces is controlled by parameter w.

3.3. Optimization

To maximize the objective function in Eq. (9) we adopt the max-sum algorithm. As mentioned in Section 2.2, that algorithm uses two types of messages (from variables to factors and from factors to variables), which are iteratively exchanged until a stable configuration is reached. The computation of these message is a crucial aspect of the algorithm, which may not be trivial due to the maximizations involved in the messages going from factors to variables (see Eq. (2)). Notice that, considering categorical variables with d possible label values and that a given factor depends on k variables, it is necessary to consider an exponential number d^k of configurations; this quickly becomes intractable even for moderate values of d and k. However, the fact that our proposed model uses binary variables and hard constraints allows deriving efficiently computable and compactly representable messages, as detailed next.

Observe that every message sent to a variable needs to be computed for all the possible configurations of that variable. The binary nature of the variables means that this can be represented in a very compact way: we can use a scalar message that expresses the difference between the values corresponding to the two possible variable allocations. Given that our FG includes three types of factors (A_{ij}, $O_{ij,tk}$, B_{jk}), we need to derive six types of messages (three in each direction).

The message from factor A_{ij} to variable c_{ij} is constant (equal to a_{ij}) and the opposite message is simply zero. The remaining four types of messages can be obtained after some mathematical manipulation (the full derivation is provided in the supplementary material), and are as follows.

1. From variable $c_{i,j}$ to factor $O_{ij,tk}$:

$$\psi_{ij}^{tk} = \sum_{\hat{tk}} \sigma_{ij}^{\hat{tk}} + \sum_{k} \eta_{ij}^{k} + \alpha_{ij}. \tag{10}$$

2. From variable $c_{i,j}$ to function $B_{i,k}$:

$$\beta_{ij}^{k} = \sum_{\hat{k}} \eta_{ij}^{\hat{k}} + \sum_{tk} \sigma_{ij}^{tk} + \alpha_{ij}. \tag{11}$$

3. From function $O_{ij,tk}$ to variable $c_{i,j}$:

$$\sigma_{ij}^{tk} = \max\Big[\min\big(wI(a_{ij}, a_{tk}), wI(a_{ij}, a_{tk}) + \psi_{tk}^{ij} \big), \min\big(-\psi_{tk}^{ij}, 0 \big) \Big]. \tag{12}$$

4. From function $B_{i,k}$ to variable $c_{i,j}$:

$$\eta_{ij}^{k} = \max\Big[\min(\Theta, I), \min(K, \Lambda) \Big]. \tag{13}$$

where

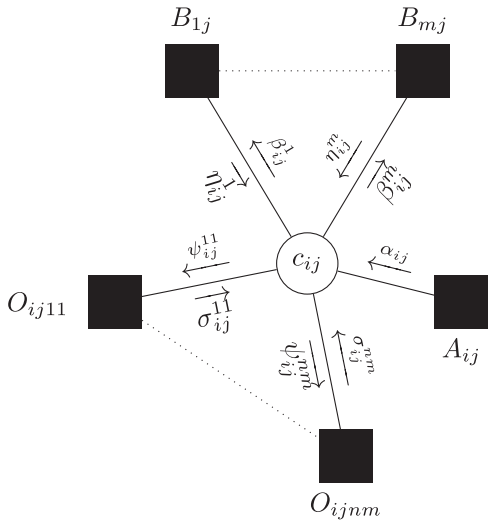

Fig. 3. Sketch of the factor graph showing the connections between one variable and all its factors.

$$\begin{aligned}\Theta &= \min\left(0, \beta_{ik}^{j}\right) + \sum_{\hat{i}} \max\left[\min(\beta_{\hat{i}j}^{k}, \beta_{\hat{i}j}^{k} + \beta_{\hat{i}k}^{j}), min(-\beta_{\hat{i}k}^{j}, 0)\right], I \\ &= \beta_{ik}^{j} + \min_{\hat{i}}(\max(0, -\beta_{\hat{i}j}^{k} - \beta_{\hat{i}k}^{j})), K = \min(0, -\beta_{ik}^{j}) \\ &\quad + \sum_{\hat{i}} \max\left[\min(0, -\beta_{\hat{i}k}^{j}), \min(\beta_{\hat{i}j}^{k}, \beta_{\hat{i}j}^{k} - \beta_{\hat{i}k}^{j})\right], \Lambda \\ &= \min_{t\neq i}\Bigg[\max(-\beta_{tk}^{j}, -\beta_{tk}^{j} - \beta_{tj}^{k}) + + \\ &\quad \sum_{l\neq t,i} (\max(\min(0, -\beta_{lj}^{k} - \beta_{lk}^{j}), \min(\beta_{lj}^{k}, -\beta_{lk}^{j})))\Bigg]\end{aligned}$$

and $\hat{i} \in N\backslash i$, $\hat{k} \in M\backslash k$ and $\hat{tk} = \{(1, 1), \ldots, (n, m)\}\backslash(t, k)$.

Fig. 3 shows a sketch of the FG connections for a given variable, and the messages exchanged. 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.

3.4. Computational complexity and scalability

Given an $n \times m$ data matrix, the model has $O(n^2 m^2)$ 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 50×50 matrices with exact max-sum update rules. We think this is an important step toward efficient FG-based biclustering, since previous methods solved up to 10×10 matrices. Although this promising result, the proposed approach cannot handle real matrices directly (e.g., the Yeast dataset has almost one million entries); in particular, the memory required by the σ messages for n genes and m conditions involves storing n^2m^2 values. To circumvent this difficulty, we adopted a scheme similar to that employed in Ref. [18]: 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 the (above mentioned) *affinity propagation* algorithm [21]; 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. (9): 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.[2]

4. Experimental evaluation

In this section we evaluate the proposed 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, also comparing it with the very recent AD^3 method [28], which in principle can provide better solutions but at a cost of slower convergence. Next, we evaluated the quality of the solutions provided by our approach in comparison with the other FG-based approaches [17,18], as well as other state-of-the-art techniques [13,35–38,25,39,40,17]. 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.

4.1. 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 [28]. Concerning the first aspect, it is known that max-sum is only guaranteed to converge in the absence of cycles [19,24], 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 Fig. 4, where it can be seen how it rapidly reaches its final value.

Concerning the second aspect, note that, as reported in previous sections, 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 [22,19,21,33,34]. However, it is important to assess how max-sum compares to other techniques, such as the recent ones based on dual decomposition [31,28,41,42,30,43]. In particular, here we compare max-sum with the recent AD^3 (*alternating directions dual*

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

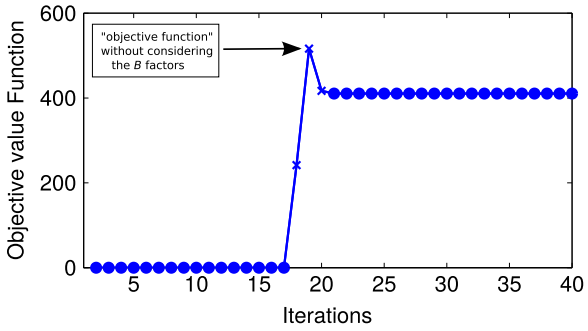

Fig. 4. Objective function evolution along the max-sum iterations. Each circle indicate a valid configurations (assignment respecting the constraints) and each cross indicates a non-valid configuration (the corresponding value of the objective function is $-\infty$ in these cases, thus we show the value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

decomposition [28][3]). 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 [28]. The authors of [28] 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 the type of constraints for which closed-form derivations (i.e. one-hot XOR, OR, OR-with-OUTPUT, negations, De Morgans laws, and AND-With-Output [28]). 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 submatrix and hence a bicluster.

In order to compare the two optimization techniques, we used the following synthetic dataset: (i) matrix A contains 20×20 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 [17]: *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 execution time, AD^3 turned out to be very slow in this problem, with a computational time 3 order of magnitude slower than that of our algorithm (as shown in Fig. 5b): 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, Fig. 5a 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.

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

4.2. Experiments on synthetic datasets

The proposed approach was compared with the two FG-based methods described in Section 2, BAP [18] and EB [17], using the same synthetic data generation method described in the previous subsection. In this case, we used 30 matrices for each noise level (for a total of 150 matrices), and each matrix has dimensions 50×50. 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 proposed in the corresponding papers, 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 proposed in Ref. [18], which groups results obtained on smaller matrices (here we used 5×5 matrices, with no overlap). Regarding similarity, we employed the negative of the Euclidean distance (as proposed in Ref. [18]) for the constant bicluster benchmark, whereas for the evolutionary bicluster benchmark, we adopted the negative of Eq. (5). [4]

Concerning EB, we used the authors' implementation.[5] of the greedy algorithm given in Ref. [17]. 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 max-sum method, the experimental details are the following:

- *Message scheduling*: even if different schedule are available [22], we adopt the most common approach of updating the messages in parallel, based on those from the previous iteration. For example, looking at the FG in Fig. 1, when computing $\mu_{f_2 \to c_3}(c_3)$ at time t, the values of $\mu_{c_2 \to f_2}(c_2)$ at time $t-1$ are used.
- *Convergence criteria*: the algorithm is stopped if the variable configurations do not change for 100 consecutive iterations.
- *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 Fig. 6, 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.

As shown in Fig. 6, the tested approaches provide similar performance on the synthetic benchmark, thus a statistical test is required to prove if the differences between the retrieved solutions are significant. We performed a paired T-test comparing other FG based algorithms against the one we propose. Specifically, in the plots, a full marker on the EBG (or BAP) curve indicates that the difference between such method and the proposed approach is statistically significant with significance level of 5%.

These results show that the proposed approach significantly outperforms the two approximate BAP and EB methods, especially for higher levels of noise, 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. (5) 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 (5) returns zero as coherence value, increasing the chance that those two

[4] This type of bicluster was not considered in Ref. .[18]
[5] Available at http://sist.shanghaitech.edu.cn/faculty/tukw/sdm11code.zip
[6] Parameter ranges are described in the code documentation.

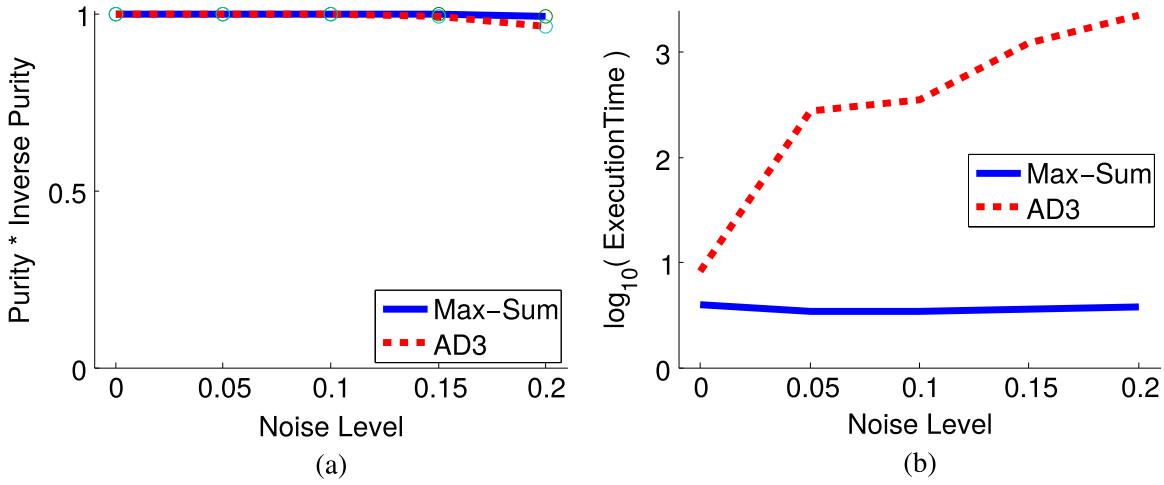

Fig. 5. The plot a shows a comparison of the Purity and Inverse Purity obtained by the two optimization algorithms on a dataset with increasing noise level; while plot b shows a comparison of the computational time occurred to obtained such results.

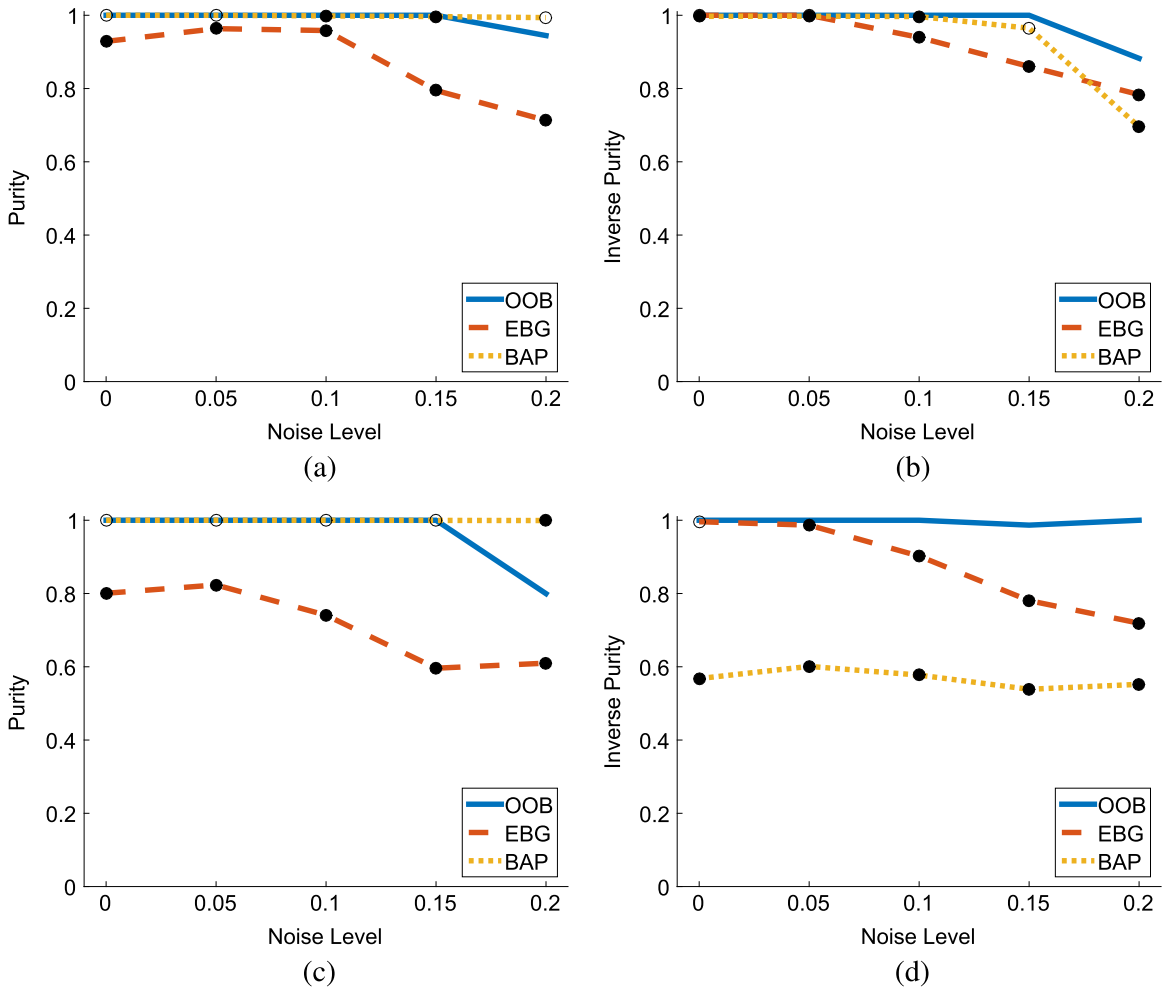

Fig. 6. Purity and inverse purity for matrices with constant and additively coherent (evolutive) biclusters.

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.

4.3. Gene expression data

The proposed algorithm was tested on two real gene expression datasets: the *Yeast* dataset [44] and the *Breast Tumor* dataset [5]. 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 behavior in a subset of

experiments. These experiments have been performed adopting the heuristic described in Section 3.4. We assess the performance on both datasets following the experimental protocol proposed in Ref. [17]: 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 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 Ref. [17], only the 100 largest biclusters (with a maximum overlap of 25%) were considered as part of the solution. Table 1 reports percentage accuracies obtained with our algorithm, for different significance levels, together with other state-of-the-art results from [17]. 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 [45,46], 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 Ref. [5]: 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 2, in comparison with all the state-of-the-art methods studied in Ref. [5]; we can conclude that our method sets a new state-of-the-art for this dataset.

Table 1
Results on Yeast dataset. Results for other approaches have been taken from Fig. 3 in Ref. [17]. Algorithms reference: Cheng and Church [25], ROCC [39], Bimax [40], EB [17].

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
Proposed	100	100	100	100	100

Table 2
Results on Breast tumor dataset. Results for other approaches have been taken from [5]. Algorithms reference: FABIA [13], ISA [35], Hierarchical [36], SAMBA [37], FLOC [38].

FABIA	ISA	Hiearc.	SAMBA	FLOC	Proposed
55%	63%	70%	73%	85%	**87.5%**

5. Algorithm variants

This section provides some guidelines on how it is possible to extend the proposed approach in order to obtain more specific models to be exploited where particular structures for the biclusters are required.

One of the advantages in using Factor Graphs is represented by the possibility of easily modifying the model by introducing or removing novel factors. As a possible extension, we firstly consider that biclustering has been largely employed to analyse temporal series in gene expression context with very good results (e.g., [48]). Concerning the analysis of gene expression temporal series, the goal is to obtain biclusters where the experimental conditions (generally represented by the data matrix columns) form a contiguous subset; and this is not a constraint involved in the basic setup of our method. The second advanced model we propose derives from the consideration that setting the parameter w in Eq. (9) may be not trivial in some situations; please note that this parameter regulates the biclusters dimensions. For this reason we introduce a novel class of factors in the model allowing the user to define a favoured size for the biclusters.

Although the model was not designed to analyse temporal series or to retrieve a prefixed dimension biclusters, by introducing a novel set of factors, such constraints can be included in the proposed approach, provided that the update rules of the max-sum messages can be computed efficiently. To devise this advanced model we resort to a recent class of factors widely known in literature as Tractable Higher Order Potentials (THOP) [47]. 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 [21,47].

5.1. Temporal series model

In this context we present an extended model devised for temporal series data which includes the *convex-set* potentials. Briefly, given a set of binary variables $x = \{x_1, \ldots, x_l\}$, the convex-set potential imposes that activated variables (variables equal to 1) must form a contiguous subset (i.e. no zeros between ones are allowed). This characteristic can be exploited in many biclustering contexts such as the analysis of temporal series.

Practically, to retrieve biclusters from the resulting FG some modifications in the messages update rules are needed. The update rules derivation for the THOP factors have been taken from [47]. Since messages going from variables to functions involve only summations, handling them is really simple. Messages γ_{ij} are the summation of all the messages incoming in c_{ij} except for ρ_{ij}. Regarding basic messages (ψ_{ij}^{tk} and β_{ij}^{k}) their summation will include γ_{ij}, as follow:

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

On the other hand, basic model messages going from factors to variables are not influenced by the introduction of such modules. New factor messages for the convex-set constraint as follows:

$$\rho_{ij} = MS(c_{1:j-1}, c_{j-1} = 1) + MS(c_{j+1:N}, c_{j+1} = 1) + - \max\Big(MS(c_{1:j-1}), MS(c_{j+1:N})\Big).$$

Considering ρ 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).

5.2. Preferred size model

To implement such model we resort to another class of THOP known as *cardinality potential*. 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 [8]). These potentials can be specified in different manners depending on the problem to solve (e.g., $\sum_i x_i = k$, $\sum_i x_i \neq k$, $\sum_i x_i > k$, $\sum_i x_i < k$, with $k \in \mathbb{N}$), here we present the general representation from which the more specific ones can be derived. The cardinality potential we present is a function $Q(\sum_i x_i)$ assigning a score to each possible cardinality value in $\{0, 1, 2, \ldots, l\}$, hence $Q: \mathbb{N} \rightarrow \mathbb{R}$.

Regarding messages updates, as for the temporal series model, the messages from variables to functions involve the summation of all the messages incoming in c_{ij} except for χ_{ij}; whereas basic

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

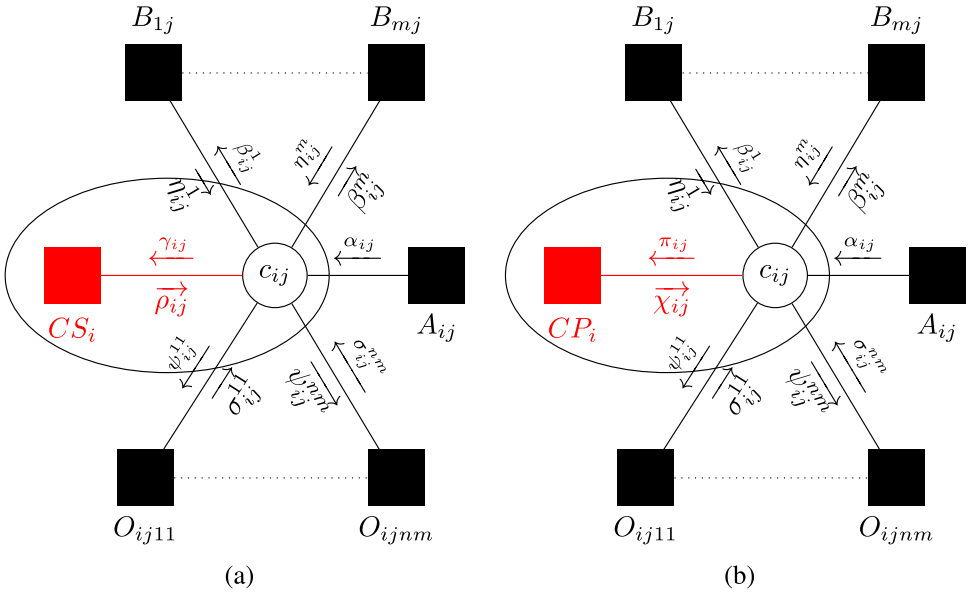

Fig. 7. Biclustering Factor Graph possible extensions. Two alternatives of the basic model where convex-set constraint (Fig. 7a) and the cardinality potential (Fig. 7b) are applied to every column. Such factors can be added on both rows or columns depending on the desired solutions.

messages will include χ_{ij}, as follow:

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

For the messages going from factors to variables the computation is slightly more complex than the temporal series model because different steps are needed. The detailed procedure is shown in Algorithm 1.

Algorithm 1. Computation of the χ_{ij}*messages*.

Require: Incoming messages $\pi_{i:}$.
1: 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
2: **for** $z \in \{0, \ldots, N\}$ **do**
3: $u_{-1}(z) = \sum_{z'=1}^{z} \left[\pi_{i_{z'}^*} + Q(z'-1) \right]$ 4:
$u_0(z) = \sum_{z'=0}^{z} \left[\pi_{i_{z'}^*} + Q(z') \right]$
5: $u_1(z) = \sum_{z'=0}^{z-1} \left[\pi_{i_{z'}^*} + Q(z'+1) \right]$
6: **end for**
7: **for** $z \in \{0, \ldots, N\}$ **do**
8: $s_1^L(z) = \max_{z' \in \{0,\ldots,z\}} c_1(z')$
9: $s_{-1}^R(z) = \max_{z' \in \{z,\ldots,N\}} c_{-1}(z')$
10: $s_0^L(z) = \max_{z' \in \{0,\ldots,z\}} c_0(z')$
11: $s_0^R(z) = \max_{z' \in \{z,\ldots,N\}} c_0(z')$
12: **end for**
13: $\chi_{ij}(0) = \max\left[s_0^L(r(j)-1), s_{-1}^R(r(j)+1) \right]$
14: $\chi_{ij}(1) = \max\left[s_1^L(r(j)-1), s_0^R(r(j)+1) \right]$
15: $\chi_{ij} = \chi_{ij}(1) - \chi_{ij}(0)$

Advanced models are presented in Fig. 7. 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.

6. Conclusions

In this paper, we proposed a novel biclustering method based on factor graph modeling and optimization tools. In particular, we propose an incremental approach where biclusters are sequentially identified, one at a time. To find each bicluster, we proposed a binary and compact factor graph depending on the data matrix entries, which is maximized via the max-sum algorithm. More specifically, we derived an approach to efficiently update the messages that constitute that algorithm, which we analyzed in terms of space and time complexity. The empirical evaluation reported in the paper showed that our approach favorably compares with the previous state-of-the-art methods on both synthetic datasets and real data, testifying for its practical significance.

Appendix A. Supplementary data

Supplementary data associated with this article can be found in the online version at http://dx.doi.org/10.1016/j.patcog.2016.08.033.

References

[1] R. Henriques, C. Antunes, S.C. Madeira, A structured view on pattern mining-based biclustering, Pattern Recognit. 48 (2015) 3941–3958.
[2] K.-O. Cheng, N.-F. Law, W.-C. Siu, Iterative bicluster-based least square framework for estimation of missing values in microarray gene expression data, Pattern Recognit. 45 (2012) 1281–1289.
[3] D. Yan, J. Wang, Biclustering of gene expression data based on related genes and conditions extraction, Pattern Recognit. 46 (2013) 1170–1182.
[4] H. Zhao, K.L. Chan, L.-M. Cheng, H. Yan, A probabilistic relaxation labeling framework for reducing the noise effect in geometric biclustering of gene expression data, Pattern Recognit. 42 (2009) 2578–2588.
[5] A. Oghabian, S. Kilpinen, S. Hautaniemi, E. Czeizler, Biclustering methods biological relevance and application in gene expression analysis, PLoS One 9 (2014) e90801.
[6] T. Pansombut, W. Hendrix, Z. Jacob Gao, B.E. Harrison, N.F. Samatova, Biclustering-driven ensemble of bayesian belief network classifiers for underdetermined problems, in: IJCAI Proceedings-International Joint Conference on Artificial Intelligence, volume 22, 2011, p. 1439.
[7] S. Madeira, A. Oliveira, Biclustering algorithms for biological data analysis a survey, IEEE Trans. Comput. Biol. Bioinforma. 1 (2004) 24–44.
[8] S. Dolnicar, S. Kaiser, K. Lazarevski, F. Leisch, Biclustering overcoming data dimensionality problems in market segmentation, J. Travel Res. 51 (2012) 41–49.
[9] A. Mukhopadhyay, U. Maulik, S. Bandyopadhyay, C.A.C. Coello, Survey of

multiobjective evolutionary algorithms for data mining Part II, Evolut. Comput. IEEE Trans. 18 (2014) 20–35.
[10] P.A. de Castro, F.O. de França, H.M. Ferreira, F.J. Von Zuben, Applying biclustering to text mining: an immune-inspired approach, in: Artificial Immune Systems, Springer, 2007, pp. 83–94.
[11] M. Kaytoue, V. Codocedo, A. Buzmakov, J. Baixeries, S.O. Kuznetsov, A. Napoli, Pattern structures and concept lattices for data mining and knowledge processing, in: Machine Learning and Knowledge Discovery in Databases, Springer, 2015, pp. 227–231.
[12] L. Lazzeroni, A. Owen, et al., Plaid models for gene expression data, Stat. Sin. 12 (2002) 61–86.
[13] S. Hochreiter, U. Bodenhofer, M. Heusel, A. Mayr, A. Mitterecker, A. Kasim, T. Khamiakova, S. Van Sanden, D. Lin, W. Talloen, et al., Fabia factor analysis for bicluster acquisition, Bioinformatics 26 (2010) 1520–1527.
[14] G. Getz, E. Levine, E. Domany, Coupled two-way clustering analysis of gene microarray data, Proc. Natl. Acad. Sci. U S A 97 (2000) 12079–12084.
[15] A. Farinelli, M. Denitto, M. Bicego, Biclustering of expression microarray data using affinity propagation, in: M. Loog, L. Wessels, M. Reinders, D. Ridder (Eds.), Pattern Recognition in Bioinformatics, volume 7036 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2011, pp. 13–24.
[16] J.A. Hartigan, Direct clustering of a data matrix, J. Am. Stat. Assoc. 67 (1972) 123–129.
[17] K. Tu, X. Ouyang, D. Han, V. Honavar, Exemplar-based robust coherent biclustering., in: SDM, SIAM, 2011, pp. 884–895.
[18] M. Denitto, A. Farinelli, G. Franco, M. Bicego, A binary factor graph model for biclustering, in: Structural, Syntactic, and Statistical Pattern Recognition, Springer, 2014, pp. 394–403.
[19] C.M. Bishop, et al., Pattern Recognition and Machine Learning, springer, New York, 2006.
[20] F. Kschischang, B. Frey, H.-A. Loeliger, Factor graphs and the sum-product algorithm, Inf. Theory IEEE Trans. 47 (2001) 498–519.
[21] B. Frey, D. Dueck, Clustering by passing messages between data points, Science 315 (2007) 972–976.
[22] S.M. Aji, R.J. McEliece, The generalized distributive law, Inf. Theory, IEEE Trans. 46 (2000) 325–343.
[23] L. O'Connor, S. Feizi, Biclustering using message passing, in: Advances in Neural Information Processing Systems, 2014, pp. 3617–3625.
[24] Y. Weiss, W. Freeman, Correctness of belief propagation in gaussian graphical models of arbitrary topology, Neural Comput. 13 (2001) 2173–2200.
[25] Y. Cheng, G.M. Church, Biclustering of expression data., in: Ismb, volume 8, 2000, pp. 93–103.
[26] A. Ben-Dor, B. Chor, R. Karp, Z. Yakhini, Discovering local structure in gene expression data the order-preserving submatrix problem, J. Comput. Biol. 10 (2003) 373–384.
[27] S. Gremalschi, G. Altun, Mean squared residue based biclustering algorithms, in: Bioinformatics Research and Applications, Springer, 2008, pp. 232–243.
[28] A.F. Martins, M.A. Figueiredo, P.M. Aguiar, N.A. Smith, E.P. Xing, Ad3 alternating directions dual decomposition for map inference in graphical models, J. Mach. Learn. Res. 16 (2015) 495–545.
[29] N. Gupta, S. Aggarwal, Mib using mutual information for biclustering gene expression data, Pattern Recognit. 43 (2010) 2692–2697.
[30] F.R. Kschischang, B.J. Frey, H.-A. Loeliger, Factor graphs and the sum-product algorithm, Inf. Theory IEEE Trans. 47 (2001) 498–519.
[31] B. Andres, J.H. Kappes, U. Köthe, C. Schnörr, F.A. Hamprecht, An empirical comparison of inference algorithms for graphical models with higher order factors using opengm, in: Pattern Recognition, Springer, 2010, pp. 353–362.
[32] N. Wiberg, H.-A. Loeliger, R. Kotter, Codes and iterative decoding on general graphs, Eur. Trans. Telecommun. 6 (1995) 513–525.
[33] A. Farinelli, A. Rogers, A. Petcu, N.R. Jennings, 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, International Foundation for Autonomous Agents and Multiagent Systems, 2008, pp. 639–646.
[34] H. Yedidsion, R. Zivan, A. Farinelli, Explorative max-sum for teams of mobile sensing agents, in: Proceedings of the 2014 International Conference on Autonomous Agents and Multi-agent Systems, AAMAS'14, International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 2014, pp. 549–556.
[35] J. Ihmels, S. Bergmann, N. Barkai, Defining transcription modules using large-scale gene expression data, Bioinformatics 20 (2004) 1993–2003.
[36] R.R. Sokal, A statistical method for evaluating systematic relationships, Univ. Kans. Sci. Bull. 38 (1958) 1409–1438.
[37] A. Tanay, R. Sharan, R. Shamir, Discovering statistically significant biclusters in gene expression data, Bioinformatics 18 (2002) S136–S144.
[38] J. Yang, H. Wang, W. Wang, P.S. Yu, An improved biclustering method for analyzing gene expression profiles, Int. J. Artif. Intell. Tools 14 (2005) 771–789.
[39] M. Deodhar, G. Gupta, J. Ghosh, H. Cho, I. Dhillon, A scalable framework for discovering coherent co-clusters in noisy data, in: Proceedings of the 26th Annual International Conference on Machine Learning, ACM, 2009, pp. 241–248.
[40] A. Prelic, S. Bleuler, P. Zimmermann, A. Wille, P. Bühlmann, W. Gruissem, L. Hennig, L. Thiele, E. Zitzler, Comparison of biclustering methods: a systematic comparison and evaluation of biclustering methods for gene expression data, Bioinformatics 22 (2006) 1122–1129.
[41] N. Komodakis, N. Paragios, G. Tziritas, Mrf optimization via dual decomposition: Message-passing revisited, in: Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on, IEEE, 2007, pp. 1–8.
[42] A. Globerson, T.S. Jaakkola, Fixing max-product: Convergent message passing algorithms for map lp-relaxations, in: Advances in neural information processing systems, 2008, pp. 553–560.
[43] V. Jojic, S. Gould, D. Koller, Accelerated dual decomposition for map inference, in: Proceedings of the 27th International Conference on Machine Learning (ICML-10), 2010, pp. 503–510.
[44] A.P. Gasch, P.T. Spellman, C.M. Kao, O. Carmel-Harel, M.B. Eisen, G. Storz, D. Botstein, P.O. Brown, Genomic expression programs in the response of yeast cells to environmental changes, Mol. Biol. Cell 11 (2000) 4241–4257.
[45] S. Rogers, M. Girolami, C. Campbell, R. Breitling, The latent process decomposition of cdna microarray data sets, Comput. Biol. Bioinform. IEEE/ACM Trans. 2 (2005) 143–156.
[46] M. Bicego, P. Lovato, A. Perina, M. Fasoli, M. Delledonne, M. Pezzotti, A. Polverari, V. Murino, Investigating topic models' capabilities in expression microarray data classification, IEEE/ACM Trans. Comput. Biol. Bioinform. (TCBB) 9 (2012) 1831–1836.
[47] D. Tarlow, I.E. Givoni, R.S. Zemel, Hop-map: Efficient message passing with high order potentials, in: International Conference on Artificial Intelligence and Statistics, 2010, pp. 812–819.
[48] S.C. Madeira, A.L. Oliveira, 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, Springer Berlin Heidelberg, Berlin, Heidelberg, 2005, pp. 39–52.

Matteo Denitto is a PhD Student at the University of Verona. His research interests involve biclustering, graphical model and factor graphs. Matteo Denitto received his bachelor and master degree in Bioinformatics from the University of Verona in 2011 and 2013 respectively.

Alessandro Farinelli is Associate Professor at University of Verona, Computer Science Department. His research interests comprise theoretical and practical issues related to the development of Artificial Intelligent Systems applied to robotics. He participated in several national and international research projects in the broad area of Artificial Intelligence for robotic systems.

Mario A.T. Figueiredo received E.E., M.Sc., and pH.D. degrees in electrical and computer engineering from Instituto Superior Tecnico, University of Lisbon, where he is now a full Professor. His interests include statistical pattern recognition, machine learning, and computer vision. He is a Fellow of the IEEE and of the IAPR.

Manuele Bicego is an Assistant Professor at the University of Verona. His research interests include pattern recognition, bioinformatics and computer vision. He actively participated to different private, national and EU-funded research projects. He served as Guest Editor of the special issue of the Pattern Recognition journal on "Similarity-based Pattern Recognition".