# A Binary Factor Graph Model for Biclustering

Matteo Denitto, Alessandro Farinelli, Giuditta Franco, and Manuele Bicego

University of Verona, Department of Computer Science, Verona, Italy

**Abstract.** Biclustering, which can be defined as the simultaneous clustering of rows and columns in a data matrix, has received increasing attention in recent years, particularly in the field of Bioinformatics (e.g. for the analysis of microarray data). This paper proposes a novel biclustering approach, which extends the Affinity Propagation [1] clustering algorithm to the biclustering case. In particular, we propose a new exemplar based model, encoded as a binary factor graph, which allows to cluster rows and columns simultaneously. Moreover, we propose a linear formulation of such model to solve the optimization problem using Linear Programming techniques. The proposed approach has been tested by using a well known synthetic microarray benchmark, with encouraging results.

## 1 Introduction

Unsupervised learning, also known as *clustering*, is an active and historically fecund research area, which offers a wide range of solution techniques [2]. In recent years, the interest of the research community has been focused also on a particular kind of clustering problems, the so-called *biclustering*, also known, in other scenarios, as co-clustering. This term encompasses a large set of techniques generally aimed at "performing simultaneous row-column clustering" [3].

Bi-clustering techniques have been applied in different scenarios, such as document analysis [4], scene categorization [5], and, most importantly, expression microarray data analysis – see the reviews [3,6,7]. In this last scenario, the - starting point is a matrix whose rows and columns represent genes and experiments, respectively. Each entry measures the expression level of a particular gene in a particular experiment. The classical analysis in this scenario is to cluster genes, with the aim of discovering which genes show the same behavior over all the experiments – this permitting the discovery of co-regulation mechanisms. However, a more interesting question can be raised: are there genes that share similar expression *only in a certain subset of experiments*? Addressing this issue, which can not be faced using a standard clustering approach, can provide invaluable information to biologists, and represents the main goal of biclustering approaches.

Different biclustering techniques have been proposed in the past [3,6,7], each one characterized by different features, such as computational complexity, effectiveness, interpretability and optimization criterion . Many of such previous approaches are based on the idea of adapting a given clustering technique to the

P. Fränti et al. (Eds.): S+SSPR 2014, LNCS 8621, pp. 394–403, 2014.

biclustering problem, for example by repeatedly performing experiments and genes clustering [8,9].

This paper follows the above-described research trend, and proposes a novel biclustering algorithm, which extends and adapts to the biclustering scenario the well known Affinity Propagation (AP) clustering algorithm [1]. This technique, which is based on the idea of iteratively exchanging messages between data points until a proper set of representatives (called exemplars) are found, has shown to be very effective (in terms of clustering accuracy) and efficient (due to its fast learning algorithm) in many different application scenarios, including image analysis, gene detection and document analysis [1]. In Affinity Propagation the clustering problem is formulated as an objective function and a set of constraints; the objective function summarizes the intracluster-similarity and the constraints guide the grouping of the points to a valid solution. Specifically, the objective function and the constraints are encoded as a binary factor graph [10], and the objective function is optimized by using the max-sum message passing algorithm [1,10].

Even if some variants of the AP approach have been applied to the microarray scenario – see for example [11,12] – its use in the biclustering context remains somehow unexplored, with few papers recently published (such as [9], and [13]). In particular, in [9] the AP model is used as the clustering module in an iterative rows and columns clustering scheme [8]: however no modifications to the basic AP model has been introduced, which is still used as a standard clustering method. In contrast, [13] proposes an exemplar-based strategy to find biclusters. However, while such approach shares many similarities with AP (e.g., it is exemplar-based and encodes the problem as a factor graph), a crucial difference is that the proposed factor graph is not binary thus drifting away from the spirit of the original AP scheme, which exploits the binary nature of the factor graph to derive efficient and fast update messages [14].

In this paper we propose an extension of the Affinity Propagation model, which i) is based on a binary factor graph, and ii) directly performs biclustering. In particular we extend the AP model in two ways: i) we consider as datapoints to be analysed the single entries of the input data matrix, instead of the classical row/column vector; ii) we add to the model a constraint which forces points belonging to the same cluster to represent a valid bicluster (namely *all* points of a subset of rows and columns). Given the new factor graph, a possible solution to optimize the objective function is to resort to the max-sum algorithm [1,10]. However, given the high number of cycles present in the factor graph, the max-sum algorithm is likely to produce poor quality solutions [15]. Therefore we derived an alternative linear formulation of the optimization problem, and use Linear Programming techniques to find the optimal solution of our model. Finally, while the space complexity of the model and the time complexity of the algorithm are both polynomial in the number of entries of the data matrix, the number of variables and constraints that our model introduces is very large (i.e., $O(n^2m^2)$ variables and $O(n^3m^3)$ functions for an input matrix with $n$ rows and $m$ columns). Hence, storing our model for typical biclustering matrices

(which can contain hundreds of rows/columns) is an issue. Consequently, we derived an aggregation methodology, which groups results obtained on smaller matrices: this allows the evaluation of the proposed approach on a standard expression microarray benchmark [6]. Obtained results confirm the potentials of the proposed method.

The remainder of paper is organized as follows: Sect. 2 presents Affinity Propagation, the starting point of our model; the proposed approach is then described in Sect. 3 and Sect. 4, whereas the experimental evaluation is given in Sect. 5; finally Sect. 6 concludes the paper.

## 2 Affinity Propagation

Affinity Propagation (AP) is a well known clustering technique recently proposed by Frey and Dueck [1]. 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 [1].

The main idea behind AP is to perform clustering by finding a set of *exemplar points* that best represent the whole data set. This is obtained by representing the input data as a factor graph [16]: a bipartite graph that encodes an objective function as an aggregation (e.g., a sum) of functions (typically called factors). In the graph, the nodes (circles) define the data points and the factors (squares) are functions defined over a subset of nodes – for details please refer to [10]. The objective function is then optimized by running an iterative message passing approach, which, in the typical task of maximizing a sum of functions, is the max-sum algorithm [10].

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 define 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:

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

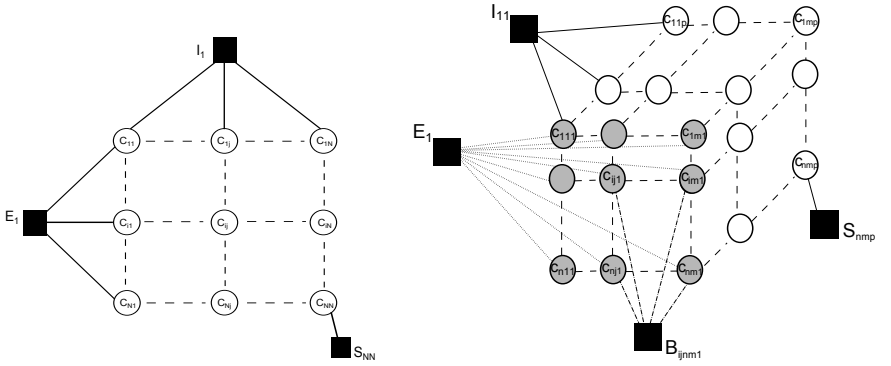
   where $n$ is the number of the points;
2. *Exemplar consistency constraint*: if a point is chosen as an exemplar by some other data point, it must choose itself as an exemplar. This constraint avoids

circular choices ("a" chooses "b", "b" chooses "c", "c" chooses "a") and can be represented by a function $E$ over $n$ nodes:

$$E_j = \begin{cases} -\infty, & \text{if} \quad c_{jj} = 0 \quad \text{and} \quad \sum_{i=1}^{n} c_{i,j} \geq 1 \\ 0, & otherwise \end{cases} \tag{2}$$

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 1(a) reports the factor graph used in AP.




(a) (b)

**Fig. 1.** Factor Graph for Affinity Propagation (a) and the proposed Factor Graph for Biclustering (b)

The objective function expressed by the AP factor graph is the sum of all the factors, i.e., the constraints expressed in Equations (1) and (2) and the sum of all similarity functions $S(i,j)$ which are defined as the similarity value $s_{i,j}$ multiplied by the variables $c_{i,j}$.

$$F = \sum_{i=1}^{n} \sum_{j=1}^{n} s_{ij} \cdot c_{ij} + \sum_{i=1}^{n} I_i + \sum_{i=1}^{n} E_j \tag{3}$$

## 3 The Proposed Approach

In this section the proposed approach is presented. In general terms, given a data matrix $D = (d_{ij})_{i \in N, j \in M}$, with $N$ set of rows ($|N| = n$) and $M$ set of columns ($|M| = m$), a bicluster $B = (d_{ij})_{i \in T, j \in K}$ is a submatrix of $D$, for $T \subseteq N$ and $K \subseteq M$, which meets specific spatial constraints ruled by a certain similarity criterion. Here we assume that different biclusters do not overlap[1].

[1] i.e. each element of the data matrix must belong to a unique bicluster.

In our approach, instead of considering as basic elements the rows and the columns, we directly consider the single entries of the input data matrix. Starting from $\{d_{ij}\}_{i\in N, j\in M}$, we look for biclusters as sets of "coherent" entries of the matrix respecting the specific spatial constraint. To obtain this, we re-define the factor graph of Affinity Propagation: in particular, we have one variable for each pair of entries of the data matrix $D$ to encode the exemplar choice; moreover, we introduce a constraint to ensure that points that belong to the same cluster represent a bicluster. In what follows, we define our model, specifying the variables, the constraints and the objective function, and motivate the use of an LP optimization approach.

### 3.1 The Model

**Variables.** Our goal is to cluster the single entries of the data matrix: therefore we encode the exemplar chosen by each entry of the data matrix $D$ with a four-dimensional Boolean matrix, where an entry $C(i, j, t, k) = c_{ijtk}$ is 1 if the point in position $(i, j)$ of the matrix chooses $(t, k)$ as its exemplar. For reasons which will be clearer later, we replace the indices of the second point with a single value $(z = 1, 2, 3, \cdots, n \cdot m)$ obtaining a three-dimensional structure $C(i, j, z)$; again, a variable $c_{ijz}$ is set to 1 if the point $(i, j)$ chooses the point $z$ as its exemplar. As in Affinity Propagation, this choice is based on a certain similarity matrix $S$, which now encodes the similarities between every pair of entries $(i, j)$ and $(t, k)$ of the input data matrix. As for $C$, we rearrange this four-dimensional matrix in a three dimensional one $S(i, j, z)$.

**Functions.** Following Affinity Propagation, we include in our model the constraint $I_{ij}$ (which is similar to (1) and encodes that one data entry should choose only one exemplar) and $E_z$ (which is similar to (2) and encodes that if $c_{i,j,z} = 1$ then $c_{\hat{i},\hat{j},z} = 1$, where $\hat{i}$ and $\hat{j}$ are the indices that correspond to $z$), which guarantee valid variable assignments. Next, we introduce an extra constraint, which ensures that the entries of the matrix which are in the same cluster do represent a bicluster. In this perspective, we observe that, given a certain value $z$, the bidimensional matrix

$$C(:,:,z) = \begin{bmatrix} c_{11z} & c_{12z} & \cdots & c_{1mz} \\ c_{21z} & c_{22z} & \cdots & c_{2mz} \\ \vdots & \vdots & \ddots & \vdots \\ c_{n1z} & c_{n2z} & \cdots & c_{nmz} \end{bmatrix} \quad \text{with } 1 \leq z \leq n \cdot m \tag{4}$$

immediately summarizes the relation between all the entries of the matrix and the entry $z$: in particular, $c_{ijz} = 1$ indicates that $(i, j)$ has chosen $z$ as its exemplar. Now, the constraints $I_{ij}$ and $E_z$ ensure that all the points in a given cluster had chosen the same exemplar, hence every matrix $C(:,:,z)$ represents a potential bicluster. However, to be a valid 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).

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:

$$B_{ijtkz} = \begin{cases} -\infty, & \text{if} \quad c_{ijz} = 1, c_{tkz} = 1 \quad \text{and} \quad c_{ikz} \cdot c_{tjz} = 0 \\ 0, & otherwise \end{cases} \tag{5}$$

Notice that the function B is defined, for every sheet $z$, on all the possible pairs of points $(i,j)$ and $(t,k)$.

**Objective Function.** Given the variables and the constraints above described – represented in Fig. 1(b) – we can now write the objective function, defined by the sum of the intra-biclusters similarity (via the matrix $C$ and $S$) and the constraints ($I$, $E$, and $B$):

$$F = \sum_{i,j,z} c_{ijz} \cdot s_{ijz} + \sum_{i,j} I_{ij} + \sum_{z} E_z + \sum_{z,i,j,t,k} B_{ijtkz} \tag{6}$$

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

### 3.2 Optimization of the Objective Function

Now, there are many possible approaches to maximize the objective function expressed by the factor graph in Fig. 1(b). In AP the binary nature of the nodes in graph is exploited to calculate an approximation of the maximum through the max-sum algorithm [1]. 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 approximated maximization algorithms degrade in such conditions [15]. Therefore we follow an alternative route, giving a linear formulation of the objective function, and using linear programming (LP) techniques [17] to find the optimal variable assignment. In general, LP approaches maximize/minimize an objective function where the constraints defined on the data points are all linear [17]. In the objective function (6), the first three addends can be easily written in a linear form; in the following we will show how to transform the biclustering integrity constraint (5) into a linear set of constraints.

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 "+" denotes 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:

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

this has to be done for all the 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.

Let us analyse the complexity of the proposed approach. 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 $O(m^3n^3)$. Even if 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. About time complexity, an Integer Programming problem is exponential in the number of constraints (in the worst case). Anyway, there are many well established methods which provide, on average, time satisfactory solutions. To overcome the scalability issue we run our algorithm on smaller matrices, extract biclusters and devise an aggregation algorithm to find biclusters in the original data matrix. We describe such aggregation algorithm in next Section.

## 4 Aggregation of Biclusters

Let a kernel be a window glass selecting a sub-matrix, we start by analyzing the data matrix by means of a fixed dimension kernel, which is shifted along the matrix, with no overlap. For every kernel, the optimal solution is retrieved (using our model and the LP approach). Once the whole matrix has been analyzed, the set of biclusters is then processed in three steps:

1. we apply a clustering algorithm on the exemplars retrieved in the different kernels, to partition the set of biclusters in groups of biclusters with coherent values. Here we adopt as a clustering algorithm the original Affinity Propagation method.

2. for every group of biclusters, we perform a hierarchical agglomerative grouping which, starting from single biclusters, repeatedly joins together the most similar groups of biclusters. Similarity between two groups of biclusters is defined as the number of rows and columns that they share – when the similarity of the nearest group is zero (no overlap) the algorithm stops. In other words, we perform a classical agglomerative clustering of biclusters by using as similarity the degree of column/rows overlap. Every group in the final partition now represents a set of biclusters with no row/column overlap with the other groups.
3. we 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.

Notice that, the third step is necessary because merging biclusters may not produce a bicluster as result. A possible alternative would be to merge only pairs of biclusters that result in a bicluster, however by so doing we would not obtain large biclusters given by the simultaneous merge of $k$ biclusters (where $k > 2$). Having described our approach, we now turn to the empirical evaluation.

## 5 Results

The methodology proposed in this paper has been tested on a set of synthetic matrices which represent a classical benchmark in the microarray scenario [6]: such set comprises synthetic expression matrices, perturbed with different schemes[2]. 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 biclustering has been assessed with the so-called *Gene Match Score* [6]: the average bicluster *relevance* reflects to what extent the generated biclusters represent a true bicluster in gene dimensions, and the average bicluster *recovery* quanties how well each of the true biclusters is recovered by the biclustering algorithm (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 [9]), which allows to retrieve only constant value biclusters. As in the original Affinity Propagation model, a proper setting of the preferences (namely the self similarities) is crucial: in our experiments we found that a good choice is represented by the first integer number below the median (which represents the standard setting [1]). The Linear Programming model was implemented and resolved using CPLEX (version 12.4).

Figures 2(a) and 2(b) report the Gene Match Scores (the recovery and the relevance values respectively – see [6]) for different levels of noise and for different dimensions of the kernel, averaged over the different repetitions (also standard deviations are displayed). As expected the approach provides better solutions as

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

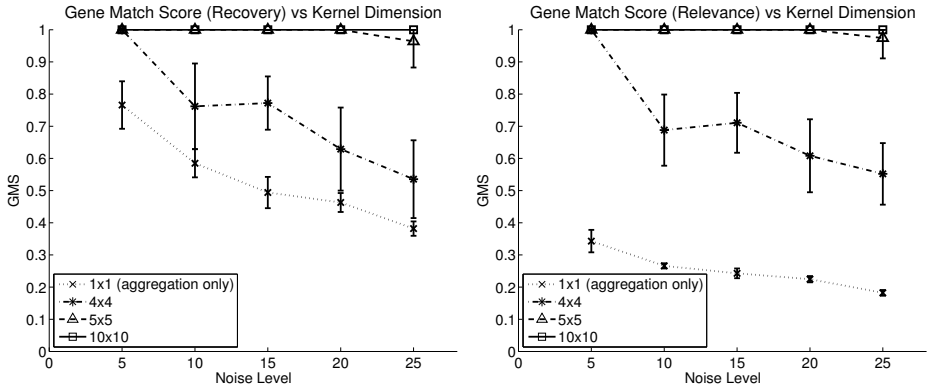



**Fig. 2.** Results for the proposed approach: (a) recovery and (b) relevance – for further information we refer to [6]

the kernel dimension increases. Please note that when using the [1x1] kernel only the aggregation algorithm described in Section 4 is employed (every data point is in its own bicluster). As we can see in Fig.2, increasing the noise completely corrupts the performances of the aggregation algorithm. Notice that, obtained results are competitive with other state of the art approaches (see figure 2 in [6], figure 1 in [18] or figure 3 in [9]), confirming the potentialities of the proposed approach.

## 6 Conclusions

In this paper we propose a novel model, inspired by Affinity Propagation [1], to retrieve biclusters from a data matrix. A key innovative element of our approach is to analyze directly the entries of the data matrix, instead of considering whole rows and columns, and to use Linear Programming techniques for computing the optimal solution [17]. The space/time complexity of the model does not allow to run our approach on typical biclustering problems, hence we partition the original data matrix in small kernels and analyse each such kernel with our approach. We then propose an aggregation approach to reconstruct the original biclusters. We evaluate our approach on standard benchmarking datasets for biclustering [6], and results show that the method is competitive with respect to other state of the art approaches.

Future work in this area includes two main research directions: first, investigate possible extensions of the approach to reduce the complexity of the data representation model, second to test the approach on real biological data sets, hence assessing the practical significance of the approach.

## References


1. Frey, B., Dueck, D.: Clustering by passing messages between data points. Science 315, 972–976 (2007)
2. Jain, A., Murty, M., Flynn, P.: Data clustering: a review. ACM Computing Surveys 21, 264–323 (1999)
3. Madeira, S., Oliveira, A.: Biclustering algorithms for biological data analysis: a survey. IEEE transactions on Computational Biology and Bioinformatics 1, 24–44 (2004)
4. Dhillon, I.: Coclustering documents and words using bipartite spectral graph partitioning. In: Proc. Int. Conf. on Knowledge Discovery and Data Mining, pp. 269–274 (2001)
5. Irie, G., Liu, D., Li, Z., Chang, S.F.: A bayesian approach to multimodal visual dictionary learning. In: Proc. Int. Conf on Computer Vision and Pattern Recognition, pp. 329–336 (2013)
6. 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 (2006)
7. Flores, J.L., Inza, I., Larrañaga, 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)
8. 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)
9. Farinelli, A., Denitto, M., Bicego, M.: Biclustering of expression microarray data using affinity propagation. In: Loog, M., Wessels, L., Reinders, M.J.T., de Ridder, D. (eds.) PRIB 2011. LNCS, vol. 7036, pp. 13–24. Springer, Heidelberg (2011)
10. Bishop, C.: Pattern Recognition and Machine Learning. Springer-Verlag New York, Inc. (2006)
11. Bayá, A., Granitto, P.: Clustering gene expression data with a penalized graph-based metric. BMC Bioinformatics 12 (2011)
12. Kiddle, S., Windram, O., McHattie, S., Mead, A., Beynon, J., Buchanan-Wollaston, V., Denby, K., Mukherjee, S.: Temporal clustering by affinity propagation reveals transcriptional modules in arabidopsis thaliana. Bioinformatics 26(3), 355–362 (2010)
13. Tu, K., Ouyang, X., Han, D., Honavar, V.: Exemplar-based robust coherent biclustering. In: SDM, pp. 884–895. SIAM (2011)
14. Givoni, I., Frey, B.: A binary variable model for affinity propagation. Neural Computation 21(6), 1589–1600 (2009)
15. Weiss, Y., Freeman, W.: Correctness of belief propagation in gaussian graphical models of arbitrary topology. Neural Computation 13(10), 2173–2200 (2001); cited By (since 1996) 168
16. Kschischang, F., Frey, B., Loeliger, H.A.: Factor graphs and the sum-product algorithm. IEEE Transactions on Information Theory 47(2), 498–519 (2001)
17. Dantzig, G.: Linear Programming and Extensions. Princeton University Press (August 1963)
18. Bicego, M., Lovato, P., Ferrarini, A., Delledonne, M.: Biclustering of expression microarray data with topic models. In: Int. Conf. on Pattern Recognition (ICPR 2010), pp. 2728–2731 (2010)