SPARSITY-COGNIZANT OVERLAPPING CO-CLUSTERING FOR BEHAVIOR INFERENCE IN SOCIAL NETWORKS

Hao Zhu[†]*, Gonzalo Mateos*[†]*, Georgios B. Giannakis*[†]*, Nicholas D. Sidiropoulos*[*]*, and Arindam Banerjee*[†]

[†]University of Minnesota, 200 Union St. SE, Minneapolis, MN 55455, USA
[*]Technical University of Crete, Dept. of ECE, 73100 Chania, Crete, Greece

ABSTRACT

Co-clustering can be viewed as a two-way (bilinear) factorization of a large data matrix into dense/uniform and possibly overlapping submatrix factors (co-clusters). This combinatorially complex problem emerges in several applications, including behavior inference tasks encountered with social networks. Existing co-clustering schemes do not exploit the fact that overlapping factors are often sparse, meaning that their dimension is considerably smaller than that of the data matrix. Based on plaid models which allow for overlapping submatrices, the present paper develops a sparsity-cognizant overlapping co-clustering (SOC) approach. Numerical tests demonstrate the ability of the novel SOC scheme to globally detect multiple overlapping co-clusters, outperforming the original plaid model algorithms which rely on greedy search and ignore sparsity.

***Index Terms*—** Clustering, overlapping co-clustering, sparsity, plaid models.

1. INTRODUCTION

The problem of co-clustering amounts to *simultaneous* clustering of a set of objects (samples) and the set of their attributes (features) into classes according to some similarity criterion; see e.g., the tutorial paper [1] and references therein. Given a data matrix with rows representing different samples and columns representing features, co-clustering reduces to finding dense, approximately constant-valued *submatrices* (see Fig. 1). The co-clustering problem is NP-hard (by reduction to ordinary clustering as in k-means), yet there is growing interest in efficient suboptimal co-clustering algorithms due to their importance in a gamut of timely applications in biostatistics, social network analysis, recommendation systems, and telecommunication networks.

One of the major challenges of social network analysis is identification of cohesive subgroups of actors within a network [2]. When the relational data is represented in the form of a *sociomatrix*, unveiling these subgroups becomes a co-clustering problem. Recently, co-clustering techniques have been also utilized for the purpose of understanding and analyzing the behavioral characteristics of Internet traffic [3]. By performing an orthogonal nonnegative matrix factorization of the adjacency matrix of the so-termed *traffic activity graphs* (TAGs), dominant host groups with strong interactions can be identified. However, orthogonality in [3] leads to non-overlapping co-clusters, a restrictive constraint for Internet traffic modeling and many other types of applications. In fact, most of the existing co-clustering algorithms, e.g., the ones in [1], consider only the case of non-overlapping clusters.

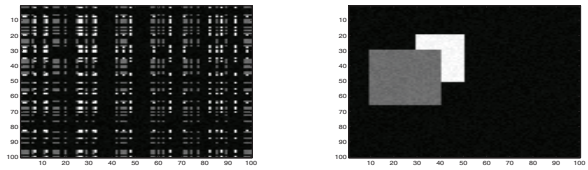

Fig. 1. Data matrix with latent overlapping co-clustering structure (left); and ideal co-clustering algorithm output where the underlying structure is revealed after suitable row/column permutations (right).

In the context of revealing interpretable biological structure in gene expression microarray data, the plaid model was put forth in [4] and is arguably amongst the most flexible co-clustering techniques. In this model, co-clusters or submatrices are referred to as *layers* and are allowed to overlap, while their union does not necessarily cover the whole data set. To fit the plaid model, a heuristic and greedy iterative algorithm was developed in [4] that includes layers one at a time until no more significant layers can be extracted. Further algorithmic improvements were reported in [5].

As data sets grow large, row/column co-cluster membership vectors are highly *sparse*, since only a few of the rows and columns determine each of the interesting submatrices sought. Incorporation of sparsity as prior information is expected to yield improved performance, encouraging parsimonious models that are more interpretable and informative. Even though sparsity-encouraging co-clustering methods are well motivated, they remain so far relatively unexplored.

Motivated by these considerations, the present paper develops a novel approach for sparsity-cognizant overlapping co-clustering (SOC). The resultant SOC algorithm builds on the plaid model and inherits its flexibility in modeling *overlapping* clusters. It encourages *sparsity* by appropriately modifying the plaid estimation criterion, through ℓ_1-norm regularization of the binary row/column (layer) co-cluster membership vectors. Estimation of the model parameters is based on an alternating block-minimization algorithm, which performs *simultaneously* cross-layer optimization instead of the greedy layer-by-layer strategy in [4] and [5]. On a per iteration basis, the algorithm: i) solves a sequence of ordinary least-squares (LS) problems to refine the layer levels; and ii) fits a Lasso-type of model with binary constraints to determine which of the rows and columns of $\mathbf{Y}$ are assigned to each layer. Numerical experiments with synthetic data show that the proposed scheme can outperform

This work was supported by the NSF grants CCF-0830480 and ECCS-0824007; and also through collaborative participation in the Communications and Networks Consortium sponsored by the U. S. Army Research Laboratory under the Collaborative Technology Alliance Program, Cooperative Agreement DAAD19-01-2-0011. The U. S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation thereon.

the baseline plaid model, highlighting its potential for real Internet traffic and sociomatrix data analysis. Diverse experiments with real data will be provided in the journal version [6].

2. PROBLEM STATEMENT AND PLAID MODELS

Consider an $n \times p$ matrix $\mathbf{Y}$ induced by two groups of interacting nodes $\{\mathcal{A}_i\}_{i=1}^n$ and $\{\mathcal{B}_j\}_{j=1}^p$, with the (i,j)-th entry $Y_{ij} \in \mathbb{R}$ measuring the strength of the relationship between nodes $\mathcal{A}_i$ and $\mathcal{B}_j$. As will become clear later on, the term 'node' should be understood as an abstract entity, possibly representing actors in a sociological study, or gene/sample descriptors in a microarray experiment. The two groups of nodes are not necessarily different, while simple dichotomous relationships can be captured through binary-valued matrices $\mathbf{Y}$.

For example, in the identification of cohesive social subgroups, the two groups coincide and $\mathbf{Y}$ becomes a sociomatrix, i.e., the $n \times n$ adjacency matrix of the graph induced by the set of actors $\{\mathcal{A}_i\}_{i=1}^n$ [2]. In this case, $Y_{ij} = 1$ if the actors i and j are related, and 0 otherwise. Another example is the traffic activity graph (TAG) for Internet traffic flow analysis. There, $\{\mathcal{A}_i\}_{i=1}^n$ represents the set of hosts within a local area network (LAN) engaged in communications with outside hosts $\{\mathcal{B}_j\}_{j=1}^p$. The value Y_{ij} tracks the traffic flow between $\mathcal{A}_i$ and $\mathcal{B}_j$ during a given time interval; or even simpler, $Y_{ij} = 1$ if there exists traffic, and 0 otherwise. Last but not least, $\mathbf{Y}$ can represent the DNA microarray data obtained from genes $\{\mathcal{A}_i\}_{i=1}^n$ and samples $\{\mathcal{B}_j\}_{j=1}^n$, where Y_{ij} measures the expression level with which gene $\mathcal{A}_i$ is expressed in sample $\mathcal{B}_j$. Relying on the data in $\mathbf{Y}$, the goal is to unveil the most descriptive relationships between the interacting node groups.

Whereas assumptions about the data and their interpretation are heavily application-dependent, a common feature in applications of co-clustering is the presence of (hidden) dense/uniform submatrices, each capturing a subset of $\{\mathcal{A}_i\}_{i=1}^n$ that has similar feature values related to a subset of $\{\mathcal{B}_j\}_{j=1}^p$. For sociomatrices, permuting the rows and columns can reveal subgroups of actors closely related to each other. For an http TAG between inside and outside hosts of a university campus, one submatrix might coincide with a research group's frequent visits to a server "farm", e.g., IEEEXplore®. In DNA microarray data, a uniform submatrix can help identify a subgroup of co-regulated genes under a subset of experimental conditions. All in all, these submatrices may reveal certain informative behavioral patterns in a concise fashion. Finding such submatrices boils down to a co-clustering problem.

Several practical concerns have to be addressed in these co-clustering applications. First, a huge amount of data is usually collected in $\mathbf{Y}$. For instance, in TAG analysis the number of hosts is typically in the order of thousands [3], and for DNA microarrays np can be over 500,000 easily [4]. Therefore, the submatrices, especially the interesting ones, are usually hidden 'sparsely' in $\mathbf{Y}$. However, traditional co-clustering algorithms are sparsity agnostic. Second, the submatrices may overlap, so that some nodes can belong to different submatrices. For example, an actor of diverse expertise can be involved with different fellow actor groups; or a gene can have multiple functions and hence co-regulate with different subsets of genes in different samples. In this context, given a data array $\mathbf{Y}$ with those characteristics, the aim of the present paper is to develop an efficient co-clustering algorithm to extract the underlying submatrices, by exploiting sparsity and accounting for possible overlapping clusters.

2.1. Plaid Models

In order to solve the co-clustering problem, one can model the data array $\mathbf{Y}$ as a superposition of multiple submatrices. This is the basic idea behind plaid models, which have been successfully used in gene expression analysis [4]. Specifically, the basic plaid model consists of multiple uniform layers, submatrices in our terminology, intended to capture the behavior patterns, and also a background layer to account for certain global effects. Given a total of k submatrices, the value Y_{ij} is expressed using the summation of all the layer effects, and a noise level ϵ_{ij}; that is,

$$Y_{ij} = \mu_0 + \sum_{l=1}^{k} \mu_l \rho_{il} \kappa_{jl} + \epsilon_{ij}, \ \ \forall i, \ \forall j \tag{1}$$

where μ_0 is the background level color, μ_l describes the level of layer l, $\rho_{il} = 1$ if $\mathcal{A}_i$ is in the l-th submatrix (0 otherwise); and similarly, $\kappa_{jl} = 1$ if $\mathcal{B}_j$ is in the l-th submatrix (0 otherwise). The binary-valued numbers ρ_{il} and κ_{jl} are the l-th submatrix membership indicator for $\mathcal{A}_i$ and $\mathcal{B}_j$, respectively. Incorporating row/column-level related effects for each layer, (1) becomes

$$\begin{aligned} Y_{ij} &= (\mu_0 + \alpha_{i0} + \beta_{j0}) + \sum_{l=1}^{k} (\mu_l + \alpha_{il} + \beta_{jl}) \rho_{il} \kappa_{jl} + \epsilon_{ij} \\ &= \sum_{l=0}^{k} \theta_{ijl} \rho_{il} \kappa_{jl} + \epsilon_{ij}, \ \ \forall i, \ \forall j \end{aligned} \tag{2}$$

where α_{il} and β_{jl} capture node-related effects of $\mathcal{A}_i$ and $\mathcal{B}_j$ within the l-th submatrix. With the plaid model of (2), we seek the optimal membership indicators to minimize the data fitting error

$$\begin{aligned} \min_{\rho_{il}, \kappa_{jl} \in \{0,1\}, \theta_{ijl}} \ & \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{p} \left(Y_{ij} - \theta_{ij0} - \sum_{l=1}^{k} \theta_{ijl} \rho_{il} \kappa_{jl} \right)^2 \\ & + \lambda \sum_{i=1}^{n} \sum_{l=1}^{k} |\rho_{il}| + \lambda \sum_{j=1}^{p} \sum_{l=1}^{k} |\kappa_{jl}| \end{aligned} \tag{3}$$

where $\lambda > 0$ controls the sparsity enforced on the chosen indicators. The first term in (3) is a classical LS fitting error criterion considered in [4]. The novelty here is in the last two sums which exploit the sparsity of the submatrices present in plaid models. Given that the number of informative submatrices and the size of them is usually much smaller compared to the problem dimension, enforcing sparsity facilitates extraction of the most representative submatrices out of the data $\mathbf{Y}$.

It is well known that many clustering problems are NP-hard; see e.g., [4]. The problem (3) is further complicated due to the binary constraints. These considerations motivate the development of efficient approximation algorithms in the next section. Before continuing, it is worth stressing that the original algorithms for plaid models [4, 5] identify the submatrices one by one; that is, they first detect the largest submatrix, then subtract it from $\mathbf{Y}$, and restart from the residual matrix to identify the next one. This way, they are likely to lose the global picture for $\mathbf{Y}$ and are prone to error propagation across different submatrices, since it is impossible to refine the submatrices once they have been determined. To address this limitation, we are going to perform simultaneous identification of all submatrices. Section 3 will elaborate on our novel sparsity-cognizant overlapping co-clustering (SOC) algorithm based on plaid models.

3. SOC ALGORITHM

To simplify exposition, suppose we have obtained the background layer levels μ_0, α_{i0}, and β_{j0} (see details in Section 3.3). The (i,j)-

th entry of the residual matrix $\mathbf{Z}$ then becomes

$$Z_{ij} = Y_{ij} - (\mu_0 + \alpha_{i0} + \beta_{j0}) = \sum_{l=1}^{k} \theta_{ijl}\rho_{il}\kappa_{jl} + \epsilon_{ij}. \quad (4)$$

Given the number of submatrices k, minimizing the fitting error in (3) amounts to substituting the term $Y_{ij} - \theta_{ij0}$ with Z_{ij}, and the SOC algorithm is developed to solve this problem. As mentioned earlier, the sparsity is enforced in (3) through the ℓ_1 norm penalty. This term necessitates cross-layer data fitting, instead of the greedy search across layers. To this end, we adopt an iterative approach where all the θ, ρ, and κ values are updated in turn per iteration cycle. Let $\boldsymbol{\theta}^{(s)}$ denote all $\theta_{ijl}^{(s)}$ values $\forall i, j, l$ at iteration s; and likewise for $\boldsymbol{\rho}^{(s)}$ and $\boldsymbol{\kappa}^{(s)}$. After selecting initial values for $\boldsymbol{\rho}^{(0)}$ and $\boldsymbol{\kappa}^{(0)}$ as detailed in Section 3.3, S update iterations follow. For $s = 1, \ldots, S$, vector $\boldsymbol{\theta}^{(s)}$ is updated from $\boldsymbol{\rho}^{(s-1)}$ and $\boldsymbol{\kappa}^{(s-1)}$, then $\boldsymbol{\rho}^{(s)}$ is updated from $\boldsymbol{\theta}^{(s)}$ and $\boldsymbol{\kappa}^{(s-1)}$, and finally $\boldsymbol{\kappa}^{(s)}$ from $\boldsymbol{\theta}^{(s)}$ and $\boldsymbol{\rho}^{(s)}$.

In addition to this iterative cycling, the membership indicators are updated with exact search on the binary-valued alphabet. This has been shown to yield better performance in [5], compared to the soft estimation of the original algorithm in [4]. However, since we update across all the submatrices, the problem (3) with binary-valued constraint will incur combinatorial complexity. Therefore, reduced-complexity algorithms will be used later to search for the optimal solution to the ℓ_1 penalized cost function on the binary lattice. First, let us look at the update of the real-valued layer level.

3.1. Updating $\boldsymbol{\theta}^{(s)}$

Given $\boldsymbol{\rho}^{(s-1)}$ and $\boldsymbol{\kappa}^{(s-1)}$, at iteration s we update the $\theta_{ijl}^{(s)}$ values by minimizing

$$\min_{\mu_l, \alpha_{il}, \beta_{jl}} \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{p}\left(Z_{ij} - \sum_{l=1}^{k}(\mu_l + \alpha_{il} + \beta_{jl})\rho_{il}^{(s-1)}\kappa_{jl}^{(s-1)}\right)^2. \quad (5)$$

This is an unconstrained quadratic programming (QP) problem, which can be solved in closed form. However, finding such solution entails inversion of a large matrix, the dimension of which equals the number of unknowns in (5). For this reason, we will apply instead the coordinate descent algorithm. Specifically, for $l = 1, \ldots, k$ suppose we have all the layer levels for the k submatrices but the l-th one, let the (i, j)-the entry of matrix $\tilde{\mathbf{Z}}^l$ be

$$\tilde{Z}_{ij}^l = Z_{ij} - \sum_{l'=1, l' \neq l}^{k} \theta_{ijl'}^{(s)}\rho_{il'}^{(s-1)}\kappa_{jl'}^{(s-1)} \quad (6)$$

and extract from $\tilde{\mathbf{Z}}^l$ of the rows where $\rho_{il}^{(s-1)} = 1$ and the columns where $\kappa_{jl}^{(s-1)} = 1$ to form the reduced matrix $\check{\mathbf{Z}}^l$. Minimizing (5) then yields

$$\begin{aligned} \mu_l^{(s)} &= \mathsf{mean}(\check{\mathbf{Z}}^l) \\ \alpha_{il}^{(s)} &= \mathsf{mean}(\check{\mathbf{Z}}_{i:}^l) - \mu_l^{(s)} \\ \beta_{jl}^{(s)} &= \mathsf{mean}(\check{\mathbf{Z}}_{:j}^l) - \mu_l^{(s)} \end{aligned} \quad (7)$$

where $\mathsf{mean}(\cdot)$ returns the mean of all the matrix entries. Note that $\alpha_{il}^{(s)} = 0$ for the rows that are not in the l-th submatrix; and similarly for $\beta_{jl}^{(s)}$. The coordinate descent algorithm updates the l-th submatrix feature levels as in (7), by cycling through $l = 1, \ldots, L$ in turn, for T cycles.

3.2. Updating $\boldsymbol{\rho}_{il}^{(s)}$ and $\boldsymbol{\kappa}_{jl}^{(s)}$

Due to the symmetry of updating the row/column membership indicators, we will only focus on detecting $\boldsymbol{\rho}^{(s)}$ using $\boldsymbol{\theta}^{(s)}$ and $\boldsymbol{\kappa}^{(s-1)}$. Notice that in this case minimization of the fitting error in (3) reduces to n subproblems, each one indexed by $i = 1, \ldots, n$

$$\{\rho_{il}^{(s)}\}_{l=1}^{k} = \min_{\rho_{il} \in \{0,1\}} \frac{1}{2}\sum_{j=1}^{p}\left(Z_{ij} - \sum_{l=1}^{k}\theta_{ijl}^{(s)}\kappa_{jl}^{(s-1)}\rho_{il}\right)^2 + \lambda\sum_{l=1}^{k}\rho_{il} \quad (8)$$

where the absolute value in the ℓ_1 norm penalty term has been dropped due to non-negativity (we enforce a $\{0, 1\}$ alphabet). With the goal of minimizing the LS cost along with a penalty λ per $\rho_{il} = 1$, the solution to (8) is expected to both be a good fit of the model and also exhibit sparsity. This accounts for the fact that only a few and relatively small informative submatrices are typically present compared to the problem dimension. The membership indicators for the i-th row, i.e., $\{\rho_{il}^{(s)}\}_{l=1}^{k}$, need to be obtained jointly. This is critical for the case of overlapping submatrices, where the cross-effects are relatively strong, thus solving $\{\rho_{il}^{(s)}\}_{l=1}^{k}$ jointly is expected to offer considerably improved performance.

The minimization problem in (8) is quadratic subject to $\{0, 1\}$ binary constraints. Mathematically, the problem is the same encountered in MIMO or multiuser detection with a binary pulse amplitude modulation (PAM) alphabet. The problem is NP-hard, yet a wide variety of advanced algorithms from the MIMO detection literature can be applied; e.g., see [7] and references therein. Among these, the *sphere decoding algorithm* (SDA), offers (near-) optimal performance at approximately cubic average complexity at medium-to-high received signal-to-noise ratio, which is related to the quality of the data matrix $\mathbf{Z}$ here. Therefore, given $\boldsymbol{\theta}^{(s)}$ and $\boldsymbol{\kappa}^{(s-1)}$, we can apply those polynomial complexity detection algorithms to obtain a (sub-)optimal $\{\rho_{il}^{(s)}\}_{l=1}^{k}$ for each $i = 1, \ldots, n$. Similarly, with $\boldsymbol{\theta}^{(s)}$ and $\boldsymbol{\rho}^{(s)}$, $\{\kappa_{jl}^{(s)}\}_{l=1}^{k}$ can be obtained efficiently $\forall j = 1, \ldots, p$.

The last two subsections complete one iteration of the novel SOC algorithm. Since the cost function in (3) is bounded below by zero, and non-increasing per iteration, the SOC algorithm is guaranteed to converge at least to a stationary point. Regularization strategies, such as pruning, have been used to refine the original algorithms in [4, 5]. They can be implemented here too, but they are not so relevant to the novelty of the SOC algorithm. Next, more details are provided on some further issues involving the implementation of the SOC algorithm.

3.3. Implementation Issues

The major part of the SOC algorithm outlined in the last two subsections exploits the sparsity inherent in typical application data, and copes with the limitations of the original greedy layer-by-layer identification. In this section, we clarify some issues related to initialization and the choice of parameters.

The initialization includes two parts: the background level fitting, and the starting values $\boldsymbol{\rho}^{(0)}$ and $\boldsymbol{\kappa}^{(0)}$ for the iterative cycles of the SOC algorithm. The purpose of background fitting is to obtain an accurate $\mathbf{Z}$ matrix in (4), such that the submatrices appear out of the large data set. It is equivalent to the l-th layer level estimation in Section 3.1, with $\mathbf{Y}$ substituting $\check{\mathbf{Z}}^l$, and all the row/column indicators equal to one. The background fitting should also be included whenever we have an improved estimate of all the submatrices after

the update of $\boldsymbol{\theta}^{(s)}$. The indicators $\boldsymbol{\rho}^{(0)}$ and $\boldsymbol{\kappa}^{(0)}$ are initialized using k-means, the 'workhorse' clustering algorithm performing either row or column partitioning; see e.g., [5] for more details.

Two parameters whose choice critically affects performance of the SOC algorithm are: i) the number of layers, k; and ii) the constant λ. In greedy layer-by-layer deflation approaches, k can be chosen to explain a certain percentage of variation which is deemed "systematic" for the type of data considered. The sparsity penalty helps to find the most representative submatrices, and we can use a similar deflation approach to determine a suitable k. The sparsity regularization parameter λ has been set by trial-and-error in our experiments; however, systematic approaches based on *bi-cross-validation* can be adopted; see e.g., [8] and the references therein.

4. PRELIMINARY NUMERICAL TESTS

In this section, a simulated test using synthetic data is presented to illustrate the merits of the SOC algorithm. Comprehensive numerical experiments on real Internet traffic and sociometric data can be found in [6], but are omitted here due to space limitations. Despite their simplicity, simulated data sets are attractive since the results are easy to visualize. For the ensuing comparison with the baseline algorithm for plain models [4], the software available from the author's website is utilized.

The simulated data set corresponds to a 100×100 binary matrix $\mathbf{Y}$ of all zeros, with the exception of two overlapping uniform submatrices, respectively supported in $S_1 := [10, 40] \times [10, 40]$ and $S_2 := [30, 60] \times [30, 60]$, such that $Y_{ij} := 1$ for $i, j \in S_1 \cup S_2$. The additive noise on each entry of $\mathbf{Y}$ is uniformly distributed between $[0, 0.5]$ independently; see Fig. 2(a). The actual data set is obtained by randomly permuting the rows and columns of the resulting matrix, so that the latent sub-block structure is no longer apparent [Fig. 2(b)].

The SOC algorithm is tested after setting $S = 20$ cycling iterations and $T = 1$ for the coordinate descent algorithm in Section 3.1, targeting for $k = 2$ layers; for both $\lambda = 0$ (no sparsity enforced) and $\lambda = 3$. We use the SDA to detect the optimal solution to (8). For the background and additive layers, the full model in (2) is adopted. For the plaid model implementation of [4], the row/column rejection strategy was utilized as well as the option to search for small intense layers rather than large diffuse ones (further details are in [4]); also three backfitting cycles were conducted to re-optimize the θ_{ijk} parameters, once the plaid model software discovered the significant layers. Upon termination of the algorithms, rows and columns of the estimated matrix (obtained as sum of fitted layers) were permuted back to the original order to inspect whether the original overlapping submatrices have been recovered.

The layer membership results are depicted in Fig. 2 (c)-(e), by showing the support of the layers found. Both the proposed algorithm without sparsity and the plaid model fail to recover the original structure. In particular, the plaid model recovers a single significant layer after deflating the background from $\mathbf{Y}$. Through ℓ_1-norm regularization of the layer row/column membership indicators, the proposed scheme with $\lambda = 3$ yields the desired result after successfully refining the initializations. In this case, $\mu_1 = 0.9349$ and $\mu_2 = 0.9063$, while the α_{ik} and β_{jk} have nonzero support for $i, j \in S_1 \cap S_2$ and $k = 1, 2$. This way, after summing layers the fitted values $\hat{Y}_{ij} \approx 1$ in the overlapping region also. Exploiting sparsity is key in determining the exact structure, while the greedy nature of the plaid model leads to a single co-cluster covering all effects but with insufficient resolution.

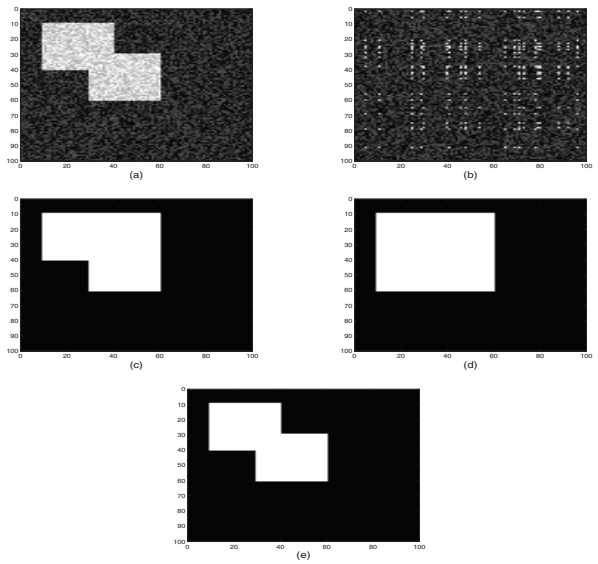

Fig. 2. Simulated data set with overlapping submatrices (a), and after row/column permutations (b). Co-clustering results: proposed scheme with $\lambda = 0$ (c), $\lambda = 3$ (e), and the plaid model (d).

5. CONCLUSIONS AND CURRENT RESEARCH

A novel sparsity-exploiting co-clustering approach was developed based on plaid models which allow for overlapping co-clusters. Preliminary numerical tests corroborated the merits of the resultant SOC algorithm relative to competing alternatives in determining the exact model structure. An interesting direction to substantiate the proposed approach includes establishing uniqueness of the solution which minimizes the LS cost regularized with the ℓ_1 norm under the binary alphabet constraints. In addition, the novel approach is currently tested on real data sets, and results will be reported in the final version of the paper as well as in [6].

6. REFERENCES

[1] S. Busygin, O. Prokopyev, and P. M. Pardalos, "Biclustering in data mining," *Computers and Operations Research*, vol. 35, pp. 2964–2987, 2008.

[2] S. Wasserman and K. Faust, *Social Network Analysis*, Cambridge University Press, 1994.

[3] Y. Jin, E. Sharafuddin, and Z. L. Zhang, "Unveiling core network-wide communication patterns through application traffic activity graph decomposition," in *Proc. of SIGMETRICS*, New York, NY, 2009, pp. 49–60.

[4] L. Lazzeroni and A. Owen, "Plaid models for gene expression data," *Statistica Sinica*, vol. 12, pp. 61–86, 2002.

[5] H. Turner, T. Bailey, and W. Krzanowski, "Improved biclustering of microarray data demonstrated through systematic performance tests," *Comput. Stat. Data Anal*, vol. 48, pp. 235–254, 2005.

[6] H. Zhu, G. Mateos, G. B. Giannakis, N. D. Sidiropoulos, and A. Banerjee, "Sparsity-exploiting overlapping co-clustering for behavior inference in social networks," (journal version in preparation).

[7] G. B. Giannakis, Z. Liu, X. Ma, and S. Zhou, *Space-Time Coding for Broadband Wireless Communications*, John Wiley & Sons, Inc., 2007.

[8] D. M. Witten, R. Tibshirani, and T. Hastie, "A penalized matrix decomposition, with applications to sparse principal components and canonical correlation analysis," *Biostatistics*, vol. 10, pp. 515–534, 2009.