HAL
open science

Application des Pattern Structures à la découverte de biclusters à changements de signes cohérents

Nyoman Juniarta, Miguel Couceiro, Amedeo Napoli

To cite this version:

Nyoman Juniarta, Miguel Couceiro, Amedeo Napoli. Application des Pattern Structures à la découverte de biclusters à changements de signes cohérents. EGC 2019 - 19ème Conférence francophone sur Extraction et Gestion des connaissances, Jan 2019, Metz, France. pp.285-290. hal-02099607

HAL Id: hal-02099607

https://inria.hal.science/hal-02099607v1

Submitted on 15 Apr 2019

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Application des Pattern Structures à la découverte de biclusters à changements de signes cohérents

Nyoman Juniarta*, Miguel Couceiro*, Amedeo Napoli*

*Université de Lorraine, CNRS, Inria, LORIA, F-54000 Nancy, France
firstname.lastname@loria.fr

Résumé. Le "biclustering" joue un rôle majeur dans beaucoup d'applications du monde réel. Il est lié au "clustering" qui regroupe des lignes similaires dans une matrice de données numériques, tandis que le biclustering cherche à regrouper simultanément des lignes et colonnes similaires, c'est-à-dire trouver des sous-matrices où émerge une corrélation entre les entrées. Le biclustering s'appuie sur un critère de similarité, et dans cet article, nous nous intéressons au biclustering "à colonnes constantes" (CC), où les valeurs numériques dans les colonnes des sous-matrices sont constantes pour chaque ligne. L'étude est ensuite étendue au biclustering à "changements de signes cohérents" (CSC), où la différence entre les valeurs de deux colonnes consécutives de la sous-matrice est du même signe pour chaque ligne.

1 Introduction

Le "biclustering" est une technique de fouille de données qui permet de retrouver des motifs sous la forme d'une sous-matrice dans une matrice ou un tableau de données à deux entrées [Madeira et Oliveira (2004)]. Il est lié au "clustering" [Arabie et al. (1996)] dont l'objectif est de regrouper des lignes de la matrice en fonction des similarités qu'elles présentent. Pour une sous-matrice considérée, la similarité entre les lignes doit alors être vérifiée pour toutes les colonnes. D'une façon alternative, le biclustering consiste à regrouper les lignes et les colonnes d'une sous-matrice simultanément, en fonction d'un certain critère de similarité. Par exemple, dans le biclustering à colonnes constantes (CC), pour chaque ligne d'une sous-matrice les valeurs dans les colonnes sont constantes et (possiblement) différentes d'une colonne à l'autre.

Pour mettre en oeuvre le biclustering, nous utilisons dans cet article l'analyse formelle de concepts (FCA pour "Formal Concept Analysis" [Ganter et Wille (1999)]) et les "Pattern Structures", une extension de la FCA pour traiter les tableaux numériques [Ganter et Kuznetsov (2001); Kaytoue et al. (2011)]. La FCA est très liée au biclustering car elle cherche aussi à regrouper les lignes et colonnes d'un tableau de données binaires en fonction d'une similarité qui consiste à partager les mêmes croix (ou 1) pour chaque ligne d'une sous-matrice.

Nous présentons ici deux méthodes pour découvrir des biclusters dits "à changements de signes cohérents" (CSC), comme défini dans [Madeira et Oliveira (2004)], à partir d'une matrice de signes. Cette matrice est obtenue en deux étapes : (i) en "échelonnant" ("scaling") la matrice numérique d'origine où ont été recherchés les biclusters à colonnes constantes (CC),

(ii) en appliquant des “partition pattern structures” dans la matrice des signes pour retrouver des partitions de lignes exhibant des changements de signes cohérents.

Le biclustering a été beaucoup utilisé pour analyser des données biologiques et en particulier des données d’expression de gènes. Ainsi l’algorithme de biclustering CSC SAMBA est proposé dans [Tanay et al. (2002)], où dans la matrice des données, les gènes sont en ligne et les conditions expérimentales en colonne. Il s’agit alors de découvrir des sous-matrices où les conditions affectent les gènes de façon cohérente, c’est-à-dire que chaque couple de conditions produit toujours le même effet, positif ou négatif, dans la sous-matrice. La matrice des données est considérée comme un graphe biparti, où les sommets se divisent en gènes et conditions, et où une arête existe entre un sommet de type “gène” g et un sommet de type “condition” c si c affecte g. Un bicluster correspond alors à une biclique du graphe biparti, ce qui est équivalent à la recherche de concepts en FCA standard.

Dans le même esprit, l’algorithme QUBIC proposé dans [Li et al. (2009)] recherche aussi tous les biclusters CSC dans une matrice de données. Un graphe pondéré est construit, où chaque gène est représenté par un sommet, et deux sommets sont reliés par une arête en fonction du degré de leur similarité. Ce degré de similarité fournit la pondération attachée à l’arête. Ensuite, la tâche de biclustering consiste à retrouver les sous-graphes de pondération maximale.

Pour notre part, nous explorons dans cet article la découverte de biclusters CSC en utilisant les techniques de FCA et l’extension des patterns structures pour traiter les données numériques. Tout d’abord nous décrivons la relation existante entre biclustering et FCA (section 2). Puis nous présentons une approche originale de recherche de biclusters à changements de signes cohérents et les expérimentations réalisées (section 3), avant de conclure le papier.

2 Biclustering et FCA

2.1 Biclustering

Dans un tableau de données numériques, considéré comme une matrice, il est intéressant de rechercher des sous-matrices où les éléments ont tous la même valeur (table 1a). Cette tâche correspond à la recherche de biclusters à valeurs constantes, ce qui revient exactement à calculer les concepts en FCA. Il peut aussi être intéressant de découvrir des biclusters dits “à colonnes constantes” (CC) où les valeurs sont identiques pour chaque colonne de la sous-matrice (table 1b), ce qui a directement des applications en recommandation où le bicluster représente un groupe d’individus (en ligne) donnant la même note à un produit (en colonne). Enfin, dans un bicluster “à changements de signes cohérents” (CSC, table 1c), les éléments de la sous-matrice sont considérés comme des symboles, soit ‘$\nearrow$’ ou soit ‘$\searrow$’. Les signes sont corrélés dans le “même sens” ou dans le “sens opposé”, par exemple la colonne 1 en table 1c) est identique à la colonne 2 et opposée à la colonne 3.

Dans cet article, nous étudions les deux types de biclusters CC et CSC, que nous appliquons à la fouille de “motifs graduels” [Di-Jorio et al. (2009)]. La recherche de biclusters de type CC en FCA a été introduite dans [Codocedo et Napoli (2014)] et s’appuie sur les “partition pattern structures”. Elle est expliquée dans la section suivante.

2	2	2	2	4	2	5	3	$\nearrow$	$\nearrow$	$\searrow$
2	2	2	2	4	2	5	3	$\nearrow$	$\nearrow$	$\searrow$
2	2	2	2	4	2	5	3	$\searrow$	$\searrow$	$\nearrow$
2	2	2	2	4	2	5	3	$\searrow$	$\searrow$	$\nearrow$
(a)				(b)				(c)		

TAB. 1 – *Trois types de biclusters : (a) "à valeurs constantes", (b) "à colonnes constantes" (CC), et (c) "à changements de signes cohérents" (CSC).*

	m_1	m_2	m_3	m_4	m_5
g_1	1	5	3	2	7
g_2	1	1	4	2	7
g_3	2	5	4	5	3
g_4	2	5	4	5	7

TAB. 2 – *Un tableau de données numériques avec* 4 *objets (en ligne) et* 5 *attributs (en colonne).*

2.2 Biclustering de type CC et "partition pattern structures"

La FCA est un formalisme mathématique qui s'appuie sur la théorie des treillis et qui est utilisé en classification et en fouille de données [Ganter et Wille (1999)]. La FCA s'applique à un tableau de données binaire, calcule les concepts formels, qui correspondent à des rectangles maximaux de croix dans le tableau binaire, et les organise en un treillis de concepts. Les "pattern structures" généralisent la FCA et sont utilisées pour traiter des données complexes comme les données numériques (entre autres).

Les "partition pattern structures" (PPS) ont été étudiées dans le cadre de la recherche de dépendances fonctionnelles [Baixeries et al. (2014)] mais aussi pour la recherche de biclusters de type CC dans une matrice numérique. Ici une partition correspond à un regroupement des objets (en ligne) en fonction des valeurs de leurs attributs (en colonne). Ainsi, un exemple de "partition pattern concept" (pp-concept) est donné par $(\{m_1, m_4\}, \{\{g_1, g_2\}, \{g_3, g_4\}\})$ dans la table 2. Deux biclusters de type CC sont obtenus, $(\{g_1, g_2\}\{m_1, m_4\})$ et $(\{g_3, g_4\}$ $\{m_1, m_4\})$. Par manque de place, nous ne pouvons détailler les calculs qui le sont en revanche dans [Codocedo et Napoli (2014)].

3 Biclustering CSC et "partition pattern structures"

Dans cette section, nous présentons deux approches pour mettre en oeuvre les biclustering CSC en s'appuyant sur les "partition pattern structures" (PPS). D'abord nous décrivons comment résoudre le problème avec un échelonnage (scaling) de la matrice des données. Ensuite, nous montrons comment les biclusters CSC peuvent être retrouvés directement avec PPS. Dans la première approche (algorithme 1), à partir d'une matrice binaire nous construisons une nouvelle matrice dans laquelle chaque colonne décrit la cohérence entre deux colonnes de la matrice d'origine. Puis nous appliquons PPS pour calculer les biclusters CC.

Considérons la matrice des signes données dans la table 3a. Cette matrice peut être échelonnée pour produire l'apposition des trois matrices binaires données dans la table 3b, où chaque couple de colonnes (c_i, c_j) constitue une nouvelle colonne. Dans une colonne (c_i, c_j), la valeur

Input : Une matrice des signes binaires S avec un ensemble de colonnes C
Output : Un ensemble de biclusters CSC extrait de S

```
all_csc := ∅ ;
foreach c_x ∈ C do
    À partir de tous les couples d'attributs (c_x, c_y), x < y, construire une matrice
    échelonnée T ;
    Calculer les biclusters à colonne constante dans T, puis les sauvegarder dans B ;
    foreach b ∈ B do
        b_r := ensemble de lignes dans T ;
        b_c := ensemble de colonnes dans T ;
        csc := une sous-matrice de S dont les lignes sont b_r et les colonnes sont les
          attributs présents dans b_c ;
        all_csc.add(csc) ;
    end
end
return all_csc ;
```

Algorithme 1 : Biclustering CSC avec échelonnage.

1 apparaît si le signe est différent pour les deux colonnes i et j, et 0 sinon (opération XOR). Ensuite, nous appliquons un biclustering CC à l'apposition des trois matrices échelonnées. Un bicluster CC résultant correspond à un bicluster CSC de la matrice d'origine (voir table 4). Les colonnes associées à un bicluster CSC proviennent de l'union des couples de colonnes associées à un bicluster CC.

	c_1	c_2	c_3	c_4
r_1	↘	↘	↗	↗
r_2	↗	↘	↗	↗
r_3	↗	↗	↘	↗
r_4	↘	↗	↘	↘

(a)

c_1c_2	c_1c_3	c_1c_4	c_2c_3	c_2c_4	c_3c_4
0	1	1	1	1	0
1	0	0	1	1	0
0	1	0	1	0	1
1	0	0	1	1	0

(b)

TAB. 3 – *(a) Une matrice binaire des signes et (b) l'apposition des trois matrices échelonnées.*

Pour éviter l'échelonnage et la combinaison des colonnes deux à deux, nous pouvons appliquer directement PPS à la matrice originale. C'est l'objet du second algorithme (algorithme 2), où les biclusters CSC sont découverts en examinant les pp-concepts engendrés.

Un pp-concept (A, d) est composé d'un ensemble d'attributs A et d'un ensemble de composants de partitions p. De plus, comme tout couple (p, A) est un bicluster CC, un bicluster CSC est soit un bicluster CC ou encore un couple de biclusters CC de signes opposés.

Par exemple, dans la matrice donnée en table 3a, trois biclusters CC sont engendrés à partir du pp-concept $(\{c_2, c_3, c_4\}, \{\{r_1, r_2\}, \{r_3\}, \{r_4\}\})$.

— $b_x = (\{r_1, r_2\}, \{c_2, c_3, c_4\})$
— $b_y = (\{r_3\}, \{c_2, c_3, c_4\})$
— $b_z = (\{r_4\}, \{c_2, c_3, c_4\})$

biclusters CC	biclusters CSC
$(\{r_2, r_4\}\{c_1c_2, c_1c_3, c_1c_4\})$	$(\{r_2, r_4\}\{c_1, c_2, c_3, c_4\})$
$(\{r_1, r_3\}\{c_1c_2, c_1c_3\})$	$(\{r_1, r_3\}\{c_1, c_2, c_3\})$
$(\{r_1, r_2, r_3, r_4\}\{c_2c_3\})$	$(\{r_1, r_2, r_3, r_4\}\{c_2, c_3\})$
$(\{r_1, r_2, r_4\}\{c_2c_3, c_2c_4\})$	$(\{r_1, r_2, r_4\}\{c_2, c_3, c_4\})$

TAB. 4 – *Quelques biclusters CC de la table 3b et leurs biclusters CSC correspondant dans la table 3a.*

Input : Une matrice binaire de signes S
Output : Un ensemble de biclusters CSC de S
1 $all_csc := \emptyset$;
2 all_ppc := tous les pp-concepts de S
3 **foreach** $ppc \in ppcs$ **do**
4 | Créer une nouvelle partition en fusionnant deux "partitions opposées" ;
5 | Ajouter les éléments de la nouvelle partition à all_csc ;
6 **end**
7 **return** all_csc ;

Algorithme 2 : Biclustering CSC sans échelonnage.

Ici, b_x est "opposé" à b_z car le signe de b_x dans chaque colonne ('$\searrow$ $\nearrow$ $\nearrow$') est l'inverse du signe correspondant dans b_z ('$\nearrow$ $\searrow$ $\searrow$'). De fait, $(\{r_1, r_2, r_4\}, \{c_2, c_3, c_4\})$ est un bicluster CSC. Quant à b_y, il est lui-même un bicluster CSC puisqu'il n'est en "opposition" avec aucun autre bicluster CC.

Dans nos expérimentations, nous avons comparé les temps d'exécution des deux approches appliquées à des jeux de données numériques générés de façon aléatoire. À partir d'un tableau de données de taille $m \times n$, les deux méthodes construisent une matrice binaire de signes de taille $C_2^m \times n$ et similaire à celle de la table 3a. Ensuite, pour 10 attributs par exemple, la méthode *"avec-scaling"* applique PPS dans des matrices de 9, 8, 7, ... et 1 attributs séparément tandis que la méthode *"sans-scaling"* applique PPS sur une seule matrice de 10 attributs.

Pratiquement, il apparaît que les méthodes *"avec-scaling"* et *"sans-scaling"* se comportent de façon quasi-identique pour des jeux de données de petite taille. Les différences de temps d'exécution n'étant pas significatives, de plus amples expérimentations doivent être menées sur des jeux de données plus conséquents avant de conclure en faveur de l'une ou l'autre méthode.

4 Conclusion

Dans cet article, nous avons introduit et discuté des approches de biclustering dans le cadre de l'analyse formelle de concepts et des "partition pattern structures". Ensuite, nous avons adapté la recherche de biclusters à colonne constante à celle des biclusters à changements de signes cohérents. Deux types d'algorithmes et d'expérimentations ont aussi été proposés, "avec-scaling" et "sans-scaling", qui ne présentent pas pour l'heure de différences significatives.

Pour le futur, il faut encore affiner la théorie et mener une étude plus fondamentale et systématique des possibilités du biclustering dans un cadre FCA. Et surtout, il faut faire beaucoup plus d'expérimentations, en particulier sur des jeux de données réels tels qu'il en existe en biologie pour l'expression de gènes, et comparer les possibilités de notre approche avec d'autres approches de biclustering ou co-clustering numériques.

Références

Arabie, P., L. Hubert, et G. D. Soete (Eds.) (1996). *Clustering and Classification*. World Scientific Publishers.

Baixeries, J., M. Kaytoue, et A. Napoli (2014). Characterizing functional dependencies in formal concept analysis with pattern structures. *Annals of Mathematics and Artificial Intelligence 72*, 129–149.

Codocedo, V. et A. Napoli (2014). Lattice-based biclustering using partition pattern structures. In *Proceedings of the Twenty-first European Conference on Artificial Intelligence*, pp. 213–218. IOS Press.

Di-Jorio, L., A. Laurent, et M. Teisseire (2009). Mining frequent gradual itemsets from large databases. In *International Symposium on Intelligent Data Analysis*, pp. 297–308. Springer.

Ganter, B. et S. O. Kuznetsov (2001). Pattern structures and their projections. In *International Conference on Conceptual Structures*, pp. 129–142. Springer.

Ganter, B. et R. Wille (1999). *Formal Concept Analysis : Mathematical Foundations* (2nd ed.). Springer.

Kaytoue, M., S. O. Kuznetsov, A. Napoli, et S. Duplessis (2011). Mining Gene Expression Data with Pattern Structures in Formal Concept Analysis. *Information Science 181*(10), 1989–2001.

Li, G., Q. Ma, H. Tang, A. H. Paterson, et Y. Xu (2009). QUBIC : a qualitative biclustering algorithm for analyses of gene expression data. *Nucleic Acids Research 37*(15), e101–e101.

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

Tanay, A., R. Sharan, et R. Shamir (2002). Discovering statistically significant biclusters in gene expression data. *Bioinformatics 18*(suppl_1), S136–S144.

Summary

Biclustering plays a crucial role in many real world applications. Related to clustering, which groups similar rows in a matrix (data table), biclustering aims at simultaneously grouping similar rows and columns, i.e. to find submatrices which exhibit a correlation among their respective cells. There are many types of biclustering based on a similarity criterion. In this paper we are interested in constant-column (CC) biclustering, where the objective is to discover submatrices whose columns have a constant value across all the rows. Then, we study an extension of CC biclustering to the so-called coherent-sign-changes (CSC) biclustering.