Extraction de biclusters contraints dans des contextes bruites

Karima Mouhoubi, Lucas Létocart, Céline Rouveirol

Extraction de biclusters contraints dans des contextes bruités

Karima Mouhoubi, Lucas Létocart, Céline Rouveirol

LIPN, UMR CNRS 7030, Université Paris 13
99 av. J.B. Clément, 93430 Villetaneuse, France
prénom.nom@lipn.univ-paris13.fr

Résumé : L'extraction de biclusters, qui consiste à rechercher un groupe d'attributs qui montrent un comportement cohérent pour un sous-ensemble d'observations dans une matrice de données, est une tâche importante dans divers domaines, telle que la biologie. Nous proposons ici un nouveau système, *COBIC*, qui combine des algorithmes de graphes avec des méthodes de fouille de données pour une recherche efficace de biclusters pertinents et susceptibles de se recouvrir. *COBIC* est fondé sur les algorithmes de flot maximal/coupe minimale et est capable de prendre en compte les connaissances d'une base exprimées sous forme d'une classification, par un mécanisme d'adaptation des poids lors de l'extraction itérative des régions denses. L'évaluation de *COBIC* sur des données réelles et la comparaison par rapport à des méthodes efficaces de biclustering montrent que *COBIC* est très performant et en particulier lorsque la qualité des biclusters s'évalue en fonction de la significativité de l'enrichissement des clusters calculés avec les fonctions cellulaires décrites dans l'Ontologie GO.

Mots-clés : Biclustering, sous graphes denses, données bruitées.

1. Introduction

La recherche de motifs ensemblistes dans des données booléennes consiste à rechercher tous les rectangles de 1 dans une matrice booléenne. Les données réelles sont souvent le résultat de traitements sur des données numériques issues de processus expérimentaux complexes, celles-ci peuvent alors contenir du bruit. Un effet du bruit va donc être de fractionner des motifs pertinents vérifiant certaines contraintes (tel que le support minimal), en un nombre exponentiel de petits fragments non pertinents. La prise en compte du bruit pour la découverte de motifs a fait l'objet d'un nombre important de travaux de recherche dans le domaine de la fouille de données. De nombreuses approches

ont été proposées (Seppänen & Mannila, 2004; Besson *et al.*, 2006; Liu *et al.*, 2006; Cheng *et al.*, 2008) afin de prendre en compte le bruit et introduisent ainsi une contrainte de *densité*. La plupart des méthodes proposées reprennent le principe de recherche par niveau de l'algorithme Apriori (Agrawal *et al.*, 1993) et sont donc limitées à l'utilisation de contraintes anti-monotones pour élaguer l'espace de recherche. Un travail qui fait exception est celui de Poernomo & Gopalkrishnan (2009) pour une contrainte de bruit proportionnelle au support du motif. Nous avons montré empiriquement dans (Mouhoubi *et al.*, 2011b,a) que HANCIM construit heuristiquement un nombre relativement restreint de régions denses qui ont une qualité comparable ou meilleure (critères supervisés ou non supervisés) que les résultats des approches de l'état de l'art proposées pour construire des (bi)-clusters à partir de données bruitées. Nous proposons ici une extension de HANCIM qui guide l'extraction de régions denses avec des connaissances, exprimées sous forme d'une classification. Cette évolution est souple : elle s'effectue en adaptant les poids des graphes bipartis pour assurer une cohérence par rapport à une classification potentiellement incomplète.

Nous introduisons dans la section suivante quelques définitions de base ainsi que le problème traité. Nous décrivons, dans la section 3., l'algorithme HANCIM et détaillons ensuite dans la section 4. notre contribution, COBIC. Nous décrivons dans la section 5. comment notre approche est évaluée et comparée et nous concluons, dans la section 6., par une brève discussion.

2. Formulation du problème

2.1. Définitions, notations

Soient A un ensemble fini d'attributs, O un ensemble fini d'observations, et R une relation binaire $R \subseteq O \times A$. R peut être modélisée par une matrice booléenne où les lignes et les colonnes correspondent respectivement aux observations et aux attributs. Un motif *m* est un sous-ensemble de A.

Une contrainte C est anti-monotone si pour toute paire E_1, E_2, $(E_1 \subseteq E_2 \wedge C(E_2)) \Rightarrow C(E_1)$ où E_1 et E_2 des ensembles d'attributs ou d'observations.

Une observation $o \in O$ supporte un motif m si tous les attributs de m apparaissent dans l'observation o. Le *support* d'un motif m, $supp(m)$, est le cardinal de l'ensemble d'observations qui supporte m. Un motif m est *fréquent*, relativement à un support minimal $minsup$, si son support est supérieur ou égal à $minsup$. Un motif fréquent est *maximal* s'il n'est sous motif d'aucun

autre motif fréquent du contexte.

Un graphe G orienté est composé d'un ensemble de sommets $V = \{v_1, v_2, \ldots, v_n\}$ et d'un ensemble d'arcs $E = \{e_1, e_2, \ldots, e_m\}$ reliant des sommets de V deux à deux. Un graphe est dit *biparti* s'il existe une partition de son ensemble de sommets en deux sous-ensembles V_1 et V_2 tel que chaque arc ait une extrémité dans V_1 et l'autre dans V_2. Nous définissons la densité d'un graphe par le rapport $\frac{|E|}{|V_1|X|V_2|}$. Un graphe est dit dense, relativement à un paramètre $\delta \in [0, 1]$, si sa densité est supérieure ou égale à δ.

La propriété suivante identifie dans un graphe biparti $G = (V_1, V_2, E)$ l'ensemble des sommets de V_1 ayant un degré important et/ou reliés à des sommets de V_2 de degrés élevés.

Définition 1
(Fortement associé) Un sommet $v_i \in V_1$ est fortement associé à un sous ensemble de sommets $v_j \in V_2$ si et seulement si
$\sum_{v_j \in V_2 \wedge d(v_j) \neq 0} (\frac{d^+(v_i)}{max_{v_k \in V_1}(d^+(v_k))} + \frac{d^-(v_j)}{max_{v_k \in V_2}(d^-(v_k))}) > max_{v_k \in V_1}(d^+(v_k))$,
tels que $d^+(v)$ et $d^-(v)$ représentent respectivement le degré entrant et sortant du sommet v.

Rappelons que notre objectif consiste à construire efficacement un nombre relativement restreint de régions denses et maximales.

Définition 2
(Région, région dense) Une région r est une paire d'ensembles (O_j, A_j) tel que $O_j \subseteq O$ et $A_j \subseteq A$ et tel que r satisfait les deux contraintes suivantes : 1) chaque attribut de A_j a une densité dans O_j supérieure ou égale à un seuil δ, 2) chaque observation de O_j est fortement associée à chaque attribut de A_j.

Notre méthode repose sur la notion de coupe minimale.

Définition 3
(st-Coupe minimale) Soit $G = (V, E)$ un graphe biparti connexe augmenté d'un sommet "source" s de degré sortant non nul et d'un sommet "destination" t de degré entrant non nul. À tout arc (x, y) est associé un entier $c(x, y)$ positif ou nul, sa capacité. Une st-coupe est une partition de V en $S \cup T$ où $s \in S$, $t \in T$. La capacité de la coupe est la somme des capacités des arcs de S vers T. Une st-coupe est minimale si sa capacité est inférieure à toute autre coupe.

2.2. État de l'art

De nombreuses approches ont été proposées afin de traiter le bruit lors de l'extraction de motifs. Ces approches peuvent être regroupées en deux sous groupes : les méthodes *complètes* et les méthodes *heuristiques*. Les méthodes complètes recherchent tous les motifs qui satisfont un ensemble de contraintes, y compris une contrainte de densité et une contrainte de support minimal, idéalement toutes les deux anti-monotones. En conséquence, ces approches se basent sur la recherche par niveau d'APRIORI et élaguent leur espace de recherche en fonction de leurs contraintes (anti-)monotones (Seppänen & Mannila, 2004; Liu *et al.*, 2006; Besson *et al.*, 2006). Signalons les travaux de Kaytoue *et al.* (2011) issus de l'Analyse Formelle de de Concepts, qui propose un méthode complète qui opère directement à partir de données numériques sur des treillis d'intervalles.

D'autres travaux (Uno & Arimura, 2008; Li *et al.*, 2008) se sont intéressés à l'énumération de toutes les bicliques maximales qui vérifient une contrainte de densité minimale, appelées quasi-bicliques. Cependant, ces méthodes s'avèrent très coûteuses en temps d'exécution et extraient un très grand nombre de résultats (redondants). Enfin, Poernomo & Gopalkrishnan (2009) définissent une contrainte de bruit proportionnelle au support du motif. Ils proposent une fonction d'élagage pour restreindre l'espace de recherche et un modèle linéaire pour le calcul du support.

Ces méthodes souffrent de limites similaires : les contraintes de densité qu'elles ont utilisées sont très strictes, le choix des paramètres reste une tâche difficile et nécessite une connaissance préalable sur les données. De plus, ces méthodes restent très coûteuses car elles sont basées sur une recherche par niveau. Enfin, elles génèrent un grand, voire énorme, nombre de motifs comportant des informations redondantes et nécessistent ainsi des étapes de post-traitement très coûteuses.

Pour surmonter ces limites, des méthodes non-complètes ou *heuristiques* ont été appliquées, tel que le biclustering (Cheng & Church, 2000; Prelic *et al.*, 2006; Deodhar *et al.*, 2009; Hanczar & Nadif, 2011).

Contrairement aux méthodes de clustering, le biclustering ne nécessite pas que les attributs d'un cluster se comportent de manière similaire sur toutes les observations. Un bicluster est défini comme un sous-ensemble d'attributs qui présentent des valeurs compatibles sur un sous-ensemble d'observations. *CC* (Cheng & Church, 2000) utilise une heuristique de recherche gloutonne pour générer des co-clusters arbitrairement positionnés et qui se recouvrent.

Leur algorithme est coûteux, il identifie séquentiellement un cocluster à la fois ce qui pourrait détériorer la qualité des biclusters résultat. L'approche *BiMax* (Prelic *et al.*, 2006) propose une méthodologie pour comparer et valider plusieurs autres méthodes de biclustering. Il propose un simple algorithme combinatoire "divide-and-conquer" qui détermine tous les groupes optimaux et maximaux, et produit ainsi un nombre de coclusters exponentiel en le nombre d'attributs et d'observations, ce qui rend son utilisation impossible dans le cas de grands jeux de données. *OPSM* (Ben-Dor *et al.*, 2002) cherche des sous-matrices dans lesquelles les niveaux d'expression de tous les gènes induisent le même ordre linéaire des expériences. Le système *ROCC* (Deodhar *et al.*, 2009), comme *CC*, génère des biclusters arbitrairement positionnés et qui se recouvrent, en utilisant une approche en deux étapes. Il élague dans un premier temps les observations non pertinentes et extrait les biclusters organisés dans une structure de grille, puis par une deuxième étape ascendante, filtre et fusionne les biclusters proches. La méthode est assez sophistiquée, mais nécessite un grand nombre de paramètres à initialiser avant d'apprendre. *SScorr* (Nepomuceno *et al.*, 2011) utilise une technique évolutionnaire s'appuyant sur une fonction d'adaptation basée sur une corrélation linéaire entre les gènes pour rechercher des biclusters susceptibles de se recouvrir. Cette approche nécessite également l'initialisation d'un nombre important de paramètres. Enfin, *Bagged Biclustering* (Hanczar & Nadif, 2011) est une autre méthode pour extraire des biclusters se recouvrant. La méthode génère d'abord K biclusters à partir de chaque ensemble de données R bootstrappé, puis construit une classification hiérarchique des biclusters obtenus. A partir de ce dendrogramme, il extrait K méta-clusters, et attribue à chacun de ces méta-clusters les exemples et les gènes qui sont fortement représentés dans ce méta-cluster. Les principales limites de cette approche est qu'elle nécessite de fixer à priori le nombre K et s'avère relativement lente, surtout pour un K relativement grand.

3. Approche proposée

Notre objectif est de construire efficacement un nombre relativement restreint de biclusters maximaux et denses sans aucune spécification sur le nombre de biclusters ni sur leur taille. Notre précédent algorithme *HANCIM* itère les deux étapes suivantes : il identifie dans un premier temps un bicluster (O_0, A_0) de densité 100% puis il utilise l'ensemble d'attributs A_0, appelé *motif graine* s, pour rechercher un bicluster dense (O_j, A_j) tel que $s \subseteq A_j$. (O_j, A_j) doit satisfaire deux contraintes : 1) chaque attribut de A_j a une densité dans O_j

supérieure ou égale à un seuil δ, 2) chaque observation de O_j est fortement associée à chaque attribut de A_j (définition 1).

Afin d'extraire un bicluster maximal dense qui inclut un motif graine donné $s \in 2^A$, un graphe biparti valué associé à s est d'abord construit, puis une s-t coupe minimale est calculée dans le graphe. Nous adaptons les capacités attribuées aux arcs du graphe de manière à récupérer, après le calcul d'une s-t coupe minimale, un sous-graphe dense qui inclut les attributs de la graine s et l'ensemble des observations O_j qui sont fortement associées à ces attributs. Lors de la prochaine étape, nous construisons le graphe correspondant aux observations O_j de manière à récupérer, après le calcul de la coupe minimale, un sous ensemble d'attributs ayant des densités supérieures à δ pour ces observations O_j. Ces deux étapes sont alternativement répétées sur les observations et les attributs jusqu'à ce que les sous-graphes denses extraits à l'étapes l et $l+1$ soient identiques ; dans ce cas, notre sous-graphe ne peut plus être étendu et le processus est arrêté. L'indice d'une étape correspond à une itération, i.e., à l'étape impaire (étape orientée attributs) et l'étape paire suivante (étape orientée observations).

3.1. Calcul des motifs graines

Nous avons proposé dans (Mouhoubi *et al.*, 2011b) une stratégie heuristique et avons démontré empiriquement que la stratégie a un meilleur comportement sur divers jeux de données. Cette stratégie vise à calculer des graines maximales avec le plus faible recouvrement possible. Afin de s'assurer que les motifs graines soient variées (dans le sens où elles sont associées à des régions non-redondantes), nous introduisons un contexte supplémentaire D_{cs}, utilisé spécifiquement pour le calcul des motifs graines. D_{cs} est initialement égal à D et est mis à jour itérativement au cours des processus d'extraction de biclusters (ligne 15 de l'alg. 1). Le motif graine s est initialisé à a, un des attributs ayant le support minimal le plus élevé dans D_{cs}. En suivant le même principe, nous recherchons itérativement un attribut a qui forme avec les attributs de s un motif de support le plus élevé possible et tel que $support(D_{cs}, s \cup a) \geq supmin$ et maximal. Le processus est arrêté lorsqu'aucun autre attribut ne peut être ajouté. À chaque fois qu'un bicluster (O_j, A_j) est calculé, chaque élément de ce bicluster est mis à zéro dans D_{cs}. Afin de minimiser la perte d'information, le support minimal dans le contexte D_{cs} est calculé de manière adaptative (ligne 18 de l'alg. 1). La densité de D_{cs} diminue, ainsi que le support des biclusters qui se recouvrent dans D_{cs}, encourageant le calcul de divers motifs

graines, et donc le calcul de biclusters denses non-redondants. Dans ce qui suit, $support(D, m)$ désigne le support du motif m dans le contexte D.

3.2. Extraction d'un bicluster maximal qui inclut un motif graine

Dans le but d'extraire un bicluster dense et maximal qui inclut un motif graine donné $s_i \in 2^A$, nous construisons dans un premier temps le graphe biparti et valué correspondant à s_i (ligne 6 de l'alg. 1), puis nous calculons une s-t coupe minimale dans le graphe, en utilisant l'algorithme de flot maximal "push-relabel" de (Cherkassky & Goldberg, 1997) puisque la valeur maximale du flot dans un graphe de s à t est égale à la capacité minimale d'une coupe séparant s de t (théorème flot-max/coupe-min).

Les capacités affectées aux arcs du graphe sont adaptées de manière à récupérer, après le calcul de la s-t coupe minimale, un sous graphe dense qui comporte les sommets attributs de s_i et un ensemble d'observations O_l qui sont fortement associées à ces attributs. Lors de la prochaine étape, nous construisons le graphe correspondant aux observations O_l (ligne 10 de l'alg. 1) de manière à récupérer, après le calcul de la coupe minimale (ligne 11), un sous ensemble d'attributs ayant des densités supérieures à δ pour ces observations O_l. Ce processus est répété jusqu'à ce que le sous graphe dense extrait à l'étape l soit identique à celui extrait à l'étape $l-1$ (ligne 8). Dans ce cas notre sous graphe ne peut plus être augmenté et le processus est arrêté. Pour plus de détails sur l'initialisation des capacités, nous invitons le lecteur à voir (Mouhoubi *et al.*, 2011b,a).

Nous verrons dans la prochaine section que ces capacités peuvent être pondérées pour tenir compte d'une classification de référence.

4. Biclustering contraint

Nous présentons dans cette section notre méthodologie, nommée *COBIC* (COnstrained BI-Clustering), pour l'extraction de biclusters pertinents dans des contextes bruités. Notre objectif dans ce travail consiste à exploiter des informations sous forme d'une classification de référence C_{ref} pour guider et adapter l'extraction des régions denses. Cette adaptatation se fait en apprenant les poids lors de la construction des graphes. Cette adaptation est souple puisqu'elle n'impose pas que les biclusters appris soient forcément en complète cohérence avec la classification de référence mais favorise plutôt l'apprentissage de biclusters cohérents par rapport à une classification potentiellement

Algorithme 1 : L'algorithme d'extraction de biclusters

input : D : Matrice de données, δ : densité minimale de chaque attribut, σ : support minimal, C_A : Classification de A, C_O : Classification de O
output : Co_Bic : Ensemble des biclusters contraints

begin
  $Co\_Bic = \emptyset$; $D_{cs} = D$; $\sigma' = \sigma$; $i = 0$; $density$ = DENSITY(D_{cs});
  s_i = COMPUTE_SEED(D_{cs}, σ');
  while *($s_i \neq \emptyset$)* **do**
    $l = 0$;
    G_l = CONSTRUCT_GRAPH_COBIC(D, s_i, δ, C_O, $\emptyset$, $\emptyset$) ; /* Algorithme 2 */
    (O_l, A_l)= MINIMAL_ST-CUT(G_l);
    while *(G_l != G_{l-1})* **do**
      l++;
      G_l=CONSTRUCT_GRAPH_COBIC(D,O_l, δ, C_A, A_{l-1}, A_{l-2}) ; /* Algorithme 2 */
      (O_l, A_l)= MINIMAL_ST-CUT(G_l);
      G_l=CONSTRUCT_GRAPH_COBIC(D,A_l, δ, C_O, O_{l-1}, O_{l-2}) ; /* Algorithme 2 */
      (O_l, A_l)= MINIMAL_ST-CUT(G_l);
    $Co\_Bic = Co\_Bic \cup (O_l, A_l)$;
    SET_TO_ZERO(D_{cs}, O_l, A_l);
    $density$' = DENSITY(D_{cs});
    if *($\frac{\sigma \times density'}{density} \geq 0.1$)* **then**
      $\sigma' = \frac{\sigma \times density'}{density}$;
    i++ ; s_i = COMPUTE_SEED(D_{cs}, σ');
end

imcomplète C_{ref}.

Définition 4
Soit Y un ensemble fini d'éléments (attributs ou observations). Une classification C_{ref} est définie comme un ensemble de classes C_i , $1 \leq i \leq n$, chaque C_i étant un sous-ensemble de Y ($C_{ref} \subseteq 2^Y$).

Aucune contrainte n'est imposée sur la structure de la classification : la classification peut être incomplète (certains éléments de Y peuvent n'appartenir à aucun cluster de C_{ref}), certaines classes de C_{ref} peuvent se recouvrir et par conséquent il se peut qu'un élément $y \in Y$ appartienne à plusieurs classes (ce qui est le cas de certains gènes dans des ontologies comme Gene Ontology (GO) (Cherry *et al.*, 1998) et Kyoto Encyclopedia of Genes and Genomes (KEGG) (Kanehisa & Goto, 2000).

4.1. Apprentissage des poids

Nous détaillons dans l'algorithme 2 la construction d'un graphe pondéré pour un ensemble d'éléments (attributs ou observations) noté X_l à la l^{eme} étape. L'objectif consiste à extraire un ensemble d'attributs satisfaisant la contrainte de densité minimale après le calcul de la coupe minimale. Lorsque $l > 2$, nous calculons une mesure de qualité pour les deux ensembles d'éléments Y_{l-1} et Y_{l-2} extraits lors des deux dernières itérations $l-1$ et $l-2$. La qualité d'un ensemble Y_i est évaluée en fonction de sa similarité avec toutes les classes $C_i \in C_Y$ définies sur Y. Rappelons que, à chaque étape, notre objectif est de guider la recherche de biclusters afin de favoriser le calcul d'ensembles d'éléments cohérents avec une classification potentiellement incomplète C_Y, et non pas avec une seule classe de C_Y.

Compte tenu du comportement de notre algorithme lors des deux dernières itérations, nous vérifions la similarité des ensembles Y_{l-2} et Y_{l-1} avec des classes de C_Y, notées $sim_{Y_{l-2}}$ et $sim_{Y_{l-1}}$ dans ce qui suit. Lorsque la similarité de l'ensemble Y_{l-1} extrait lors de l'itération $l-1$ est meilleure que la similarité de Y_{l-2}, l'algorithme se comporte bien et dans ce cas nous construisons le graphe correspondant à X_l comme nous le faisions dans ***HANCIM*** (ligne 24 de l'algorithme 2). Dans le cas contraire, nous utilisons les valeurs de similarité pour pondérer les capacités des arcs du graphe. En effet, si la similarité de Y_{l-1} est inférieure à celle de Y_{l-2}, cela signifie que notre solution s'éloigne de la classification C_Y ce qui est dû soit : 1) à des éléments de Y_{l-2} qui ont été retirés dans Y_{l-1} et/ou 2) à de nouveaux éléments ajoutés à Y_{l-1}. Dans ce cas, à l'étape l, en pondérant les poids des arcs incidents aux sommets y_j appartenant à $Y_{l-1} \setminus Y_{l-2}$ par la valeur de la similarité $sim_{Y_{l-1}}$, on pénalise ces sommets y_j, et en pondérant les poids des arcs incidents aux sommets y_k appartenant à Y_{l-2} par la valeur de la similarité $sim_{Y_{l-2}} > sim_{Y_{l-1}}$, nous indiquons notre préférence pour ces sommets y_k comparés aux sommets y_j.

Pour ce faire, nous pondérons les poids des arcs (x_i, y_j) tel que $y_j \in Y_{l-1} \setminus Y_{l-2}$ par la valeur de la similarité $sim_{Y_{l-1}}$ et les arcs (x_i, y_j) tel que $y_k \in Y_{l-2}$ par la valeur de la similarité $sim_{Y_{l-2}}$. Sachant que les poids des arcs lors de la construction d'un graphe associé à un ensemble d'attributs diffèrent de ceux utilisés lors de la construction d'un graphe pour un ensemble d'observations, les poids originaux, proposés dans (Mouhoubi *et al.*, 2011b) sont définis comme suit :

1. si $X_l \subseteq O$:
   - $W_{x_i y_j} = \frac{100}{|X_l|}$ (lignes 14, 17, 19 de l'algorithme 2) et

– $W_{y_j t}$=2 × (100 × δ) - $weight^{-}(y_j)$ (ligne 22 de l'algorithme 2).

2. si $X_l \subseteq A$:
   – $W_{x_i y_j}$=($\frac{d^{+}(x_i)}{max_{x_k \in X_l}(d^{+}(x_k))}$ + $\frac{d^{-}(y_j)}{max_{y_k \in Y_j}(d^{-}(y_k))}$) x $\frac{100}{|X_l|}$ (lignes 14, 17, 19 de l'algorithme 2) et
   – $W_{y_j t}$= $max_{y_k \in Y_j}(d^{-}(y_k))$ × $\frac{200}{|X_l|}$ - $weight^{-}(y_j)$ (ligne 22 de l'algorithme 2).

Comme $sim_{Y_{l-1}} < sim_{Y_{l-2}}$, en pondérant les poids des arcs (x_i, y_j) tel que $y_j \in Y_{l-1} \setminus Y_{l-2}$ par la valeur de la similarité $sim_{Y_{l-1}}$, les poids de ces arcs (x_i, y_j) sont considérablement réduits et donc la possibilité de couper ces arcs et ainsi de supprimer les sommets $y_j \in Y_{l-1} \setminus Y_{l-2}$ de Y_l est augmentée.

Algorithme 2 : CONSTRUCT_GRAPH_COBIC

input : $D = (O, A)$: matrice des données, X_l : Ensemble de sommets ($X_l \subseteq A$ ou $X_l \subseteq O$) et $l > 2$, δ : densité minimale, C : Classification de Y (si $X_l \subseteq A$ alors $Y = O$, sinon $Y = A$), $Y_{l-1} \subseteq Y$: l'ensemble de sommets extraits à l'étape $l - 1$, $Y_{l-2} \subseteq Y$: l'ensemble de sommets extraits à l'étape $l - 2$
output : $G(V, E)$: le graphe construit

```
1  begin
2      sim_{Y_{l-1}} = SIMILARITY(Y, C_Y, Y_{l-1});
3      sim_{Y_{l-2}} = SIMILARITY(Y, C_Y, Y_{l-2});
4      if (sim_{Y_{l-1}} < sim_{Y_{l-2}}) then
5          V= X_l ∪ {s, t} ;
6          forall x_i ∈ X_l do
7              E=E ∪ (s, x_i);
8              weight(s, x_i)= +∞ ;
9          forall x_i ∈ X_l do
10             forall y_j s.t. D[x_i][y_j] == 1 do
11                 V=V ∪ y_j ;
12                 E=E ∪ (x_i, y_j) ;
13                 if ( y_j ∈ Y_{l-2}) then
14                     weight(x_i, y_j)= W_{x_i y_j} × sim_{Y_{l-2}} ;
15                 else
16                     if ( y_j ∈ Y_{l-1}) then
17                         weight(x_i, y_j)= W_{x_i y_j} × sim_{Y_{l-1}} ;
18                     else
19                         weight(x_i, y_j)= W_{x_i y_j} ;
20         forall y_j ∈ V\(X_l ∪ {s, t}) do
21             E=E ∪ (y_j, t) ;
22             weight (y_j, t)= W_{y_j t};
23     else
24         CONSTRUCT_GRAPH_HANCIM (cf (Mouhoubi et al., 2011b));
25 end
```

4.2. Calcul de similarité

Comme indiqué précédemment, afin de guider la recherche de biclusters vers des solutions pertinentes, nous exploitons une classification (par exemple,

GO ou KEGG ontologies fonctionnelles). Nous présentons ici notre méthodologie pour calculer la similarité moyenne entre un ensemble d'attributs (respectivement d'observations) et certaines classes d'attributs (respectivement d'observations). Considérons Y un ensemble d'éléments (attributs ou observations), C_Y une classification de Y (au sens de la définition 4) et Sub_Y un sous-ensemble de Y. Pour calculer la similarité entre deux ensembles, nous utilisons la similarité de Jaccard. Comme la similarité de Jaccard est définie pour être appliquée entre des partitions, nous calculons notre similarité entre $Sub_Y \subset Y$ et $C_i \in C_Y$ comme la similarité entre les partitions $(Sub_Y, Y \setminus Sub_Y)$ et $(C_i, Y \setminus C_i)$. Ainsi, nous définissons la similarité de Jaccard comme suit :

$$Sim\_JACCARD(Y, Sub_Y, C_i) = \frac{N_{01} + N_{10}}{N_{01} + N_{01} + N_{11}}$$

où $N_{11} = |Sub_Y \cap C_i|$, $N_{01} = |C_i \setminus Sub_Y|$, $N_{10} = |Sub_Y \setminus C_i|$ et $N_{00} = |(Y \setminus Sub_Y) \cap (Y \setminus C_i)|$.

Comme indiqué précédemment, nous utilisons plusieurs classes de C_Y pour guider notre recherche, nous calculons la moyenne des k meilleures valeurs de similarités pour tous les clusters de la classification. Nous avons fixé empiriquement k à 5 dans nos expérimentations.

5. Expérimentations et résultats

Tous les tests ont été réalisés sur un PC sous Linux avec un processeur Intel(R) Pentium(R) 4 (3 GHz) et une RAM de 2Go. Dans ce travail, nous avons choisi d'évaluer la performance de *COBIC* sur deux jeux de données d'expression de levure, les données de Gasch (Gasch *et al.*, 2000) et les données de Lee (Lee *et al.*, 2004). Le choix de ces jeux de données est motivé par la disponibilité de classifications de référence correctes pour guider le processus de biclustering, ainsi que par des objectifs de comparaison avec l'état de l'art en biclustering. Le jeu de données $Gasch$ mesure l'expression de 6152 gènes de levure sous 173 conditions environnementales de stress. Les données de Lee comporte les valeurs d'expression de 5612 gènes de la levure pour 592 expériences.

Pour évaluer la pertinence biologique des biclusters extraits, à partir de ces données réelles, nous nous sommes appuyés sur différentes mesures mesurant la cohérence par rapport à des ontologies de référence comme GO (Cherry *et al.*, 1998) et KEGG (Kanehisa & Goto, 2000) :

– l'enrichissement des biclusters extraits avec des termes GO, comme

dans (Birmele *et al.*, 2008), où une p-value mesure la probabilité qu'une liste de gènes de même taille tirée aléatoirement présente la même proportion de gènes ayant une fonction GO donnée. Un enrichissement est d'autant plus significatif que cette p-value est basse.

– le pourcentage des gènes de la matrice des données qui appartiennent à au moins un bicluster.

Nous avons comparé les performances de *COBIC* avec des algorithmes performants de biclustering, tel que *HANCIM* (Mouhoubi *et al.*, 2011b), *SS-corr* (Nepomuceno *et al.*, 2011), Bagged Biclustering (Hanczar & Nadif, 2011) et *ROCC* (Deodhar *et al.*, 2009). Grâce à des expérimentations approfondies, Prelic *et al.* (2006) ont déjà montré que *OPSM* (Ben-Dor *et al.*, 2002) et *BiMax* (Prelic *et al.*, 2006) obtiennent de meilleurs résultats que les autres algorithmes les plus connus de biclustering. Récemment, Deodhar *et al.* (2009) ont montré que *ROCC* donne de meilleurs résultats que *OPSM* et *BiMax* ; enfin Mouhoubi *et al.* (2011b) ont montré que les résultats obtenus par *HANCIM* sont meilleurs que ceux obtenus avec *BiMax* que ce soit sur des données synthétiques ou réelles.

Comme dans (Deodhar *et al.*, 2009), afin de se comparer à *ROCC* et *SS-corr*, nous sélectionnons nos 200 meilleurs résultats (avec les plus petites p-values, la dernière colonne des tableaux 1, 2 et 3). Pour *COBIC* et *HANCIM*, nous avons utilisé le modèle de discrétisation décrit dans (Prelic *et al.*, 2006) et un seuil fixé à $e_{min} + (e_{max} - e_{min})/2$ où e_{min} et e_{max} représentent les valeurs minimales et maximales d'expression des gènes dans la matrice des données. Nous avons appliqué *COBIC* et *HANCIM* sur Gasch discrétisé et Lee avec un support minimal de 20% et une densité minimale de 80%.

Nous avons choisi KEGG comme classification de référence pour la phase d'adaptation des poids de *COBIC* pour les données de Gasch et de Lee, et GO pour les données de cancer du poumon. Pour tous les jeux de données, nous avons évalué les résultats évaluant l'enrichissement en terme GO. GO contient plus de clusters que KEGG : pour la levure, la base GO utilisée contient 4227 clusters de taille moyenne 14 ($min = 2$ et $max = 254$), tandis que KEGG contient 99 clusters de taille moyenne 37 ($min = 2$ et $max = 253$) ; pour l'humain, GO contient 6803 clusters de taille moyenne 12 ($min = 2$ et $max = 353$). Cela signifie que des informations incomplètes mais correctes sont suffisantes pour améliorer les résultats par rapport à *HANCIM*.

Comme prévu, les temps de calcul sont assez faibles, de l'ordre de quelques minutes (3 min pour Gasch restreint et 35 min pour Gasch complet), sauf pour les données de Lee où *COBIC* a pris plus de temps (280 min). En effet,

le nombre de résultats sur les données de Lee est presque le double que celui obtenu pour les données Gasch, et le nombre moyen d'itérations pour converger vers chaque résultat est plus important sur les données de Lee. De plus, on remarque que chaque itération de *COBIC* prend en moyenne plus de temps sur la matrice des données de Lee que sur celle de Gasch puisque les données Lee contiennent presque $3,5$ fois plus de gènes que les données de Gasch. En général, nous avons constaté que le temps supplémentaire induit par la phase d'apprentissage des poids est limité à environ 20% du temps de calcul total pour tous les jeux de données testés.

5.1. Résultats sur les données de Gasch

Nous avons comparé dans un premier temps *COBIC* avec *HANCIM* sur le jeu de données Gasch restreint (2993×173) utilisé dans (Prelic *et al.*, 2006). La table 1 résume le pourcentage de résultats enrichis avec les termes de l'ontologies GO, dans lesquels au moins un terme GO est sur-représenté pour les différents niveaux de p-values. Comme on peut le constater à partir du tableau 1, *COBIC* a obtenu de meilleurs résultats par rapport à *HANCIM*.

	HANCIM	*COBIC*	*COBIC* 200 meilleurs
# results	548	554	200
$<e^{-2}$	94%	97.3%	100%
$<e^{-3}$	48%	58%	100%
$<e^{-4}$	28%	33.7%	93.5%
$<e^{-5}$	18%	24%	66.5%
$<e^{-10}$	7%	10%	28%
$<e^{-20}$	4%	6.5%	18%
Meilleure p-value	e^{-38}	e^{-64}	e^{-64}

TABLE 1: Résultats d'*HANCIM* et *COBIC* sur les données de Gasch restreint

Nous observons le même comportement si l'on compare *COBIC* avec les résultats de *SScorr*, de plus, on remarque que *COBIC* est considérablement meilleur que *SScorr*. Sur ces données de Gasch restreint, on trouve que pour les 100 meilleurs résultats de *SScorr*, le pourcentage de biclusters enrichis avec une p-value inférieure à 0.01 (resp. 0.001) est inférieur à 30% (resp. 20%). Si nous analysons maintenant nos 200 meilleurs biclusters (voir tableau 1), 100% de nos biclusters enrichis ont une p-value inférieure à 0.001.

Nous présentons à présent les résultats obtenus sur les données de Gasch complet (6152×173). Comme on peut le voir, les résultats obtenus sur ces données sont meilleurs que ceux obtenus sur le jeu de Gasch restreint. Toutes les p-values associées aux 200 meilleurs biclusters obtenus avec *COBIC* sont

inférieures à e^{-10} et la meilleure p-value est égale à e^{-98}. Notons que le pourcentage de gènes de la matrice des données qui sont apparus dans les 200 meilleurs résultats extraits par ***ROCC*** est de 0.9%, alors que nous avons regroupé avec ***COBIC*** 51.4% des gènes dans nos 200 meilleurs résultats.

	COBIC	*COBIC* 200 best
# results	1075	200
$<e^{-2}$	96%	100%
$<e^{-3}$	60%	100%
$<e^{-4}$	41.5%	100%
$<e^{-5}$	35.6%	100%
$<e^{-10}$	25%	100%
$<e^{-20}$	15%	83%
Meilleure p-value	e^{-98}	e^{-98}

TABLE 2: Résultats de *COBIC* sur Gasch complet

Comme on peut le voir, les résultats obtenus sur le jeu de données de Gasch complet sont meilleurs que ceux obtenus sur le jeu de Gasch restreint. Toutes les p-values associées aux 200 meilleurs biclusters obtenus avec ***COBIC*** sont inférieures à e^{-10} et la meilleure p-value est égale à e^{-98}. Notons que le pourcentage de gènes de la matrice des données qui sont apparus dans les 200 meilleurs résultats extraits par ***ROCC*** est de 0.9%, alors que nous avons regroupé avec ***COBIC*** 51.4% des gènes dans nos 200 meilleurs résultats.

5.2. Résultats sur les données de Lee

La table 3 donne le pourcentage de biclusters enrichis pour ***COBIC***. Comme nous pouvons le voir, toutes les p-values associées aux 200 meilleurs biclusters obtenus avec ***COBIC*** sont inférieures à e^{-03} et les meilleures valeurs de p-value, $5.46e^{-43}$ et $1.58e^{-34}$. Notons que le pourcentage de gènes représentés dans au moins un bicluster parmis les 200 meilleurs résultats de ***ROCC*** est de 16.5% pour les données Lee, tandis 30% des gènes sont représentés dans les 200 meilleurs résultats de ***COBIC***.

6. Conclusion

Nous avons présenté dans cet article une nouvelle approche, $COBIC$, pour la recherche de biclusters contraints. La méthode proposée est basée sur les algorithmes de flot maximal/coupe minimale et est guidée par une base de connaissances exprimée sous forme d'une classification. L'approche a été

	COBIC	*COBIC* 200 meilleurs
# résultats	1969	200
$<e^{-2}$	86%	100%
$<e^{-3}$	34.5%	100%
$<e^{-4}$	11%	77%
$<e^{-5}$	6%	40%
$<e^{-10}$	1.5%	8.5%
$<e^{-20}$	0.2%	2%
Meilleure p-value	e^{-43}	e^{-43}

TABLE 3: Résultats de *COBIC* sur les données de Lee

testée sur divers jeux de données réels de la bioinformatique. Une comparaison de l'algorithme proposé avec des méthodes de biclustering concurrentes été effectuée et démontre que $COBIC$ est très compétitif par rapport aux meilleurs méthodes de biclustering existantes. En guise de perspectives, nous comptons étudier l'applicabilité de notre approche dans d'autres domaines, comme par exemple l'extraction de communautés dans les réseaux sociaux.

Références

AGRAWAL R., IMIELINSKI T. & SWAMI A. N. (1993). Mining association rules between sets of items in large databases. In *Proc. SIGMOD*, p. 207–216.

BEN-DOR A., CHOR B., KARP R. & YAKHINI Z. (2002). Discovering local structure in gene expression data : the order-preserving submatrix problem. In *Proc. RECOMB*, p. 49–57.

BESSON J., ROBARDET C. & BOULICAUT J.-F. (2006). Mining a new fault-tolerant pattern type as an alternative to formal concept discovery. In *Proc. ICCS*, p. 144–157.

BIRMELE E., ELATI M., ROUVEIROL C. & AMBROISE C. (2008). Identification of functional modules based on transcriptional regulation structure. *BMC Proc.*, **2**(Suppl 4), S4.

CHENG H., YU P. S. & HAN J. (2008). Approximate frequent itemset mining in the presence of random noise. *Soft Computing for Knowledge Discovery and Data Mining*, p. 363–389.

CHENG Y. & CHURCH G. (2000). Biclustering of expression data. In *Proc. Int. Conf. Intell. Syst. Mol. Biol.*, p. 8 :93–103.

CHERKASSKY B. V. & GOLDBERG A. V. (1997). On implementing the push-relabel method for the maximum flow problem. *Algorithmica*, **19**(4), 390–410.

CHERRY J. M., ADLER C., BALL C. & CHERVITZ S. A. (1998). SGD : Saccharomyces genome database. *Nucleic Acids Research*, **26**(1), 73–79.

DEODHAR M., GUPTA G., GHOSH J., CHO H. & DHILLON I. S. (2009). A scalable framework for discovering coherent co-clusters in noisy data. In *Proc. ICML*, p.31̃.

GASCH A., SPELLMAN P., KAO C., CARMEL-HAREL O., EISEN M., STORZ G., BOTSTEIN D. & BROWN P. (2000). Genomic expression programs in the response of yeast cells to environmental changes. *Mol. Biol. Cell*, **11**(12), 4241–57.

HANCZAR B. & NADIF M. (2011). Using the bagging approach for biclustering of gene expression data. *Neurocomputing*, **74**(10), 1595–1605.

KANEHISA M. & GOTO S. (2000). KEGG : Kyoto encyclopedia of genes and genomes. *Nucleic Acids Res*, **28**(1), 27–30.

KAYTOUE M., KUZNETSOV S. O., NAPOLI A. & DUPLESSIS S. (2011). Mining gene expression data with pattern structures in formal concept analysis. *Inf. Sci.*, **181**(10), 1989–2001.

LEE I., DATE S., ADAI A. & MARCOTTE E. (2004). A probalistic functionnal network of yeast genes. *Science*, **306**(5701), 1555–1558.

LI J., SIM K., LIU G. & WONG L. (2008). Maximal quasi-bicliques with balanced noise tolerance. In *Proc. SDM 2008*, p. 72–83.

LIU J., PAULSEN S., SUN X., 0010 W. W., NOBEL A. B. & PRINS J. (2006). Mining approximate frequent itemsets in the presence of noise : Algorithm and analysis. In *Proc. SDM 2006*, p. 405–416.

MOUHOUBI K., LÉTOCART L. & ROUVEIROL C. (2011a). Extraction de motifs ensemblistes dans des contextes bruités. In *CAP 2011*, p. 311–326.

MOUHOUBI K., LÉTOCART L. & ROUVEIROL C. (2011b). Itemset mining in noisy contexts : A hybrid approach. In *Proc. ICTAI 2011*, p. 33–40.

NEPOMUCENO J., LORA A. T. & AGUILAR-RUIZ J. (2011). Biclustering of gene expression data by correlation-based scatter search. *BioData Min.*, **4**(3).

POERNOMO A. K. & GOPALKRISHNAN V. (2009). Towards efficient mining of proportional fault-tolerant frequent itemsets. In *Proc. KDD*, p. 697–706.

PRELIC A., BLEULER S., ZIMMERMANN P., WILLE A., BÜHLMANN P. & GRUISSEM W. (2006). A systematic comparison and evaluation of biclustering methods for gene expression data. *Bioinformatics*, **22**(9), 1122–1129.

SEPPÄNEN J. K. & MANNILA H. (2004). Dense itemsets. In *Proc. KDD*, p. 683–688.

UNO T. & ARIMURA H. (2008). Ambiguous frequent itemset mining and polynomial delay enumeration. In *Proc. PAKDD 2008*, p. 357–368.