

Mémoire de Master Recherche
Une nouvelle méthode de clustering appliquée à un projet
bioinformatique

Alexis Lamiable

2003-2004

Université de Bordeaux 1

Encadrants :

Isabelle Dutour (LaBRI) et Antoine de Daruvar (CBiB, Bx 2)

Table des matières

1	Introduction	2
2	Modélisation	4
2.1	Règles d'association	4
2.2	Graphes de règles d'association	5
2.3	Analyse des graphes	6
3	Première approche : connexité	8
3.1	Composantes fortement connexes	8
3.2	Génération de candidats	8
3.3	Critique des composantes fortement connexes	10
4	Résultats	13
4.1	Jeux de données et validation	13
4.2	Requêtes sur des données mixtes	13
4.3	Requêtes sur les classifications systématiques	16
4.4	Requêtes sur les données d'expression	18
4.5	Requêtes basées sur les améliorations proposées	20
4.6	Conclusion	21
5	Autres approches envisageables	22
5.1	Heuristiques de "déplacement"	22
5.2	Algorithmes de clustering	24
5.3	Modifications du graphe de règles d'association	24
6	Conclusion	26
	Références	27

1 Introduction

Le projet de bioinformatique BlastSets [1] a pour objectif l'intégration d'informations hétérogènes au sein d'un système informatique, dans le but de faire apparaître des correspondances entre ces informations. Les données y sont représentées de manière unifiée sous la forme de groupes de gènes, rassemblés dans une base de données. Un groupe peut être formé, par exemple, de tous les gènes qui produisent des enzymes intervenant au sein d'une même chaîne de réactions biochimiques. La comparaison entre un *groupe requête* et un sous-ensemble de cette base permet d'en faire ressortir les groupes significativement ressemblants à ce groupe requête.

La comparaison entre deux groupes ayant n gènes en commun est une réponse à la question « est-il probable que ces n gènes communs soient dus au hasard ? ». Le test utilisé est basé sur la loi hypergéométrique, qui représente le tirage sans remise de deux sous-populations. Cette comparaison produit une *pvalue* qui est d'autant plus faible que la similarité entre les deux groupes est forte. La comparaison entre un groupe requête et un ensemble de groupes cibles produit donc un ensemble de *hits* dont seule une partie est significative.

La base de données est constituée de classifications, qui sont des ensembles de groupes constitués à partir du même critère biologique (complexes protéiques, position voisine sur les chromosomes, même voie métabolique, même fonction, etc.). Les groupes sont liés les uns aux autres au sein de leur classification, suivant une structure de graphe orienté sans cycle (DAG), dans lequel un sommet représente un groupe, et un arc représente une relation d'inclusion. Ainsi, au sein d'une même classification, les groupes n'ont pas des compositions indépendantes. Lorsque l'on effectue une requête contre une classification, le groupe requête est comparé à l'ensemble des groupes de cette classification.

L'utilisation « typique » de BlastSets consiste à soumettre un groupe de gènes afin de faire ressortir ceux de la base de données qui lui sont similaires, pour établir des correspondances entre ces données hétérogènes. Les gènes du groupe requête ont été réunis en fonction d'un critère (même voie métabolique, même fonction, même localisation de leur produit dans la cellule), et l'on voudrait savoir quelles autres propriétés ils ont en commun. Ainsi, un groupe requête constitué des gènes codant pour l'hémoglobine (assemblage de protéines, groupe constitué à partir des informations d'interaction physique) pourrait apparaître similaire à un groupe de la base annoté comme participant au transport de l'oxygène (annotation fonctionnelle, groupe constitué à partir des mots-clés communs à chacune des protéine).

L'objectif de ce mémoire est d'utiliser cet ensemble d'informations hétérogènes, et le fait que nous possédons un moyen de les mettre en relation les unes par rapports aux autres, pour faire émerger de la base de données des groupes n'en faisant pas partie, mais ayant une pertinence biologique, ceci de manière automatique, en nous appuyant sur des techniques de data mining (fouille de données) ainsi que sur la théorie des graphes. En effet, la diversité des critères utilisés pour la formation des groupes engendre une certaine redondance dans la composition des groupes, et les gènes ayant le plus de d'importance pour la cellule devrait être rassemblés « souvent » pour des raisons variées. De plus, les gènes étant regroupés pour des raisons différentes, on s'attend à trouver des ensembles de gènes ayant une relation forte mais regroupés avec des gènes « satellites » dûs au critère de regroupement.

Pour cette raison, nous recherchons aussi bien des groupes se trouvant déjà dans la base de données que de nouveaux groupes, ne s'y trouvant pas mais présentant une forte similarité avec les groupes présents. Nous espérons d'une part réussir à mettre en évidence de manière

automatique des groupes déjà bien caractérisés par les biologistes, d'autre part confirmer ou compléter des groupes hypothétiques, et enfin extraire des groupes encore inconnus des biologistes et correspondant à des systèmes biologiques non documentés.

Nous avons fait notre étude sur le génome de la levure, qui est un organisme de référence. Ce génome comprenant 5 786 gènes, il n'est pas envisageable de faire une recherche exhaustive sur les 2^{5786} groupes possibles. Il s'agit donc de définir des méthodes permettant en un temps raisonnable de construire des groupes candidats à partir des informations contenues dans la base de données, puis de valider ces méthodes.

Dans une première partie, nous décrirons la formalisation d'un cadre dans lequel nous pourrions représenter des associations entre les gènes, à l'aide de la théorie des graphes. Puis, nous développerons une première approche de génération de groupes de candidats à partir de propriétés de ces graphes. Ensuite, nous analyserons la pertinence des candidats produits au regard des groupes similaires extraits par BlastSets et, enfin, nous étudierons les différentes manières de dépasser les limitations de cette approche.

Remerciements : Isabelle Dutour, Antoine de Daruvar et Roland Barriot pour leur aide et leurs conseils.

2 Modélisation

On peut imaginer deux approches permettant de créer ou extraire des groupes intéressants à partir de la base de données. D'une part, une approche descendante (*top-down*) consisterait à partir d'un groupe contenant l'ensemble des gènes de l'organisme étudié, et à améliorer ce groupe en enlevant des gènes par étapes successives. Cela nécessiterait de définir un critère permettant de dire qu'un gène ne fera pas partie d'un « bon » groupe, et d'avoir une vision précise de l'évolution (en termes de qualité) d'un groupe lorsqu'on lui retire un gène.

D'autre part, une approche ascendante (*bottom-up*) consisterait à partir de petits groupes, et à les améliorer en y insérant des gènes. Il est par exemple envisageable de commencer avec des groupes contenant chacun un des gènes de l'organisme étudié, et de réunir des groupes entre eux par étapes successives si leur réunion améliore la « qualité » du groupe, c'est à dire d'utiliser un algorithme de clustering.

Pour pouvoir appliquer l'une ou l'autre de ces méthodes, nous devons disposer d'un moyen d'estimer la pertinence du lien qui unit deux gènes, ou deux ensembles de gènes. Pour cela, nous allons avoir recours à une technique de *data mining*, les *règles d'association*, pour construire un graphe représentant ces liens.

2.1 Règles d'association

Les règles d'association [2, 3] permettent de détecter des associations entre valeurs dans de grands ensembles de données. Les associations mises en évidence de cette manière sont du type « les clients qui ont acheté le produit A ont souvent également acheté le produit B ». Dans le cas qui nous préoccupe, l'existence d'une règle associant un gène A à un gène B indique que le gène A est souvent accompagné du gène B , où le terme « souvent » doit être interprété comme le *niveau de confiance* de la règle.

Soient deux gènes A et B . Le *support* de A est la fréquence d'apparition de A dans la base de données de groupes, et le support de (A, B) est la fréquence de co-apparition de A et de B dans un même groupe (ou la probabilité qu'un groupe contiennent A et B).

$$Supp(A) = \frac{\#A}{|Base|}$$

La *confiance* de (A, B) est la probabilité d'avoir B dans un groupe sachant qu'on a A dans ce groupe.

$$Conf(A \rightarrow B) = \frac{Supp(A, B)}{Supp(A)}$$

En fixant un support et une confiance minimaux s et c , on établit une règle d'association entre A et B (notée $A \rightarrow B$) si $Supp(A, B) \geq s$ et $Conf(A, B) \geq c$.

Prenons par exemple la liste de groupes suivante :

Groupe	Liste de gènes
1	A, B, C
2	A, C
3	A, D
4	B, E, F

Après un parcours de la table, on ne conserve que les ensembles d'éléments dont le support est au moins égal au seuil fixé (ici, 50 %). Nous obtenons alors la liste suivante :

Ensembles fréquents	Support
A	75 %
B	50 %
C	50 %
A, C	50 %

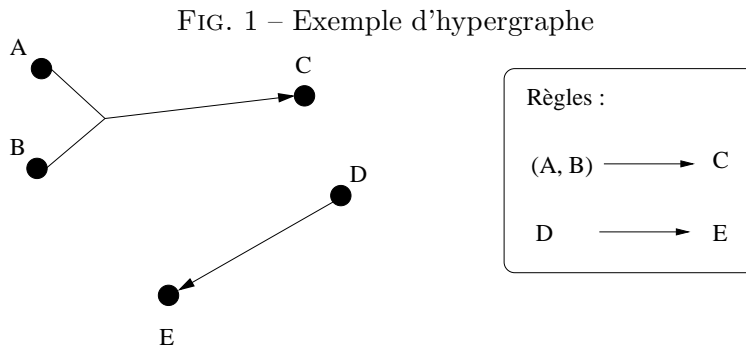
Afin de savoir s'il existe une règle d'association ($A \rightarrow C$), nous calculons la confiance de cette règle :

$$\begin{aligned}
 Conf(A \rightarrow C) &= \frac{Supp(A,C)}{Supp(A)} \\
 &= \frac{0.5}{0.75} \\
 &= 66.67 \%
 \end{aligned}$$

Si cette valeur est supérieure au seuil de confiance que nous avons fixé, nous retenons cette règle d'association.

2.2 Graphes de règles d'association

Pour un support et une confiance minimaux donnés, il est possible de modéliser l'ensemble des règles d'association de la base de données sous la forme d'un hypergraphe orienté où chaque gène est représenté par un sommet, et où il existe un hyperarc entre deux ensembles de sommets a et b si et seulement si il existe une règle d'association ($a \rightarrow b$).



Toutefois, afin de simplifier le calcul des règles et l'analyse du graphe, nous nous limiterons aux règles d'associations binaires (associant un gène à un autre). De cette manière, le calcul des règles peut se faire en une seule passe dans la base de données, et le graphe obtenu n'est pas un hypergraphe mais un simple graphe orienté.

Nous allons maintenant voir un exemple de création d'un graphe de règles d'association, et son évolution en fonction des paramètres de support et de confiance minimaux. Nous avons la base de données suivante, composée de dix groupes, et contenant au total cinq gènes nommés de A à F :

A, D, E	A, B, D
A, B, C	B, C, D
B, C	E, B, C, F
A, B	C, E, F
A, D	E, D, F, C

Tout d'abord, nous calculons le support de toutes les paires de gènes dans la base, ce qui nous donne la matrice suivante :

	A	B	C	D	E	F
A	0.5	0.3	0.1	0.3	0.1	0.0
B	0.3	0.6	0.4	0.2	0.1	0.1
C	0.1	0.4	0.6	0.2	0.2	0.3
D	0.3	0.2	0.2	0.5	0.2	0.1
E	0.1	0.1	0.2	0.2	0.4	0.3
F	0.0	0.1	0.3	0.1	0.3	0.3

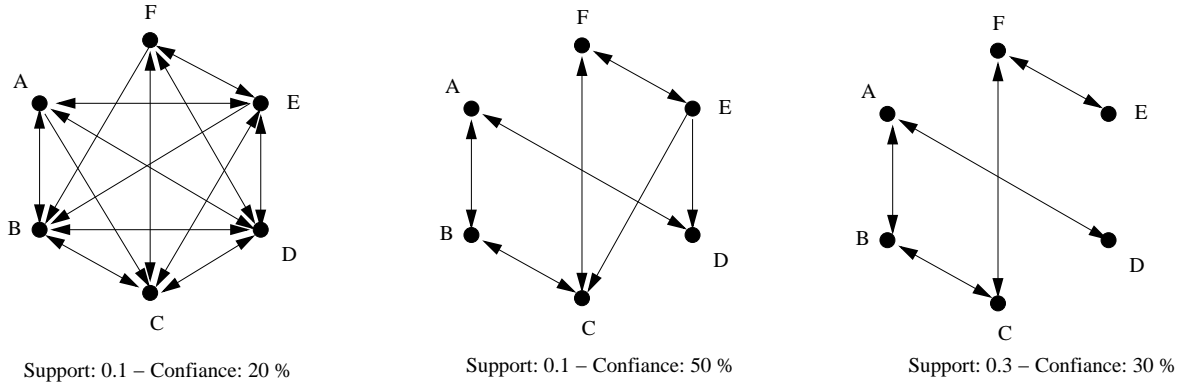
Ensuite, nous fixons les seuils s et c de support et de confiance, qui vont déterminer le nombre d'arcs du graphe que nous allons obtenir. Nous conservons les paires de gènes (x, y) qui apparaissent ensemble avec un support supérieur ou égal à s , et nous calculons la confiance des règles $(x \rightarrow y)$ et $(y \rightarrow x)$. Nous conservons les règles dont la confiance est supérieure ou égale à c , et nous les utilisons pour construire le graphe d'associations. La figure 2 montre l'évolution de ce graphe lorsqu'on augmente la confiance (pour les deux premiers graphes), et le support (pour le troisième graphe).

2.3 Analyse des graphes

Une fois les règles d'associations calculées, nous allons analyser le graphe en découlant et nous intéresser aux caractéristiques de ce graphe permettant de trouver des groupes intéressants.

Est-il justifié de déduire quelque chose d'une chaîne de règles d'associations (ie. d'un chemin dans le graphe) du type $(a \rightarrow b)$, $(b \rightarrow c)$? Ce genre de chaîne peut s'interpréter de la manière suivante : « Les gens ayant acheté le produit a ont souvent aussi acheté le produit b . Les gens ayant acheté b ont souvent aussi acheté c ». On peut en déduire qu'un nombre raisonnable des gens ayant acheté a ont aussi acheté c , sans que cela implique nécessairement qu'il existe une règle d'association entre a et c (si la confiance

FIG. 2 – Évolution du graphe en fonction des paramètres



dans cette règle est trop basse). Il ne faut cependant pas confondre cela avec une règle $(a, b \rightarrow c)$, qui a une signification différente.

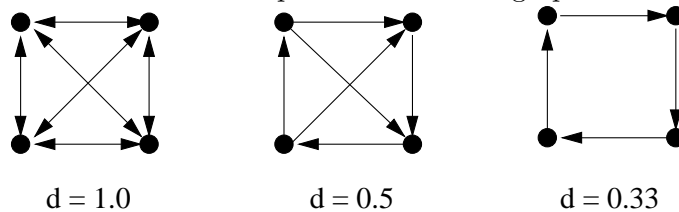
La première caractéristique du graphe d'associations qui semble intéressante sont ses cliques (une clique est un sous-graphe complet maximal). En effet, si les gènes a , b , c et d forment une clique du graphe, cela signifie qu'ils sont *tous* souvent ensemble dans la base, et ce de manière significative (c'est à dire qu'ils sont plus souvent ensemble qu'avec d'autres gènes). Nous serions donc intéressés par l'extraction des cliques d'un graphe mais aussi, d'une manière générale, par les sous-graphes « très connectés », ou très « denses ».

Pour cela, nous avons besoin d'une mesure de la densité d'un graphe, qui permette de comparer deux graphes entre eux, même si leurs nombres de sommets ou d'arêtes diffèrent. Nous définissons la densité d'un graphe $G = (V, E)$ de la manière suivante :

$$d = \frac{|E|}{|V|. (|V| - 1)}$$

Il s'agit du degré moyen d'un graphe, ramené entre 0 et 1 pour permettre une comparaison entre deux graphes. Pour une clique, nous avons $d = 1$, pour un graphe où un arc seul relie chaque paire de sommets, nous avons $d = 0.5$, etc.

FIG. 3 – Exemples de densité de graphes

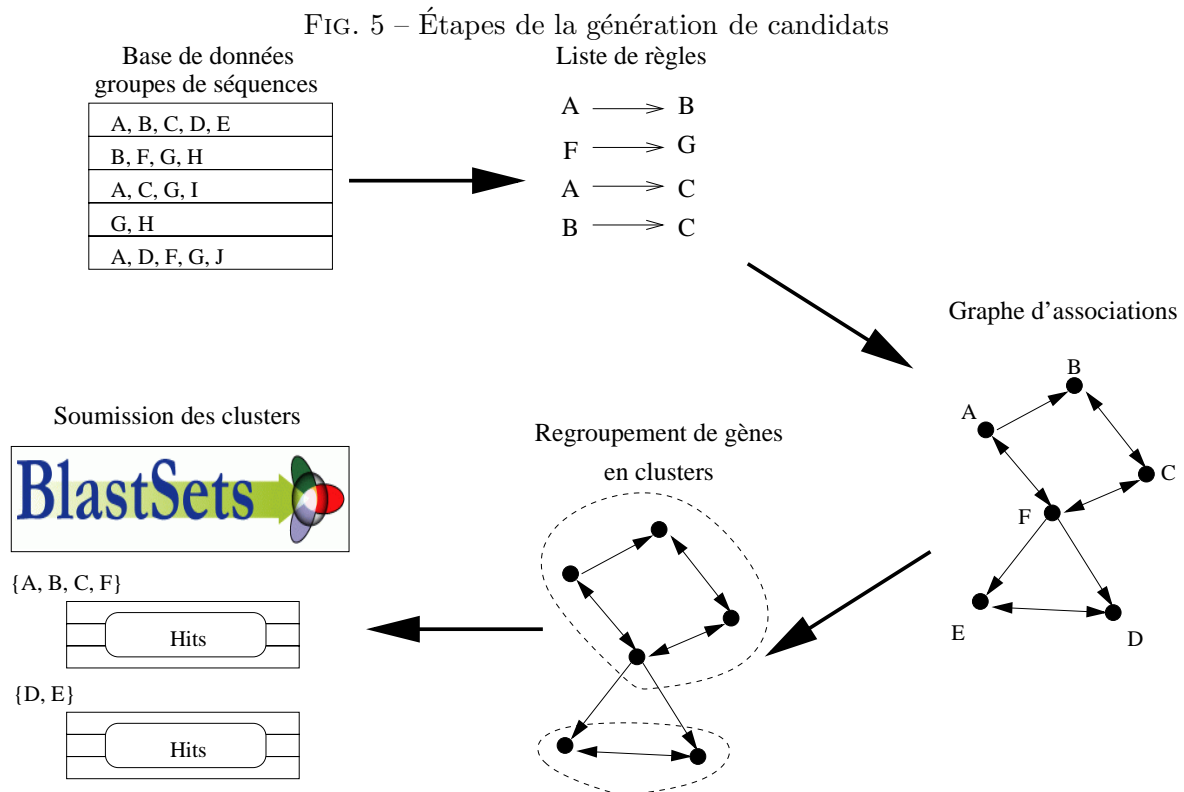


Nous allons donc chercher à partitionner un graphe à partir de ce critère de densité. Le partitionnement de graphe est, dans le cas général, un problème NP-complet.

sera utilisé pour réduire le nombre de cas à considérer, mais sera placé relativement bas.

Nous allons nous intéresser d'une part à l'intégralité de la base de données, et d'autre part à deux sous-ensembles de cette base de données qui regroupent les classifications systématiques et les données expérimentales du transcriptome (données d'expression). Nous allons donc générer trois matrices contenant les fréquences de co-apparition des paires de gènes dans ces bases de données. Une fois les matrices générées, nous calculons de manière automatique les règles d'associations en faisant varier les paramètres, entre 1.10^{-4} et 1.10^{-3} pour le seuil de support, et entre 60 et 100 % pour le seuil de confiance. Enfin, nous construisons les graphes associés à chaque ensemble de règles d'associations, donc un graphe par couple de valeurs de support et de confiance.

Une fois ces listes de graphes générées, nous pourrons y appliquer différentes méthodes décrites par la suite afin d'en extraire des *groupes candidats*. Ces groupes candidats sont destinés à être soumis à BlastSets en tant que requêtes. Afin d'effectuer une grande quantité de requêtes de manière automatique, nous avons développé une bibliothèque en langage Ruby permettant de s'interfacer aux services web de BlastSets conçus par Roland Barriot [4].



Nous avons généré des candidats à partir des composantes fortement connexes de taille supérieure ou égale à deux, et de densité supérieure ou égale à 0.05, et effectué les requêtes correspondant à tous ces candidats.

3.3 Critique des composantes fortement connexes

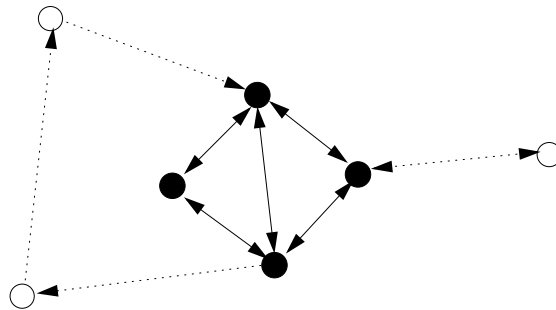
Si le critère de forte connexité semble intuitivement intéressant, il peut toutefois se révéler insuffisant de deux manières opposées.

Critère trop tolérant

Tout d'abord, il se peut qu'à cause de la structure du graphe obtenu, nous obtenions de grosses composantes fortement connexes, de faibles densités. Ce risque sera d'autant plus grand que le seuil de confiance fixé sera bas (cf. figure 6).

Une fois une composante fortement connexe obtenue, si sa densité est trop basse, nous pouvons essayer de « l'améliorer » en retirant des sommets pour augmenter cette densité au moyen d'une heuristique. Nous avons développé un algorithme très simple qui retire des sommets parmi ceux de plus faible degré tant que cela améliore la densité de la composante. Nous avons appliqué cet algorithme aux composantes de densité inférieure à un certain seuil, d'une part en conservant la forte connexité, et d'autre part en abandonnant ce critère, afin d'observer si cela améliore les résultats.

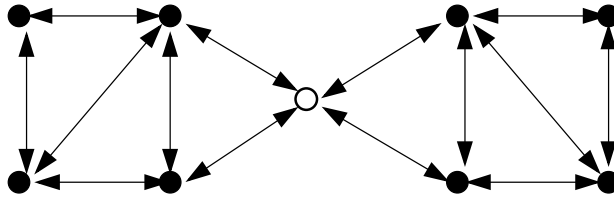
FIG. 6 – Retirer des sommets peut améliorer la densité



De plus, il est possible qu'un gène fasse partie de deux modules théoriquement intéressants apparaissant dans le graphe sous la forme de deux composantes fortement connexes très denses. La présence de ce gène commun aux deux composantes fait que l'union des deux composantes est toujours fortement connexe. Par conséquent, nous risquons une fois de plus d'obtenir des composantes « trop grosses » que l'algorithme décrit ci-dessus ne pourra pas améliorer (cf. figure 7).

Nous voudrions pouvoir trouver, en plus du groupe comprenant tous les sommets de cette composante, les deux groupes intéressants de gènes, chacun contenant le gène « blanc », sans nécessairement les réunir, ce qui nécessite l'emploi d'un algorithme permettant de « dupliquer » un gène pour pouvoir le mettre dans plusieurs groupes. Cela sera discuté dans la section 5.

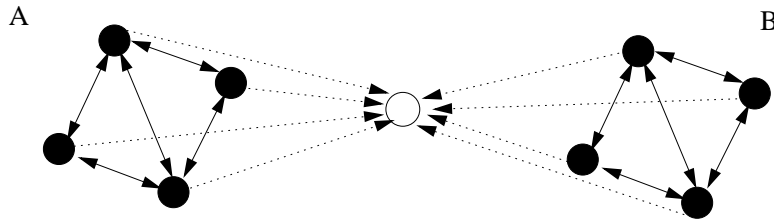
FIG. 7 – Deux composantes réunies « à tort »



Critère trop strict

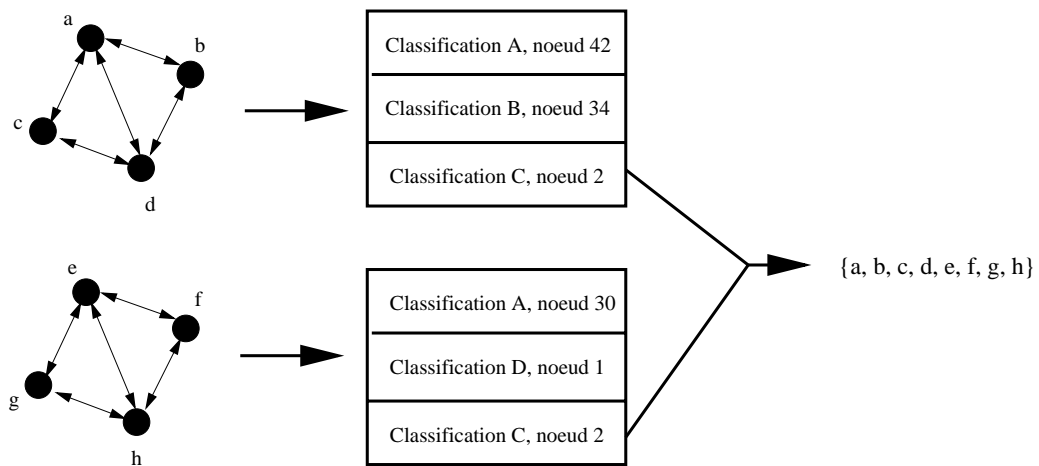
Il se peut également que favoriser la forte connexité par rapport à la densité conduise à « manquer » certains sommets. Prenons l'exemple hypothétique de la figure 8, dans lequel un sommet fait figure de point d'articulation entre deux composantes fortement connexes, mais ne fait partie d'aucune d'entre-elles. Dans ce cas de figure très schématique, il n'y a pas de règles d'association de ce sommet vers A car leur confiance était trop basse (50 %) à cause de B et de même, il n'y a pas pas de règles vers B , à cause de A . Cependant, ce sommet semble intéressant à la fois dans un candidat formé des sommets de A et dans un candidat formé des sommets de B .

FIG. 8 – Un “point d'articulation”



Enfin, il est possible que le critère de forte connexité soit trop strict à des niveaux de confiance élevés, et donc qu'il produise des candidats trop petits pour être intéressants. Nous avons donc tenté, une fois les composantes fortement connexes utilisées comme groupes requête, de fusionner celles qui avaient des résultats en commun, et de ré-effectuer des requêtes avec ces nouveaux groupes (cf. figure 9).

FIG. 9 – Fusion de deux groupes candidats qui ont un hit commun



4 Résultats

4.1 Jeux de données et validation

Nous pouvons distinguer deux grandes catégories parmi les classifications de la base de données. D'une part, des classifications systématiques représentant des informations connues des biologistes (complexes protéiques, voies métaboliques, etc.), et d'autre part des données expérimentales issues de profils d'expression. Dans un premier temps, nous avons considéré l'ensemble de la base de données, toutes classifications confondues. Puis, nous avons effectué une distinction entre les classifications correspondant aux données d'expression et celles correspondant aux classifications systématiques, afin de faire ressortir le rôle joué par les informations avérées dans la génération de candidats, par rapport aux données expérimentales. Voici la composition de ces trois jeux de données :

Composition	Nombre de groupes de gènes
Données mixtes	159 925
Classifications systématiques	6 603
Données d'expression	220 309

Entre le moment où nous avons étudié les données mixtes et celui où nous avons distingué les deux types de classifications, la composition de la base avait évolué, ce qui explique que la somme du nombre de groupes des classifications systématiques et des données d'expression soit supérieure au nombre de groupes des données mixtes. c pas honk

Nous espérons trouver une confirmation de la validité de notre approche en observant dans les résultats la formation de groupes dont la fonction est déjà connue. Les ribosomes sont des organites impliqués dans la traduction de l'ARN messager. [5] et [1] indiquent que le ribosome est toujours co-régulé, nous nous attendions à faire ressortir un ou plusieurs groupes correspondant au ribosome dans les candidats générés à partir des données d'expression.

De plus, les classification systématiques sont dominées par la Gene Ontology, qui représente un sixième des groupes. Cette classification a une structure de DAG de grande profondeur. Un groupe se retrouvant dans tous ses parents, nous nous attendons à ce que la Gene Ontology ressorte dans nos résultats lors de l'étude sur les classifications systématiques.

4.2 Requêtes sur des données mixtes

Candidats générés

Nous avons appliqué la procédure décrite dans la figure 5 (page 9) aux données mixtes. Tout d'abord, voici un tableau présentant le nombre de règles d'association que l'on obtient pour différentes valeurs des paramètres de support et de confiance, sur les

données mixtes de la base :

Support	Confiance	Nombre de règles
0.0001	60 %	182 437
0.0001	65 %	74 986
0.0001	70 %	32 218
0.0001	75 %	12 762
0.0001	80 %	3 725
0.0001	85 %	1 241
0.0001	90 %	297
0.0001	95 %	10
0.003	40 %	5 321 782
0.003	50 %	1 222 535
0.003	60 %	142 140
0.003	70 %	15 592
0.003	80 %	540

Le nombre de règles obtenu représente le nombre d'arcs du graphe sur lequel nous allons travailler. Nous pouvons constater que pour certaines valeurs du seuil de confiance, le graphe contient une telle quantité d'arcs que son analyse sera difficile. Dans la première partie du tableau, nous pouvons observer que le nombre de règles est assez important pour un niveau de confiance de 60 %, mais diminue assez vite. Fixer un seuil de confiance supérieur 80 % semble sans intérêt étant donné le faible nombre d'arcs que nous obtiendrons.

Dans la deuxième partie du tableau, nous avons augmenté sensiblement le support, et nous pouvons constater que le nombre de règles du graphe varie beaucoup lorsque l'on modifie le seuil de confiance de 10 %. Pour des valeurs de ce seuil inférieures à 50 %, le graphe contiendra trop d'arcs pour être analysé rapidement, et nous pouvons craindre que ce grand nombre d'arcs ne produise des composantes fortement connexes trop grosses. Toutefois, à ce niveau de support, le nombre d'arcs diminue vite, et pour un niveau de confiance de 80 %, nous n'avons plus que 540 règles, au lieu de 3 725 précédemment. Nous préférons par la suite favoriser un seuil de support assez bas, et un niveau de confiance plus haut.

Le tableau suivant résume le nombre de candidats (c'est à dire de composantes fortement connexes) obtenus à partir des graphes correspondant à ces listes de règles, en fonction de la confiance.

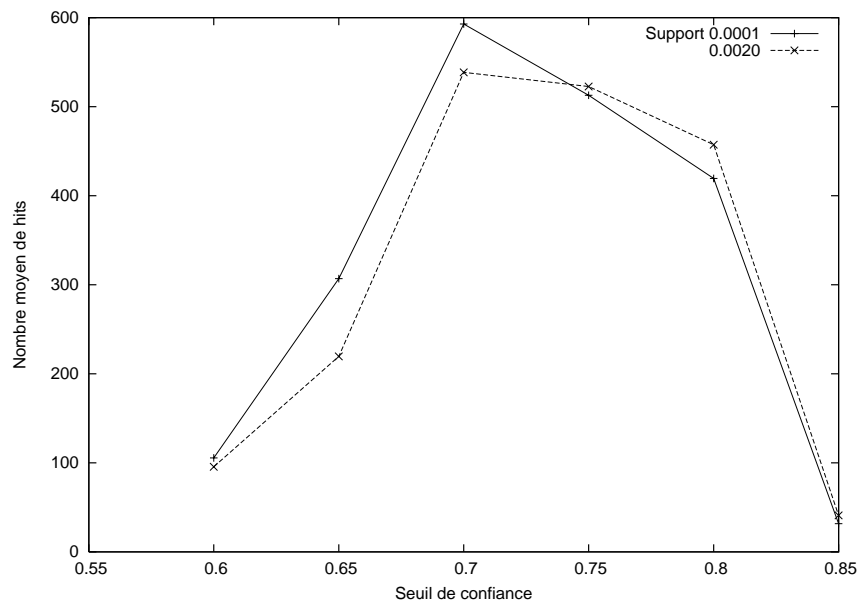
Support	Confiance	Nbre de composantes
0.0001	60 %	102
0.0001	65 %	127
0.0001	70 %	44
0.0001	75 %	26
0.0001	80 %	15

Nous pouvons constater que, bien que le nombre de composantes connexes diminue généralement lorsque l'on augmente le seuil de confiance, il augmente en passant de 60 à 65 %. Cela est dû à l'éclatement d'une grosse composante peu dense.

Résultats

La figure 10 représente le nombre moyen de hits des candidats générés, lorsque l'on effectue des requêtes avec comme cible l'ensemble des classifications de la base. Nous proposons la quantité de hits comme critère de qualité des groupes, et nous pouvons constater que la moyenne du nombre de hits d'un candidat est à son maximum pour une confiance de 70 %, reste élevée jusqu'à 80 %, pour chuter ensuite, ce qui semble cohérent avec l'idée qu'une confiance assez haute donne de bons résultats.

FIG. 10 – Nombre moyen de hits - toute la base



Afin de résumer la grande quantité de résultats obtenus, nous allons donner un exemple d'un candidat correspondant à des informations avérées, et un autre ne correspondant à priori à rien de connu, et pouvant être un sujet d'investigations biologiques. Les exemples donnés ont été pris parmi les candidats obtenus avec un niveau de confiance de 70 %.

Parmi ces résultats, nous pouvons retrouver un candidat correspondant au ribosome, sous la forme d'une composante de 98 gènes, de densité 0.45 et générant 8057 hits, dont voici les meilleurs dans les classifications systématiques :

pValue	Communs	Sur	Classification	Description
1.150e-181	97	128	KEGG SCe	Ribosome
2.466e-178	97	127	GO - cellular component	cytosolic ribosome
1.191e-157	98	195	FUNCTIONAL CATEGORY	ribosome biogenesis
1.647e-153	96	185	GO - molecular function	structural constituent of ribosome
2.632e-113	96	402	GO - biological process	protein biosynthesis
1.210e-105	98	537	SUBCELL	cytoplasm

Ces résultats sont très satisfaisant, d'autant plus que les gènes qui font partie du hit sur KEGG mais ne font pas partie de notre groupe candidat correspondent au ribosome mitochondrial, et n'apparaissent donc pas dans les données du transcriptome.

D'autre part, nous pouvons remarquer une autre composante, qui comprend 47 gènes et qui a une densité de 0.11. Cette composante génère 1869 hits, mais aucun hit informatif dans des classifications systématiques, ce qui pousse à s'intéresser à la raison pour laquelle ces 47 gènes ont été réunis :

pValue	Communs	Sur	Classification	Description
6.459e-05	2	2	GO - molecular function	ceramidase
0.0009489	2	6	SWISSPROT_Keywords	keyword swissprot cAMP
0.0022423	2	9	KEGG SCe	Pentose and glucuronate interconversions

Cette composante ainsi que d'autres du même type peuvent être de bons candidats pour une investigation biologique afin de savoir s'il existe une raison à leur rapprochement.

4.3 Requêtes sur les classifications systématiques

Candidats générés

La première chose que nous pouvons remarquer en observant les règles obtenues à partir des classifications systématiques est qu'elles sont bien plus nombreuses, à niveau de confiance égale, qu'avec un jeu de données mixtes (cf. tableau suivant). Cela nous empêchera d'obtenir des candidats intéressants à des niveaux de confiance bas, néanmoins nous pouvons espérer obtenir de bons résultats à des niveaux de confiance élevés.

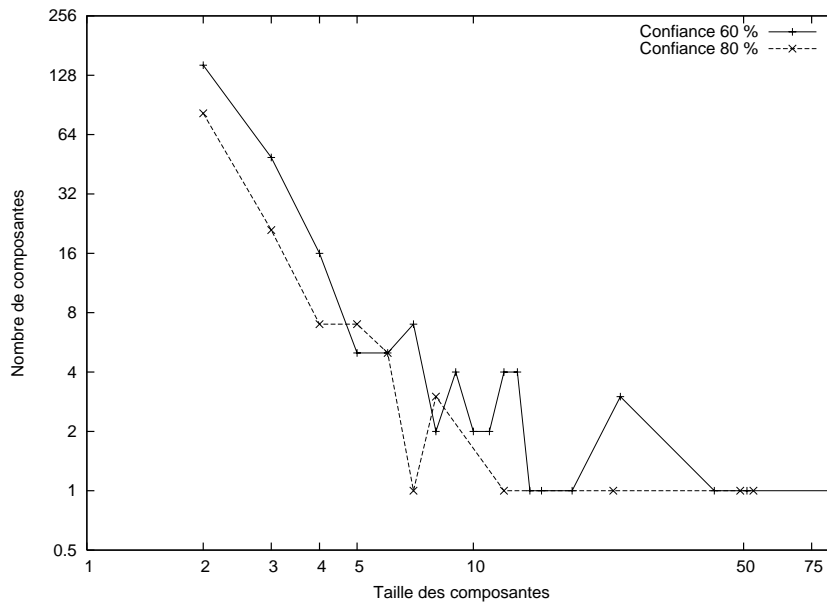
Support	Confiance	Nombre de règles
0.0001	60 %	3 249 411
0.0001	65 %	2 322 298
0.0001	70 %	1 659 579
0.0001	75 %	1 067 892
0.0001	80 %	716 335
0.0001	85 %	329 389
0.0001	90 %	285 174
0.0001	95 %	118 101

Le tableau suivant donne un aperçu des candidats obtenus à partir de ces règles. Nous pouvons remarquer la présence d'une très grosse composante fortement connexe, qui englobe plus de la moitié des gènes de la levure (qui sont 5 786) pour le niveau de confiance de 60 %. Étant donné la taille de cette composante, même si les candidats que nous trouvons sont intéressants, nous serons certains d'avoir manqué beaucoup de bons groupes potentiels.

Support	Confiance	Nbre de composantes	Taille max
0.0001	60 %	254	3210
0.0001	65 %	328	1912
0.0001	70 %	277	1788
0.0001	75 %	188	1644
0.0001	80 %	131	1421
0.0001	85 %	108	964
0.0001	90 %	85	921
0.0001	95 %	176	71

Cette grosse composante est considérablement réduite lorsque l'on passe le seuil de confiance à 65 %, mais reste encore très grosse. On constate naturellement une augmentation du nombre de composantes fortement connexes, dû à l'éclatement de cette grosse composante. C'est à ce niveau de confiance que se situe le pic du nombre de candidats. Notons que la plus grosse composante ne s'est pas divisée en deux, mais plutôt en petites composantes, comme nous pouvons le voir en observant la distribution des tailles des candidats, figure 11 : il n'y a pas d'apparition de grosses composantes lorsque l'on passe d'un niveau de confiance de 60 % à 80 %.

FIG. 11 – Taille des composantes obtenues - classifications systématiques



Lorsque l'on continue à augmenter la confiance, la taille de la plus grosse composante diminue, mais le nombre de candidats diminue également. Ceci est dû au fait que le nombre de règles diminue (augmentation de la confiance) et que l'on n'a pas de séparation de grosse composante qui se produit.

Enfin, nous observons l'éclatement de la plus grosse composante en passant de 90 à 95 % de confiance, mais à ce niveau, les règles risquent d'être trop strictes pour produire des résultats vraiment intéressants.

Résultats

Le nombre moyen de hits par composante est représenté sur la figure 13 (page 20), en fonction de la confiance. Lors de l'analyse en séparant les classifications systématiques des données d'expression, nous n'avons effectué les requêtes que contre ces classifications systématiques, afin de comparer les candidats générés aux informations connues, ce qui explique le faible nombre de hits par candidat en comparaison des premiers résultats.

Les hits obtenus à des niveaux de confiance très élevés ne sont pas intéressants, et correspondent à des niveaux proches de la racine des structures hiérarchiques des classifications. Autour de 70 %, on trouve des résultats qui ont potentiellement plus de sens, mais les candidats sont essentiellement de petits groupes. Il ne faut pas oublier qu'à cause de la très grosse composante fortement connexe du graphe, on exclue une partie importante des gènes de notre génération de candidats. Nous retrouvons néanmoins le ribosome encore une fois, avec 99 gènes en commun sur 128.

4.4 Requêtes sur les données d'expression

Candidats générés

Les données du transcriptome étaient dominantes dans la constitution de la première base de groupes de gènes sur laquelle nous avons travaillé, nous ne nous étonnons donc pas de trouver des résultats assez similaires à ceux détaillés plus haut. Pour un même niveau de confiance, nous avons légèrement moins de règles d'associations :

Support	Confiance	Nombre de règles
0.0001	60 %	133 614
0.0001	65 %	56 688
0.0001	70 %	24 635
0.0001	75 %	9 089
0.0001	80 %	2 964
0.0001	85 %	719
0.0001	90 %	126
0.0001	95 %	37

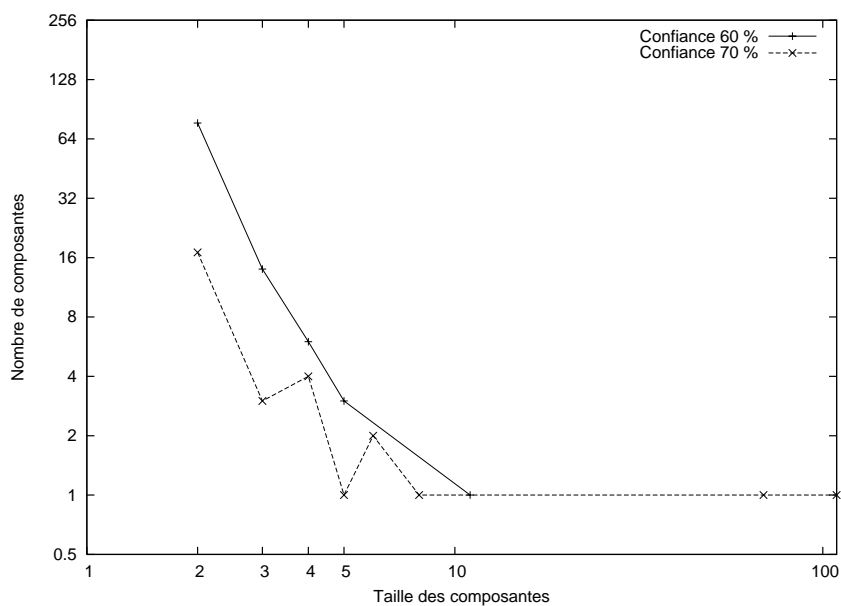
Nous rencontrons encore une fois le même problème : une très grosse composante se forme pour des niveau de confiance bas, mais cette fois, sa densité est tellement faible qu'elle n'a pas été sélectionnée comme candidat. Elle comporte là encore plus de la

moitié des gènes. Toutefois, nous pouvons remarquer que le nombre de candidats, à niveau de confiance égal, est semblable. De la même manière, la taille des candidats est plus faible, comme le montre la figure 12.

Support	Confiance	Nbre de composantes
0.0001	60 %	102
0.0001	65 %	80
0.0001	70 %	30
0.0001	75 %	21
0.0001	80 %	5

La principale différence entre ce tableau et celui présentant le nombre de composantes pour le jeu de données mixtes est le fait que nous n'avons pas de pic autour de 65 % de confiance. Cela indique que nous n'avons pas l'éclatement d'une grosse composante, comme nous avons précédemment.

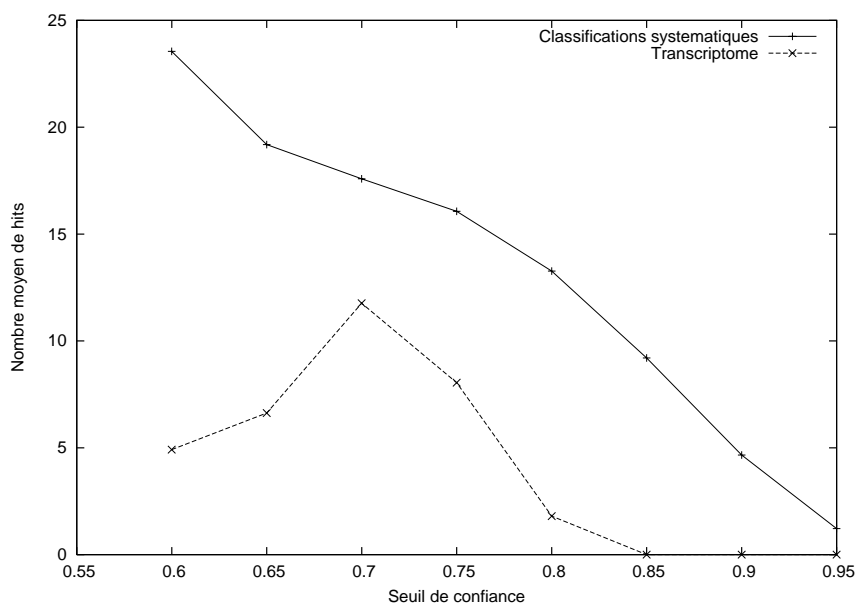
FIG. 12 – Taille des composantes obtenues - transcriptome



Résultats

Nous avons là encore un groupe candidat correspondant au ribosome, composé de 108 gènes et renvoyant 54 hits, dont les meilleurs sont :

FIG. 13 – Nombre moyen de hits - par catégories



pValue	Communs	Sur	Classification	Description
2.297e-182	101	128	KEGG SCe	Ribosome
1.282e-179	100	127	GO - cellular component	cytosolic ribosome
1.594e-158	103	195	FUNCTIONAL CATEGORY	ribosome biogenesis
6.250e-153	100	185	GO - molecular function	structural constituent of ribosome
3.616e-123	105	402	GO - biological process	protein biosynthesis
8.250e-117	108	537	SUBCELL	cytoplasm

4.5 Requêtes basées sur les améliorations proposées

Amélioration de la densité

Nous avons appliqué l'algorithme décrit paragraphe 3.3 aux composantes de densité inférieure à 0.5, dans le premier jeu de données (classifications systématiques et données d'expression mélangées). Nous avons donc produit une nouvelle liste de groupes candidats basée sur ces composantes. Ces candidats étaient plus petits, mais formaient des composantes fortement connexes plus denses.

Dans tous les cas, le nombre de hits du groupe « dense » a été plus faible que celui du groupe d'origine, et la qualité des hits produits a elle aussi été moindre (c'est à dire que leur pvalue a été plus élevée). Par conséquent, l'augmentation de la densité n'est pas un critère suffisant pour améliorer les candidats.

Fusion des composantes

Nous avons appliqué l'algorithme décrit paragraphe 3.3 (figure 9) pour fusionner les candidats qui avaient des points communs parmi leurs meilleurs hits, d'une part sur les classifications systématiques, et d'autre part sur les données du transcriptome. Toutefois, cette méthode ne semble pas concluante, car elle a tendance à fusionner des groupes qui ont un hit en commun qui est un très gros groupe peu intéressant, par exemple celui représentant des gènes de fonction moléculaire inconnue.

Une alternative serait de se restreindre aux groupes ayant leur meilleur hit en commun, mais cela ne semble pas justifié d'un point de vue logique : admettons qu'une requête renvoie deux hits, un ayant dix gènes en commun sur dix, et l'autre en ayant sept sur dix. Le premier hit sera le meilleur (c'est à dire aura la pvalue la plus faible), mais nous serions tentés de « compléter » le deuxième hit, en fusionnant ce groupe requête avec un autre qui comporterait trois gènes en commun sur dix.

Par conséquent, cette méthode ne nous semble pas applicable sans un critère supplémentaire pour décider de la fusion ou non de deux groupes candidats.

4.6 Conclusion

Ces trois séries d'expériences confirment la pertinence du modèle basé sur un graphe de règles d'association que nous avons défini, car nous avons obtenu des résultats conforme à nos attentes. Toutefois, nous avons mis en évidence une limitation dans l'utilisation des composantes fortement connexes : à des niveaux de confiance qui sont potentiellement intéressants pour nous, la présence d'une grande quantité d'arcs provoque une agglomération trop importante des sommets en grosses composantes fortement connexes, qui masque probablement des informations.

Pour contourner cette limitation, nous devons trouver un moyen de partitionner le graphe d'une manière différente, qui nous permettrait d'explorer les ensembles très denses que l'on peut trouver au sein d'une composante fortement connexe. Nous allons donc développer les différentes approches envisageables dans la section suivante.

5 Autres approches envisageables

5.1 Heuristiques de “déplacement”

Pour capturer un concept plus vague que celui de forte connexité, il est nécessaire d’avoir recours à des heuristiques. Nous allons nous intéresser aux manières de partitionner un graphe de manière à obtenir des sous-ensembles « denses » de sommets [6].

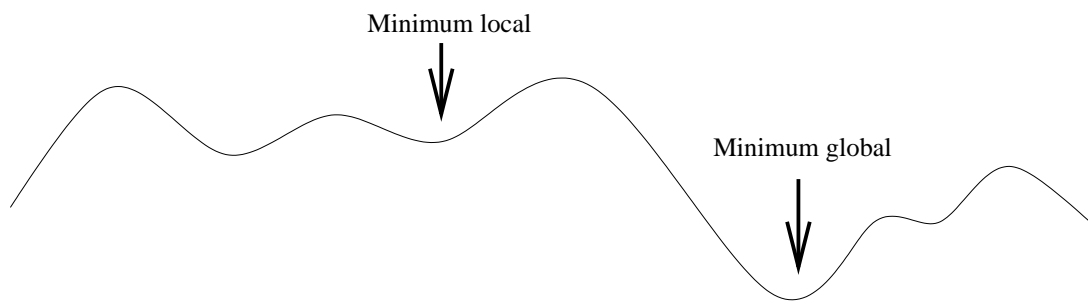
Tout d’abord, nous allons étudier les heuristiques qui consistent en un « déplacement » dans un espace de solutions. Il faut tout d’abord définir la structure de cet ensemble de solutions, qui doit contenir une notion de voisinage. Pour notre problème de partitionnement, nous devons partir d’une solution initiale aléatoire, c’est à dire d’un partitionnement des sommets du graphe en clusters. À partir de cette solution initiale, nous pouvons par exemple nous déplacer dans l’ensemble des solutions en déplaçant un sommet d’un cluster à un autre d’une manière systématique, éventuellement en le plaçant seul dans un nouveau cluster.

Nous avons également besoin d’une fonction qui nous permet d’évaluer la qualité d’une solution, et que nous devons maximiser. Nous pouvons pour cela utiliser une fonction basée sur la densité, éventuellement favorisant un grand nombre de sommets (mieux vaut 10 sommets et une densité de 0.6 plutôt que 3 sommets et une densité de 1.0) ou la forte connexité.

Algorithme glouton

Le moyen le plus simple et intuitif d’améliorer la solution initiale est de procéder par améliorations successives, et à s’arrêter lorsqu’aucun déplacement dans le voisinage immédiat n’améliore la solution. Cependant, ce type d’algorithme se contente de trouver un minimum local (cf. figure 14), et pas nécessairement une bonne solution du point de vue global.

FIG. 14 – Minimums locaux et globaux



Pour contourner ce problème et éviter de rester bloquer dans un minimum local, il faut un algorithme qui permette de « remonter » dans certains cas, c’est à dire qui puisse accepter de prendre une mauvaise décision à l’instant t dans l’espoir d’obtenir un meilleur résultat à l’instant $t+n$. Certains algorithmes gloutons se basent sur une notion

de voisinage *étendu* leur permettant de sortir de minimums locaux, les plus connus étant l'algorithme de Kernighan-Lin, et son amélioration par Fiduccia-Mattheyses [7].

Recuit simulé

Le recuit simulé [8] est une approche inspirée d'un procédé métallurgique consistant à faire fondre une substance et à réduire lentement sa température afin d'obtenir une configuration d'atomes plus stable.

L'algorithme du recuit simulé se déplace d'une solution à une solution voisine si celle-ci représente une amélioration, mais il acceptera aussi de se déplacer vers une solution plus mauvaise avec une probabilité dépendant de la *température* du système, un paramètre dont la valeur initiale et l'évolution sont fixées par l'utilisateur.

Le recuit simulé est une méthode efficace pour trouver de bonnes solutions, mais son temps d'exécution peut s'avérer très long, en particulier à cause de nombreux déplacements aléatoires inutiles, et à cause du fait que plus la température refroidit, plus les déplacements aléatoires sont rejetés. Toutefois, notons qu'une très grande partie du temps de calcul est dominé par l'évaluation de la qualité d'une solution. Notre critère de densité étant rapide à évaluer, cette méthode semble intéressante.

Recherche avec tabou

La recherche avec tabou est un mécanisme similaire aux algorithmes gloutons, mais qui autorise des *remontées*, et conserve une liste des r déplacements les plus récents, afin d'éviter de revenir sur ses pas et de se bloquer dans un minimum local. De plus, une fonction d'*aspiration* permet de passer outre la liste taboue, si la solution semble « assez bonne ».

La recherche avec tabou présente un avantage par rapport au recuit simulé : alors que celui-ci fait ses déplacements complètement aléatoirement, et donc peut perdre du temps à ré-explore des solutions déjà visitées, la recherche avec tabou peut explorer l'espace de solutions plus rapidement.

Algorithmes génétiques

Les algorithmes génétiques s'inspirent du mécanisme de sélection naturelle, où les individus les plus adaptés à leur environnement ont une descendance plus importante, et où des mutations aléatoires interviennent de temps en temps.

Un algorithme génétique est défini par deux paramètres, la fonction de *crossover* et la fonction de *mutation*. Le crossover est l'analogie de la reproduction sexuée dans le monde vivant : il produit un mélange des caractéristiques des deux « parents » pour produire une descendance. La mutation permet quant à elle d'introduire des modifications aléatoires dans une solution. Enfin, le *mécanisme de sélection* définit quels sont les « individus » aptes à se reproduire ou à disparaître.

L'algorithme démarre avec une population initiale, qui est un ensemble de solutions, c'est à dire un ensemble de partitionnements du graphe en groupes de sommets et, suivant les critères définis par le mécanisme de sélection, on calcule une deuxième génération de solutions, en produisant de nouvelles solutions à partir de la génération précédente. Ces nouvelles solutions sont majoritairement basées sur les meilleurs partitionnements de la solution précédente. L'opération est alors répétée un certain nombre de fois, puis la meilleure solution est sélectionnée, et considérée comme le résultat de l'algorithme.

Des algorithmes génétiques ont été utilisés et se sont montrés efficaces pour des problèmes de partitionnement de graphes [9], il semble donc intéressant d'explorer cette voie en développant des fonctions de crossover et de mutation qui correspondent à notre problème et permettent à un gène de se retrouver, éventuellement, dans plusieurs groupes d'une solution.

5.2 Algorithmes de clustering

Alors que les heuristiques que nous avons vues partent d'un partitionnement initial, et essaient de l'améliorer, les algorithmes de clustering utilisent une approche ascendante (*bottom-up*), et partent d'un partitionnement où chaque sommet est un cluster, en progressant par fusions successives de clusters. On peut distinguer les algorithmes agglomératifs, où un seul cluster est formé à chaque fois, et les algorithmes hiérarchiques, où plusieurs clusters peuvent être formés en une seule fois. Un certain nombre de ces algorithmes [6, 10] ont été utilisés sur des critères proches de ceux qui nous intéressent. Toutefois, peu d'entre-eux ont été conçus pour permettre à un sommet de se trouver dans plusieurs clusters, cas qui nous intéresse potentiellement.

5.3 Modifications du graphe de règles d'association

Graphe de règles d'association pondéré

Jusqu'à présent, nous avons décidé de fixer un seuil minimal de support et de confiance, et de considérer comme arc du graphe toute règle dont la confiance était supérieure au seuil fixé. Cela nous permet, entre autres, d'observer l'évolution du graphe lorsque l'on augmente la confiance minimale des règles. Toutefois, nous pouvons imaginer d'affiner la modélisation des associations entre gènes, en incluant la confiance en tant que poids des arcs du graphe. Cela nous permettrait en particulier d'utiliser la distance entre deux gènes au sein des algorithmes précédemment évoqués.

La manière la plus instinctive de faire cela est d'utiliser l'inverse de la confiance comme poids des arcs. De cette manière, plus la confiance de la règle ($a \rightarrow b$) est proche de 100 % et plus le poids de l'arc reliant le sommet a au sommet b est proche de un. Ainsi la distance entre deux gènes a et b devient la longueur du plus petit chemin entre les sommets a et b du graphe, mais on peut aussi s'intéresser à d'autres critères :

- Le diamètre d'un sous-graphe, c'est à dire la distance maximale entre deux sommets de ce sous-graphe, peut remplacer (ou s'ajouter à) la mesure de densité que nous utilisons pour en estimer la qualité.
- la distance moyenne entre deux sommets d'un sous-graphe peut elle aussi nous intéresser, car nous désirons obtenir des sous-graphes dont tous les sommets sont relativement proches les uns des autres.

Assouplissement par itérations

Il est également possible d'utiliser le fait que, pour un support fixé et des seuils de confiance variables, le graphe d'associations obtenu conserve la même « apparence », mais est moins dense (voir figure 2 page 7). En nous basant sur cette observation, nous pourrions envisager de générer le graphe correspondant à un niveau de confiance minimal élevé (disons, 80 %), lui appliquer une heuristique de notre choix pour en produire un partitionnement, puis générer le graphe correspondant à un seuil de confiance plus bas (50 ou 60 %), et appliquer la même heuristique en utilisant comme solution initiale le partitionnement obtenu pour le premier graphe.

En effet, la plupart des heuristiques que nous avons évoquées ont été prouvées être plus efficaces lorsqu'elles sont initialisées avec une solution non optimale, mais malgré tout assez bonne (provenant d'un minimum local), produite par un algorithme glouton [6].

6 Conclusion

Au cours de notre travail, nous avons proposé une formalisation de la notion de « relation » entre des gènes, au moyen d'une technique de data mining et de la théorie des graphes. Nous avons défini des critères sur ces graphes permettant de dire qu'un ensemble de gènes est fortement lié, et étudié une première approche de partitionnement basée sur l'extraction de composantes fortement connexes, afin de construire des groupes que nous avons appelé « candidats ». Puis nous avons soumis ces groupes candidats à BlastSets afin de connaître leur similarité par rapport aux groupes présents dans la base de données, et de savoir s'ils avaient une signification biologique connue. Cette méthode a confirmé que notre modèle était pertinent, en produisant des groupes candidats correspondants à des entités connues, et que nous attendions de trouver, ainsi qu'en produisant des candidats dont le critère de rapprochement n'est pas apparent. Toutefois, nous avons mis en évidence d'importantes limitations dans cette première approche, qui réunit des ensembles de sommets trop importants. Nous avons donc étudié différentes méthodes de partitionnement de graphe qui pourraient s'appliquer à notre formalisme et contourner ces limitations.

Références

- [1] R. Barriot, J. Poix, A. Groppi, A. Barré, N. Goffard, D. Sherman, I. Dutour, and A. de Daruvar. New strategy for the representation and the integration of biomolecular knowledge at a cellular scale. *Nucleic Acids Research*. accepted.
- [2] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules in large databases. In *Proceedings of the 20th International Conference on Very Large Data Bases*, pages 487–499. Morgan Kaufmann Publishers Inc., 1994.
- [3] Amihood Amir, Ronen Feldman, and Reuven Kashi. A new and versatile method for association generation. In *Principles of Data Mining and Knowledge Discovery*, pages 221–231, 1997.
- [4] R. Barriot and A. Lamiable. Web services and client packages as a framework for data mining on the neighborhood of organized sets of biological sequences. *ECCB - ISMB 2004 / Glasgow, Angleterre*.
- [5] R. Jansen, D. Greenbaum, and M. Gerstein. Relating whole-genome expression data with protein-protein interactions. *Genome Res*, 12 :37–46, 2002.
- [6] Charles J. Alpert and Andrew B. Kahng. Recent directions in netlist partitioning : A survey. *Integr. VLSI J.*, 19(1-2) :1–81, 1995.
- [7] C. M. Fiduccia and R.M. Mattheyses. A linear time heuristic for improving network partitions. *Proc. ACM/IEEE Design Automation Conf.*, pages 175–181, 1982.
- [8] S. Kirkpatrick, Jr. C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220 :671–680, 1983.
- [9] R. Chandrasekharam, S. Subhranian, and S. Chaudhury. Genetic algorithm for node partitioning problem and applications in vlsi design. *IEE Proceedings E (Computer and Digital Techniques)*, 140(5) :255–260, 1993.
- [10] R. Shamir and R. Sharan. Algorithmic approaches to clustering gene expression data. In Y. Xu T. Jiang, T. Smith and M. Q. Zhang, editors, *Current Topics in Computational Biology*. MIT press, 2001. To appear.