Université de Montréal

Analyse des données d'expression de gènes

par

Éric Paquet

Département d'informatique et de recherche opérationnelle

Faculté des arts et des sciences

Mémoire présenté à la Faculté des études supérieures
en vue de l'obtention du grade de Maîtrise ès sciences (M. Sc.)
en Informatique

Avril, 2005

Université de Montréal
Faculté des études supérieures

Ce mémoire intitulé:

Analyse des données d'expression de gènes

présenté par:

Éric Paquet

a été évalué par un jury composé des personnes suivantes:

Nadia El-Mabrouk
président-rapporteur

François Major
directeur de recherche

Miklós Crürös
membre du jury

Mémoire accepté le 8 juin 2005

Résumé

Les données provenant des biopuces sont de plus en plus nombreuses et nous en sommes maintenant au développement d'outils nous permettant d'extraire de l'information pertinente de ces données. Le but de ce mémoire est d'apporter une contribution au développement de ces outils. Le premier chapitre traite de l'accélération d'une méthode de recherche de la structure optimale d'un réseau bayesien. Nous présentons les éléments d'une implantation efficace de la méthode MCMC (Monte Carlo Markov Chain). L'importance de certains éléments d'implantation sera démontrée. Le second chapitre utilise la méthode développée au chapitre un pour vérifier l'impact des connaissances *a priori* au niveau de la structure sur la précision de reconstruction du réseau véritable. Nous démontrons que ces connaissances n'améliorent pas significativement les performances de la méthode. Le troisième chapitre présente l'analyse de données provenant du thymus de souris leucémiques et non-leucémiques. Nous présentons les gènes différemment exprimés dans 6 analyses effectuées sur 5 triplicatas différents. Le quatrième chapitre traite d'une analyse au niveau du génome humain utilisant des données disponibles sur l'Internet. Nous présentons une nouvelle métrique pour identifier des régulations négatives potentielles entre deux gènes. De plus, nous présentons les résultats intéressants obtenus lors de cette analyse.

Mots-clés : bioinformatique, apprentissage machine, réseaux Bayesien, biopuces, connaissances a priori, gènes différemment exprimés

Abstract

Microarray data are widely available resulting in the need for tools to extract meaningful information from them. The topic of this thesis is the development of such tools. The first chapter is dedicated to the description of crucial implementation aspects of the MCMC (Markov Chain Monte Carlo) to retrieve optimal Bayesian networks structure. We show that computational aspects are primordial to acquire reasonable running time. In the second chapter, we use implementation strategies developed in chapter one to verify the importance of structural prior knowledge over the accuracy of the search for the optimal network structure. We show that this knowledge does not improve significantly the accuracy of the method. In the third chapter, we discuss an analysis at the genome level of the differentially expressed genes from the thymus of leukemic and none leukemic mice. We show the top differentially expressed genes found in that analysis. In the fourth chapter, we present the results of the application of a new metric we have developed to retrieve significant inhibitor pattern at the genome scale level using all human microarray data publicly available, and we discuss interesting results we have generated

Keywords : bioinformatics, machine learning, Bayesian networks, microarray, prior knowledge, differentially expressed genes

Table des matières

Liste des équations

Liste des tableaux

Liste des figures

À Sophie et Persil

Remerciements

Pendant les deux dernières années, j'ai côtoyé des gens qui ont fait de mon passage au LBIT un moment fort agréable. Les lignes qui suivent ont pour but de leur rendre hommage.

Je tiens sincèrement à remercier Martin « But du fond » Larose pour sa grande connaissance des langages de programmation, son aide avec Linux, de m'avoir fait connaître Pennac et aussi pour toutes nos discussions. Philip « Diagonale » Thibault pour tous les outils qu'il m'aura fait découvrir, les commandes shells qu'il m'aura fournies ainsi que son aide avec Word. Anita Boisgontier pour ses connaissances de MySQL et pour toutes les discussions du 6ième pendant le lunch. Vincent Devloo pour sa grande sagesse et aussi pour avoir développé en moi un infime intérêt pour la politique étrangère et américaine. Yannick Pomerleau pour le soccer et les nombreuses discussions. Emmanuelle Permal pour ses connaissances biologiques. Paul Dallaire pour nos nombreuses discussions et sa grande connaissance de la biologie. Chabane Tibiche pour nos discussions. Marc Parisien pour ses critiques et nos nombreuses discussions de protéines. Le hongrois Tamás Marcinkovics pour sa compagnie dans le petit local, son aide avec la langue de Shakespeare et ses questions embêtantes sur la langue de Molière. Une pensée toute spéciale pour Marc Hallé. Marc qui aura été la personne avec laquelle j'aurai échangé des idées de « protein folding » les plus ésotériques. Bon voyage mon « chum » tu me manqueras. Les « luncheux » du midi pour tous les débats et les critiques littéraires, cinématographiques et autres. Tous les joueurs de babyfoot du labo sans qui ces deux dernières années n'auraient pas été aussi distrayantes. Mon fidèle compagnon de travail, Sulley, pour avoir travaillé sans se tromper et se fatiguer. Finalement, François Major, pour m'avoir accepté dans son laboratoire, subventionné, lu, corrigé et permis de rencontrer tous ces gens que je n'oublierai jamais.

À vous tous merci.

CHAPITRE 1

Mise en situation

Présentation du mémoire

Ce mémoire à pour but de présenter différentes méthodes permettant l'analyse des données de biopuces. Le premier chapitre a pour but de définir les bases essentielles à la bonne compréhension de ce mémoire. Les sections qui suivent décrivent le travail qui a été effectué dans le cadre de cette maîtrise. Le second chapitre traite de façon précise d'une implantation permettant de parcourir efficacement l'espace des structures acycliques des réseaux bayesiens. De plus, il présente une méthode inédite de calcul du nombre de réseaux acycliques accessibles à partir d'un réseau courant. Le troisième chapitre traite de l'évaluation de l'ajout de certaines connaissances *a priori* sur la précision de l'apprentissage des réseaux bayesiens. Par la suite, l'analyse effectuée sur des données Affymetrix provenant de souris normales et cancéreuses est présentée, le but étant l'extraction de gènes différemment exprimés. L'avant dernier chapitre présente une analyse à l'échelle du génome humain permettant d'extraire des relations inhibiteurs-inhibés à partir de données d'expression de gènes.

Introduction

L'humain adulte est constitué d'environ 1014 types cellulaires différents [1]. Parmi ceux-ci nous pouvons énumérer les cellules de type sanguin, musculaire, de la peau et beaucoup d'autres. Ces types cellulaires diffèrent tant au niveau de leurs formes que de leurs fonctionnements internes. Si nous considérons qu'à notre création, nous provenons d'une seule cellule, il est tout à fait légitime de s'interroger sur les mécanismes qui permettent à cette fameuse première cellule de former un organisme pluricellulaire aussi complexe que l'humain. L'état actuel de la science ne nous permet pas de lever un voile complet sur toute cette instrumentation, mais elle nous permet d'en expliquer certaines parties que nous tenterons de mettre en lumière dans les sections qui vont suivre. De plus, nous aimerions ajouter, que le but de ce travail est d'apporter un peu plus d'information au sujet de cette instrumentation en utilisant les données d'expression de gènes.

Contexte biologique

L'acide desoxiribo-nucléique(ADN)

À la naissance, nous recevons 21 chromatides de notre père et 21 de notre mère. Les paires homologues s'assemblent aux environs de leur centre pour former 21 chromosomes. La molécule principale des chromatides, et donc des chromosomes, est l'acide désoxiribo-nucléique, plus communément appelée ADN. Cette molécule est constituée de quatre éléments de bases appelés nucléotides. Nous retrouvons donc dans l'ADN deux groupes de nucléotides, les purines : adénine (A) et guanine (G) et les pyrimidines : cytosine (C) et thymine (T). La structure de l'ADN a pendant longtemps été un mystère, mais en 1953, Watson et Crick publiaient la célèbre structure de la double hélice [2]. En effet, l'ADN est formée de deux brins qui s'entortillent pour former une structure ayant l'apparence d'un ressort. Les brins d'ADN sont formés d'une chaîne de nucléotides et chaque nucléotide dans un brin est associé à un autre dans l'autre brin selon la règle d'appariement A - T , C – G et inversement.

L'ADN emmagasine toutes les instructions pour produire des outils qui seront utilisés par la cellule pour réagir à différentes situations. Une liste d'instructions correspond à une partie de l'ADN de taille variable et est couramment appelé gène. Selon Jean-Michel Claverie de l'Institut de Biologie Structurale et Microbiologie de Marseille, il y a environ 30,000 gènes répartis au travers des chromosomes chez l'être humain [3]. De ces 30,000 gènes, environ 24,500 codent pour des protéines [4]. Un gène est une séquence d'ADN, donc de nucléotides, qui contient l'information pour produire sa protéine correspondante.

Les eucaryotes

Les eucaryotes, organismes auxquels appartiennent les humains, les plantes et les champignons, ont comme point en commun des cellules ayant un noyau. De plus, les gènes des eucaryotes sont composés de plusieurs parties de deux types différents, les introns et les exons. Les parties que nous appelons codantes, car elles contiennent les instructions pour la construction de la protéine, sont nommées exons. Les parties non codantes, qui ne sont pas utilisées pour produire la protéine, sont appelées introns. Longtemps, les introns ont été appelés la matière non fonctionnelle de l'ADN, mais depuis un certain temps, les scientifiques se questionnent sur le rôle qu'a à jouer cette matière. En effet, plus de la moitié

de notre ADN ne code pas pour des protéines. À quoi peut bien servir cette partie d'ADN sachant très bien qu'il est improbable, la nature étant ce qu'elle est, qu'elle ne serve à rien ? Nous ne traiterons pas en détail de ce domaine dans ce mémoire mais un lecteur intéressé pourrait consulter des articles pertinents de Bejerano, Haussler, Blanchette et Down, Hubbard [5][6].

Les procaryotes

Les procaryotes, pour la plupart des bactéries, ont la caractéristique de posséder des cellules ne possédant pas de noyau. Ces organismes ont des gènes constitués essentiellement d'exons.

De l'ADN à la protéine

Maintenant que nous avons une idée de ce qu'est l'ADN, nous allons voir dans les sections qui suivent, comment cette super molécule peut être utilisée pour amener la vie.

Comme mentionné précédemment, les gènes codent pour des protéines. Une protéine est plus précisément une chaîne d'acides aminés ayant une forme bien particulière dans son environnement. Un des principes fondamental de la biologie est que la forme de la protéine dicte sa fonction. La chaîne de la protéine peut être formée de 20 unités de bases différentes, les acides aminés. Ces acides aminés peuvent être regroupés en 4 groupes : les hydrophobes, les hydrophiles, les acides et les basiques. Toute la variété que nous retrouvons dans les structures des protéines provient, entre autre, des caractéristiques physico-chimiques distinctes de chacun des acides aminés. Le processus permettant à une partie d'ADN de produire une chaîne d'acides aminés sera décrit dans la section suivante.

La première étape du processus transformant un gène en protéine est effectuée par une enzyme ayant le nom d'acide ribonucléique (ARN) polymérase. Cette molécule, essentielle à la vie, fut découverte indépendamment par Sam Weiss et Jerard Hurwit en 1960. Elle a pour fonction d'effectuer la transcription qui consiste à copier une partie de l'ADN double brin en ARN. L'ARN ressemble à l'ADN à la différence qu'elle est simple brin et qu'au lieu de la thymine, on retrouve plutôt l'uracile (U). Suite à cela, si nous sommes chez les eucaryotes, la molécule d'ARN sera épissée. L'action d'épisser consiste à l'élimination des introns de l'ARN. Cette action est effectuée par un complexe d'ARN et de protéines que

l'on appelle spliceosome. Cette grosse molécule sait faire la différence entre introns et exons grâce à une séquence de nucléotides spécifiques à chacun [7]. Une fois l'ARN épissé, une chaîne contenant seulement des exons est obtenue. Cette molécule est communément appelée ARN messager (ARNm). À noter, dans le cas des procaryotes, nous obtenons directement de l'ARNm, car ces organismes n'ont pas d'introns. Ensuite, une guanine modifiée est ajoutée à un des bouts de l'ARNm pour permettre la reconnaissance de cette molécule par le ribosome (décrit plus loin). De plus, une longue séquence d'adénine est ajoutée à l'autre bout de l'ARNm pour que cette dernière puisse vivre plus longtemps à l'intérieur de la cellule. Cette étape très importante, car elle permet à l'ARNm de produire plus d'une protéine avant d'être détruite, s'appelle la polyadénylation. Dans le cas des eucaryotes, la nouvelle molécule d'ARNm est transportée à l'extérieur du noyau, dans le cytoplasme, par des protéines chaperonnes.

Une fois l'ARNm dans le cytoplasme, elle ne tardera pas à rencontrer le ribosome, un très gros complexe de molécules d'ARN et de protéines ribosomales. Cette macro molécule a comme rôle de lire l'ARNm par section de 3 nucléotides appelés codons. Chaque codon correspond à un acide aminé particulier. Le ribosome lit donc les codons et, une autre molécule, l'ARN de transfert (ARNt), amène les acides aminés correspondant aux codons. Le ribosome s'occupe aussi d'assembler les acides aminés entre eux de façon à constituer la protéine, acide aminée par acide aminée. Par la suite, l'ARNm est relâchée et il y a production d'une protéine.

Les réseaux de régulation

Tout ce que nous avons vu préalablement explique comment un gène peut donner une protéine, mais ça n'explique pas pourquoi la cellule « décide » de produire cette protéine. En effet, il est bien connu que la cellule est très structurée et organisée et qu'elle ne produit que les protéines donc elle a besoin. Comme la cellule n'est pourvue d'aucune forme d'intelligence, il y a donc des processus biologiques et chimiques qui lui confèrent ces capacités de produire les bonnes protéines au bon moment. L'état de la science actuelle ne réussit pas à expliquer tous les mécanismes, mais les réseaux de régulation, qui seront présentés subséquemment, sont pour le moment ce que nous connaissons le mieux.

Pour expliquer ce qui permet à la cellule de produire à un moment donné les protéines essentielles, nous utiliserons un exemple bien connu de régulation : l'opéron lactose chez le procaryote *Escherichia coli* (*E. coli*), une bactérie indispensable à la digestion chez les animaux et les mammifères. Quand son environnement le permet, elle utilise le glucose qui l'entoure pour produire son énergie. Nous verrons au cours des lignes qui vont suivre, comment cette bactérie réagit lorsqu'il n'y a plus de glucose dans son environnement, mais seulement du lactose.

Cette bactérie dispose principalement de trois enzymes pour utiliser le lactose : la β-galactosidase, la β-galactoside et la β-perméase. La première et la deuxième permettent à la bactérie de dégrader la molécule de lactose pour produire de l'énergie et la dernière permet de faire entrer plus rapidement le lactose à l'intérieur de la cellule. En temps normal, quand il n'y a pas de lactose et seulement du glucose dans l'environnement d'*E. coli*, les gènes codant pour les trois protéines sont inaccessibles à l'ARN polymérase. L'élément qui empêche la polymérase d'accéder à la partie d'ADN utile à l'emploi du lactose est une protéine spéciale ayant le nom de facteur de transcription. Un facteur de transcription est une protéine qui s'accroche à l'ADN pour influencer, positivement ou négativement, l'action de la transcription par l'ARN polymérase. En effet, ces protéines peuvent autant inhiber qu'accélérer l'effet de la polymérase. Dans notre cas, le facteur de transcription accroché à l'ADN qui empêche la transcription de l'ARNm des protéines aidant à la dégradation du lactose, est la protéine produite par le gène lac I.

Nous pouvons constater que la cellule réagit de façon très efficace à la situation dans laquelle elle est. En effet, il n'y a pas de lactose et elle ne produit pas les protéines utilisées pour l'utilisation du lactose. Que se passe-t-il maintenant quand il n'y a plus de glucose et seulement du lactose dans l'environnement *d'E. coli* ? Ce qui se passe c'est que le facteur de transcription accroché à l'ADN et empêchant l'ARN de transcrire a aussi de l'affinité pour la molécule de lactose. Quand une molécule de lactose passe au environ du facteur de transcription, celui-ci change de conformation pour se coller au lactose relâchant ainsi le de brin d'ADN. Nous pouvons voir ce phénomène comme une clef et un cadenas. Le lactose est la clef qui débarre le cadenas bloquant l'accès à la polymérase. Une fois le groupe de gènes codant pour les protéines dégradant le lactose accessible à l'ARN polymérase, la bactérie peut utiliser le lactose. De plus, le manque de glucose a aussi pour effet la production d'un complexe de protéines pouvant s'accrocher en amont de la partie d'ADN codant pour les

gènes du lactose. Ce complexe a comme effet d'attirer la polymérase accélérant ainsi la production des protéines. Encore une fois, nous constatons que la bactérie réagit de façon adéquate à toutes ces situations. Pas étonnant que nous la croyons intelligente !

Pour résumer, nous disons donc que la protéine empêchant la polymérase de lire la partie de l'ADN codant pour les protéines du lactose est une inhibitrice et le complexe de protéines produit par le manque de glucose est un activateur. Nous pouvons représenter ces relations sous forme d'un réseau. Dans ce réseau, les nœuds représentent les gènes et les liens entre les gènes représentent les relations entre ceux-ci : inhibition ou activation. Habituellement, l'activation est représentée par une flèche tandis que l'inhibition est représentée par un lien avec une ligne perpendiculaire au bout. Nous appelons ce réseau un réseau de régulation, car il montre la façon dont les gènes sont régulés.

Au tout début de cette section, nous parlions du grand nombre de cellules différentes chez l'humain. Très loin de nous, au niveau de la quantité de types cellulaires, la levure possède 3 types différents de cellules [8]. Chacun de ces types cellulaires diffère des autres par l'ensemble unique des gènes qu'il exprime. Ceci est dû à des facteurs des transcriptions qui sont répartis dans le génome et qui bloquent ou activent des gènes. Sans vouloir simplifier à tord la complexité cellulaire humaine, il ne serait pas bête de prétendre qu'un même système pourrait être utilisé chez l'humain pour expliquer la pluricellularité. Par exemple, les éléments cellulaires qui différencieraient le neurone du muscle pourraient se limiter uniquement aux facteurs de transcription. Bien évidemment, tout n'est pas aussi simple et des mécanismes comme les micro ARN viennent s'ajouter aux facteurs de transcription pour tenter de clarifier les mystères de la diversité cellulaire chez l'humain [9][10].

Note : Pour plus d'information au sujet des notions biologiques traitées dans la partie précédente, nous référons le lecteur à un livre de biologie cellulaire écrit par Lodish, Baltimore, Berk, Zipursky, Matsudaira et Darnell [8].

Les biopuces à ADN

Depuis un certains nombres d'années, il y a un outil que les biologistes utilisent de plus en plus lors de leurs expériences. Cet outil est typiquement utilisé pour connaître l'état d'un groupe de cellules à un moment donné. L'outil en question porte le nom de biopuce à

ADN ou « microarray » en anglais. Dans les lignes qui suivent, nous allons décrire brièvement ce qu'est une biopuce et comment elle fonctionne. Nous allons ensuite effleurer les sujets de recherche en bioinformatique s'y rattachant.

Description des biopuces

Pour débuter, décrivons comment il est possible d'obtenir le niveau d'expression d'un gène grâce aux biopuces. La première des choses à savoir au sujet des biopuces est que nous recherchons la quantité d'ARN messager (ARNm) associée à un gène de la cellule à un moment donné. L'hypothèse de base de cette technologie est que la quantité d'ARNm nous informe sur le niveau d'expression d'un gène. Voici maintenant, dans l'ordre, les étapes pour connaître la quantité d'ARNm associée à un gène.

Pour débuter, il faut obtenir une copie de l'ARNm que l'on veut étudier. Suite à cela, il faut obtenir le brin complémentaire à cet ARNm. Par exemple, si le brin d'ARNm que nous voulons étudier est A-G-G-C-T-G, nous allons avoir besoin du brin complémentaire T-C-C-G-A-C. Une des choses importantes à savoir au sujet de ce brin complémentaire est que c'est lui qui a le plus d'affinité avec le brin que nous voulons étudier. En d'autres mots, si le brin complémentaire et l'ARNm que nous voulons étudier flottaient séparément dans le même liquide, les chances sont très fortes pour que les deux s'associent après un certain temps. C'est grâce à ce principe d'attraction mutuelle entre brins complémentaires que les biopuces fonctionnent. L'idée est de fixer le brin d'ARN complémentaire sur une plaque et de faire baigner cette plaque dans une solution contenant potentiellement l'autre partie complémentaire.

Dans les faits, on ne place pas seulement un brin complémentaire sur la plaque mais un très grand nombre regroupés dans le même secteur (qu'on appelle point ou « spot »). Une fois la plaque construite, on s'en sert pour extraire les brins d'ARNm présents dans la cellule. Une fois les brins extraient de la cellule, il faut les marquer à l'aide d'un composé. Ce composé est bien spécial, car lorsqu'il est excité à une longueur d'onde spécifique, il devient fluorescent et il est possible de l'observer. Une fois les brins marqués, nous les mettons en contact avec la plaque précédemment formée dans une solution pendant un certain temps. Le temps que les brins complémentaires puissent se retrouver. Pour les aider, il faut agiter la solution dans laquelle se trouve les ARNm. Suite à cela, il faut purger la solution et garder

seulement la plaque. Dans la solution, il reste les brins d'ARNm non appariés, parce qu'ils n'ont pas été en mesure de trouver leur brin complémentaire ou tout simplement qu'ils n'étaient pas à l'étude dans cette expérience.

Par la suite, la plaque est excitée grâce à un laser émettant à la longueur d'onde spécifique au composé utilisé pour marquer les ARNm. Une photo du résultat de l'excitation est prise et ceci constitue le résultat de l'expérience. Cependant, une photo donne beaucoup d'information qu'il faut d'abord analyser pour qu'elle devienne accessible, ce qui n'est pas vraiment pratique. Le traitement de la photo sera élaboré un peu plus en détails plus loin, car c'est un champ de recherche en bioinformatique.

Un fait intéressant au sujet des plaques est qu'il est possible de faire de la comparaison de niveau d'expression entre deux cellules en marquant l'ARNm d'une cellule d'une couleur et l'ARNm de l'autre cellule d'une autre couleur. On marque la première cellule avec un marqueur vert et l'autre avec un marqueur rouge, on prend une photo en excitant la plaque avec un laser vert et une photo en excitant avec le laser rouge et on superpose les images. Les endroits où il y aura du jaune seront les endroits où le niveau d'expression est identique.

Il est à noter qu'il y a deux façons de faire de l'analyse à l'aide de biopuces; il y a la méthode Synteni/Stanford et la méthode utilisée par la compagnie Affymetrix. Les différences entre les méthodes se situent au niveau de la façon dont les plaques sont construites et aussi au niveau de ce qui est déposé sur celles-ci. Dans le cas des plaques Affymetrix, la photolithographie est utilisée pour la construction et ce n'est pas un brin complet d'ARNm complémentaire qui est déposé sur la plaque, mais bien plusieurs petits fragments complémentaires ainsi que plusieurs petits fragments pratiquement complémentaires. Les brins pratiquement complémentaires sont introduits pour évaluer le taux d'erreur.

La construction des biopuces est vraiment très intéressante et il aurait été possible d'en parler pendant plusieurs pages. Le lecteur intéressé trouvera plus d'information à ce sujet en consultant deux sites Internet [11][12].

Survol de la bioinformatique dans le domaine des biopuces

Il existe plusieurs problèmes reliés aux biopuces qui intéressent les bioinformaticiens. Certains de ces problèmes seront décrits brièvement dans les lignes qui suivent.

Tout d'abord, une des premières choses à faire suite à une expérience avec les biopuces, est d'analyser le résultat obtenu. Pour cela, il faut prendre la photo du résultat et en extraire la quantité d'ARNm se retrouvant à un point donné de l'image. En effet, les molécules d'ARN complémentaires sont regroupées par point sur les plaques. Les techniques d'analyse d'images ont été appliquées et adaptées pour déterminer l'intensité de fluorescence d'un point. La reproduction d'une photo d'une puce qui suit peut nous en donner une idée :

Figure 1 : Reproduction d'une partie de biopuce. Nous pouvons voir le bruit introduit par la méthode. À noter que les zones ne sont pas parfaitement rondes et ceci représente la réalité. Habituellement, un spécialiste ou un programme informatique déterminera les zones d'intérêts. Chacune des zones n'est pas uniforme et ceci provient des bruits introduits par la méthode expérimentale. En temps normal, une moyenne de l'intensité d'une zone est utilisée pour évaluer la quantité d'ARNm et dans le cas où l'écart type est trop grand, la valeur est annotée comme improbable.

Comme on peut le voir à la figure 1, dans le cas du point en bas à droite, l'intensité n'est pas uniforme et la taille de chacun des points n'est pas fixe. Un article du groupe de Terry Speed résume les techniques existantes adressant ce problème [13]. Dans cet article, l'auteur divise le procédé d'extraction de l'intensité en 3 parties. Premièrement, il faut identifier les points d'intérêts sur l'image. Il est possible pour un humain de faire cette tâche, mais étant donné le grand nombre de point sur une photo, certains programmes informatiques ont été développés pour adresser ce problème. Deuxièmement, il faut être en mesure de déterminer l'intensité réelle et celle de fond (bruit). La dernière partie est celle de l'extraction du niveau d'intensité comme telle.

Une fois l'intensité de fluorescence identifiée, il faut déterminer à quelle quantité d'ARNm cette intensité peut correspondre. Pour cela, une expérience de calibration de l'instrument est effectuée [14]. Dans le cas de l'article de Hubbell, Liu et Mei, il est question des biopuces Affymetrix, mais la méthode peut aussi s'appliquer aux biopuces de Synteni/Stanford [14]. Pour ce faire, il faut préparer une expérience spéciale où les niveaux d'expression en ARNm sont connus à l'avance. L'équipe de Hubbell a fait l'expérience en utilisant 14 concentrations d'ARNm différentes variant de 0 à 1024 {0, 0.25, .5, 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024}. Ils ont aussi utilisé une méthode de carré latin pour construire 14 expériences avec ces concentrations. La méthode de carré latin pour l'expérience se résume à changer la concentration d'un seul ARNm à la fois. Le carré latin est utilisé pour être en mesure de faire une analyse statistique convenable au sujet de la concentration et de la variabilité d'une expérience à l'autre [15]. Une fois les expériences effectuées, comme nous connaissons les concentrations, il est possible d'estimer un modèle de calibration permettant d'associer une concentration à une intensité.

Une fois que nous connaissons la concentration en ARNm, nous pouvons utiliser cette concentration comme le niveau d'expression du gène associé à cet ARNm. Encore une fois, beaucoup de recherche est effectuée pour extraire de l'information pertinente de ces concentrations de gènes. Cependant, dans ce domaine, une seule expérience (concentration d'un ensemble de gènes) ne suffit pas. Il nous faut un ensemble de concentrations provenant d'expériences différentes. Il y a plusieurs types d'expériences différents. Nous pouvons faire des expériences séquentielles dans le temps sur la même cellule dans la même situation. Dans ce cas, il est question de suivre l'évolution du niveau d'expression d'un même gène dans le temps. Il peut aussi s'agir de varier les conditions dans laquelle se trouve la cellule. Par exemple, il n'est pas rare de voir des expériences comparant la concentration des gènes de la cellule dans un environnement normal et dans un environnement anormal. Par environnement anormal, il peut être question de chocs thermiques (chaud, froid) ou da mutation (inactivation) d'un gène. Pour plus de détail à ce sujet, nous suggérons la lecture des articles du groupe de Futcher et du groupe de Sherlock [16][17].

Une fois l'ensemble des résultats obtenus, il est possible d'utiliser plusieurs méthodes pour les analyser. Une façon est de regrouper les gènes selon une évaluation de leur niveau de ressemblance au niveau de leur expression. Il est possible d'y arriver en utilisant des mesures comme la distance euclidienne, la corrélation de Pearson (degré de relation linéaire entre

deux variables [18]) ou la corrélation Spearman [19]. L'hypothèse à la base de ce regroupement est que deux gènes ayant un niveau de ressemblance élevé auront une fonction similaire. Cette hypothèse permet de déterminer certaines fonctions pour des gènes inconnus en essayant de trouver le gène ayant une fonction connue y ressemblant le plus. Un grand nombre de techniques de forage de donnée « data mining » ont été utilisées dans le but d'extraire des liens entre les gènes. Des techniques comme l'analyse des composantes principales (PCA) [20], les réseaux de neurones [21] et plusieurs algorithmes de regroupement « clustering » [22][23] ont tous été appliqués à ce problème.

Contexte informatique

Dans le cadre de cette recherche, certains outils d'analyse, comme les réseaux Bayesiens, ont été utilisés. Les lignes qui suivront traiteront plus en détail de cet outil et de l'état de la recherche dans ce domaine et ceci de manière à être en mesure de bien situer l'apport de notre étude.

Les réseaux Bayesiens

Aux environs de 1763, Thomas Bayes publiait un essai intitulé : « An Essay towards solving a Problem in the Doctrine of Chances ». Sans le savoir, il venait, à ce moment, de créer les fondations de la théorie bayesienne. La base de sa théorie repose sur l'équation 1, toute simple, mais combien puissante.

$$P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)}$$

De plus, si l'on suppose n évènement exclusifs et exhaustifs pour i allant de 1 à n :

$$P(A_i \mid B) = \frac{P(B|A_i)P(A_i)}{P(B|A_1)P(A_1) + P(B|A_2)P(A_2) + \ldots + P(B \mid An)P(A_n)}$$

Où

$$P(A \mid B) = \frac{P(A,B)}{P(B)}$$

Équation 1 : Notions de base sur les probabilités conditionnelles.

Pour bien comprendre cette l'équation 1, nous allons supposer qu'A est une variable nous informant sur le fait qu'il ait plu ou non – elle peut donc prendre deux valeurs « il a

plu » et « il n'a pas plu »- et B nous informe sur si le gazon est mouillé à l'extérieur – encore là deux valeurs, « le gazon est mouillé » et « le gazon n'est pas mouillé ». Dans le théorème de Bayes, P(A,B) est une probabilité jointe et doit être lue comme étant la probabilité d'obtenir A et B. Dans notre exemple, cela peut être vu comme la probabilité d'obtenir à la fois une des quatre combinaisons possibles ({« il a plu », « il n'a pas plu »}x{« le gazon est mouillé », « le gazon n'est pas mouillé »}).

Nous pouvons aussi observer une probabilité conditionnelle P(A|B) qui doit être lue comme la probabilité de A étant donné B. Encore une fois nous pouvons observer quatre cas . Par exemple, quelle est la probabilité « qu'il ait plu » si nous observons que le gazon est mouillé ? Chacun des termes du théorème a un nom précis. La probabilité conditionnelle P(A|B) est la probabilité à posteriori. Ceci peut être vu comme la probabilité d'observer A en utilisant l'information que nous avons sur B. Par exemple, la probabilité qu'il ait plu sera plus élevée si nous savons que le gazon est mouillé que s'il ne l'est pas P(« il a plu »| « le gazon est mouillé ») > P(« il a plu »| « le gazon n'est pas mouillé »). Le terme P(A) est appelé la probabilité *a priori*. Ce terme nous informe donc sur la probabilité de A supposant que nous n'avons aucune connaissance de B. Dans notre exemple, ceci pourrait être la valeur de probabilité donnée par les météorologues à la télé ou la radio. Il arrive parfois que nous ayons la valeur de P(A) directement, mais quelque fois, il faut marginaliser la probabilité jointe pour obtenir ce que l'on veut. L'équation 2 montre comment il est possible de marginaliser.

$$P(A=a)=\sum_{B,C,D} P(A=a,B,C,D)$$

Équation 2 : Marginalisation de la probabilité jointe.

P(B|A) ou L(A|B) est la fonction de vraisemblance. La différence entre P(A|B) et P(B|A) est la position de la variable que nous fixons. Dans notre cas nous fixons B. Quand la variable fixée est à droite de | nous appelons la probabilité une probabilité *a posteriori* et dans le cas où elle est à gauche de | nous parlons de vraisemblance. Nous en sommes maintenant rendu au dernier terme, P(B) qui est le terme de normalisation. L'équation 3 résume la théorie de Bayes.

$$\text{posteriori} = \frac{\text{vraisemblance} * \text{priori}}{\text{constante}}$$

Équation 3 : Résumé théorie de Bayes.

Pour tenter de démontrer l'intérêt et la puissance du théorème de Bayes, nous allons utiliser un exemple tiré de Neapolitan [31].

Supposons un homme qui débute un emploi dans une compagnie. Comme il en est souvent le cas, la compagnie demande un examen médical avant embauche. L'homme passe donc un examen au rayon-X pour détecter la présence de cancer. Les résultats lui reviennent et il apprend qu'il a un résultat positif pour le cancer du poumon. La panique s'empare donc de l'homme, mais a-t-il raison ? Le problème est qu'il ne connaît pas la précision du test. Quand il apprend que la précision du test n'est pas absolue, il décide de vérifier à quel point il est juste de s'alarmer. Il apprend alors que le test a un taux de vrais positifs de .6 et un taux de faux positif de .02. Les lignes qui suivent résument alors ses connaissances :

$$P(\text{Test} = \text{positif} \mid \text{Cancer du poumon} = \text{présent}) = .6$$

$$P(\text{Test} = \text{positif} \mid \text{Cancer du poumon} = \text{absent}) = .02$$

Nous pouvons remarquer dans ces deux probabilités conditionnelles, deux variables : « Test » qui représente la valeur obtenue lors du test, ayant comme valeur positif et négatif, ainsi que « Cancer du poumon » qui nous donne l'état du patient, ayant comme valeur présent ou absent. Le .6, représente le fait que le test n'est pas sûr à 100%, est réconfortant, mais à quel point est-il probable que l'homme ait le cancer ? La probabilité qui nous intéresse à ce moment est la probabilité d'avoir le cancer étant donné que nous avons un résultat positif. Cette information est représentée par P(Cancer du poumon = présent | Test = positif) et la valeur de cette probabilité conditionnelle n'est pas donnée directement. En utilisant le deuxième format du théorème de Bayes, nous pouvons donc écrire cette probabilité en fonction de celles que l'ont connaît :

$$P(\text{présent} \mid \text{positif}) = \frac{P(\text{positif} \mid \text{présent})P(\text{présent})}{P(\text{positif} \mid \text{présent})P(\textit{présent}) + P(\textit{positif} \mid \textit{absent})P(\textit{absent})}$$

Nous remarquons qu'il semble nous manquer une valeur soit la probabilité *a priori* d'avoir le cancer du poumon. Dans le cas de notre homme, étant donnée son état et qu'il est non-fumeur, il évalue ses chances a 1 sur 1000 d'avoir le cancer. Nous avons maintenant tous les éléments pour résoudre l'équation :

$$P\langle \text{présent} | \text{positif} \rangle = \frac{(.6)(0.001)}{(.6)(0.001)+(.02)(.999)} = .029$$

La valeur .029 correspond donc à la chance que l'homme a d'avoir le cancer. 0.29 étant une valeur peu élevée, l'homme peut dormir sur ses deux oreilles et ceci grâce à Thomas Bayes !

Du théorème de Bayes au réseaux bayesiens

Précédemment, nous avons vu un exemple concernant seulement 2 variables. Il est très rare en réalité que nous rencontrons des problèmes aussi simples. Par exemple, les réseaux de régulation peuvent être composés d'une centaine voir même d'un millier de variables. Dans ces situations, nous avons besoin d'un outil nous permettant de façon simple et rapide de calculer la probabilité jointe sur l'ensemble des variables *i.e.* P(A=a,B=b,…,Z=z). C'est donc précisément pour calculer la valeur de cette probabilité jointe que nous utilisons les réseaux bayesiens.

Tout d'abord, les réseaux bayesiens sont des graphes dirigés. Un graphe dirigé peut être définit à l'aide d'une liste de nœuds N={N1,N2,…,Nn} et d'une liste de liens dirigés L entre les noeuds. Nous disons donc que la structure de notre réseau bayesien peut être exprimée de façon précise en utilisant seulement le couple de variables {N, L}. De plus, le réseau bayesien associe à chacun des nœuds une probabilité conditionnelle $P(N1|Pa_{N1})$ où Pa_{N1} représente les nœuds **Pa**rents du nœud Ni. Si un nœud n'a pas de parents, la probabilité conditionnelle est une probabilité *a priori* P(Ni) pour le nœud en question. La figure 2 montre un réseau bayesien de 5 variables. Elle montre aussi un tableau de probabilité conditionnelle pour la variable C. Il est a noté que dans ce cas chacune des variables est binaire, elle peut prendre la valeur 0 ou 1. Un réseau bayesien est une façon graphique de représenter des notions de dépendances et d'indépendances entre des variables. Par exemple, la valeur que prendra la variable C dans le réseau de la figure 2 dépend directement de la valeur de la variable A et B.

P(C|A,B)			
		C	
A	B	0	1
0	0	0.75	.25
0	1	0.5	0.5
1	0	0.40	0.60
1	1	0.9	0.1

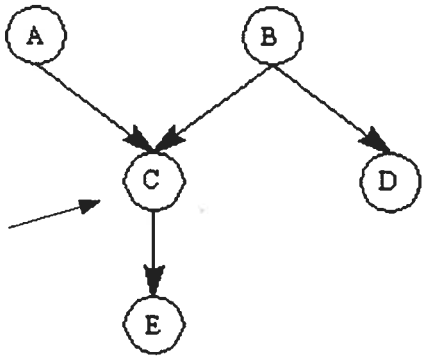


Figure 2 : Structure d'un réseau bayesien de 5 variables avec la table de probabilité conditionnelle pour une des variables. Réseau bayesien de 5 variables binaires {A-E} pouvant prendre la valeur 0 ou 1. La structure du graphe peut être décrite en utilisant N={A,B,C,D,E} et L={(A,C),(B,C),(B,D),(C,E)}. La liste des parents pour chacun des nœuds est PaA={},PaB={},PaC={A,B},PaD={B} et PaE={C}. Pour calculer la probabilité jointe P(A,B,C,D,E), nous utilisons les notions d'indépendance conditionnelle décrite par le réseau. Donc P(A,B,C,D,E) = P(A)P(B)P(C|A,B)P(D|B)P(E|C). Nous pouvons aussi observer la table de probabilité conditionnelle de la variable C. À noter que tous les nœuds du réseau ont aussi des tables, mais dans le but d'alléger l'image, elles ont été omises.

Nous pouvons aussi affirmer que les variables A et B ne dépendent d'aucune autre variable. Dans un réseau bayesien, il est simple d'extraire les notions de dépendances entre les variables, car une variable A dépend directement de ses parents P_A s'ils sont observés. En effet, il peut arriver, dans certaines situations, que les parents ne soient pas tous observés. Dans le cadre de cette recherche, nous allons toujours utiliser des données où toutes les variables sont observées, les nœuds dépendront donc toujours directement de leurs parents. Cette supposition simplifie grandement le calcul de la probabilité jointe. L'équation 4 nous montre comment calculer cette probabilité.

$$P(N_1, N_2, \ldots, N_n) = \prod_{i=1}^{n} P(N_i \mid Pa_{N_i})$$

Équation 4 : Calcul de la probabilité jointe sur toutes les variables du réseau bayesien. À noter que nous supposons que toutes les variables sont observées et ceci nous permet de limiter les probabilités conditionnelles entre une variable et ses parents.

La structure d'un réseau bayesien n'est pas forcée d'être acyclique. Cependant, le fait qu'il n'y ait pas de cycles dirigés garantie que nous obtiendrons des distributions de probabilités cohérentes [32]. Dans le cadre de cette recherche, nous nous intéressons aux

dépendances directes entre les variables de notre réseau. Pour cette raison, nous ne décrirons pas le concept de d-séparation qui nous permet de déterminer les indépendances conditionnelles entre les variables. Nous référons néanmoins le lecteur à Neapolitan [31] pour plus de détails sur ce concept et sur les réseaux bayesiens en général.

Apprentissage de la structure d'un réseau bayesien à partir de données

Il y a plusieurs raisons qui poussent les chercheurs à utiliser les réseaux bayesiens plutôt que d'autres méthodes pour découvrir des dépendances entre des variables. La première est qu'il est possible de travailler avec des données manquantes. En effet, il est possible de déterminer la valeur de la probabilité jointe quand nous n'observons pas toutes les variables. Nous ne traiterons pas de ces méthodes dites d'inférence, mais nous pouvons en nommer quelques unes comme la méthode d'inférence exacte utilisant les arbres de jonction, ainsi que les méthodes d'inférence approximatives d'échantillonnage logique et les méthodes abductives [31].

Il n'y a pas seulement pour l'inférence qu'il est possible d'utiliser les réseaux bayesiens en présence de données manquantes. Il est aussi possible d'apprendre la structure. Des techniques comme EM (Expectation Maximization) permettent d'y arriver. Voici les grosses lignes d'un algorithme permettant d'apprendre la structure en pareille situation [36]:

Procédure permettant l'apprentissage de la structure en présence de données manquantes :

1-Nous initialisons les données manquantes aléatoirement

2-Nous apprenons la structure du réseau optimal en utilisant les algorithmes que nous allons voir subséquemment.

3-Nous attribuons à nouveau des valeurs aux données manquantes, mais cette fois en utilisant la structure apprise pour déterminer les valeurs optimales.

4-Recommencer les étapes 2 et 3 tant que nous réussissons à améliorer la vraisemblance de la structure du réseau.

Figure 3 : Pseudo-code pour l'apprentissage de la structure en présence de données manquantes.

Une autre facette intéressante des réseaux bayesiens est qu'il est facile d'y ajouter nos connaissances *a priori*. Par exemple, si nous connaissons un lien entre deux variables, il est possible d'insister pour que la structure optimale contienne absolument ce lien. Par exemple,

en associant une probabilité *a priori* de zéro à un réseau ne contenant pas cette dépendance. En dernier lieu, les méthodes statistiques bayesiennes en conjonction avec les réseaux bayesiens forme un couple permettant d'éviter le surentraînement [33].

Principe de base

Lorsque nous utilisons les réseaux bayesiens pour apprendre les notions de dépendance entre des variables, nous essayons de déterminer la structure qui obtiendra la probabilité *a posteriori* la plus élevée étant donnée les données observées. Supposons donc que nous avons un ensemble de variables V={v1,v2,…,vn} que nous observons dans des ensembles de données D={d1,d2,…,dm}. Par exemple, pour l'ensemble de donnée d1, nous avons une valeur pour chacune des variables. Dans les cas biologiques qui nous intéressent, un ensemble de donnée dm correspond aux valeurs réelles d'expression pour chacun des gènes sur une biopuce. Passons maintenant à une définition plus formelle. Nous cherchons la structure d'un réseau bayesion $R=\{N_{rb},L_{rb}\}$, qui aura la plus grande probabilité *a posteriori* étant donné les données. L'équation 5 décrit formellement la fonction à optimiser.

$$\hat{R} = \arg\max_{R} score(R, D)$$

$$score(R, D) = \prod_{i=1}^{n} P(R \mid d_i)$$

Équation 5 : Formalisme d'optimisation. Nous devons déterminer la structure optimale $\hat{R}$ qui optimisera la probabilité *a posteriori* du réseau étant donné les données.

Nous devons donc parcourir l'espace de toutes les structures de réseau possibles à la recherche de cet optimum. Ceci peut se faire assez efficacement en partant d'un réseau de départ quelconque et en appliquant une modification de base qui fait que le score(R,D) associé à ce réseau s'améliore. Nous appliquons des modifications tant que le score s'améliore. Les modifications que nous pouvons appliquer sont l'addition, la soustraction ou le retournement d'un lien. Une modification est jugée valide si suite à l'application de cette modification au réseau courant, celui-ci demeure acyclique. L'algorithme peut se résumer ainsi :

Procédure de recherche de la structure optimale:

1-R ← réseau de départ acyclique (souvent avec une structure aléatoire)
2-Tant que score(R,D) s'améliore
3- LM ← liste des modifications valides
4- M ← modification dans LM améliorant le plus score(R,D)
5- R ← M, nous appliquons cette modification à R
6-Retourner R

Figure 4 : Pseudo-code pour l'apprentissage de la structure optimale d'un réseau bayesien.

Cet algorithme se nomme K2 et a été décrit par le group de Nie Friedman [25]. Malheureusement, dans les cas qui nous intéressent, ce ne peut être aussi simple. Le problème est que le nombre de structures croît de manière super-exponentielle [26][39]. Pour donner une idée au lecteur, un réseau de 8 variables possède un nombre à douze chiffes de réseaux acycliques possibles. De plus, comme nous sommes intéressés par des problèmes qui possèdent plus de variables que de données, il arrive qu'il y ait plusieurs structures optimales. Nous allons donc présenter deux méthodes de recherche basées sur celle vue précédemment, mais permettant de gérer le problème de dimensionnalité.

Méthode de rééchantilonnage (Bootstraping)

Cette méthode à été décrite par Nir Friedman [38]. Le principe de base est assez simple. Supposons que nous sommes intéressés par des critères dont nous pouvons évaluer la présence dans un réseau bayesiens. Par exemple, choisissons comme critère la présence où l'absence d'un lien allant de A à B. Si ce critère est vraiment pertinent (si B dépend réellement de A), à chaque fois que nous appliquerons l'algorithme K2 (

figure 4) avec un réseau de départ aléatoire, nous devrions, en pratique, toujours retrouver le lien allant de A à B dans le réseau optimal. La même chose devrait se produire si nous choisissons aléatoirement un sous-ensemble de nos données pour apprendre le réseau optimal. En effet, si le lien de dépendance est fort, peu importe les données que nous utilisons, nous devrions quand même le retrouver. Cette méthode nous permet donc d'associer un niveau de confiance au critère que nous évaluons. Le critère de confiance est décrit dans l'équation 6.

$$\text{Niveau de confiance du lien allant d'A à B} = \frac{1}{N_{\hat{r}}} \sum_{i=1}^{N_{\hat{r}}} p(A,B,\hat{R}_i)$$

Équation 6 : Calcul du niveau de confiance associé à un lien. $N_{\hat{r}}$ est le nombre de réseaux optimaux que nous utilisons. La fonction $p(A,B,\hat{R}_i)$ vaut 1 si le lien allant de A à B est présent dans le réseau $\hat{R}_i$ et 0 sinon. Cette équation nous donne donc la proportion des réseaux optimaux contenant l'arc allant de A à B.

Suite à cela, nous n'avons qu'à fixer un seuil sur la confiance et à retrouver le réseau contenant les arcs ayant un niveau de confiance au dessus du seuil.

Méthode MCMC (Monte Carlo Markov Chain)

Cette méthode est basée sur une chaîne de Markov sur la structure des réseaux bayesiens. Chaque nœud de la chaîne est relié à un autre nœud avec une probabilité plus grande que 0 si et seulement si il existe une modification de base (addition, soustraction ou retournement d'un arc) qui permet de passer d'une structure à l'autre. Chaque nœud représente donc la structure d'un réseau bayesien acyclique. Voici un gros plan sur la chaîne nous permettant de mieux comprendre :

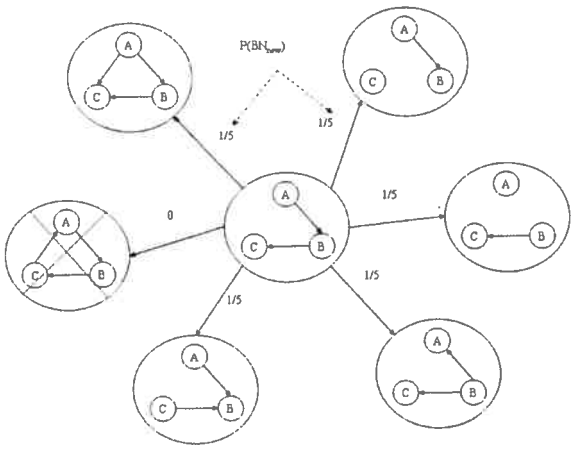


Figure 5 : Chaîne de Markov. Gros plan sur une partie de la chaîne de Markov d'un réseau bayesien de 3 nœuds. Nous pouvons voir qu'il est possible d'atteindre les réseaux en périphérie en appliquant une des opérations de base (addition, soustraction, retournement d'un arc), sur le réseau du milieu. À noter que le réseau contenant le cycle obtient une probabilité de 0 et que les autres obtiennent une probabilité égale d'être visités.

Nous utilisons la chaîne de Markov comme une raille nous permettant de nous diriger vers un sous-espace intéressant (un sous-espace contenant des réseaux ayant des structures à probabilité élevée étant donné nos données). Certaines conditions décrites par Husmeier [40] nous permettent d'être assuré que notre chaîne convergera vers ce sous-espace intéressant. Voici le pseudo-code dérivé d'Husmeier [40] nous permettant de nous diriger vers le sous-espace intéressant :

Procédure de parcours de ma chaîne de Markov:

1-R_{old} ← Choisir un réseau de départ aléatoirement
2-Tant que nous somme dans la phase de réchauffement
3- R_{new} ← Choisir un nouveau réseau aléatoire dans l'entourage du réseau R_{old} dans la chaîne de Markov
4- R_{new} deviendra R_{old} avec une probabilité de P_{MH}
5-Tant que nous somme dans la phase d'échantillonnage
6- R_{new} ← Choisir un nouveau réseau aléatoire dans l'entourage du réseau R_{old} dans la chaîne de Markov
7- R_{new} deviendra R_{old} avec une probabilité de P_{MH}
8- Emmagasiner R_{old}
9-Retourner tous les réseaux emmagasinés

Figure 6 : Pseudo-code pour le parcours de la chaîne de Markov des réseaux bayesiens. À noter que le seul élément différenciant la boucle 2, 3, 4 de la boucle 5, 6, 7, 8 est le point 8 où nous emmagasinons le réseau.

Dans le pseudo-code de la

figure 6, nous pouvons apercevoir la variable P_{MH} qui est le critère de Metropolis-Hasting [41]. Ce critère peut être calculé à l'aide de l'équation 7.

$$P_{MH} = \min\left\{1, \frac{P(R_{new} \mid D)}{P(R_{old} \mid D)} * \frac{N(R_{old})}{N(R._{new})}\right\}$$

Équation 7 : Critère de Metropolis-Hasting. Dans cette équation, P(R|D) est la probabilité du réseau étant donné les données et N(R) correspond au nombre de réseau bayesien acyclique accessible à partir de R.

Pour une valeur de nombre de réseaux bayesiens accessibles égale $N(R_{new})/N(R_{old}) = 1$ nous pouvons observer que la probabilité sera plus élevée d'aller vers R_{new} si $P(R_{new}|D) > P(Rold|D)$ et l'inverse dans le cas où $P(R_{new}|D) < P(R_{old}|D)$. Maintenant si nous fixons les

probabilités $P(R_{new}|D) / P(R_{old}|D) = 1$. Nous observons qu'il est plus probable d'aller vers R_{new} s'il y a moins de réseaux accessibles à partir de cette structure.

Une fois que nous obtenons notre liste de réseaux à l'aide de la phase d'échantillonnage, nous pouvons utiliser l'équation 6 pour déterminer les liens qui nous intéressent.

Choix de la méthode de recherche utilisée

Dans les cas où nous possédons un nombre élevé de données, la méthode de rééchantillonnage et la méthode MCMC obtiennent des résultats comparables. Cependant, comme nous ne possédons pas en quantité suffisante des données de biopuces, il ne serait pas logique d'utiliser la méthode de rééchantillonnage et ainsi mettre de côté certaines données que nous avons. C'est pour cette raison que cette recherche sera effectuée en utilisant la méthode MCMC pour rechercher les structures le plus probables étant donné les données.

Famille de distribution utilisée

Dans cette section, nous allons expliquer plus en détails comment il est possible de calculer l'équation 5.

Pour modéliser l'interaction entre un nœud et ses parents, nous pourrions utiliser un grand nombre de familles de distribution. Cependant, les distributions doivent être régulières au sens décrit dans le papier du groupe de David Heckerman [26]. Deux familles de distribution respectent ces règles de régularités : la famille de distribution multinomiale et la famille de distribution Gaussienne linéaire. Chacun de ces modèles a ses forces et ses faiblesses. La distribution multinomiale a comme désavantage d'utiliser des valeurs discrètes. Comme les valeurs obtenues des biopuces sont réelles, nous devons donc discrétiser les données et ainsi perdre de l'information. Cependant, le modèle d'interaction multinomiale permet de modéliser des relations non linéaires entre un nœud et ses parents. La distribution linéaire Gaussienne utilise directement les données réelles, donc ne souffre pas d'une perte d'information. Cependant, elle ne peut que modéliser des interactions linéaires. Dans le cadre de cette recherche, nous allons utiliser les distributions multinomiales, car nous ne voulons pas restreindre les interactions à un niveau uniquement linéaire. L'équation 8 montre

comment il est possible de calculer la probabilité *a posteriori* d'un réseau étant donné les données.

$$P(R \mid D) = \frac{P(D \mid R)P(R)}{P(D)} \rightarrow \text{Théorème de Bayes}$$

$$\log(P(R \mid D)) = \log(P(D \mid R)) + \log(P(R)) - \log(P(D)) \rightarrow \text{Propriété du log}$$

$$\log(P(D \mid R)) = \sum_{i=1}^{n} P(v_i \mid Pa_{v_i})$$

Équation 8 : Probabilité à posteriori. Développement plus détaillé de la façon de calculer la probabilité à posteriori d'un réseau étant donné les données. Comme nous ne pouvons calculer directement la première probabilité, nous devons utiliser le théorème de Bayes. Par la suite, nous appliquons le log pour diminuer l'ampleur des données. À noter, que la valeur P(D) sera constante pour toutes les probabilités calculées. Elle peut être omise. P(R) est la probabilité *a priori* que nous associons au réseau. Nous n'assumons aucune connaissance *a priori* donc P(R) sera constant pour toutes les structures générées. La valeur de $P(v_i \mid Pa_{v_i})$ dépend du modèle d'interaction utilisé *i.e.* multinomiale ou Gaussienne linéaire.

La probabilité d'un nœud étant donnée la valeur de ses voisins, dans le cas où nous travaillons avec des données discrètes, peut être calculée facilement en utilisant la distribution multinomiale exposée dans l'équation 9.

$$P(V_i \mid Pa_{V_i}) = \prod_{a=1}^{C(Pa_{V_i})} \frac{\Gamma(N_a)}{\Gamma(N_a + M_a)} \prod_{b=1}^{D(V_i)} \frac{\Gamma(p_{ab} + u_{ab})}{\Gamma(p_{ab})}$$

Équation 9 : Distribution multinomiale. Cette équation permet de calculer la probabilité d'observer les valeurs d'un nœud Vi étant donnée les valeurs des nœuds parents. C() est une fonction qui retourne le nombre de Configurations possibles que peuvent prendre les parents de Vi. Par exemple, si Vi à 3 parents pouvant prendre 2 valeurs différentes, $C(Pa_{V_i})$ = 23. a itérera donc sur chacune des configurations des parents. D() est la fonction qui retourne le nombre de valeurs Discrètes que peut prendre le nœud Vi. Si Vi peut prendre 2 valeurs alors D(Vi) = 2. b itérera donc sur les valeurs que peut prendre le nœud Vi. Ma correspond au nombre de fois que nous observons les nœuds parents en configuration a dans les données. pab est un nombre réel > 0 qui correspond au connaissance *a priori* que nous avons sur le fait d'observer les parents dans l'état a et le nœud dans l'état b. Comme nous n'avons aucune connaissance *a priori*, cette valeur égalera toujours 1. uab est le nombre de fois que nous observons les parents ayant une valeur a et le nœud ayant une valeur b dans les données. Finalement, Na= $\sum_{b=1}^{D(V_i)} p_{ab}$. $\Gamma(x)$ représente la fonction gamma[42]. Pour plus de détails nous référons le lecteur au livre de Neapolitan [31].

Méthode de discrétisation

Comme nous avons choisi d'utiliser les distributions multinomiales, ceci implique que nous devions discrétiser nos données. Nous avons donc utilisé deux méthodes différentes pour discrétiser nos données. La première étant la méthode par intervalle et la deuxième étant la méthode par quantile [76].

La méthode par intervalle divise l'espace des valeurs d'expression en n blocs de dimension égale. La valeur discrète associée à une valeur réelle est déterminée par le numéro du bloc dans lequel il se trouve.

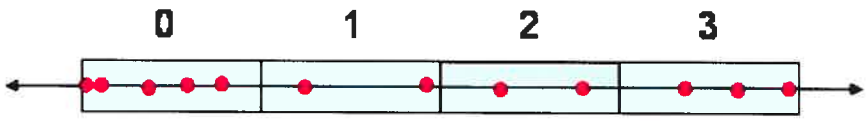


Figure 7 : Méthode de discrétisation par intervalle. Les chiffres au-dessus des blocs de même dimension représentent la valeur discrète qu'auront les valeurs réelles contenues dans le bloc. Cet exemple est pour un n de valeur 4 (pour 4 blocs).

Pour déterminer la dimension d'un bloc, il suffit de faire $(\max(x_i)-\min(x_i))/n$.

La méthode par quantile permet de discrétiser les données en sous groupe de même dimension. Pour y arriver, il suffit de déterminer le nombre de données que nous aurons par sous-groupe en faisant $t=|x_i|/n$. Ensuite, nous trions nos données et nous associons les t premières valeurs réelles à la valeur discrète 0, les t suivantes à la valeur discrète 1 et ainsi de suite. Voici les résultats obtenus sur le même ensemble de données que le cas par intervalle.

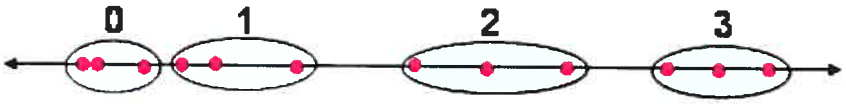


Figure 8 : Méthode de discrétisation quantile. Les chiffres au-dessus des sous-ensembles représentent la valeur discrète qu'auront les valeurs réelles contenues dans le sous-ensemble. Cet exemple est pour un n de valeur 4 pour 4 blocs.

Il est important de constater que les deux méthodes de discrétisation ne produisent pas exactement les mêmes valeurs discrètes.

Réseaux bayesiens dynamiques

Comme il a été mentionné précédemment, les réseaux bayesiens doivent être acyclique pour avoir des distributions de probabilité cohérentes. Cependant, les réseaux de régulation contiennent très souvent des cycles. À ce point de vue, l'approche bayesienne peut sembler limitative. Pour résoudre ce problème, il est possible de dérouler le réseau dans le temps de manière à associer une série de nœuds pour le temps T et une autre série de nœuds pour le temps T+1(voir figure 9).

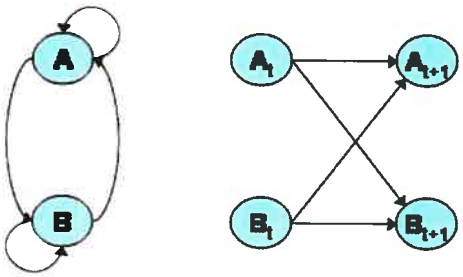


Figure 9 : Élimination des cycles. Passage d'un réseau contenant des cycles à un réseau bayesien sans cycle en déroulant le réseau sur deux tranches dans le temps T et T+1.

Ensuite, nous fixons comme contrainte que les liens peuvent seulement partir du temps T pour aller au temps T+1 et non pas l'inverse. De cette façon nous doublons le nombre de nœuds, mais nous permettons la modélisation de réseau contenant des cycles. Concrètement, un réseau bayesien dynamique est ni plus ni moins qu'un réseau bayesien avec des contraintes sur les arcs qui peuvent être utilisées. Toutes les méthodes décrites auparavant s'appliquent donc directement à ce type de réseau. Cependant, ce type de réseau peut seulement être déterminé à l'aide de données séquentielles. À noter qu'il n'existe pas autant de données séquentielles que de données non séquentielles, mais ces données sont quand même plus intéressantes du fait qu'elles donnent une information de plus, car elles nous présentent la variation de l'expression des gènes dans le temps.

CHAPITRE 2

Accélération de la recherche dans l'espace des structures de réseaux bayesiens

Résumé

Plusieurs méthodes existent pour chercher la structure d'un réseau bayesien optimale représentant les dépendances entre variables. Ces méthodes parcourent l'espace en utilisant les trois modifications de base (l'addition, la soustraction et le retournement d'un lien). Comme mentionné précédemment, pour obtenir des distributions de probabilités cohérentes, nous devons avoir une structure n'ayant aucun cycle dirigé. Pour cette raison, après l'application d'une modification de base, nous devons vérifier la présence de cycles. Évidemment, il n'est pas question d'utiliser à chaque fois un algorithme qui vérifie en entier tout le réseau. Giudici et Castelo ont décrit dans leur article une méthode efficace pour déterminer à l'avance si une modification créera un cycle ou non [43]. Cependant, leur article ne traite pas des détails d'implantation cruciaux comme de l'utilisation de vecteurs de bits ou l'ajout de critères préalables pour le retournement et l'utilisation du tri topologique en ligne [45]. De plus, nous avons développé une méthode efficace pour calculer le nombre de réseaux acycliques accessibles à partir d'un réseau courant. Ce nombre, utilisé dans le critère de Metropolis-Hasting, est parfois éliminé dans le cas de réseaux de grandes tailles, car il est trop coûteux à calculer. À notre connaissance, aucune publication à ce jour ne traite de ces détails d'implantation et c'est pour cette raison que nous nous sommes appliqués à corriger cette lacune.

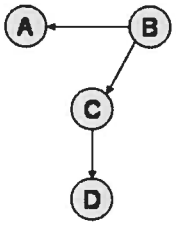
Introduction

La plupart des méthodes utilisées pour déterminer la ou les structures représentant le plus adéquatement les dépendances entre des variables à partir de données utilisent la même méthodologie de recherche. En effet, les algorithmes utilisent l'addition, la soustraction et le retournement d'un lien pour passer d'une structure à une autre [32][40][38]. Comme la structure cherchée par ces méthodes doit être acyclique, il est impératif d'avoir une façon efficace de vérifier si l'application d'une modification de base crée un cycle. À ce sujet, la méthode présentée par Giudici et Castelo permet d'accélérer grandement la vérification [43].

L'idée de base est de maintenir à jour une matrice d'ancêtres et d'utiliser des opérations simples pour vérifier s'il est possible d'appliquer la modification.

Matrice d'ancêtres

Une matrice d'ancêtres est une matrice booléenne non symétrique qui a pour valeur 1 en position [A,B] si le nœud A est l'ancêtre du nœud B et 0 sinon. Nous disons que B est ancêtre de A s'il existe un moyen de passer de B à A en suivant un chemin dirigé. Un exemple de matrice d'ancêtres pour un graphe simple est montré à la figure 10.



	A	B	C	D
A	0	0	0	0
B	1	0	1	1
C	0	0	0	1
D	0	0	0	0

Figure 10 : La structure d'un réseau et sa matrice d'ancêtres associée. La valeur 1 à la case [B,D] signifie qu'il y a un chemin menant de B à D. Ce chemin passe de B à C et ensuite de C à D. La valeur 0 à la case [A,C] signifie qu'il n'y a aucun chemin allant de A à C.

Tri topologique

Le tri topologique est une façon d'ordonnancer les nœuds de manière à ce que tous les liens du réseau partent d'un nœud ayant un ordre plus bas vers un nœud ayant un ordre plus haut. Pour l'exemple de la figure 10, il existe plusieurs tris topologiques valident et B, A, C, D en est un.

Méthode précédente

Maintenant que nous avons introduit la matrice d'ancêtres et le tri topologique, nous allons décrire les règles définies par Giudici et Castelo nous permettant d'utiliser cette matrice pour accélérer la détection des cycles [43]. La

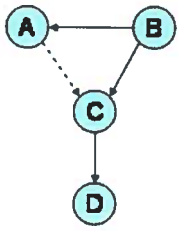
figure 11 présente la procédure décrite par Giudici et Castelo pour l'ajout d'un lien.

Procédure d'addition d'un lien allant de A à B :

1-Si B n'est pas ancêtre de A (*matrice_ancêtres*[B,A]==0)
2- Nous pouvons ajouter le lien sans créer de cycle
3- Nous devons mettre à jour *matrice_ancêtres*
4- *matrice_ancêtres*[A,B]=1
5- Tous les ancêtres de A deviennent les ancêtres de B

Figure 11 : Pseudo-code à exécuter pour l'addition d'un lien allant de A à B.

Exemple d'addition d'un lien qui ne crée pas de cycle :



Avant

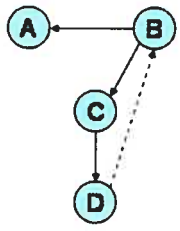
	A	B	C	D
A	0	0	0	0
B	1	0	1	1
C	0	0	0	1
D	0	0	0	0

Après

	A	B	C	D
A	0	0	1	1
B	1	0	1	1
C	0	0	0	1
D	0	0	0	0

Figure 12 : Exemple d'addition du lien allant de A à C. La case rouge dans la matrice d'ancêtres désigne la case que nous devons vérifier pour s'assurer que l'ajout du lien ne crée pas de cycle. Les cases vertes sont les cases qui seront potentiellement modifiées par la propagation des ancêtres de A à C.

Exemple d'addition d'un lien qui crée un cycle :



Avant

	A	B	C	D
A	0	0	0	0
B	1	0	1	1
C	0	0	0	1
D	0	0	0	0

Après

	A	B	C	D
A	0	0	0	0
B	1	0	1	1
C	0	0	0	1
D	0	0	0	0

Figure 13 : Exemple d'addition d'un lien allant de D à B. Nous pouvons voir dans la matrice d'ancêtres que B est déjà ancêtre de D alors nous pouvons immédiatement savoir que l'ajout du lien crée un cycle. C'est pour cette raison que nous n'avons pas à mettre à jour la matrice.

Si nous regardons la complexité de l'opération d'ajout d'un lien, le test pour vérifier si l'ajout du lien crée un cycle se fait en temps constant, soit dans $O(1)$, car l'accès à la valeur dans la matrice est directe. La mise à jour de la matrice dépend du nombre d'ancêtres que possède le nœud d'origine et prend au plus un temps dans $O(n)$.

Procédure d'élimination d'un lien allant de A à B

La soustraction d'un lien ne crée jamais de cycle. Cependant, la matrice d'ancêtres doit recevoir une sérieuse mise à jour. En effet, enlever un lien de A à B nécessite qu'une partie des ancêtres de A doivent être enlevés des ancêtres de B et des ancêtres des descendants de B (tous les nœuds dont B est un ancêtre).

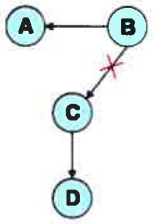
1-Réinitialiser les ancêtres de B aux parents directs de B

2-Ajouter tous les ancêtres des parents de B comme ancêtres de B

3-Parcourir en ordre topologique les descendants de B et refaire les étapes 1 à 3 pour chacun des descendants

Figure 14 : Pseudo-code à exécuter pour l'élimination d'un lien allant de A à B.

Exemple d'élimination d'un lien :



Avant

	A	B	C	D
A	0	0	0	0
B	1	0	1	1
C	0	0	0	1
D	0	0	0	0

Après

	A	B	C	D
A	0	0	0	0
B	1	0	0	0
C	0	0	0	1
D	0	0	0	0

Figure 15 : Exemple de soustraction du lien allant de B à C. Il est possible de voir en vert les cases de la matrice d'ancêtres qui devront être mises à jour. Pour la mise à jour, les ancêtres des nœuds C et D seront mis à jour selon l'algorithme présenté à la figure 14.

La complexité pour la soustraction d'un lien est dans $O(n^2)$. En effet, dans le pire des cas, nous aurons à mettre à jour la matrice d'ancêtres pour tous les nœuds et, toujours au pire, ces nœuds auront comme parents directs les n-1 autres nœuds.

Procédure de retournement d'un lien allant de A à B

1-Le retournement est en fait une suite de deux opérations, la soustraction du lien de A à B et l'addition du lien de B à A.

Figure 16 : Pseudo-code à exécuter pour le retournement d'un lien allant de A à B.

La complexité du retournement dépend grandement de la soustraction qui prend un temps dans $O(n^2)$, donc la complexité du retournement est dans $O(n^2)$.

Ceci complète la description détaillée de la méthode présentée par Giudici et Castelo pour une implantation efficace de la méthode des chaînes de Markov Monte Carlo sur les

structures de réseaux Bayesiens [43]. La section qui suit présente les améliorations que nous avons apportées à la méthode et une section subséquente présente des résultats montrant l'amélioration des performances.

Améliorations proposées

Cette section a pour but de présenter des modifications à apporter à la méthode de Giudici et Castelo pour permettre un parcours plus rapide de la chaînes de Markov [43]. La rapidité est un facteur déterminant de cette méthode puisque nous devrons générer de nombreux réseaux avant d'atteindre la zone d'échantillonnage. De plus, la méthode de Giudici et Castelo ne traite aucunement du calcul du nombre de réseaux acycliques accessibles à partir d'un réseau courant [43]. Nous proposerons donc une solution à ce problème.

Utilisation des vecteurs de bits

L'article de Giudici et Castelo n'avait pas pour but de présenter une implantation spécifique dans un langage de programmation mais, selon nous, il est primordial que la matrice d'ancêtres soit implantée à l'aide de vecteurs de bits. Ceci accélère grandement la propagation des ancêtres d'un nœud à un autre, car il ne suffit que d'un seul OU logique, (|), entre deux vecteurs pour faire la propagation. Par exemple et tel qu'illustré dans la figure 12, quand vient le temps de propager les ancêtres de C à D, il suffit d'effectuer l'opération 1100 | 0110 = 1110. Nous suggérons aussi d'ajouter une nouvelle matrice, soit la matrice des parents\enfants directs à un nœud. Dans cette matrice, l'entrée [A,B] aura la valeur 1 si A est parent de B (B est enfant de A) et 0 sinon. Cette matrice porte habituellement le nom de matrice d'incidence. Les raisons pour ajouter cette matrice deviendront plus claires plus loin dans cette section. L'utilisation de vecteurs de bits pour représenter les matrices en plus de permettre des opérations binaires efficaces diminue l'espace mémoire occupée par les matrices. Dans le cadre de cette recherche, nous avons implanté des vecteurs de bits utilisant des blocs de 64 bits. Notez que les nouveaux processeurs (AMD64) permettent des opérations binaires directement sur 64 bits. Ici, nous profitons des fonctionnalités de cette architecture. Ce qui veut dire que toutes les opérations binaires sur des vecteurs de 64 bits ou moins (donc de réseaux de 64 nœuds ou moins) se font en temps constant, $O(1)$, et les opérations sur des

vecteurs de plus de 64 bits se font en un temps de $(\lfloor (n-1)/64 \rfloor + 1)$ qui est dans l'ordre d'*O(n)*.

Utilisation d'un tri topologique en ligne

La méthode décrite par Giudici et Castelo mentionne qu'à la suite de la soustraction d'un lien, il faut parcourir en ordre topologique les descendants du nœud cible du lien supprimé [43]. Le logiciel Bayes Net Toolbox (BNT) [44], programmé en Matlab et implantant la méthode de Giudici et Castelo fait un tri topologique complet à chaque soustraction d'un lien. Le coût d'un tri topologique complet sur une seule machine est dans $O(n^2)$ [46], où n représente le nombre de nœuds du réseau. Ceci peut s'avérer très coûteux quand la soustraction d'un lien survient fréquemment. Nous proposons donc l'utilisation d'un tri topologique en ligne qui ajuste la topologie à chaque addition d'un lien. À noter que seul l'addition ou le retournement d'un lien peut amener un désordre topologique. La méthode décrite par Pearce et Kelly a été utilisée à cette fin [45]. Globalement, cette méthode limite l'espace du tri topologique aux nœuds impliqués par l'addition et à leurs descendants.

Utilisation d'un critère pour vérifier le retournement

La méthode décrite pour le retournement est simple mais nécessite à chaque fois une soustraction d'un lien pour vérifier si l'addition sera valide ou non. Dans certains cas, il arrive que la soustraction, l'opération la plus coûteuse, soit effectuée et que l'addition crée un cycle, rendant la soustraction inutile. Nous proposons donc une vérification pré retournement pour éviter ces soustractions inutiles.

L'idée est simple, supposons qu'il existe un lien allant de A à B et que nous voulions le retourner, le seul cas où cette opération crée un cycle est si les parents de B ont comme ancêtre A (voir la figure 11).

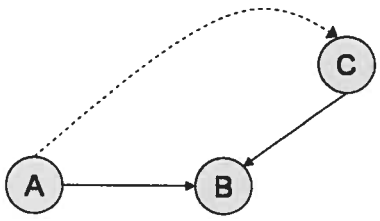


Figure 17 : Exemple où le retournement du lien allant de A à B crée un cycle. Le lien pointillé représente la notion d'ancêtre.

Voici maintenant le pseudo-code de la vérification à effectuer pour s'assurer de la validité du retournement d'un lien allant de A à B :

Procédure de validation du retournement

```
1-Reversible=vrai;
2-Itérer sur les parents de B
3-      Si le nœud courant a pour ancêtre A
4-              Reversible=faux
```

Figure 18 : Pseudo-code à exécuter pour valider le retournement d'un lien allant de A à B.

Cet algorithme prendra au plus un temps dans $O(n)$ dans le cas où le nœud B aurait tous les autres nœuds comme parents.

Calcul efficace du nombre de réseaux acycliques accessibles

Pour calculer la probabilité que nous avons d'accepter ou non la configuration en cours lors de la méthode Monte Carlo, nous devons calculer le critère de Metropolis-Hasting. Pour évaluer ce critère, nous devons compter le nombre de réseaux acycliques accessibles à partir du réseau courant. Cette problématique n'a pas été traitée dans l'article de Giudici et Castelo [43]. Une méthode utilisée pour résoudre ce problème consiste à essayer toutes les configurations accessibles à partir du réseau courant en appliquant une opération de base (addition, soustraction ou retournement d'un lien) et de compter les réseaux atteints qui sont acycliques. Cette méthode est très fastidieuse car elle effectue beaucoup de manipulations inutiles de la matrice d'ancêtres. Les lignes qui suivent décrient une méthode efficace permettant de calculer ce nombre.

Le calcul s'effectue en trois étapes. Premièrement, nous devons calculer le nombre d'additions valides (ne créant pas de cycle) que nous pouvons effectuer. Pour ce faire, nous

allons compter le nombre de liens valides (ne créant pas de cycle) pouvant partir de chacun des nœuds. Le nombre de liens pouvant partir d'un nœud est égal au nombre de zéros dans le vecteurs de bits résultant du OU logique entre le vecteur de bits des ancêtres du nœud et le vecteur de bits des enfants du nœud moins 1. Ceci revient à calculer le nombre d'ancêtres que le nœud n'a pas moins les liens qu'il a déjà moins le lien sur lui-même. Pour accélérer cette opération, nous avons trouvé une façon efficace de compter le nombre de bits à 0 dans un vecteur de bits. L'idée est de précalculer un tableau contenant les 2^{16} possibilités pour un vecteur de 16 bits. Par exemple, dans notre tableau précalculé, la valeur à l'index [1000111011100110] sera de 7, car il y a 7 zéros. Une fois le tableau précalculé, nous avons seulement à fragmenter le vecteur de bits en parties de 16 bits pour calculer rapidement le nombre total de 0. Comparativement à une méthode itérative conventionnelle utilisant le décalage et le test sur un bit pour chacun des 64 bits, cette méthode prend 64/4 = 16 fois moins de temps. Pour un nœud donné, cette opération prendra le temps d'un OU logique (|) entre les ancêtres du nœud courant et ses enfants, soit n/64 plus un compte rapide du nombre de 0 (n/16) et une opération de soustraction pour le lien sur le nœud lui-même. La somme de ces termes se fait en temps (n/64 + n/16 + 1) qui est dans $O(n)$.

Par la suite, nous calculons le nombre de liens que nous pouvons enlever. Cette opération est très rapide si nous connaissons le nombre de lien du réseau courant, car toutes les soustractions sont valides. Il suffit donc d'ajouter le nombre de liens au nombre total de configurations accessibles.

Pour terminer, nous devons calculer le nombre de retournements valides que nous pouvons effectuer. Pour y arriver, nous allons utiliser la méthode développer plus haut pour déterminer efficacement si un lien allant de A à B peut être retourné. Tel que mentionné plus haut, le lien pourra être retourné seulement si le nœud A ne fait pas partie des ancêtres des parents de B. Une solution simple serait donc d'itérer sur tous les liens pointant sur un nœud et d'utiliser l'astuce développée plus haut à chacun des liens. Dans le pire des cas, nous devrons faire dans $O(n^2)$ opérations. Le premier n-1 vient de la complexité de la méthode pour vérifier le retournement et le deuxième du fait que le nombre de liens pointant sur un nœud ne peut être plus grand que n-1. Cependant, il est possible de faire mieux ! Il suffit de créer un vecteur indiquant tous les ancêtres des parents du nœud courant. Une fois que nous avons ce vecteur, les 1 indiquent les nœuds pour lesquels si nous avons un lien partant de ce nœud en direction du nœud courant, il sera impossible de le retourner sans créer un cycle.

Nous devons donc déterminer quels parents du nœud courant ont une valeur 0 dans le vecteur indiquant tous les ancêtres des parents du nœud. Comme nous disposons d'une matrice d'incidence, il est simple d'obtenir un vecteur de parents pour un nœud donné. Ensuite, nous voulons une opération qui nous permette de connaître le nombre de liens pointant sur le nœud courant pouvant être retournés. Pour y arriver, nous proposons une suite d'opérations. La première consiste à faire la négation du vecteur des ancêtres parentaux. Cette opération à pour but de transformer les 0 (nœuds dont nous pouvons retourner les liens) en 1. Ensuite nous faisons le ET logique de ce vecteur avec le vecteur des parents. En appliquant cette dernière opération, nous obtenons un nouveau vecteur ayant des 1 aux positions des liens que nous pouvons retourner. Une fois ce dernier vecteur obtenu, il suffit de compter le nombre de 1 pour obtenir la valeur désirée. Dans le pire des cas, nous aurons (n-2) (un de moins que les n-1 nœuds parentaux maximum) OU logique à faire pour créer le vecteur d'ancêtres parentaux, une négation et un compte rapide du nombre de 1. Nous avons donc n-1 opérations prenant un temps n/64 et une opération prenant un temps n/16. Le temps maximum pour un nœud est donc de (n/64*(n-1) + n/16) ce qui est dans $O(n^2)$.

Les lignes qui suivent décrivent de manière détaillée l'algorithme présenté dans les paragraphes précédents. À noter, *MA*[x] retourne un vecteur de bits des ancêtres du nœud x. Par exemple, si nous utilisons le réseau de la figure 15 , chacun des nœuds (A,B,C,D) aura un bit lui étant attribué dans le vecteur de bit. Sans perte de généralité, le nœud A sera représenté par le premier bit, le nœud B par le deuxième et ainsi de suite. *MA*[x] retourne donc la colonne de la matrice associée au nœud x. *E*[x] retourne aussi un vecteur de bits mais celui-ci contient un masque des enfants du nœud x. Finalement, *P*[x] retourne un masque des parents de x.

Procédure pour calculer le nombre de réseaux acycliques accessibles

1-*Nombre_de_réseaux*=0
2-Itérer sur tous les nœuds *n*
3- *Nombre_de_réseaux*+=compte_rapide_0(*MA*[*n*] | *E*[*n*])-1
4- *vecteur_ancêtres_parentaux*=000..000 //Vecteur de dimension n vide
5- Itérer sur les parents (*p*) du nœud courant *n*
6- *vecteur_ancêtres_parentaux* |= *MA*[*p*]
7- *Nombre_de_réseaux*+=compte_rapide_1(!*vecteur_ancêtres_parentaux* & P[n])
8-*Nombre_de_réseaux* +=*nombre_de_liens_courants*

Figure 19 : Pseudo-code à exécuter pour calculer le nombre de réseaux acycliques accessibles en appliquant toutes les opérations de bases sur le réseau courant.

Ici la fonction compte_rapide_0 utilise à répétition le vecteur de dimension 2^{16} contenant le nombre de 0 précalculés. À noter que la fonction compte_rapide_1 retourne (n-compte_rapide_0). Voici donc ce que donne l'application de cet algorithme au réseau de la figure 15 avant la soustraction :

Matrice d'ancêtres			
A	B	C	D
0	0	0	0
1	0	1	1
0	0	0	1
0	0	0	0

Matrice d'incidence			
A	B	C	D
0	1	0	0
0	0	0	0
0	1	0	0
0	0	1	0

Figure 20 : Matrice d'ancêtres (*MA[n]*) et d'incidence du réseau de la figure 15 avant la soustraction. Les colonnes de la matrice d'incidence représentent les enfants (*E[n]*) et les lignes les parents (*P[n]*). Ces matrices sont utilisées dans l'exemple de la procédure pour calculer le nombre de réseaux acycliques accessibles.

Nombre_de_réseaux=0

Nœud A :

Étape 3 :

Nombre_de_réseaux += compte_rapide_0(*MA*[*A*] | *E*[*A*])-1
Nombre_de_réseaux += compte_rapide_0(0100 | 0000) -1
Nombre_de_réseaux += compte_rapide_0(0100) -1
Nombre_de_réseaux += 3 -1

Commentaire : Il est possible d'ajouter les liens A→ C et A→ D

Étape 6 :

vecteur_ancêtres_parentaux = 0000 | *MA*[*B*] //Un seul parent (B)
vecteur_ancêtres_parentaux = 0000 | 0000
vecteur_ancêtres_parentaux = 0000

Étape 7 :

Nombre_de_réseaux += compte_rapide_1(!*vecteur_ancêtres_parentaux* & P[A])
Nombre_de_réseaux += compte_rapide_1(!*0000* & 0100)
Nombre_de_réseaux += compte_rapide_1(0100)
Nombre_de_réseaux += 1

Commentaire : Il est possible de retourner le lien B→ A

À la fin de l'itération pour le nœud A, *Nombre_de_réseaux* vaut 0 + 2 + 1 = 3

Nœud B :

Étape 3 :

Nombre_de_réseaux += compte_rapide_0(*MA*[*B*] | *E*[*B*])-1
Nombre_de_réseaux += compte_rapide_0(0000 | 1010) -1
Nombre_de_réseaux += compte_rapide_0(1010) -1
Nombre_de_réseaux += 2 -1

Commentaire : Il est possible d'ajouter le lien B→ D

Étape 6 :

vecteur_ancêtres_parentaux = 0000 //Aucun parent

Étape 7 :

Nombre_de_réseaux += compte_rapide_1(!*vecteur_ancêtres_parentaux* & P[B])
Nombre_de_réseaux += compte_rapide_1(!*0000* & 0000)
Nombre_de_réseaux += compte_rapide_1(0000)
Nombre_de_réseaux += 0

Commentaire : Il n'y a aucun lien entrant à retourner

À la fin de l'itération pour le nœud B, *Nombre_de_réseaux* vaut 3 + 1 + 0 = 4

Nœud C :

Étape 3 :

Nombre_de_réseaux += compte_rapide_0(*MA*[*C*] | *E*[*C*])-1
Nombre_de_réseaux += compte_rapide_0(0100 | 0001) -1
Nombre_de_réseaux += compte_rapide_0(0101) -1
Nombre_de_réseaux += 2 -1

Commentaire : Il est possible d'ajouter le lien C→ A

Étape 6 :

vecteur_ancêtres_parentaux = 0000 | *MA*[*B*] //Un seul parent (B)
vecteur_ancêtres_parentaux = 0000 | 0000
vecteur_ancêtres_parentaux = 0000

Étape 7 :

Nombre_de_réseaux += compte_rapide_1(!*vecteur_ancêtres_parentaux* & P[C])
Nombre_de_réseaux += compte_rapide_1(!*0000* & 0100)
Nombre_de_réseaux += compte_rapide_1(0100)
Nombre_de_réseaux += 1

Commentaire : Il est possible de retourner le lien B→ C

À la fin de l'itération pour le nœud C, *Nombre_de_réseaux* vaut 4 + 1 + 1 = 6

Nœud D :

Étape 3 :

Nombre_de_réseaux += compte_rapide_0(*MA*[*D*] | *E*[*D*])-1
Nombre_de_réseaux += compte_rapide_0(0110 | 0000) -1
Nombre_de_réseaux += compte_rapide_0(0110) -1
Nombre_de_réseaux += 2 -1

Commentaire : Il est possible d'ajouter le lien D→ A

Étape 6 :

vecteur_ancêtres_parentaux = 0000 | *MA*[*C*] //Un seul parent (C)
vecteur_ancêtres_parentaux = 0000 | 0100
vecteur_ancêtres_parentaux = 0100

Étape 7 :

Nombre_de_réseaux += compte_rapide_1(!*vecteur_ancêtres_parentaux* & P[D])
Nombre_de_réseaux += compte_rapide_1(!*0100* & 0010)
Nombre_de_réseaux += compte_rapide_1(0010)
Nombre_de_réseaux += 1

Commentaire : Il est possible de retourner le lien C→ D

À la fin de l'itération pour le nœud C, *Nombre_de_réseaux* vaut 6 + 1 + 1 = 8

Étape 8 :

nombre_de_liens_courants = 3
Nombre_de_réseaux += 3

À la fin de la procédure, *Nombre_de_réseaux* vaut 8 + 3 = 11

Cet algorithme prendra dans le pire des cas un temps un temps de $\frac{n^3}{128}+\frac{17n^2}{128}+\frac{15n}{16}+1$ dans $O(n^3)$ (voir l'annexe A pour la démonstration).

Résultats

En pratique, toutes les propositions de la section précédente ont pour but d'accélérer le parcours aléatoire dans la chaîne de Markov des réseaux acycliques. Pour s'en assurer, nous avons fait deux tests en combinant les approches pour déterminer celles offrant les meilleures performances. Nous avons donc mis à l'épreuve une méthode, nommée ST, qui est la méthode de Giudici et Castelo à laquelle nous avons seulement ajouté les vecteurs de bits. Nous avons aussi testé une méthode, nommée C pour complète, contenant toutes les améliorations apportées, soit les vecteurs de bits, le tri topologique en ligne, la vérification pré retournement et le compte de réseaux acycliques accessibles proposés. Toutes les autres méthodes testées dérivent de la méthode C et en diffèrent par seulement une amélioration. Il s'agit de la méthode SCR (Sans le Compte Rapide), SVR (Sans la Vérification pré Retournement) et STTEL (Sans Tri Topologique En Ligne). Tester individuellement des méthodes qui diffèrent par seulement une amélioration sert à mesurer l'apport de chacune des propositions sur l'accélération. Les lignes qui suivent décrivent les tests effectués et présentent les résultats.

Le premier test effectué évalue la rapidité de génération d'un certain nombre de réseaux aléatoires x pour des réseaux de tailles fixes. Nous avons choisi de faire varier le nombre de réseaux à générer de 10 à 100 en utilisant des pas de 10 pour les méthodes lentes. Pour les méthodes rapides, nous varions le nombre de réseaux à générer de 100 à 1100 avec des pas de 200. Nous avons fixé la taille des réseaux à 70 noeuds car de cette façon nous utilisons deux vecteurs de 64 bits pour représenter les réseaux en mémoire. Si nous avions utilisé des réseaux de taille 64, on aurait pu nous accuser de favoriser, à tord, les méthodes utilisant les vecteurs de bits. Les résultats obtenus sont présentés à la figure 21.

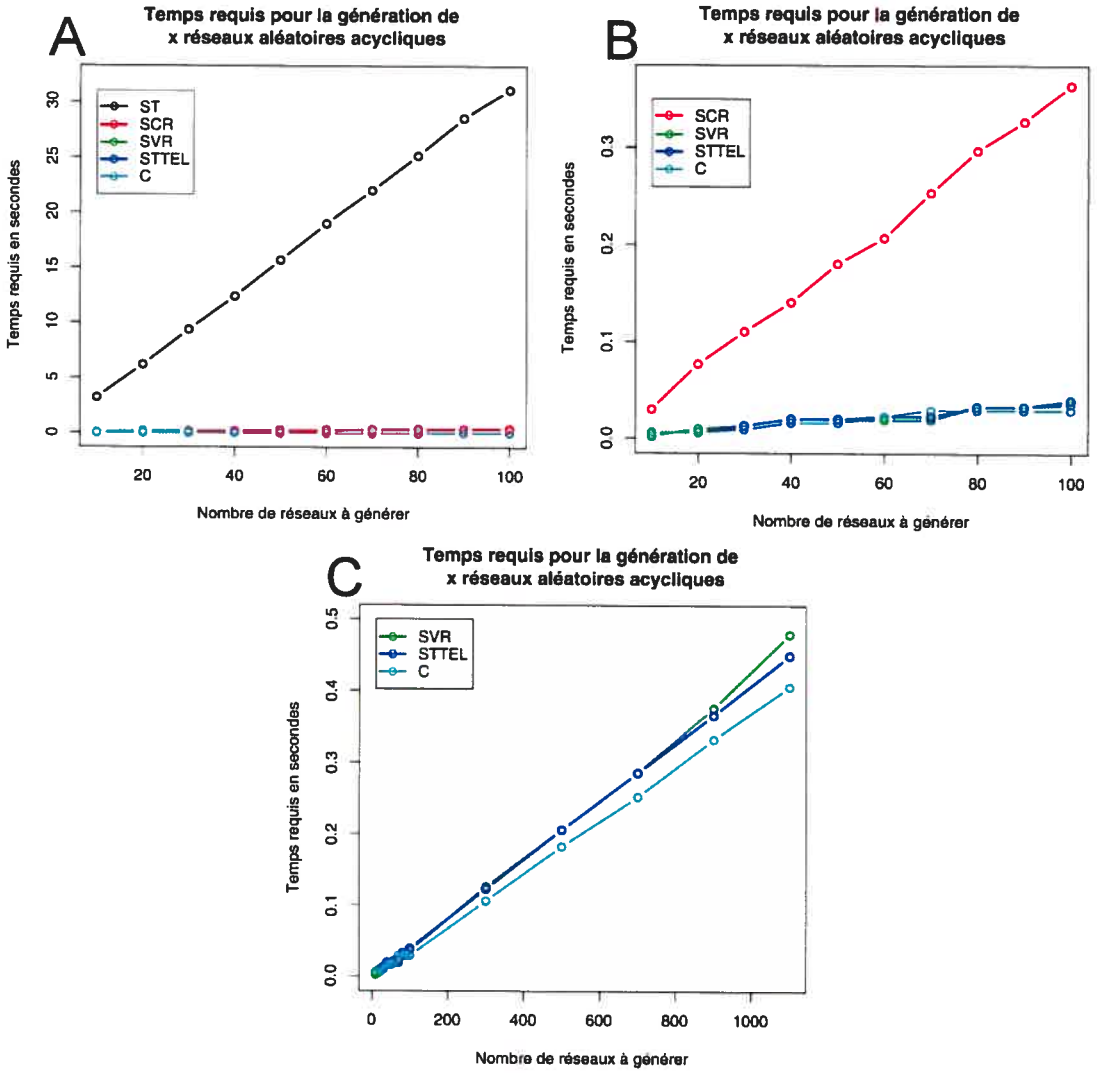


Figure 21 : Graphiques présentant les résultats du premier test. Les notations utilisées dans les graphiques correspondent à ST(méthode de Giudici et Castelo + les vecteurs de bits), C(méthode comprenant toutes les améliorations proposées), SCR(méthode C sans le compte rapide),SVR(méthode C sans vérification pré retournement) et STTEL(méthode C sans tri topologique en ligne). Le test consiste à calculer le temps pris par les méthodes pour générer x réseaux aléatoires acycliques en fixant la taille du réseau à 70 nœuds. Les graphiques A et B diffèrent par la soustraction de la courbe ST dans B. Le graphique C diffère du graphique A par la soustraction de la courbe ST et SCR et des valeurs plus élevées en x.

Le second test effectué est appliqué pour vérifier l'effet de la taille du réseau sur la génération d'un nombre fixe de réseaux acycliques aléatoires. Pour le deuxième test, nous avons donc fixé le nombre de réseaux à générer à 100 et nous faisons varier la taille des réseaux de 10 à 100 en utilisant des pas de 10 noeuds. En suivant la même idée que pour le

premier test, pour les méthodes rapides nous avons ajouté des tailles de réseaux de 100 à 350 en faisant des pas de 50 noeuds. Les résultats obtenus sont présentés à la figure 22.

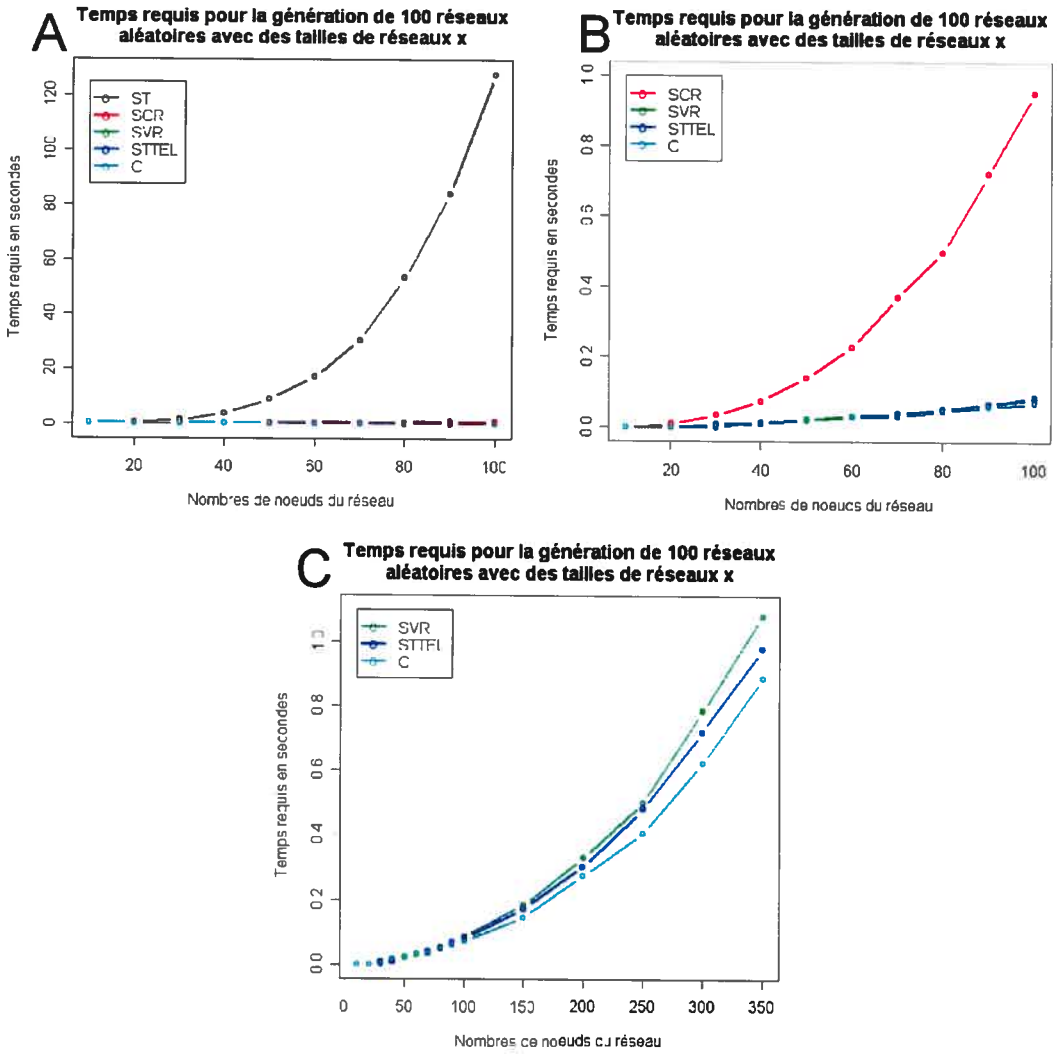


Figure 22 : Graphique présentant les résultats du deuxième test. Les notations utilisées dans les graphiques correspondent à ST(méthode de Giudici et Castelo + les vecteurs de bits), C(méthode comprenant toutes les améliorations proposées), SCR(méthode C sans le compte rapide),SVR(méthode C sans vérification pré retournement) et STTEL(méthode C sans tri topologique en ligne). Le test consiste à calculer le temps pris par les méthodes pour générer 100 réseaux aléatoires acycliques pour un réseau de x nœuds. Les graphiques A et B diffèrent par la soustraction de la courbe ST dans B. Le graphique C diffère du graphique A par la soustraction de la courbe ST et SCR et des valeurs plus élevées en x.

Analyse des résultats

Les graphiques des figures 21 et 22 montrent que les améliorations apportées augmentent grandement l'efficacité de la méthode tant lors de l'augmentation du nombre de réseaux générés qu'au niveau de la taille des réseaux utilisés. Cette affirmation devient évidente quand on compare la courbe ST et la courbe C dans les graphiques A des figures 21 et 22. L'écart entre les deux courbes est énorme ce qui montre bien l'amélioration. En utilisant les graphiques, nous sommes en mesure de classer les améliorations par ordre d'importance (de celle accélérant le plus la méthode à celle l'accélérant le moins). En effet, les courbes SCR, SVR et STTEL se situent toutes au dessus de la courbe C à des distances différentes. Plus la distance à la courbe C est grande, plus l'amélioration que nous avons enlevée accélère la méthode. En utilisant cette observation, l'amélioration la plus importante est donc le compte rapide des réseaux acycliques. Ceci n'a rien de très surprenant étant donné que Giudici et Castelo ne traitait pas de cette problématique dans leur article, nous avons donc utilisé la méthode observée dans le code MATLAB de BNT *i.e.* essayer toutes les possibilités et ajouter un si la modification de base ne crée pas de cycle. De plus, comme nous nous devons de calculer le nombre de réseaux accessibles à chaque nouveau pas dans la simulation MCMC, cette opération est donc critique. La deuxième amélioration la plus importante est la vérification pré retournement. Ceci montre qu'il est avantageux de tester le retournement pour ne pas faire une soustraction de lien inutile. La troisième amélioration la plus importante est le tri topologique en ligne. Bien que cette amélioration ne soit pas énorme, nous voyons clairement à l'aide des graphiques que cette amélioration à une influence. Il est à noter que nous n'avons pas testé une implantation sans vecteurs de bits pour vérifier l'importance de cette amélioration. Cependant, nous croyons que l'implantation utilisant les vecteurs de bits est très importante, car si nous n'utilisons pas cette approche, nous devons utiliser des listes et cette structure de donnée est beaucoup plus coûteuse d'utilisation qu'un vecteur de bits.

Conclusion

Suite à nos expériences, il est clair que les modifications à la méthode de Giudici et Castelo apportent des gains très significatifs en terme de temps d'exécution pour le parcours de la chaîne de Markov des réseaux acycliques. Cette nouvelle implantation efficace sera

utilisée dans les sections suivantes pour atteindre des zones d'échantillonnage intéressantes lors de nos simulations Monte Carlo. Le code C++ développé lors de cette recherche est disponible en ligne à l'adresse www.iro.umontreal.ca/~paqueeri.

CHAPITRE 3

Étude de l'influence des statistiques *a priori* sur l'inférence d'un réseau bayesien à partir de données simulées de réseaux génétiques

Introduction

Un article de Dirk Husmeier paru dans la revue "Bioinformatics" montre de façon convaincante qu'il est difficile, en utilisant seulement des données de biopuces et les réseaux bayesiens, de déterminer avec précision la structure d'un réseau de régulation [40]. Fort de cette observation, les scientifiques en sont maintenant à utiliser d'autres sources d'information pour améliorer les performances de l'approche bayesienne. Des chercheurs de l'Université Stanford, membres du groupe d'Eran Segal, utilisent les séquences de nucléotides en amont des gènes pour identifier des séquences promotrices communes aux gènes étant régulés par le même régulateur et ceci en plus des données de biopuce [74]. Alexander Hartemink, de l'Université Duke, utilise, quant à lui, des données de localisation des protéines dans la cellule ainsi que des données sur l'arrimage protéine-protéine [75].

Dans cette même optique, nous voulons vérifier s'il est possible d'utiliser les réseaux de régulation que nous connaissons déjà et de s'en inspirer pour construire des réseaux ayant des structures semblables. Dans le contexte des réseaux bayesiens, les connaissances *a priori* peuvent facilement être ajoutées en utilisant le terme P(R) présenté à l'équation 8 de la page 22 du chapitre 1. Voyons maintenant ce qui a déjà été fait pour ajouter des connaissances en utilisant le terme P(R).

Différentes méthodes pour utiliser P(R)

Une des méthodes utilisées pour déterminer la valeur de P(R) est tout simplement de ne pas essayer de la déterminer et d'associer la même probabilité à tous les réseaux. Cette méthode a comme avantage de très bien généraliser dans tous les domaines d'inférences de réseaux bayesiens. Cependant, comme elle associe une probabilité uniforme à tous les réseaux, il est possible de croire que cette méthode peut être améliorée en ayant un minimum de connaissance du domaine. À ce sujet, certaines techniques interagissent avec l'utilisateur

pour limiter l'espace de recherche à un petit nombre de réseaux. Cependant, une fois l'espace limité, ces méthodes associent aussi des probabilités uniformes à chacun des réseaux.

Une autre méthode, qui ressemble à celle énoncée précédemment, est de donner en entrée à la méthode d'optimisation un réseau contenant les liens que nous connaissons déjà dans le réseau. Suite à cela, les probabilités P(R) sont déterminées en calculant une mesure de différence entre le réseau courant et le réseau en entrée. Pour plus d'information sur ces techniques, le lecteur peut consulter l'article de Heckerman, Geiger, et Chickering [26].

Dans le cadre de ce projet nous répondons à la question: « Comme nous connaissons déjà un certains nombres de réseaux, est-il possible d'utiliser cette information pour aider à l'inférence de nouveaux réseaux? » En effet, bien avant l'arrivée des biopuces, les biologistes se sont intéressés à déterminer les interactions entre les gènes. Les expériences des biologistes ont permis de générer un grand nombre de réseaux représentant ces interactions. Il est intéressant de se demander si en combinant des données de biopuces avec les connaissances acquises auparavant par les biologistes, il est possible d'améliorer l'inférence automatique de réseaux génétiques.

Méthodologie

La première chose à faire pour vérifier notre hypothèse est de récupérer tous les réseaux de régulation connus jusqu'à ce jour. Ensuite, de l'ensemble des réseaux, nous extrayons des fonctions de densité discrètes pour des statistiques simples comme le nombre de nœuds n'ayant pas d'enfants, le nombre d'enfants moyen, le nombre de nœuds sans parents et le nombre moyen de parents par nœuds.

Il existe bien d'autres statistiques mais, dans le cadre de ce projet, nous nous limiterons à ces quatre. Par la suite, nous récupérons le plus de données de biopuces possibles pour tenter la reconstruction d'un réseau de régulation concernant des gènes d'intérêt. Nous testons toutes les combinaisons de statistiques possibles comme, le nombre de nœuds n'ayant pas d'enfants seulement, le nombre de nœuds n'ayant pas d'enfants plus le nombre de nœuds sans parents plus le nombre moyen de parents par nœuds, etc.

Ensuite, nous comparons le résultat de la reconstruction avec le réseau connu de la littérature et nous vérifions si l'une des combinaisons de statistiques performe mieux que les

autres. Toute cette expérimentation est réalisable, mais il y a un petit problème, nous ne pouvons évaluer la reconstruction car le réseau de régulation de référence n'est pas complet.

Pour régler ce problème, nous devons nous rabattre sur la simulation. Nous allons donc générer aléatoirement un ensemble de réseaux de régulation connus selon des critères réalistes décrits dans la littérature. Nous générerons aussi un réseau à partir de ces mêmes critères et nous simulerons des données de biopuces réalistes à partir de ce réseau. En utilisant notre ensemble de réseaux connus sur lequel nous calculerons nos statistiques et les données de biopuces simulées, nous tenterons de reconstruire le réseau de départ. De cette façon, il est possible d'évaluer la précision de la reconstruction. Il est à noter que toutes les reconstructions seront effectuées en utilisant la méthode MCMC développée au premier chapitre en utilisant une phase de réchauffement de 50 000 réseaux et une phase d'échantillonnage de 10 000. La figure 23 montre un schéma présentant la méthodologie.

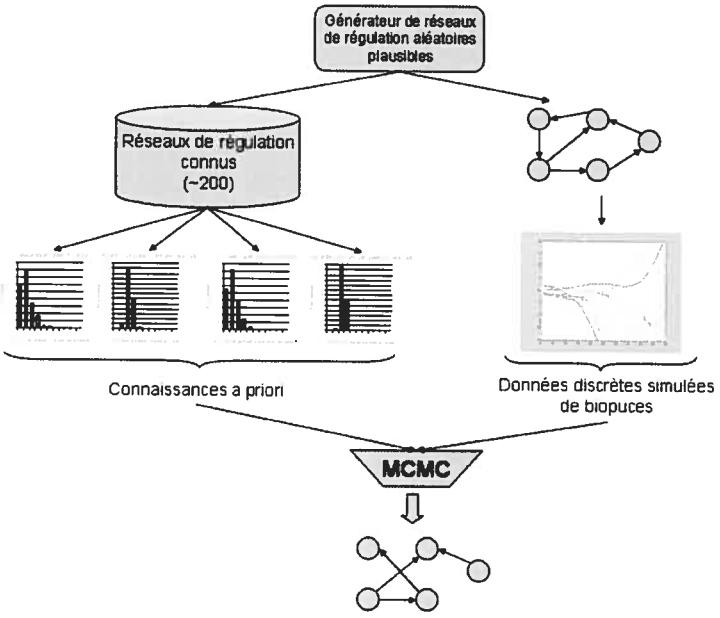


Figure 23 : Méthodologie utilisée. Schéma présentant la méthodologie utilisée pour vérifier l'impact de connaissances *a priori* sur la précision de la méthode de reconstruction MCMC. Nous pouvons voir, en haut complètement, le générateur de réseaux de régulation aléatoires plausibles. De ce générateur, si nous suivons la flèche de droite, nous allons générer un réseau. De ce réseau, nous allons générer des données discrètes de biopuces. Si nous suivons la flèche de gauche à partir du générateur, nous allons générer un ensemble de réseaux de régulation plausibles. De ces réseaux, nous allons extraire de statistiques simples que nous appelons nos connaissances *a priori*. Le trapèze inversé, représente la combinaison des connaissances *a priori* et des données discrètes de biopuces à l'aide de la méthode de recherche de réseau bayesien optimal MCMC. Le résultat obtenu de cette méthode, est un réseau contenant des arcs présents (représentés en rouge) et des arcs non-présents (représentés en bleu) dans le réseau de départ.

Génération de réseaux de régulation réalistes

Comme nous voulons que nos données simulées de biopuces ressemblent le plus possible à la réalité, nous nous devons de respecter certains critères importants établis par les biologistes au sujet de la structure des réseaux de régulation. Les lignes qui suivent traitent de ces critères.

Le premier critère à respecter est celui de faible connectivité. En effet, il a été démontré que les gènes ont en moyenne une vingtaine d'interactions [35]. Pour respecter ce critère, un générateur de nombres aléatoires suivant une loi normale de moyenne 2.5 et de variance 1.1 sera utilisé pour déterminer le nombre de parents qu'aura un gène. Le choix de

ces constantes (2.5 et 1.1) est justifié par le fait que nous allons simuler de petits réseaux de 10 gènes.

Il aurait été impossible de faire une étude approfondie en utilisant de grands réseaux, car le temps de calcul aurait été trop long. Donc, les paramètres de la loi normale ont été adaptés au cas de réseaux plus petits. Comme les valeurs retournées par le générateur sont réelles et que nous voulons un nombre de voisins positif et entier, les valeurs seront arrondies et ramenées dans un intervalle allant d'un à quatre inclusivement.

Le second critère est celui des noeuds sans parents. Pour simuler cette contrainte, 10% des noeuds du réseau n'auront pas de parents [35]. Pour y arriver, une valeur aléatoire d'une loi uniforme(0,1) sera générée pour chaque noeud et si cette valeur est plus petite que .1, alors ce noeud n'aura pas de parents.

Une fois que la structure du réseau respecte ces critères, il faut maintenant ajouter des valeurs sur les arcs reliant les gènes pour pouvoir simuler le réseau. La signification de ces valeurs à un sens bien précis en biologie. Par exemple, supposons un arc allant du gène A au gène B. Si la valeur associée à cet arc est négative, cela signifie que le gène A inhibe l'expression du gène B. Dans le cas où la valeur de l'arc est positive, cela signifie que le gène A fait augmenter l'expression du gène B. Pour générer ces valeurs, une loi uniforme sur l'intervalle (-0.1, 0.1) est utilisée et les |valeurs| > 0.02 sont associées aux arcs.

Voici, pour résumer la méthode de création du réseau, le pseudo-code de la fonction qui a été programmée en matlab :

Algorithme pour la génération d'un réseau plausible de gènes

Entrées:
n = nombre de gènes(dans notre cas 10)
probOfZero = probabilité qu'un gène n'ait pas de parents(dans notre cas .1)
mean = moyenne de parents d'un noeud(dans notre cas 2.5)
var = variance du nombre de parents d'un noeud(dans notre cas 1.1)
min = minimum du nombre de parents pour les noeuds qui ont des parents(dans notre cas 1)
max = maximum du nombre de parents pour les noeuds qui ont des parents(dans notre cas 4)
Sortie :
R(n, e) = un réseau de gènes plausibles

```
1-Pour tous les noeuds i
2-      Si random_Uniforme(0,1) > probOfZero //cas où le noeud à des voisins
3-              nombreDeParents = round(random_Normale(mean, var))
4-              Si nombreDeParents < min, nombreDeParents = min //Ramener à minimum
5-              Si nombreDeParents > max, nombreDeParents = max //Ramener à maximum
6-              Pour j allant de 1 à nombreDeParents
7-                      pasEncoreParent = noeuds n'étant pas encore parents de i excluant le noeud i
8-                      indexNouveauParent=
                        round(random_Uniforme(0.5,length(pasEncoreParent)+.5)) //choix du
                        nouveau parent
9-                      actionSurEnfant = random_Uniforme(-0.1,0.1) //Choix de l'action du parent
                        sur l'enfant
10-                     tant que |actionSurEnfant| < 0.02, actionSurEnfant = random_Uniforme(-
                        0.1,0.1)
11-                     Reseau(pasEncoreParent[indexNouveauParent],i) = actionSurEnfant
12-             Fin pour
13-     Fin si
14-Fin pour
15-retourner Reseau
```

Figure 24 : Pseudo-code présentant l'algorithme utilisé pour générer un réseau plausible de gènes.

Voici aussi un exemple de ce que retourne cette fonction pour un réseau de 10 gènes :

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0.0407	0.0938	0.0882	0
0	0	0	0	0.0704	0	0	0	0.0865	0.0736
-0.0776	0.0417	0	0	-0.0267	0	0	-0.0902	0	-0.0853
0.0444	0.0806	0	0.0436	0	0.0632	0	0	0	-0.0838
0	0	0	0	-0.0825	0	0.0866	0	0	0
-0.0971	0	0	0	0	0	0	0	0.0695	0
0	-0.0437	0	0	0	0	-0.0544	0	0	-0.0361
0	0.0972	0.0658	0	0	0	0	0.0790	0	0
0	0	0	0	0	-0.0972	0	0	0	0

Tableau 1 : Résultat de la fonction de génération d'un réseau. Tableau montrant un exemple d'une matrice de coefficients d'interaction entre les gènes dans un système de taille 10.

Les éléments inscrits dans cette matrice représentent les valeurs des arcs entre les gènes. Par exemple, la valeur 0.0417 en position 4,2 signifie que le gène 4 stimule la production du gène 2 à un niveau 0.0417.

Simulation de données plausibles de biopuces

Maintenant que nous avons la matrice qui représente les interactions entre les gènes de notre réseau, il est possible de simuler le réseau. Pour cela, nous allons supposer que l'effet d'un gène sur un autre peut être modélisé par une fonction linéaire. Donc, pour déterminer la valeur d'un gène x_i au temps t, il suffit d'utiliser l'équation 10 .

$$x_{i,t} = x_{i,t-1} + \sum_{j \neq i}^{n} M_{j,i} x_{j,t-1}$$

Équation 10 : Modèle utilisé pour la simulation des données.

Dans l'équation 10 , Mj,i représente la valeur j,i dans la matrice et xi,t-1 représente le niveau d'expression du gène x_i au temps t-1. Pour simuler le réseau, il suffit de déterminer des valeurs de départ pour les gènes et d'utiliser l'équation pour obtenir les niveaux d'expressions des gènes pour les temps suivants. Cependant, en utilisant cette méthode de simulation, il arrive assez rapidement que les niveaux d'expressions divergent. Pour éviter la divergence, le niveau d'expression d'un gène est ramené dans l'intervalle -4,4.

Voici le résultat d'une simulation de 50 itérations du réseau précédant en utilisant une loi uniforme (-0.5, 0.5) pour générer les niveaux d'expressions de départ des gènes. Remarquez que la plupart des gènes ont déjà atteint la limite d'expression de -4 ou de 4.

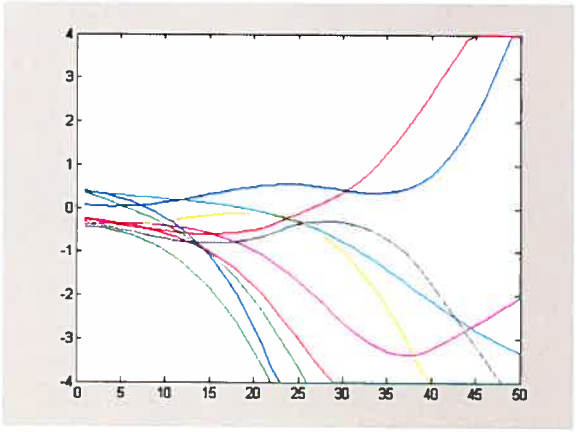


Figure 25 : Simulation de données. Figure montrant un exemple d'une simulation de données de biopuces. Chacune des courbes représente le niveau d'expression d'un gène dans le temps.

Toutefois, ce n'est pas ces données qui vont être utilisées pour inférer le réseau. En effet, ces données sont dépendantes dans le temps. Comme la méthode utilisée nécessite que les données soient indépendantes, il faut en tenir compte lors de la génération des données. Pour simuler des données indépendantes, il suffit pour chacune des données de choisir un état de départ aléatoire, de faire 10 itérations de la simulation et de sauvegarder l'état des gènes à la fin de cette simulation. Ensuite, pour assurer encore plus d'indépendance, il est possible de brasser les données (ce qui a été fait dans ce projet). Ceci règle le cas de l'indépendance des données, mais nos données ne sont toujours pas discrètes. À ce sujet, nous avons utilisé la méthode de discrétisation par intervalle présentée à la figure 7 de la page 23 du chapitre 1. La

figure 26 présente le pseudo-code pour la génération de données indépendantes.

Algorithme pour la génération de données indépendantes pour un réseau donné

Entrées :

nb = nombre de données désirées

R = un réseau pour lequel nous voulons des données

Sortie :

D = des données

1-Pour i allant de 1 au nombre de données désirées nb
2- Pour chacun des gènes du réseau, choisir un niveau d'expression de départ à l'aide d'une loi uniforme (-0.5,0.5)
3- Simuler le réseau R sur 10 itérations
4- Données(i) = niveau d'expression des gènes du réseau après 10 itérations
5-fin pour
6-Brasser les données
7-Pour chacun des gènes j
8- m = moyenne de l'expression du gène j dans les données
9- v = variance du l'expression du gène j dans les données
10- Pour chacune des données k associées au gènes j
11- If k < (m – v), k = -1;
12- elseif k > (m – v), k = 1;
13- else k = 0;
14- fin si
15- fin pour
16-fin pour
17-retourner les données

Figure 26 : Pseudo-code présentant l'algorithme utilisé pour générer des données indépendantes à partir d'un réseau donné.

Une chose très importante à noter est que le réseau créé contient des cycles. Ceci représente la réalité biologique, mais il est impossible d'inférer des réseaux contenant des cycles à l'aide d'une méthode d'inférence de réseaux bayesiens. Comme nous voulons connaître la performance de la méthode sur les réseaux de gènes, nous traiterons les réseaux ayant des cycles en sachant très bien que certains des liens ne pourront pas être déterminés par la méthode.

Description des tests effectués

L'un des problèmes de l'inférence de réseaux de gènes à partir de données de biopuces, est le fait qu'il en coûte très cher, en réalité, pour produire des données. Par le fait même, des tentatives de reconstruction de réseaux de 800 gènes ont été effectuées par le

groupe de Nir Friedman en utilisant seulement 76 données [25]. Dans ce projet, si nous avions voulu simuler le manque de données en proportion, nous aurions été obligés d'utiliser seulement une donnée. Ceci n'aurait pas vraiment eu de sens, donc à ce sujet, il a fallu utiliser une autre méthode pour tester les différentes modifications sur les probabilités *a priori*. Pour y arriver, une banque de données de test, contenant 20 réseaux différents avec 300 données indépendantes pour chacun des réseaux a été générée.

Il est à noter que les réseaux et les données sont produites en utilisant les méthodes décrites précédemment. Pour tester une méthode, elle est soumise à des conditions d'inférence variables. La variabilité se situe au niveau du nombre de données disponibles pour la reconstruction du réseau. Dans les tests effectués dans ce projet, des dimensions de 5, 8, 10, 15, 20, 25, 30, 35, 40, 50 et 60 ont été utilisées. Pour chacune de ces dimensions, 2 reconstructions sont effectuées. Certaines statistiques sont calculées à partir du résultat de la reconstruction, mais ces statistiques seront décrites plus en détails plus loin dans le texte.

Description des méthodes de calcul des probabilités *a priori*

La première méthode testée est celle étant la plus utilisée et la plus simple pour attribuer une probabilité à un réseau. Cette méthode associe une probabilité identique à tous les réseaux. Le but de ce projet étant de vérifier s'il est possible de faire mieux que cette méthode en utilisant de l'information sur les réseaux déjà connus.

Pour faire suite, voici la description des méthodes utilisées pour extraire l'information pertinente dans les réseaux connus. Il est à noter qu'en réalité, le nombre de réseaux connus est limité et pour simuler cette contrainte, seulement 200 réseaux seront utilisés.

La première méthode est celle qui utilise le nombre de noeuds n'ayant pas d'enfants comme critère. La première chose à faire est de générer 200 réseaux à l'aide de l'algorithme de la

figure 24 et de faire des statistiques sur le nombre de noeuds n'ayant pas de parents dans ceux-ci. Un exemple fictif serait qu'il y ait 60 réseaux sur 200 qui aient un seul noeud sans enfant. Supposons ensuite qu'un réseau ayant 1 seul noeud sans enfant soit généré par la

méthode MCMC. Ce réseau obtiendra alors une probabilité 60/200 (.3). Il est à noter que pour un réseau de 10 gènes, il y a 11 possibilités pour cette probabilité allant de 0 à 10. Voici le graphique représentant les probabilités associées à chacune des valeurs :

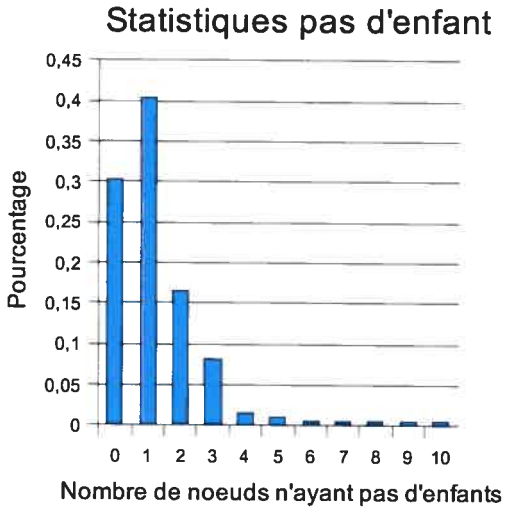


Figure 27 : Distribution de la statistique sur le nombre de nœuds sans enfant.

La deuxième méthode fait des statistiques sur le nombre d'enfants moyen pour les noeuds d'un réseau. Les 200 réseaux aléatoires sont donc utilisés pour générer une distribution représentant ce qui est observé dans ce qu'on connaît déjà. À noter, comme la moyenne a rarement une valeur discrète, celle-ci est arrondit. Voici la distribution qui a été utilisée dans ce projet :

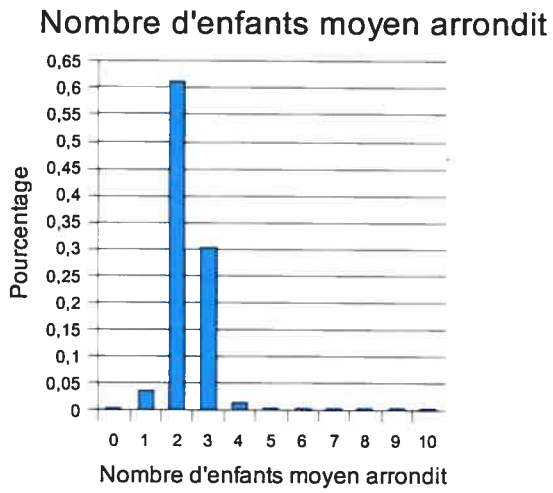


Figure 28 : Distribution du nombre d'enfants moyens par nœud.

La troisième méthode ressemble à la première, à la différence qu'au lieu de faire des statistiques sur les enfants, elle fait la même chose, mais sur les parents. Voici la distribution déterminée à l'aide des 200 réseaux.

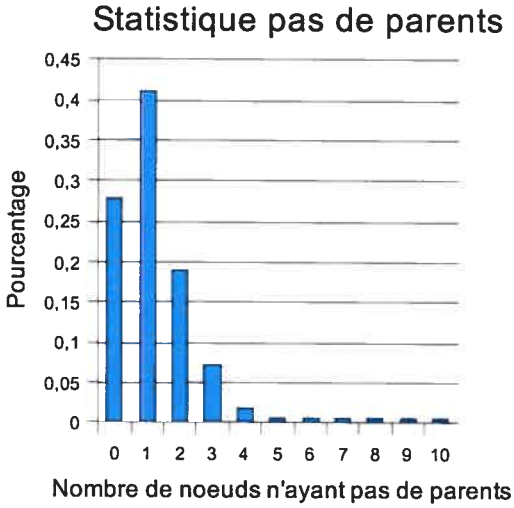


Figure 29 : Distribution du nombre de nœuds n'ayant pas de parents.

La quatrième méthode ressemble pour sa part à la deuxième. Les statistiques sont faites sur les parents au lieu des enfants. Voici les résultats obtenus :

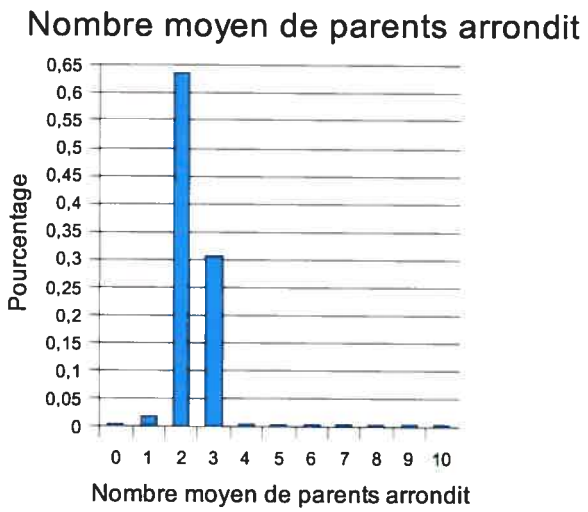


Figure 30 : Distribution du nombre moyen de parents.

Par la suite, il est possible de mélanger ces méthodes entre elles pour obtenir plus ou moins de richesse dans nos probabilités *a priori*. En effet, supposons qu'un réseau ait 1 noeud sans enfant, une moyenne de 2 enfants par noeud, 0 noeud sans parent et une moyenne de 3 parents par noeud. Pour connaître la probabilité d'un tel réseau, il suffit de multiplier les probabilités de chacune des méthodes. Dans notre exemple il serait question de $P_1(1)*P_2(2)*P_3(0)*P_4(3)$ où P_i signifie la probabilité associée à la méthode i. Voici donc tous les agencements de méthodes qui ont été étudiés dans ce projet : (à noter, un x signifie que cette méthode est présente dans l'agencement)

Méthode	*1*	*2*	*3*	*4*
a	X			
b		X		
c			X	
d				X
e	X	X		
f	X		X	
g	X			X
h	X	X	X	
i	X	X		X
j	X		X	X
k	X	X	X	X
l		X	X	
m		X		X
n		X	X	X
o			X	X

Tableau 2 : Différentes combinaisons de statistiques *a priori* testées lors de cette expérimentation.

Résultats

Pour être en mesure de vérifier la validité d'une reconstruction, il faut calculer une valeur qui nous donne une idée de la ressemblance entre le réseau reconstruit et le vrai réseau. Pour ce faire, la notion de sensibilité et spécificité sera utilisée. Tout d'abord, pour calculer ces deux valeurs, il faut définir les notions de vrais positifs (VP), faux positifs (FP), vrais négatifs (VN) et faux négatifs (FN). Voici un tableau permettant de bien résumé :

Réseau de référence\Réseau obtenu	Lien de i à j présent	Lien de i à j non-présent
Lien de i à j présent	VP	FP
Lien de i à j non-présent	FN	VN

Tableau 3 : Vrai/Faux Positifs/Négatifs. VP (vrais positifs), FP (faux positifs), FN (faux négatifs) et VN (vrais négatifs).

Pour déterminer la sensibilité il suffit de faire VP / VP+FN et pour déterminer la spécificité, il suffit faire VN / VN+FP.

Une fois que ces métriques sont bien définies, il est maintenant possible de montrer le meilleur résultat de cette expérimentation obtenu par la méthode g du tableau 2. Cette méthode combine les distributions sur le nombre de nœuds sans enfants avec celles sur le nombre moyen de parents par nœuds.

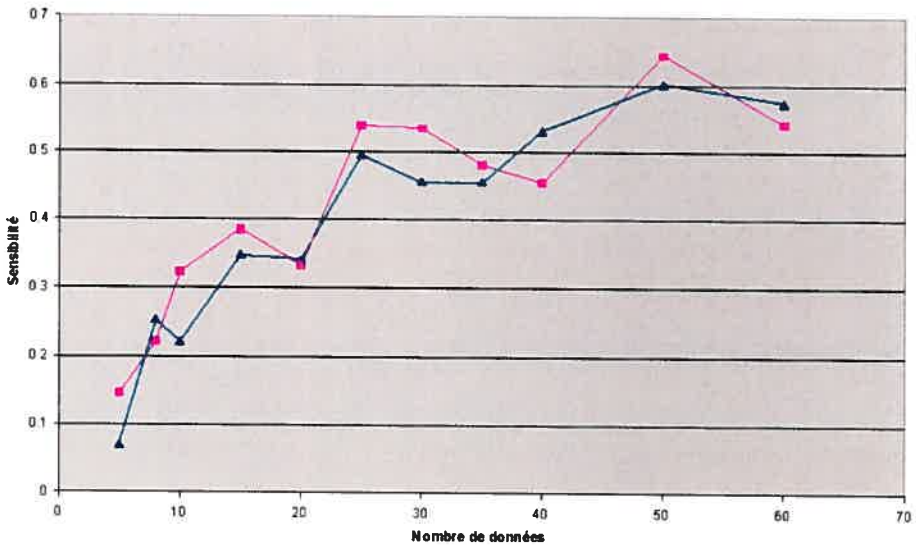


Figure 31 : Sensibilité de la méthode g utilisant les probabilités *a priori* (courbe rose) versus une méthode qui ne les utilise pas (courbe bleue) en fonction du nombre de données utilisées pour l'inférence.

Ce graphique montre la sensibilité moyenne obtenue pour les tests décrits dans la section précédente. Il est possible de constater que pour certains nombres de données comme

10, 15, 25, 30, 35 et 50, la sensibilité de la méthode g est supérieur à la sensibilité de la méthode uniforme. Cependant, il est à noter que les résultats présentés ici sont les meilleurs et que, dans plusieurs cas, la sensibilité était équivalente à celle des probabilités uniformes et dans certain cas pire. Les causes possibles de ces résultats seront énoncées dans l'analyse. Voici les variances associées à ce graphique :

Méthode\Nombre de données	***5***	***8***	***10***	***15***	***20***	***25***	***30***	***35***	***40***	***50***	***60***
Variances méthode uniforme	0,15	0,16	0,18	0,17	0,18	0,19	0,21	0,18	0,17	0,16	0,15
Variances méthode g	0,04	0,18	0,15	0,2	0,19	0,2	0,19	0,17	0,19	0,15	0,15

Tableau 4 : Variances sur la sensibilité. Obtenues pour la méthode uniforme et la méthode g.

Pour faire suite, comme il est question d'une mesure de sensibilité et de spécificité, voici le second graphique correspondant à la spécificité :

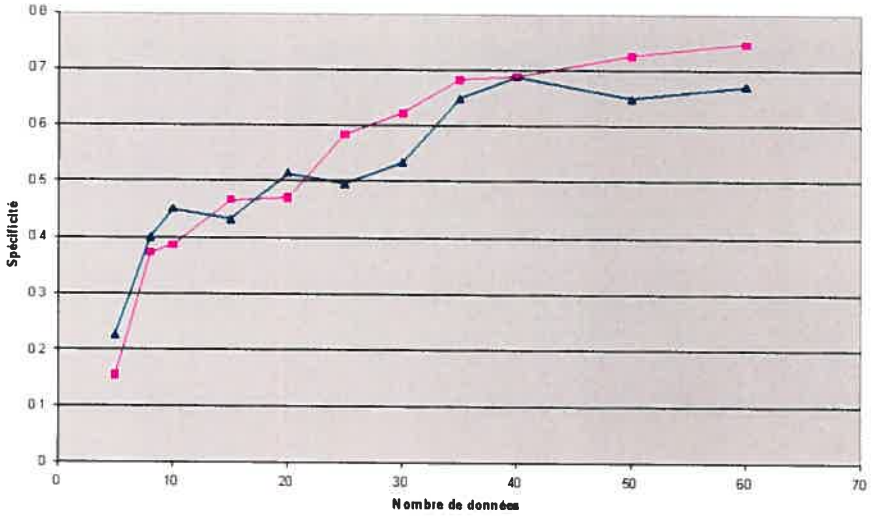


Figure 32 : Spécificité de la méthode g (courbe rose) versus une méthode n'utilisant pas de statistique *a priori* (courbe bleue) en fonction du nombre de données utilisées pour l'inférence.

Encore une fois, il est possible de constater que pour un certain nombre de données comme 15, 25, 30, 35, 50 et 60, la spécificité de la méthode g est supérieure à la spécificité de la méthode uniforme. Cependant, comme il est mentionné précédemment, ce n'est pas généralisé pour toutes les méthodes. Voici la variance associée à ces données :

Méthode\Nombre de données	***5***	***8***	***10***	***15***	***20***	***25***	***30***	***35***	***40***	***50***	***60***
Variances méthode uniforme	0,12	0,15	0,14	0,14	0,13	0,14	0,13	0,08	0,06	0,08	0,07
Variances méthode g	0,1	0,16	0,15	0,15	0,15	0,12	0,1	0,07	0,07	0,05	0,03

Tableau 5 : Variances obtenues pour la spécificité de la méthode uniforme et celle de la méthode g.

Voici un exemple d'une reconstruction effectuée par la méthode g en utilisant un ensemble de données de dimension 30 :

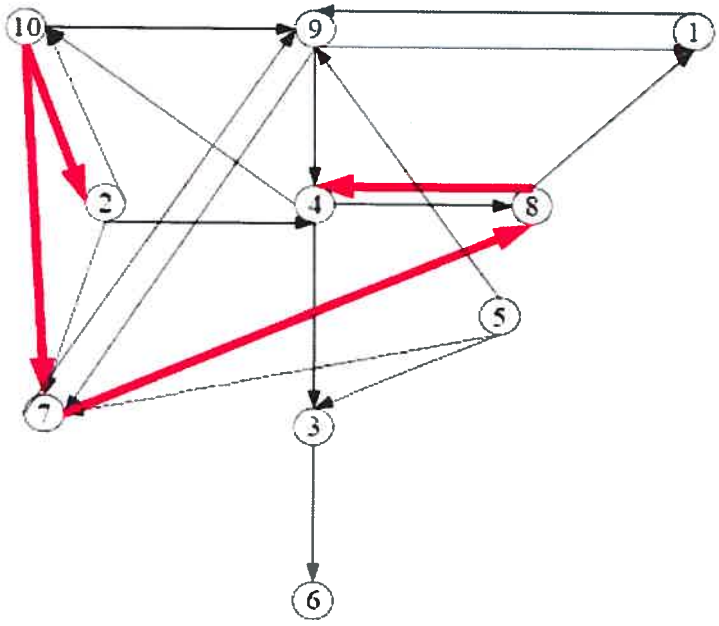


Figure 33 : Résultat d'une inférence. Les ars en rouges correspondent aux arcs retrouvés par la méthode. À noter la présence de plusieurs cycles dans le vrai réseau.

Analyse des résultats

Les résultats obtenus lors de cette expérimentation sont un peu surprenants. En effet, nous nous attendions à ce que les méthodes non-uniformes affichent des sensibilités et des spécificités nettement supérieures à la méthode uniforme. Cependant, les meilleurs résultats montrés précédemment prouvent le contraire.

À ce sujet, il est bon de se questionner sur la cause de cette piètre performance. L'explication que nous avons à ce sujet est que les probabilités *a priori* utilisées dans ce projet, ne sont pas adaptées à la méthode d'optimisation MCMC. Nous voulons dire par là, que les statistiques relevées sur les 200 réseaux sont des statistiques sur des réseaux complets et la méthode commence avec un réseau vide incomplet. Les probabilités ne sont pas adaptées à la méthode, celles-ci devraient tenir compte du fait que le réseau est en construction. Par exemple, pour la statistique sur le nombre de parents moyens pour un noeud, il est impossible que la première itération de la méthode produise une réseau ayant une moyenne de 2 parents par noeud, car dans le meilleur des cas, elle aura ajouté un parent à tous les noeuds et la moyenne sera de 1.

Il faudrait, pour corriger cette situation, faire des statistiques en fonction de l'état du réseau. Par exemple, si le nombre moyen de parents moyens est plus petit qu'un certain seuil, il faudrait utiliser une probabilité plutôt qu'une autre. En utilisant ce genre de nouvelle probabilité, nous croyons qu'il serait possible d'obtenir de meilleurs résultats que ceux montrés dans ce rapport.

Conclusion

Il a été démontré à l'aide de simulation que d'utiliser des statistiques simples comme le nombre de noeuds sans parents, le nombre de parents moyens, le nombre de noeuds sans enfants et le nombre d'enfants moyens n'améliore pas significativement l'inférence de réseaux de gènes à l'aide de la méthode bayesienne. Cependant, certaines hypothèses comme d'utiliser des probabilités tenant compte de la méthode de recherche pourraient peut-être améliorer les résultats. À ce sujet, des expérimentations futures pourraient être faites pour vérifier cette hypothèse.

CHAPITRE 4

Analyse statistique de l'expression des gènes pour la découverte de gènes différemment exprimés

Introduction

Avec l'avènement des biopuces à ADN, les scientifiques disposent maintenant d'un nouvel outil pour étudier la maladie. En effet, dans le cas des maladies génétiques[1], certains gènes subissent des variations au niveau de leur expression qui, si elles se produisent au mauvais moment dans la vie de la cellule, peuvent nuire grandement à son développement. Les cancers ne faisant pas exception à cette règle, des chercheurs de l'Université de Montréal (en particulier dans l'équipe du Dr.Trang Hoang) tentent d'identifier des gènes pouvant causer un type méconnu de leucémie. Pour y arriver, ils utilisent des souris mutantes en un gène spécifique (transgéniques) pour simuler une leucémie. En comparant les niveaux d'expression de ces souris à ceux de souris normales (ne possédant pas la leucémie), ils pourraient être en mesure de comprendre de manière plus approfondie cette leucémie.

Pour y arriver, ils doivent appliquer une méthode informatique pour extraire les gènes significativement et différemment exprimés à partir des données de biopuces des souris normales et des souris leucémiques. Ce chapitre traite d'une méthode que nous avons utilisée pour effectuer cette tâche. Nous expliquerons les choix que nous avons faits et présenterons les détails de cette méthode.

Mise en situation

Le problème que nous voulons résoudre se pose comme suit : supposons que nous ayons n réplications de données de biopuces provenant d'une condition A et m autres réplications de données de biopuces provenant d'une condition B. Nous voulons extraire de ces données les éléments qui subissent la plus grande variation en passant d'une condition à l'autre tout en ayant des réplications se ressemblant. La ressemblance des réplications est un point important, car les données de biopuces possèdent énormément de bruit.

---

[1] Il peut s'agir de mutations au niveau du génome, de translocations de bout d'ADN ayant pour effet de sur exprimer des gènes normalement exprimés faiblement, etc.

En termes plus formels, nous voulons donc une différence en valeur absolue entre les moyennes des réplications (moyenne réplications A – moyenne réplications B) de chacune des conditions élevées ainsi qu'une variance sur les réplications qui est basse.

Méthodes existantes

Cela fait déjà un certains nombres d'années que les statisticiens apportent des solutions à ce problème. Une des conséquences de cette situation est qu'il existe une pléthore de méthodes toutes aussi meilleures les unes que les autres. Les non initiés au domaine doivent donc parcourir la littérature à la recherche de la méthode qui convient le mieux au problème qu'ils veulent résoudre. Les lignes qui suivent décriront brièvement les méthodes que nous retrouvons le plus fréquemment et expliqueront le choix qui a été effectué.

Test t

Il existe plusieurs versions au test t. Les versions diffèrent selon que nous sommes face à un problème pour lequel nous pouvons supposer ou non que les variances des données sont égales. Comme nous travaillons avec des données provenant de biopuces et que la nature de ces données est bruitée, nous ne pouvons supposer une variance égale. De plus, l'équipe du Dr. Hoang nous fournit des triplicatas de données, et ceci est considéré comme étant un nombre peu élevé [48]. L'idée du test t est assez simple, il suffit de calculer une valeur t pour laquelle nous connaissons la distribution. La valeur t se calcule en utilisant l'équation 11.

$$\text{valeur t} = \frac{\overline{X_A} - \overline{X_B}}{\sqrt{{s_A}^2 / K_A + {s_B}^2 / K_B}}$$

Équation 11 : Calcul de la valeur t.

Dans l'équation 11, X_A représente les valeurs d'expression de tous les gènes de la condition A et X_B représente les valeurs d'expression de tous les gènes de la condition B. La barre au-dessus signifie la moyenne des expressions. ${s_A}^2$ représente la variance des expression de la condition A et ${s_B}^2$ représente la variance des expression de la condition B. K_A et K_B représente respectivement le nombre de données de la condition A et de la condition B. Ensuite, nous devons attribuer un « p-value » à la valeur t observée. Pour ce faire, nous devons connaître la distribution des valeurs t. En supposant que les expressions X suivent une

distribution normale, il a été démontré que les valeurs t suivront une distribution t ayant le nombre de degrés de liberté calculé à l'aide de l'équation 12 [48][49]:

$$\text{degré de liberté} = \frac{(s_A{}^2 / K_A + s_B{}^2 / K_B)^2}{(s_A{}^2 / K_A)^2 / (K_A - 1) + (s_B{}^2 / K_B)^2 / (K_B - 1)}$$

Équation 12 : Calcul du degré de liberté de la distribution t .

Voici maintenant un graphique montrant un exemple des valeurs t en fonction de la moyenne de l'expression de tous les gènes

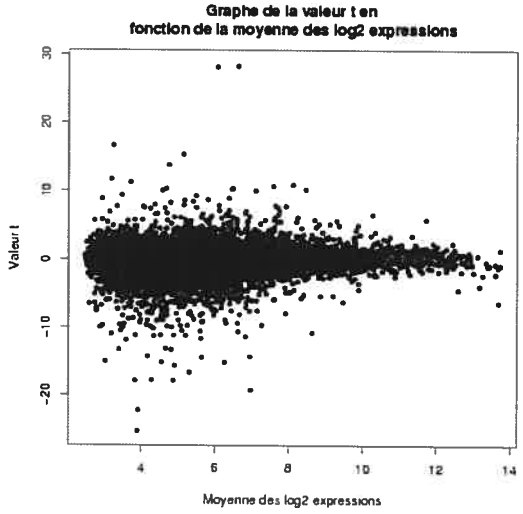


Figure 34 : La valeur t obtenue avec le test t en fonction de la moyenne de toutes les expressions pour un gène donné.

Si nous regardons attentivement la figure 34 , nous pouvons observer que la valeur t n'est pas indépendante de la moyenne. En effet, nous observons beaucoup plus de valeurs t extrêmes, donc intéressantes, quand la moyenne est base (entre 4 et 7). Nous ne voulons pas de cette dépendance et c'est pour cette raison que le test t ne sera pas utilisé ici.

Significance Analysis of Microarray (SAM)

Cette méthode décrite par Tusher, Tibshirani et Chu à pour but de corriger les défauts du test t [50]. En effet, le test t a le défaut de favoriser les gènes ayant des niveaux d'expression bas car les variances de ces gènes sont généralement plus basses que celles des gènes ayant des niveaux d'expressions plus élevés.

Pour y arriver, les auteurs comparent les valeurs t obtenues de la même façon que celles du test t avec d'autres valeurs t calculées en appliquant F permutations aléatoires entre les classes. Par exemple, le triplicata 2 de la condition A pourrait devenir le triplicata 3 de la condition B et vice-versa. Ensuite, ils ordonnent les valeurs t obtenues pour chacune des F permutations et ils font la moyenne à chaque position. De cette façon, ils obtiennent une idée des valeurs t attendues. En résumé, on ajoute un facteur au dénominateur pour corriger l'effet de dépendance observé dans le test t. Pour plus de détail nous vous référons à l'article de Tusher, Tibshirani et Chu [50].

Pour visualiser l'effet de la correction apportée par la méthode SAM, voici un graphique superposant les valeurs t en fonction de la moyenne des expressions des deux méthodes (test t et SAM)

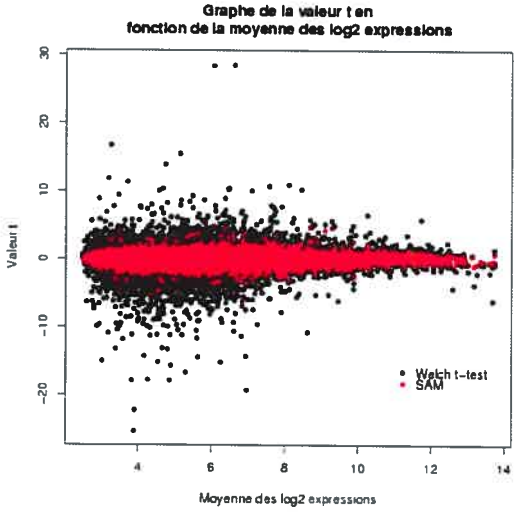


Figure 35 : La valeur t en fonction de la moyenne de toutes les expressions pour le test t (noir) et SAM (rouge).

Le graphique de la figure 35 montre clairement l'effet de la méthode SAM sur la valeur t. Nous observons des valeurs t extrêmes distribuées pratiquement uniformément sur l'ensemble de la moyenne des expressions.

Bayesienne empirique

Avec cette méthode on vise le même but que la méthode précédente, soit de diminuer la dépendance entre la valeur moyenne des expressions et la valeur t. Bien évidemment, la

méthode utilisée pour y arriver n'est pas la même. Contrairement à la méthode précédente, Gordon K. Smyth utilisent un modèle linéaire pour déterminer si un gène est différemment exprimé [60].

Soit K = {1,2,...k1,k1+1,k1+2,...k1+k2} où k1 et k2 représentent respectivement le nombre de réplications dans l'ensemble de données A et l'ensemble de données B. Posons aussi j allant de 1 à n où n représente le nombre de gènes ciblés par la puce. Voici maintenant le modèle mathématique pour lequel nous devons déterminer les paramètres a_j et b_j :

$$Y_{jk}=a_j + b_j x_k + \epsilon_{jk}$$

Équation 13 : Modèle utilisé pour représenter la problématique des gènes différemment exprimés.

Dans l'équation 13, Y_{jk} correspond à la donnée pour le gène j que nous observons dans l'ensemble K et x_k vaut 1 pour les ensembles plus petits ou égaux à k1 et 0 pour les autres. Le terme ϵ_{jk} représente le bruit induit par la puce. Il est important de remarquer que les variables a et b dépendent de j et que les variables x et Y dépendent de k. Nous voulons donc déterminer les valeurs à donner à a_j et b_j pour minimiser la différence entre le Y_{jk} observé et le modèle que nous tentons de déterminer : $a_j + b_j x_k$. Par exemple, supposons que les données que nous observons dans la condition A (k1=3, 1, 2, 3) pour le gène j=1 sont (9.225333, 9.849712, 9.883079) et les données pour la condition B (k2=3, 4, 5, 6) pour le même gène sont (7.159841, 7.350615, 7.702326). Le système d'équations que nous devons résoudre est le suivant :

$$9.225333= a_j + b_j + \epsilon_{j1}$$

$$9.849712= a_j + b_j + \epsilon_{j2}$$

$$9.883079= a_j + b_j + \epsilon_{j3}$$

$$7.159841= a_j + b_j + \epsilon_{j4}$$

$$7.350615= a_j + b_j + \epsilon_{j5}$$

$$7.702326= a_j + b_j + \epsilon_{j6}$$

La résolution de ce système donne les valeurs $a_j = 7.404261$ et $b_j = 2.248447$ et les erreurs, dans l'ordre, sont : -0.42737500, 0.19700400, 0.23037100, -0.24441967, -0.05364567, 0.29806533. Comme $b_j >> 0$, il est probable que ce gène soit différemment

exprimé, mais nous ne pouvons pas en être assuré. L'idée de la méthode empirique est donc de déterminer si la valeur b_j est une valeur rare comparativement aux autres valeurs. C'est ce qui doit être fait par la suite en faisant des statistiques sur les valeurs b_j. Pour plus de détails à ce sujet, nous suggérons la lecture de l'article de Smyth [60].

Voici les résultats de la méthode empirique comparativement aux deux autres méthodes.

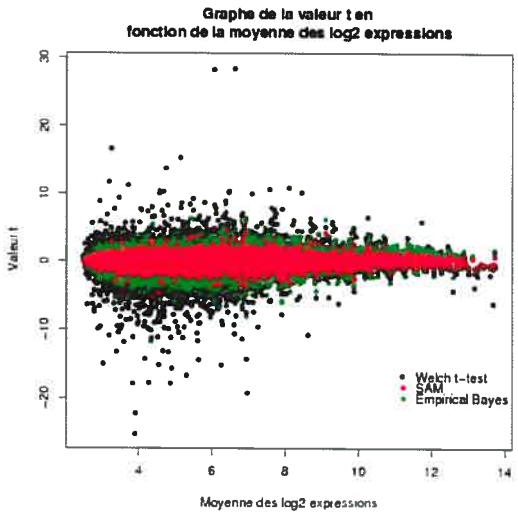


Figure 36 : La valeur t en fonction de la moyenne de toutes les expressions pour le test t (noir) et SAM (rouge) et bayesienne empirique (vert).

Méthode choisie

Parmi toutes les méthodes décrites précédemment, notre choix s'est arrêté sur la méthode bayesienne empirique. Nous justifions ce choix par les excellentes performances de cette méthode tel que discutées dans plusieurs articles [51][52]. Ces articles présentent les résultats obtenus par différentes méthodes pour détecter les gènes différemment exprimés sur des données dont nous connaissons à l'avance les résultats attendus.

Pré-traitement des données

Avant de faire l'analyse proprement dite des données, nous devons procéder à un pré-traitement de manière à enlever toutes sources de variations non-biologiques qui peuvent être dues à des saletés sur les puces ou à des variations subtiles dans les conditions

expérimentales. Il existe, ici aussi, un grand nombre de méthodes pour le pré-traitement. Notre choix s'est arrêté sur la méthode Robust Microarray Analysis (RMA) en se basant sur les résultats présentés dans plusieurs articles [53][54][55][56].

La méthode RMA procède en trois étapes. La première étape consiste à enlever le bruit de fond de chacune des données. Ensuite, les données sont normalisées en utilisant une normalisation par quantile qui revient à ajuster la distribution des valeurs observées de façon à ce qu'elles aient la même distribution. La dernière étape consiste à ajuster un modèle linéaire pour déterminer la vraie valeur. Pour plus de détails au sujet de cette méthode de pré-traitement, nous référons le lecteur à l'article du groupe de Terry Speed [57].

Description des données

Pour l'analyse, l'équipe du Dr. Hoang nous a fourni des triplicatas de données de 5 types différents. À noter que toutes les données proviennent de la puce MOE43A GeneChip® d'Affymetrix qui s'applique sur le génome de la souris [59].

Les données sont divisées en deux groupes : les pré-leucémiques et les leucémiques. La différence entre les deux est le moment dans la vie de la souris où les échantillons cellulaires du thymus ont été recueillis. Le thymus est une glande qui se situe dans la partie supérieure antérieure de la cavité située entre le cou et l'estomac. Le thymus joue un rôle important dans le développement du système immunitaire en permettant la maturation des lymphocytes en cellules T (T pour thymus). Ces mêmes cellules T se retrouvent alors dans le sang où elles jouent un rôle immunitaire important.

L'équipe du Dr. Hoang s'intéresse à la leucémie lymphoblastique sévère. Cette leucémie est caractérisée par la production anormale de lymphoblastes malins dans la moelle osseuse d'où l'étude du thymus. La figure 37, montre les différents types des données ainsi que les analyses qui ont été effectuées :

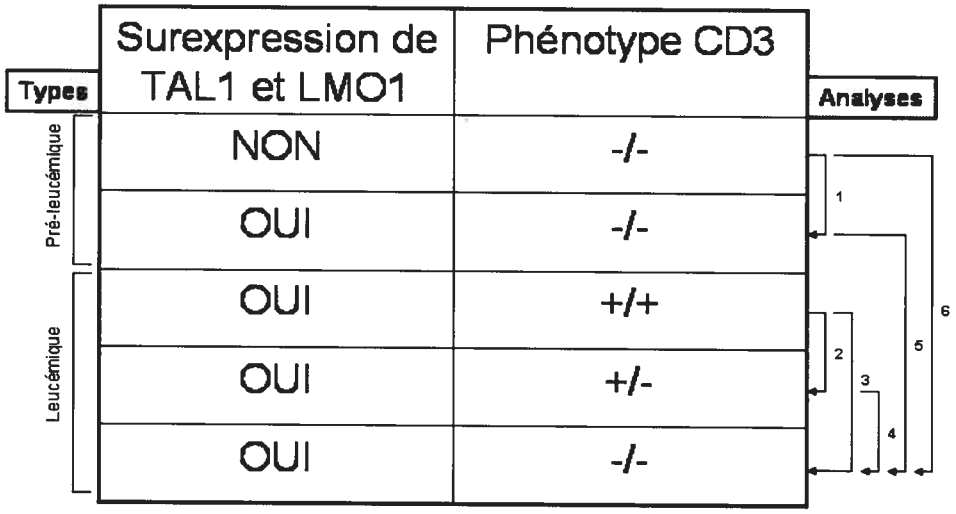


Figure 37 : Les différents types de données ainsi que les analyses effectuées. De gauche à droite le type de données, les gènes TAL1 et LMO1 surexprimés ou non chez les souris, le phénotype du gène CD3. À droite, les analyses qui ont été effectuées. Le gène TAL1 est décrit comme étant le gène « T-cell acute lymphocytic leukemia 1 », le gène LMO1 est « LIM domain only 1 » et le gène CD3 est les gènes « CD3 antigen, epsilon polypeptide ».

Analyses effectuées

Toutes les analyses effectuées ont été réalisées à l'aide de l'outil R (version GNU de S-Plus). La librairie BioConductor contient le package affy contenant la fonction rma() qui nous permet de pré-traiter les données à l'aide de la méthode RMA [58]. Il y a aussi le package limma qui contient les fonctions lm.series() et ebayes() permettant de déterminer les gènes différemment exprimés à l'aide de la méthode bayesienne empirique. Voici maintenant un exemple du code à utiliser pour effectuer l'analyse numéro 1 à l'intérieur de R. Nous lançons le programme dans le répertoire où se trouvent les données CEL produites par les puces Affymetrix.

```
>library(affy) #Nous utilisons le package affy

>allDataNoPreTreat=ReadAffy() #Nous lisons tous les fichier CEL et les mettons dans la variable allData

>allData=exprs(rma(allDataNoPreTreat)) #Nous appliquons la méthode RMA aux données et allons chercher toutes les valeurs en utilisant la fonction exprs

>WTmmFiles=c(« 1264 », « 1458 », « 1459 ») #nom des fichiers de type WT -/-

>TAL1_LMO1_mmFiles=c(« 1298 », « 1299 ») #nom des fichiers de type TAL1/LMO1 -/-
```

```
>library(limma) #nous utilisons le package limma

>pId=row.names(allData) #Allons chercher les noms des probe ID

>presentPIdWTmm=union(union(pId[allData[,WTmmFiles[1]]>log2(100)],
pId[allData[,WTmmFiles[2]]>log2(100)]), pId[allData[,WTmmFiles[3]]>log2(100)]) #Nous
conservons les probes ID ayant au moins une valeurs au-dessus de log2(100) pour au moins
un ensemble de données

>presentPIdTAL1_LMO1mm=union(union(pId[allData[,TAL1_LMO1_mmFiles
[1]]>log2(100)],pId[allData[,TAL1_LMO1_mmFiles[2]]>log2(100)]),
pId[allData[,TAL1_LMO1_mmFiles [3]]>log2(100)]) #Nous conservons les probes ID ayant
au moins une valeurs au-dessus de log2(100) pour au moins un ensemble de données

>selectedPId=union(presentPIdWTmm, presentPIdTAL1_LMO1mm)

>selData=allData[selectedPId,cbind(WTmmFiles, TAL1_LMO1_mmFiles)] #Voici les
données qui seront utilisées pour l'analyse

>library(limma) #Package utilisé pour l'analyse

>fit=lm.series(selData,design=cbind(c(1,1,1,1,1),c(1,1,1,0,0))) #Ajustement au modèle décrit
dans la section bayesienne empirique

>AnalysisResults=ebayes(fit) #Application de la méthode bayesienne empirique

>top100ProbeId=row.names(selData)[order(AnalysisResults$p.value[,2])[1:100]] #La
variable top100ProbeId contient maintenant les 100 probes Id les plus différemment exprimés
entre les souris de type WT -/- et TAL1/LMO1 -/-

>library(moe430a) #Annotation de la puce MOE43A

>mget(top100ProbeId,env=moe430aSYMBOL) # Nom des gènes associés aux probe ID
```

Figure 38 : Le code à exécuter dans R pour déterminer les gènes différemment exprimés.

Résultats

Une façon efficace de présenter graphiquement les résultats des analyses est d'utiliser des graphiques volcans. Dans ces graphiques, l'axe des x correspond à la différence entre les moyennes de chacune des conditions. Par exemple, pour l'analyse numéro 1, il est question de la moyenne des données de types WT -/- moins la moyenne des données TAL1/LMO1 -/-. Une différence positive signifie que le niveau d'expression dans la condition WT -/-, par exemple, est plus élevé que le niveau d'expression dans la condition TAL1/LMO1 -/-. Nous disons donc que le gène pour lequel la différence est positive est sous-exprimé dans la

condition TAL1/LMO1 -/- (surexprimé dans la condition TAL1/LMO1 -/-). Dans le cas où la différence est négative, cela signifie que le niveau d'expression est plus élevé du côté de TAL1/LMO1 -/-, soit que le gène en question est surexprimé dans la condition TAL1/LMO1 -/- (sous-exprimé dans la condition WT -/-).

L'axe des y représente la valeur absolue du log en base 10 du P-Value. Donc, plus la valeur est élevée, plus ce que nous observons est rare. Les résultats intéressants des analyses se retrouvent en haut du graphique. Pour chacune des analyses, nous avons indiqué le nom des 20 gènes ayant les P-Values les plus bas.

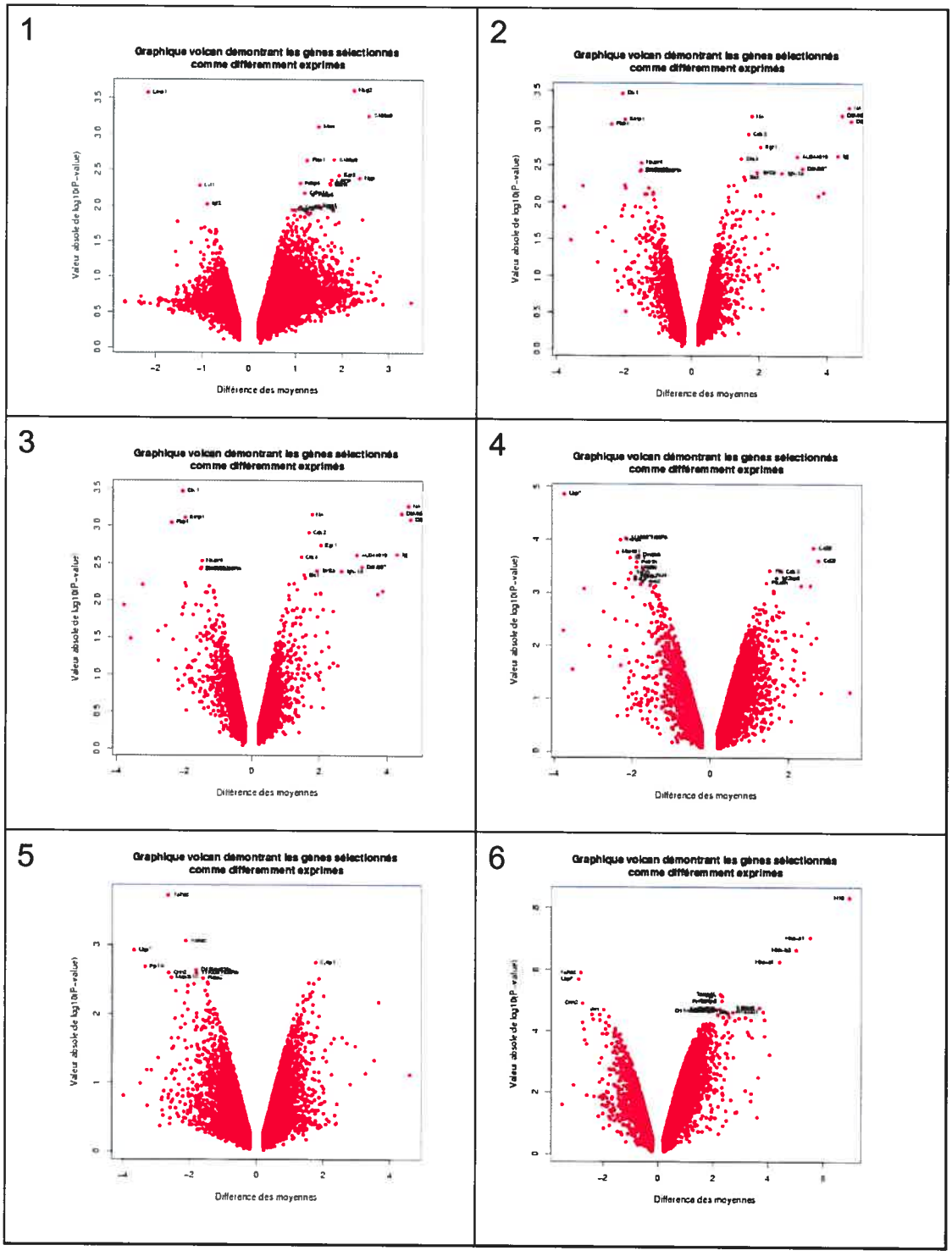


Figure 39 : Les résultats obtenus pour chacune des 6 analyses sous forme de graphique volcan.

Conclusion

L'analyse préliminaire fait ressortir des gènes intéressants comme par exemple les gènes LMO1 et LYL1 dans l'analyse 1 et CD28 dans l'analyse 4. Il y a aussi certaines observations rassurantes comme l'apparition de plusieurs probe ID correspondant au même gène. C'est le cas pour les gènes Hba-a1 (analyse 6), CD28 analyse (4), Ywhaz (analyse 5). Certains gènes apparaissent aussi dans plusieurs analyses comme les gène Ywhaz et Usp7. Notez ici qu'une analyse plus approfondie par les membres de l'équipe du Dr. Hoang apportera beaucoup plus d'information au sujet des résultats.

CHAPITRE 5

Analyse globale de l'expression des gènes chez l'humain pour la découverte de paires inhibiteur-inhibé significatives

Introduction

Depuis un certain nombre d'années, un grand nombre de bases de données d'expression de gènes ont vu le jour sur l'Internet. Parmi celles-ci figure la Stanford Microarray Database SMD ainsi que la Gene Expression Omnibus GEO au NCBI. Ces bases de données (BD) contiennent des expressions de gènes provenant d'expériences de biopuces faites sur plusieurs types cellulaires ainsi que sur plusieurs organismes. La quantité d'information contenue dans ces BD est phénoménale et le besoin en outils efficaces pour les analyser commence de plus en plus à se faire sentir.

Comme la quantité de données est très grande, des méthodes tels les réseaux Bayesiens ne sont pas applicables car elles sont trop coûteuses en temps de calcul. Nous devons donc nous rabattre vers des méthodes plus simples qui sauraient efficacement extraire de l'information pertinente. Dans ce chapitre, nous décrirons la méthode utilisée pour extraire des relations inhibiteurs-inhibés apparaissant de façon marquée dans le génome humain. Pour y arriver, nous utiliserons toutes les données disponibles dans SMD et GEO.

Mise en situation

Dans le cadre de cette recherche, nous nous intéressons seulement aux relations inhibiteurs-inhibés qui peuvent survenir entre les gènes. Au point de vue biologique, nous voulons connaître le gène A codant pour un élément qui aurait la capacité de diminuer la production du gène B. Nous avons utilisé le terme général élément dans la phrase précédente, car le gène A pourrait produire autant un facteur de transcription qu'un microARN [9][10].

Dans le cadre d'un projet de recherche automatisée de microARN à travers plusieurs organismes, nous avons eu l'idée d'inclure les données d'expressions pour fournir des candidats potentiels. L'hypothèse de base étant que l'augmentation de l'expression du gène A, codant pour un microARN, aura pour effet de diminuer l'expression du gène B considéré

comme étant sa cible. Cette inhibition devrait apparaître dans les données d'expression. Nous recherchons donc des couples de gènes dans le génome pour lesquels nous observons un patron d'inhibition fort dans les données de biopuces. Nous produirons donc une liste de candidats microARN-cible potentiels. Cette liste devra, par la suite, être analysée pour retrouver les candidats ayant le plus de chance d'être un couple microARN-cible selon des critères décrits dans la littérature.

Le présent chapitre traitera donc de la méthode utilisée pour créer une liste de candidats inhibiteurs-inhibés intéressants à partir des BD SMD et GEO. À noter que l'analyse pour extraire de la liste les candidats microARN-cible potentiel ne sera pas traiter dans ce chapitre. Ceci fera l'objet d'un projet futur d'un autre étudiant du laboratoire.

Cueillette des données

L'étape la plus laborieuse de cette recherche consiste à la récupération de toutes les données d'expression de chacune des deux BD et d'associer ces données à des gènes. En effet, aucune des deux BD n'offre une façon simple de recueillir toutes les données disponibles pour un gène et un organisme donnés. Pour cette recherche, nous avons limité notre travail à l'humain.

La première étape est de trouver une façon de faire des requêtes précises pour des gènes spécifiques. Si nous prenons l'exemple du gène TAL1 chez l'humain, il existe 3 autres façons de le nommer : SCL, TCL5 et tal-1. Comme nous ne voulons pas faire le travail 4 fois, il faut s'assurer d'avoir le nom qui identifie de façon unique le gène TAL1. Pour ce faire, nous avons utilisé la liste des noms uniques de gènes rendue disponible par le comité de nomenclature des gènes HUGO[61]. La méthodologie pour produire cette liste est décrite en détail dans les articles de Wain et collègues [62][63].

L'étape suivante consiste à utiliser la liste des noms uniques de gènes et de s'en servir pour faire des requêtes pour obtenir 2 choses : 1) l'identifiant décrivant d'où provient la donnée et 2) la valeur de l'expression pour ce gène. En date du 7 février 2005, il n'est pas possible de faire ce genre de requête sur aucune des deux BD. Nous devons donc nous même faire l'association entre les données des BD et les gènes. Les prochains paragraphes traiteront de la méthodologie développée pour obtenir l'information recherchée.

Gene Expression Omnibus GEO

Les développeurs de GEO ont décidé de regrouper les données d'expression en groupes statistiquement comparables qu'ils ont nommés « GEO datasets (GDS)» [64]. Les données d'un même groupe proviennent donc d'une même technologie et ont été corrigées et normalisées en utilisant les mêmes critères. Ces opérations appliquées aux données font que des analyses peuvent être effectuées directement sur un GDS. Il est possible de télécharger à même le site FTP du NCBI les fichiers GDS pour tous les groupes annotés disponibles. Cependant, il faut être en mesure de savoir dans quels fichiers GDS un gène se retrouve. Nous avons donc créer un petit script qui envoie une requête au site GEO et retrouve une liste de fichiers GDS dans lequel le gène de la requête se retrouve. Nous avons appliqué ce petit script à tous les gènes de l'humain pour retrouver les fichiers GDS à télécharger du site FTP du NCBI. Par la suite, nous avons extrait les valeurs d'expressions des fichiers GDS pour chacun des gènes de manière à générer un fichiers contenant à chaque ligne toutes les données pour un seul gène.

Stanford Microarray Database (SMD)

L'organisation de la base de données SMD est quelque peu différente de celle de GEO [65]. En effet, contrairement à GEO, SMD regroupe les données provenant d'un même auteur pour une seule étude. Ce critère est un peu plus restrictif que celui de GEO où les données sont regroupées aussitôt qu'elles proviennent de la même technologie. Lors de la récupération des données du site SMD, il faut faire attention de ne pas récupérer des données que nous aurions déjà trouvées sur le site GEO. Nous avons donc récupéré les données qui n'étaient pas dans la BD GEO et nous avons ensuite extrait les données correspondant à chacun des gènes et nous les avons ajouté à la suite des autres dans le fichier précédemment créé pour la BD GEO.

Au total, nous avons été en mesure de recueillir 5193 expériences provenant de différents tissus humains. Une expérience semblable a été effectuée par l'équipe d'Homin K. Lee en 2004 sur un total de 3924 données [66]. Malheureusement cette équipe n'a pas jugée pertinent de mettre leurs données disponibles à la communauté scientifique ce qui fait que nous avons dû refaire au complet le même processus qu'eux.

Voici un tableau montrant le nom des données utilisées ainsi que le nombre d'expériences dans chacune. À noter que nous avons recueillis beaucoup de données (6950 pour un total de 318 ensembles de données différents), mais pour cette analyse nous avons décidé d'utiliser les ensembles de 20 expériences ou plus.

Données	#	Données	#	Données	#	Données	#	Données	#
GDS337	20	GDS423	24	GDS144	30	GDS558	38	garber	73
GDS400	20	GDS424	24	GDS171	30	GDS183	40	GDS534	75
GDS504	20	GDS425	24	GDS604	30	GDS257	42	GDS84	84
GDS591	20	GDS426	24	GDS707	30	GDS601	42	whitfield	85
GDS703	20	GDS533	24	GDS714	30	GDS546	42	GDS181	85
GDS482	21	GDS478	24	GDS73	31	GDS532	44	sorlie	85
GDS612	21	GDS563	24	GDS184	31	GDS232	46	GDS715	87
GDS609	21	GDS614	24	GDS395	32	detweiler	53	GDS182	90
GDS605	22	GDS640	24	GDS470	32	GDS427	53	GDS619	91
GDS610	22	GDS586	24	GDS618	32	GDS234	54	rosenwald	102
GDS611	22	GDS690	24	GDS495	33	GDS711	57	GDS330	120
GDS686	22	GDS90	26	GDS74	35	GDS488	59	GDS217	123
GDS544	23	GDS52	27	GDS41	36	GDS724	62	GDS594	158
GDS564	23	GDS625	27	GDS172	36	GDS89	63	GDS596	158
GDS40	24	GDS10	28	GDS408	36	GDS539	63	GDS8	171
GDS241	24	GDS260	28	GDS449	36	GDS200	64	GDS9	171
GDS251	24	GDS391	28	GDS579	36	GDS204	66	GDS531	173
GDS268	24	GDS578	28	GDS641	36	GDS365	66	boldrick	182
GDS405	24	GDS214	29	GDS638	36	GDS75	67	chen	207
GDS360	24	GDS266	29	GDS639	36	GDS331	70		
GDS422	24	GDS267	29	GDS651	37	GDS541	73	Total	5193

Tableau 6 : Les données et le nombre d'expériences contenues dans chacune pour l'analyse effectuée.

Méthode

L'idée de base est de tester toutes les paires de gènes possibles (i.e. $n(n-1)/2$ où n est le nombre de gènes utilisés dans l'analyse. Dans notre étude, nous avons utilisé un n égal à 13944, représentant le nombre de gènes humains pour lesquels il existe au moins un ensemble de données le contenant dans les BD GEO et SMD.

La comparaison entre deux gènes est possible si et seulement si il existe au moins un ensemble de données dans lequel les deux gènes sont présents. Il arrive donc fréquemment qu'il soit possible de comparer les deux gènes en fonction de plusieurs ensembles de données. Le but de l'analyse est donc de fournir une liste triée de paires de gènes ayant une relation inhibiteur-inhibé forte au travers de tous les ensembles de données. Pour déterminer si une

paire de gènes possède une relation d'inhibition forte, nous avons développé une fonction utilisant le Positive and Negative Co-regulated Gene Cluster (PNCGC) décrit dans l'article de Liping et Kian-Lee [67].

Plusieurs équipes utilisent la corrélation de Pearson pour déterminer des relations entre les gènes [68][66]. Avec cette métrique, une valeur près de -1 ou de 1 signifie, respectivement, qu'il existe une corrélation négative forte ou positive forte entre l'expression des deux gènes à l'étude. Le problème de cette métrique, selon nous, est qu'elle suppose que la relation entre les valeurs est linéaire. En effet, la corrélation de Pearson calcule la distance des points à la droite passant au milieu des points [69].

Comme nous ne voulons pas poser de contrainte sur la forme que devrait avoir la relation entre nos gènes, nous nous devons d'utiliser un métrique qui ne fait aucune supposition à ce sujet. La métrique de Ji et collègues s'applique parfaitement à notre situation [67]. Supposons que le gène A soit présent dans le même ensemble de données que le gène B. Le gène A possède les données {a1, a2, a3, a4,..., am} et le gène B les données {b1, b2, b3, b4,..., bm} où m représente le nombre d'expériences contenues dans l'ensemble de données que les gènes ont en commun. Nous filtrons d'abord les données pour obtenir des couples pour lesquels les deux gènes ont une valeur présente. Cette règle a été ajoutée, car il arrive fréquemment que des données soient considérées manquantes (car trop bruitées) dans les données de biopuces. Nous obtenons donc deux nouveaux ensembles de données a'={a'1, a'2, a'3, ..., a'k} et b'={b'1, b'2, b'4, ..., b'k} où k <= m.

En utilisant ces k données, nous voulons savoir si une augmentation significative du gène A implique une diminution significative du gène B, ou à l'inverse si une diminution significative du gène A implique une augmentation significative du gène B. Pour y arriver, nous étudions les k(k-1)/2 couples de données possibles ((a'1,b'2),(a'1,b'3),(a'1,b'4),...(a'1,b'k)...(a'k-1,b'k)) et nous observons le comportement des deux gènes d'une donnée à l'autre. Pour chacun des couples, nous aurons deux valeurs pour le gène A et deux valeurs pour le gène B. Pour déterminer les augmentations/diminutions significatives d'une donnée d'un couple à l'autre, nous utilisons la fonction décrite à l'équation 14.

$$Sdij = \begin{cases} -1 \text{ si } \dfrac{di - dj}{|dj|} \leq -\dfrac{1}{3} \\ 1 \text{ si } \dfrac{di - dj}{|dj|} \geq \dfrac{1}{3} \\ 0 \text{ sinon} \end{cases}$$

Équation 14 : Fonction utilisée pour déterminer les augmentations/diminutions significatives.

Dans l'équation 14, d peut prendre la valeur a ou b et ij prendra les valeurs de toutes les combinaisons possibles des k valeurs ((1,2), (1,3), (1,4), …, (1,k), …, (k-1,k)) où i < j. En utilisant cette équation, nous obtenons k(k-1)/2 valeurs discrétisées pour $A(S_{aij})$ et la même chose pour $B(S_{bij})$. La valeur -1 signifie que le niveau d'expression du gène a augmenté significativement de i à j, la valeur 1 signifie que la valeur du gène a diminué significativement de i à j et la valeur 0 signifie qu'il n'y a pas eu de variation marquée de 1 à j.

L'étape suivante consiste à comptabiliser le nombre de cas positifs ((S_{aij} == -1 et S_{bij} == 1) ou (S_{aij} == 1 et S_{bij} == -1)) dans le k(k-1)/2 S_{dij}. Nous obtenons ainsi une valeur Pabx représentant le nombre de cas positifs pour la comparaison des gènes a et b dans l'ensemble de données x. Potentiellement, nous obtiendrons (13944*(13944 - 1) /2)*103 = 10 012 691 388 de ces valeurs. Si nous utilisions un octet pour représenter ce résultat, nous aurons besoin d'environ 9.5 gigaoctets de mémoire vive pour analyser ces résultats. Comme nous ne possédons pas une machine avec autant de capacité, nous avons développé un moyen de contourner ce problème.

Premièrement, nous devons associer à chacune de nos valeurs Pabx une valeur p(p-value). En utilisant des notions de probabilités de bases, nous sommes en mesure de déterminer une équation nous donnant la valeur p. Il suffit de voir le compte des cas positifs comme un cas de lancement de k(k-1) /2 dés à 9 faces. Supposons que notre k vaut 4. Nous observerons donc 4(3)/2=6 lancements de dés. Supposons que nous observons 2 résultats positifs. Ce 2 peut provenir de PNNPNN comme de PPNNNN. Nous voulons donc savoir quelle est la probabilité d'obtenir 2 P en 6 lancés de dés. Nous supposons que l'ordre n'a pas d'importance et que les lancements de dés sont des évènements indépendants entre eux.

La raison de l'utilisation du dé à neuf faces est que nous pouvons observer 9 cas (-1,-1), (-1,0), (-1,1), (0,-1), (0,0), (0,1), (1,-1), (1,0), (1,1) avec les valeurs Sdij. Les valeurs qui nous intéressent sont (1,-1) et (-1,1) (i.e. 2/9 des chances d'obtenir P et 7/9 des chances d'obtenir N). Revenons maintenant à l'observation de 2 positifs en 6 lancés. La probabilité de cette observation se calcule de la façon suivante :

$$\binom{6}{2}\frac{2}{9}^{2}\frac{7}{9}^{6-2} = \frac{6!}{(6-2)!2!}\frac{2}{9}^{2}\frac{7}{9}^{6-2} = 0.271074306$$

La formule générale pour calculer la valeur p pour un nombre de cas positifs cp et un k donné est exposée à l'équation 15 :

$$\frac{k!}{(k-cp)!cp!}\frac{2}{9}^{cp}\frac{7}{9}^{k-cp}$$

Équation 15 : Formule générale pour calculer la valeur p.

À l'aide de l'équation 15 nous sommes en mesure d'associer à chaque valeur Pabx une valeur p représentant la chance que nous avions de faire cette observation. Cependant, ceci ne diminue d'aucune façon l'espace requis pour faire l'analyse. Pour diminuer l'espace, nous allons utiliser des notions de statistique de l'ordre (order statistics). Pour se faire, nous allons traiter tous les cas de comparaisons d'un seul gène (A) avec tous les autres (B, C, D, ...). Ensuite, nous allons classer les valeurs p obtenues de ces comparaisons {(A,B),(A,C),(A,D),...} en ordre croissant de la moins probable à la plus probable. À noter, comme nous avons plusieurs ensembles de données, il sera possible de comparer 2 gènes dans plusieurs ensembles de données et ainsi obtenir plusieurs valeurs pour un couple (A,Y) où Y est élément de {A,B,C,...}. Pour un couple (A,Y) donné, nous allons conserver les 4 meilleurs rangs associés aux valeurs p dans la liste triée en ordre croissant. Nous allons faire de même pour tous les couples. Pour chacun des couples {(A,B),(A,C),(A,D),...} nous obtenons un ensemble de 4 rangs dans la liste triée. Pour chacun des ensembles, nous allons calculer la statistique d'ordre y étant associée. Ensuite, nous conservons les 100 gènes Y ayant les statistiques d'ordre les plus basses parmi tous les couples (A,Y). Ces 100 gènes correspondent aux 100 meilleurs candidats d'inhibition du gène A.

Voici comment calculer la statistique d'ordre. Premièrement, nous devons trier en ordre croissant les 4 valeurs p. Une fois que ceci est fait, nous calculons la probabilité que nous avions d'observer un tel ordonnancement à l'aide de l'équation 16 [68] :

$$P(p_1, p_2, p_3, ..., p_n) = \sum_{i=1}^{n} (p_{n-i+1} - p_{n-i}) P(p_1, .., p_{n-i}, p_{n-i+2}, p_n)$$

Équation 16 : Formule récursive pour le calcul de la statistique d'ordre.

Dans l'équation 16, p_0 vaut 0 et la dernière probabilité $P(p_1, .., p_{n-i}, p_{n-i+2}, p_4)$ signifie un appel récursif sans la valeur p_{n-i+1}. Dans notre cas, nous calculons pour $P(p_1,p_2,p_3,p_4)$. Cette probabilité nous donne une idée de la rareté de ce que nous observons. Par exemple, P(0.1,0.1,0.1,0.1) sera plus rare que P(0.5,0.5,0.5,0.5). Ce qui fait qu'un couple de gènes ayant une $P(p_1,p_2,p_3,p_4)$ très basse sera un candidat très intéressant pour une relation inhibiteur-inhibé significative. À noter que la relation que nous trouvons est commutative. Par exemple, si nous trouvons le couple A,B ayant une probabilité $P(p_1,p_2,p_3,p_4)$ basse, cela ne veut pas nécessairement dire qu'A inhibe B. Il peut aussi s'agir de B qui inhibe A.

Résultats

Le résultat que nous obtenons de cette analyse est une liste de paires de gènes triée selon la valeur de la probabilité P(p1,p2,p3,p4). Tout d'abord, le

tableau 7 montre les 50 relations inhibiteurs-inhibés pour lesquelles nous obtenons la valeur de probabilité la plus faible.

Gène 1	Gène 2	P(p1,p2,p3,p4)	Gène 1	Gène 2	P(p1,p2,p3,p4)
ADK	PTP4A3	0.000354976	IFNGR1	VEGFB	0.454411
ITGB3	PECAM1	0.00941972	MGLL	ZNF292	0.455297
IRF4	PECAM1	0.030402	GCSH	PTP4A3	0.458699
GATA6	GRB2	0.110761	CBFA2T2	IL6ST	0.459004
MYO1B	PECAM1	0.148741	GLA	ZNF292	0.461907
BCL2A1	TXNIP	0.172361	FOXO1A	TRIM11	0.463166
MBNL2	PIK3R2	0.26074	TNFRSF6	TRIM11	0.465716
FTHFSDC1	ZNF336	0.276509	SNX27	TNFRSF6	0.473532
LCAT	PEA15	0.293376	MBD2	RAP1A	0.474429
BCAT2	DTX1	0.300173	FCGRT	MYO1B	0.476571
ALDH9A1	PTP4A3	0.319297	ARHE	PPOX	0.478715
FTL	KCNA3	0.347534	NDUFB7	PELI1	0.479562
CACNA1I	GATA6	0.347851	IRF4	LGALS1	0.481351
PIK3R2	TNFRSF6	0.372545	DIO3	UBE2Q	0.482646
KCNA3	SULT1A3	0.380291	C9orf46	CX3CL1	0.483378
ADAM15	XPC	0.382716	AGPAT1	SERPINB8	0.488135
SNX1	TNFRSF6B	0.388717	CHES1	PDGFD	0.489502
ID2	TRIM11	0.396711	ADAM15	HADHB	0.491134
SLC27A6	UBE2Q	0.403319	ARHN	CCNH	0.492618
GCH1	PPP2CB	0.404288	FOSB	GMNN	0.493024
IQGAP1	MBNL3	0.413404	FGFR3	TRIM14	0.495253
SPINK5	TFF1	0.419314	IMPG1	NUMB	0.495913
HDAC11	TNFRSF6	0.429846	LCAT	TRIM11	0.496352
RGS1	TXNIP	0.445834	GCH1	PSMF1	0.496888
CYFIP2	TIMP1	0.453747	MARCKS	MTHFD1	0.498435

Tableau 7 : Les 50 paires inhibiteurs-inhibés les plus intéressantes retrouvées par cette analyse.

Nous avons testé quel pourcentage des 300 premières interactions contenait au moins un gène ayant une description de « negative regulation, regulation, inhibit ou transcription » dans la Gene Ontology [70] et 192 paires sur 300 (i.e. 64 % respecte ce critère). Nous retrouvons 64 % de ce que nous cherchons dans le top 300, ce qui est très intéressant. En effet, dans le 36 % qui reste, il y a plusieurs gènes pour lesquels il n'y a pas d'entrée dans l'ontologie. Nous pouvons donc conjecturer qu'il s'agit d'une relation d'inhibition.

La figure 40 montre le réseau obtenu en utilisant les 300 premières paires. Nous avons généré ce réseau en utilisant l'application GraphViz [71]. Pour permettre la visualisation de la haute connectivité du réseau, nous avons agrandi la partie centrale. La figure 41 montre cet agrandissement.

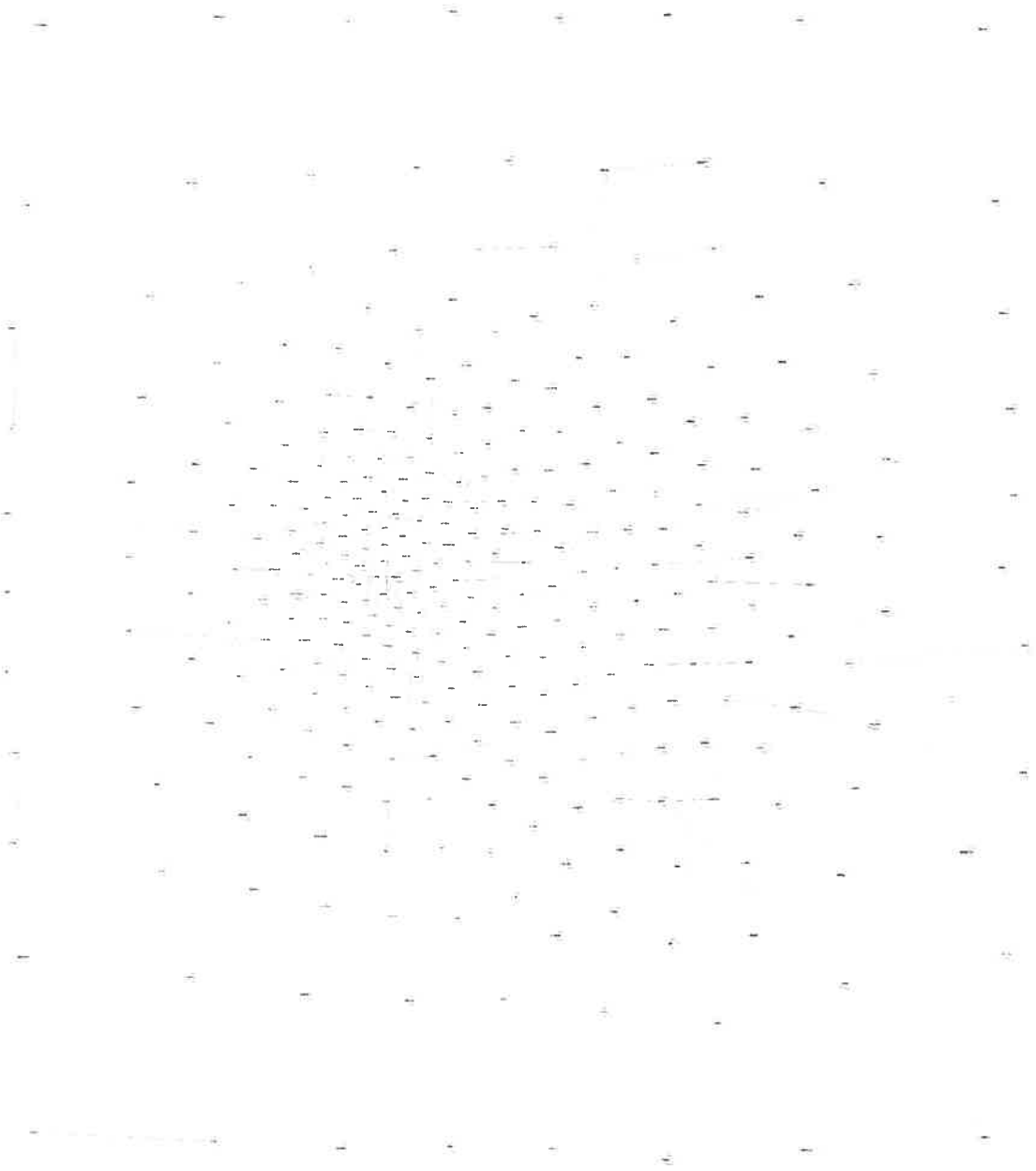


Figure 40 : Le réseau obtenu en utilisant les 300 paires les plus intéressantes.

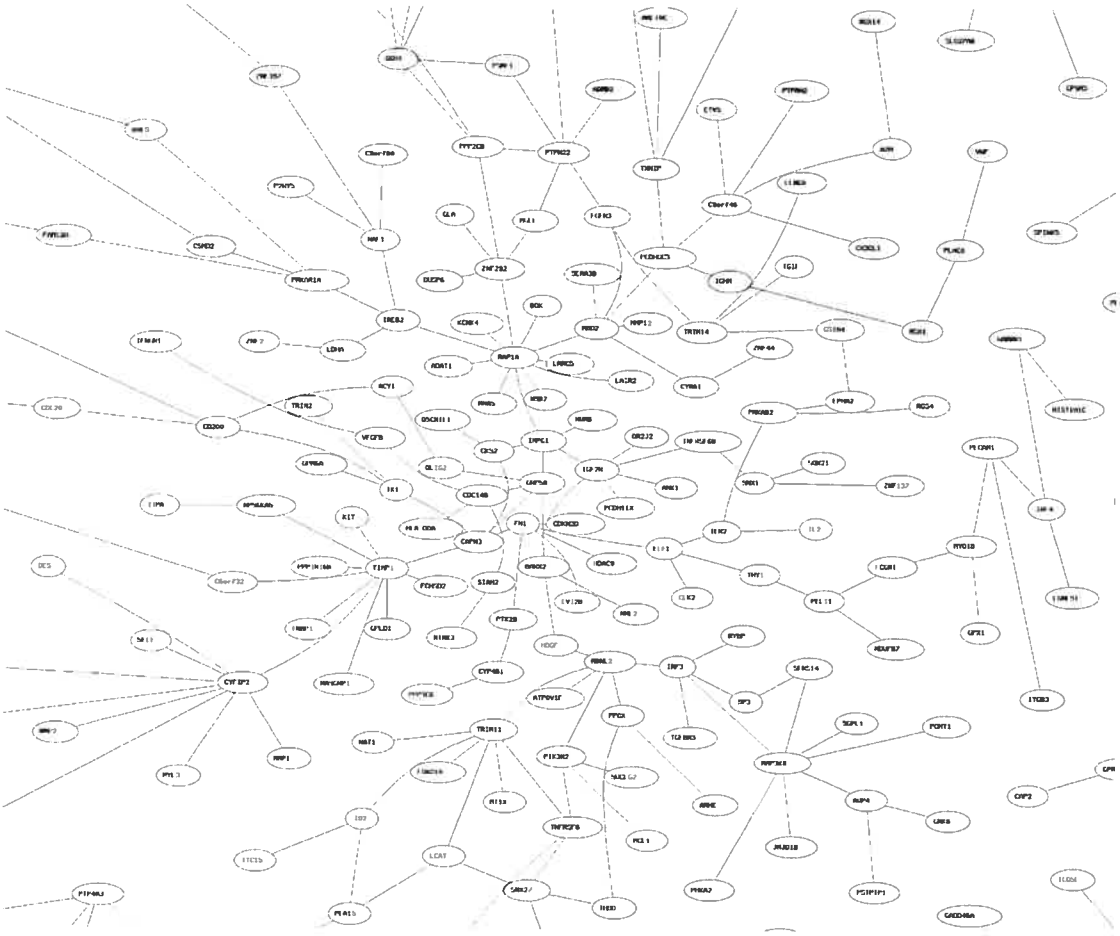


Figure 41 : Agrandissement de la figure 40 montrant à quel point le réseau est inter connecté.

Nous avons aussi fait le cumulatif du nombre de fois qu'un gène apparaît dans une paire. Nous obtenons donc une liste de gènes avec le nombre de fois que ce gène apparaît dans les paires. De cette façon, il est possible de cibler les gènes ayant un haut potentiel d'inhibition (i.e. ayant plusieurs cibles). Nous avons généré une liste à partir des 300 premières interactions et nous présentons dans le

tableau 8 le classement des 100 premiers gènes ayant comme attributs « negative regulation, regulation, inhibit ou transcription » dans la GO.

Il est très impressionnant de constater qu'en première position nous obtenons le gène RAP1A qui est responsable de la régulation négative du cycle cellulaire, donc un gène ayant été expérimentalement confirmé comme étant un inhibiteur du cycle cellulaire. En deuxième

position, nous obtenons le gène TIMP1 qui régule négativement la production de protéines membranaires. L'énumération des résultats intéressants pourrait être fait textuellement, mais rien ne vaut un coup d'œil au

tableau 8 pour constater à quel point la méthode d'analyse cible les gènes ayant un potentiel d'inhibition élevé. La figure 42 montre les gènes RAP1A, TIMP1, MBD2 et FOSB en relation avec leurs paires correspondantes dans le top 300.

Rang	Nom du gène	Description du gène	Gene Ontology
1	RAP1A	RAP1A, member of RAS oncogene family	negative regulation of cell cycle
2	TIMP1	Tissue inhibitor of metalloproteinase 1 (erythroid potentiating activity, collagenase inhibitor)	negative regulation of membrane protein ectodomain proteolysis
9	MBD2	Methyl-CpG binding domain protein 2	negative regulation of transcription
15	FOSB	FBJ murine osteosarcoma viral oncogene homolog B	negative regulation of transcription from Pol II promoter
19	IRF3	Interferon regulatory factor 3	transcription factor activity
22	BARX2	BarH-like homeobox 2	transcription factor activity
25	ELF1	E74-like factor 1 (ets domain transcription factor)	transcription factor activity
26	GATA6	GATA binding protein 6	transcription factor activity
27	IREB2	Iron-responsive element binding protein 2	negative regulation of translation
30	PIK3R2	Phosphoinositide-3-kinase, regulatory subunit 2 (p85 beta)	negative regulation of anti-apoptosis
32	PRKAR1A	Protein kinase, cAMP-dependent, regulatory, type I, alpha (tissue specific extinguisher 1)	regulation of transcription from Pol II promoter
37	TNFRSF6	Tumor necrosis factor receptor superfamily, member 6	regulation of apoptosis
43	CDK8	Cyclin-dependent kinase 8	regulation of transcription
44	CKS2	CDC28 protein kinase regulatory subunit 2	regulation of cyclin dependent protein kinase activity
48	IRF4	Interferon regulatory factor 4	positive regulation of interleukin-2,4,13,10
49	KLF12	Kruppel-like factor 12	regulation of transcription from Pol II promoter
51	MYB	V-myb myeloblastosis viral oncogene homolog (avian)	regulation of transcription
53	PEA15	Phosphoprotein enriched in astrocytes 15	negative regulation of glucose import
60	A2M	Alpha-2-macroglobulin	wide-spectrum protease inhibitor activity
66	BCL2A1	BCL2-related protein A1	regulation of apoptosis
67	BIRC2	Baculoviral IAP repeat-containing 2	regulation of apoptosis

Rang	Nom du gène	Description du gène	Gene Ontology
71	CBFA2T2	Core-binding factor, runt domain, alpha subunit 2; translocated to, 2	transcription factor activity
72	CCNG1	Cyclin G1	regulation of cyclin dependent protein kinase activity
73	CCNH	Cyclin H	regulation of cyclin dependent protein kinase activity
75	CDC20	CDC20 cell division cycle 20 homolog (S. cerevisiae)	regulation of cell cycle
76	CHES1	Checkpoint suppressor 1	transcription factor activity
77	CRK	V-crk sarcoma virus CT10 oncogene homolog (avian)	regulation of transcription from Pol II promoter
81	CYR61	Cysteine-rich, angiogenic inducer, 61	regulation of cell growth
85	FADS1	Fatty acid desaturase 1	regulation of cell differentiation and transcription
89	GABBR1	Gamma-aminobutyric acid (GABA) B receptor, 1	negative regulation of adenylate cyclase activity
90	GAS6	Growth arrest-specific 6	regulation of cell growth
94	HDAC11	Histone deacetylase 11	regulation of transcription, DNA-dependent
97	IQGAP1	IQ motif containing GTPase activating protein 1	GTPase inhibitor activity

Tableau 8 : Les gènes les plus connectés triés en ordre de connectivité.

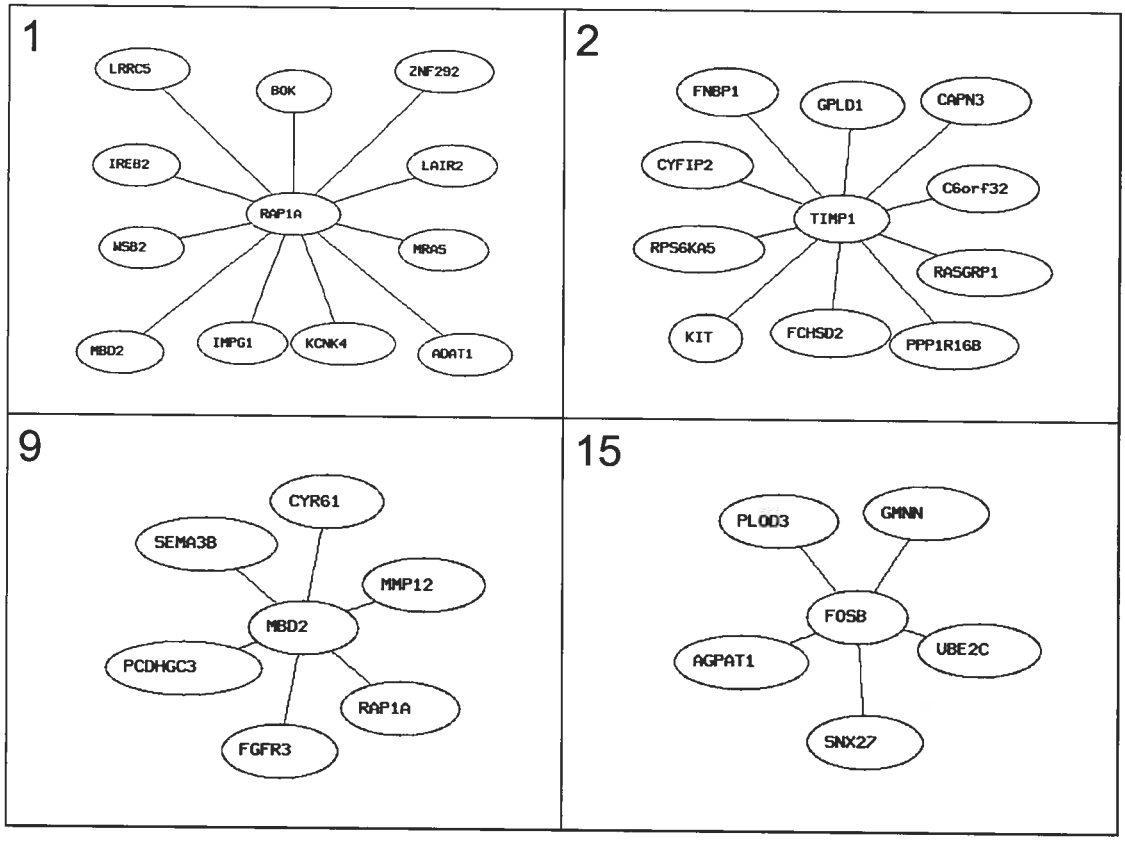


Figure 42 : Les 4 gènes les plus connectés avec leur position respective dans le tableau 8.

Conclusion

Les résultats obtenus lors de cette analyse montrent clairement que la méthode utilisée réussit à extraire des relations inhibiteurs-inhibés au niveau du génome humain à partir de données de biopuces prises sur des bases de données publiques. Maintenant que nous disposons d'une liste de candidats intéressants, nous pourrons vérifier si certaines des inhibitions pourraient être expliquée par un mécanisme de microARN. Nous pourrions aussi faire vérifier les paires se situant dans le haut de la liste par un laboratoire pour confirmer ou infirmer les résultats. Somme toute, cette méthode d'analyse semble prometteuse et nous songeons l'appliquer à d'autres cas.

Dans le futur, nous pensons appliquer la même méthode pour trouver des relations d'activation entre les gènes. À cette fin, nous devrions modifier la fonction PNCGC qui devrait compter les cas positifs (1,1) au lieu de (-1,1) et (1,-1). Il faudrait aussi changer 2/9 pour 1/9 et 7/9 pour 8/9 dans le calcul de la valeur p. Nous pourrions aussi, comme l'a fait le

groupe de Stuart Kim, croiser des données d'expressions de plusieurs espèces pour faire ressortir des relations d'inhibtion/activation conservées lors de l'évolution [68].

CHAPITRE 6

Conclusion

Le but premier de ce mémoire était l'analyse des données de biopuces. Nous y avons consacré pratiquement la totalité du document à l'exception du dernier chapitre qui traite de la problématique du placement discret. Les données ont été analysées à différents niveaux à l'aide de différents outils et nous avons démontré qu'il est possible d'extraire de l'information pertinente de ces données bruitées.

Retour sur les chapitres et perspectives

Le premier chapitre présente une mise en contexte du domaine biologique à l'étude dans ce mémoire, une description de la technologie des biopuces et une introduction aux méthodes d'apprentissage de la structure optimale d'un réseau bayesien à partir de données.

Le deuxième chapitre présentait les éléments cruciaux d'une implantation efficace de la méthode MCMC pour la recherche de structures de réseaux bayesiens. Nous avons montré qu'en utilisant les vecteurs de bits pour représenter les matrices d'ancêtres et d'incidence, un tri topologique en ligne au lieu d'un tri topologique standard, un critère à vérifier avant l'opération de retournement pour éviter les retournements inutiles et un calcul du nombre de réseaux acycliques accessibles efficace, nous obtenions une accélération significative de la méthode. L'efficacité a été démontrée à l'aide de deux tests différents, l'un testant la génération d'un nombre variable de réseaux aléatoires en fixant la taille des réseaux à 70 et l'autre fixant le nombre de réseaux aléatoires à générer à 100 et faisant varier la taille des réseaux. Dans les deux cas, la méthode utilisant toutes les améliorations obtenait des temps d'exécution nettement plus rapide que la méthode référence de Giudici et Castelo [43].

Le troisième chapitre utilisait la méthode MCMC efficace développée au premier chapitre pour vérifier l'impact de l'ajout de connaissances a priori au sujet des structures sur la précision des résultats obtenus. Nous avons développé une méthodologie utilisant un générateur de réseaux aléatoires représentant la réalité biologique ainsi qu'un simulateur de données de biopuces pour estimer l'apport des connaissances. Nous avons montré que l'utilisation de statistiques simples comme le nombre de noeuds sans parents, le nombre de parents moyens, le nombre de noeuds sans enfants et le nombre d'enfants moyens n'améliore

pas significativement la précision de la méthode MCMC. Nous avons cependant conjecturé que les statistiques que nous avons utilisées ne se prêtaient pas à la méthode de recherche utilisée. Les perspectives de ce projet seraient donc dans l'élaboration de méthodes pouvant s'adapter à l'état actuel du réseau. Bien évidemment, plusieurs autres métriques ainsi que plusieurs autres sources d'information additionnelles pourraient aussi être ajoutées pour tenter d'améliorer la méthode utilisant les réseaux bayesiens.

Le quatrième chapitre présentait plusieurs techniques pour la découverte de gènes différemment exprimés entre deux jeux de données différents de biopuces. Nous avons présenté le test t, la méthode SAM et la méthode bayesienne empirique. Nous avons justifié le choix de la méthode bayesienne empirique par ses bonnes performances démontrées dans de nombreux articles. Nous avons aussi survolé les techniques d'ajustement des données et nous avons choisi la méthode RMA. Ici encore, nous justifions les choix de cette technique par ses bonnes performances dans de nombreux papiers. Nous avons ensuite appliqué la méthode bayesienne empirique et RMA à l'analyse de 5 triplicatas différents pris à différents stades chez la souris par l'équipe du Dr Hoang. Nous avons présenté les résultats obtenus lors de ces analyses en utilisant des graphiques volcans. Nous avons fait ressortir certains gènes intéressants différemment exprimés dans l'une ou l'autre des analyses. Il est question des gènes LMO1, LYL1, CD28, Ywhaz et Usp7. En travaillant en collaboration avec des chercheurs d'autres domaines, comme dans ce projet, il est plus facile de voir les outils qu'ils seraient intéressants de développer pour faciliter le travail de ceux-ci. Par exemple, nous avons constaté, qu'il serait intéressant de développer un outil permettant de faire, facilement et intuitivement, des opérations comme l'union, l'intersection et la différence entre des listes de gènes provenant de différentes analyses. Cet outil pourrait être l'objet de développements futurs.

Le cinquième chapitre présentait une analyse au niveau du génome de l'humain pour retrouver des paires inhibiteurs-inhibés présentes de façon significative dans les données de biopuces disponible dans les BD GEO et SMD. Nous avons mis au point une métrique permettant d'extraire les relations intéressantes. Cette métrique est basée sur la méthode PNCGC décrite dans l'article de Liping et Kian-Lee [67]. Nous utilisons aussi les statistiques de l'ordre pour associer un rang aux relations que nous retrouvons. Les résultats que nous obtenons sont très intéressants et font ressortir des gènes que nous connaissons déjà comme étant des gènes inhibiteurs importants. Par exemple, la méthode fait ressortir les gènes

RAP1A et TIMP1 en tête de la liste des gènes inhibant le plus. Comme mentionné à la fin du chapitre 4, il serait intéressant d'appliquer la même méthode pour la découverte de pairs activateur-activés.

Le sixième chapitre ne traite pas de l'analyse des biopuces, mais plutôt d'une méthode permettant de déterminer le placement optimal de pièces pouvant être discrétisées sur une surface de travail. Nous avons présentés un algorithme générique pouvant traiter toutes sortes de pièces ainsi que toutes sortes de surfaces. L'algorithme produit en sortie un système de programmation linéaire pouvant être résolu en utilisant le logiciel commercial CPLEX. Nous avons présentés plusieurs résultats d'application de cette méthode à plusieurs problèmes jouets pour démontrer le fonctionnement de la méthode. La perspective pour le futur serait d'appliquer cette méthode pour résoudre des problèmes d'industrie concrets.

Annexe A

Preuve que le nombre de liens maximum d'un réseau est de *n(n-1)/2*

Rappelons tout d'abord qu'il est possible de trier en ordre topologique les nœuds d'un graphe dirigé acyclique. Supposons donc un ordre quelconque qui associe, pour simplifier, le nœud 1 à la position 1, le nœud 2 à la position 2, etc. Selon la définition du tri topologique, un nœud ne peut pas avoir de descendants se retrouvant avant lui dans l'ordre topologique. Suite à cette définition, on peut voir que le nombre de liens au maximum partant du nœud 1 est de n-1, représentant le cas où le nœud 1 pointe sur tous les nœuds qui le suivent dans l'ordre topologique. Si nous appliquons cette construction à tous les autres nœuds, nous obtenons un graphe ayant l'allure suivante :

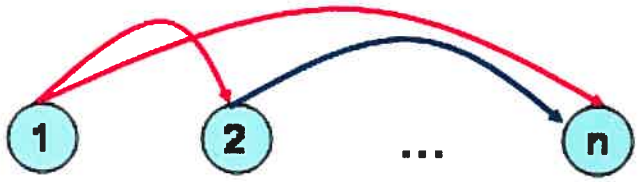


Figure 43 : Tri topologique des nœuds montrant la façon d'obtenir un nombre de liens maximal.

En résumé, nous obtenons *(n-1) + (n-2) + (n-3) + ... + 1* liens maximum, soit :

$$\sum_{i=n}^{2}(i-1)=\frac{n(n-1)}{2}\blacksquare$$

Preuve de la complexité du calcul de retournement valide en temps $\frac{n^2(n-1)}{128}+\frac{n(n-1)}{16}$

Il a été montré au chapitre 2 que le nombre d'opérations nécessaires pour calculer le nombre de retournements possibles pour un nœud ayant n-1 parents était de (n/64*n + n/16). On peut généraliser et dire que si le nœud possède m parents, alors l'opération pour compter le nombre de retournements possibles pour ce nœud prendra un temps de *(n/64*m + n/16)*.

Comme mentionné à la section précédente, le nombre de liens maximal est de $\sum_{i=n}^{2}(i-1)=\frac{n(n-1)}{2}$. Si nous regardons le réseau ainsi généré du point de vue des parents des nœuds et que nous commençons par le nœud à la fin du tri topologique, ce réseau n'aura aucun enfant, mais il aura *n-1* parents. Le nœud le précédent dans le tri aura *n-2* parents et ainsi de suite pour terminer par le premier nœud qui aura *0* parent. À l'aide de l'équation (n/64*m + n/16) et de la constatation que nous venons de faire, nous pouvons dire que la complexité totale en terme de nombre d'opération pour le calcul du nombre de retournements valides pour tous les nœuds prend un temps de $\sum_{i=1}^{n-1}\left(\frac{ni}{64}+\frac{n}{16}\right)$. En simplifiant quelque peu nous obtenons :

$$\begin{aligned}
&\sum_{i=1}^{n-1}\left(\frac{ni}{64}+\frac{n}{16}\right)\\
&=\sum_{i=1}^{n-1}\frac{ni}{64}+\sum_{i=1}^{n-1}\frac{n}{16}\\
&=\frac{n}{64}\sum_{i=1}^{n-1}i+\frac{n(n-1)}{16}\\
&=\frac{n^2(n-1)}{128}+\frac{n(n-1)}{16}
\end{aligned}$$

■

Complexité totale du calcul du nombre de réseaux acycliques accessibles

Maintenant que nous connaissons le coût de l'opération la plus complexe, nous pouvons ajouter le calcul pour l'addition et celui pour la soustraction. Comme mentionné au chapitre 2, l'addition prendra, pour un nœud donné, un temps de n/64 + n/16 + 1. Si nous appliquons cette opération à tous les nœuds du réseaux, nous devons multiplier par *n* pour obtenir $n^2/64 + n^2/16 + n$. La soustraction étant seulement l'addition du nombre de liens courants présents dans le réseau, elle prend donc une temps constant, soit dans *O(1)*. Si nous regroupons les trois termes, soit l'addition, la soustraction et le retournement, nous obtenons

un temps d'exécution de $\frac{n^2}{64}+\frac{n^2}{16}+n+\frac{n^2(n-1)}{128}+\frac{n(n-1)}{16}+1$ ou, en appliquant quelques simplifications, un temps de $\frac{n^3}{128}+\frac{17n^2}{128}+\frac{15n}{16}+1$ qui est dans l'ordre $O(n^3)$.

Bibliographie

[1] http://en.wikipedia.org/wiki/Cellular_differentiation#Mammalian_cell_types.

[2] Watson, JD et Crick, FHC, Molecular structure of Nucleic Acids. Nature 171, 737-738 (1953).

[3] Jean-Michel Claverie, Gene Number. What If There Are Only 30,000 Human Genes? Science 291, 1255–1257 (2001).

[4] Elizabeth Pennisi, A Low Number Wins the GeneSweep Pool. Science 300, 1484 (2003).

[5] Bejerano G, Haussler D, Blanchette M., Into the heart of darkness: large-scale clustering of human non-coding DNA. Bioinformatics Aug 4;20 Suppl 1, I40-I48. (2004).

[6] Thomas A Down and Tim JP Hubbard, What can we learn from noncoding regions of similarity between genomes? BMC Bioinformatics. Sep 15;5(1):131 (2004).

[7] Konarska MM, Recognition of the 5' splice site by the spliceosome. Acta Biochim Pol 45(4):869-81 (1998).

[8] Lodish, Baltimore, Berk, Zipursky, Matsudaira, Darnell, Biologie Moléculaire De La Cellule.

[9] Benjamin P. Lewis, I-hung Shih, Matthew W. Jones-Rhoades, David P. Bartel, and Christopher B., Burge Prediction of Mammalian MicroRNA Targets. Cell 115: 787-798 (2003).

[10] David P. Bartel, MicroRNAs: Genomics, Biogenesis, Mechanism, and Function. Cell 116: 281-297 (2004).

[11] www.affymetrix.com

[12] www.gene-chips.com

[13] YH Yang, MJ Buckley, S Dudoit, TP Speed, Comparison of methods for image analysis on cDNA microarray data. Journal of Computational and Graphic Statistics 11:108-136 (2002).

[14] Hubbell, Liu et Mei, Robust estimators for expression analysis. Bioinformatics 18:1585-1592 (2002).

[15] Affymetrix technical report, New statistical algorithms for monitoring gene expression on GeneChip probe arrays, Affymetrix:1-5.

[16] Spellman PT, Sherlock G, Zhang MQ, Iyer VR, Anders K, Eisen MB, Brown PO, Botstein D, Futcher B, Comprehensive Identification of Cell Cycle-regulated Genes of the Yeast Saccharomyces cerevisiae by Microarray Hybridization. Molecular biology of the cell Vol. 9, Issue 12, 3273-3297 (1998).

[17] Gollub J, Ball CA, Binkley G, Demeter J, Finkelstein DB, Hebert JM, Hernandez-Boussard T, Jin H, Kaloper M, Matese JC, Schroeder M, Brown PO, Botstein D, Sherlock G, The Stanford Microarray Database: data access and quality assessment tools. Nucleic Acids Res Vol. 31, No. 1 94-96 (2003).

[18] http://davidmlane.com/hyperstat/A34739.html

[19] http://mathworld.wolfram.com/SpearmanRankCorrelationCoefficient.html

[20] Jean-Philippe Vert and Minoru Kanehisa, Graph-driven features extraction from microarray data. arXiv:physics/0206055 v1 17 Jun (2002).

[21] Alvaro Mateos, Joaquín Dopazo, Ronald Jansen, Yuhai Tu, Mark Gerstein, and Gustavo Stolovitzky, Systematic Learning of Gene Functional Classes From DNA Array Expression Data by Using Multilayer Perceptrons. Genome Research 12: 1703-1715 (2002).

[22] James Lyons-Weiler1,2, Satish Patel and Soumyaroop Bhattacharya, A Classification-Based Machine Learning Approach for the Analysis of Genome-Wide Expression Data. Genome Research Vol 13, Issue 3, 503-512 (2003).

[23] Michael B. Eisen, Paul T. Spellman, Patrick O. Browndagger, and David Botstein., Cluster analysis and display of genome-wide expression patterns. PNAS Vol. 95,14863–14868 (1998).

[24] Shoudan Liang, Stefanie Fuhrman Et Roland Somogyi, Reveal, A General Reverse Engineering Algorithm for Inference Of Genetic Network Architectures. Pac Symp Biocomput. 18-29 (1998).

[25] Friedman N., Linial M., Nachman I. et Pe'er D., Using bayesian networks to analyze expression data. RECOMB (2000).

[26] Heckerman D., Geiger D. et Chickering D.M., Learning Bayesian Networks: The Combination of Knowledge and Statistical Data. Technical report Microsoft 197-243 (1995).

[27] Michiel J.L. De Hoon, Seiya Imoto, Kazuo Kobayashi,Naotake Ogasawara Et Satoru Miyano, Inferring Gene Regulatory Networks From time-Ordered Gene Expression Data Of Bacillus subtilis Using Differential Equations. Pacific Symposium on Biocomputing 17-28 (2003).

[28] M.K. Stephen Yeung, Jesper Tegnér et James J.Collins, Reverse engineering gene networks using singular value decomposition and robust regression. PNAS 99(9), 6163-6168 (2002).

[29] Guillaume Bourque, David Sankoff, A comparative approach for multiple gene network inference using time-series gene expression data http://www.ima.umn.edu/complex/abstracts/bourque/Bourque_IMAposter.pdf

[30] http://marketing.byu.edu/htmlpages/books/pcmds/REGRESS.html

[31] Richard E. Neapolitan, Learning Bayesian Networks, Pearson (2004).

[32] Nir Friedman, Inferring Cellular Networks Using Probabilistic Graphical Models. Science vol. 13,799-805 (2004).

[33] David Heckerman, A Tutorial on Learning With Bayesian Networks. (1995).

[34] Cooper and Herskovits, A Bayesian method for the induction of probabilistic networks from data. Machine Learning 9:309-347 (1992).

[35] Jeong, S. P. Mason, A.-L. Barabási, Z. N. Oltvai, Lethality and centrality in protein networks. Nature 411:41-42 (2001).

[36] Nir Friedman, The Bayesian Structural EM Algorithm. Fourteenth Conf. on Uncertainty in Artificial Intelligence (UAI) (1998).

[37] Kevin Murphy, Dynamic Bayesian Networks : Representation, Inference and Learning. PhD Thesis (2002).

[38] Nir Friedman, Data Analysis with Bayesian Networks: A Bootstrap Approach. Proc. Fifteenth Conf. on Uncertainty in Artificial Intelligence (UAI) (1999).

[39] D.M Chickering, Learning Bayesian networks is NPcomplete. In Fisher,D. and Lenz,H.J. (eds), Learning from Data. Artificial Intelligence and Statistics Vol. 5, Springer, New York pp. 121–130 (1996).

[40] Dirk Husmeier, Sensitivity and specificity of inferring genetic regulatory interactions from microarray experiments with dynamic Bayesian networks. Bioinformatics Vol. 19 no. 17, pages 2271–2282 (2003).

[41] W.K. Hastings, Monte Carlo sampling methods using Markov chains and their applications. Biometrika 57, 97–109 (1970).

[42] http://mathworld.wolfram.com/GammaFunction.html

[43] Paolo Giudici, Robert Castelo, Improving Markov Chain Monte Carlo Model Search for Data Mining. Machine Learning, 50, 127–158 (2003).

[44] Kevin Murphy, The Bayes Net Toolbox for Matlab. Computing Science and Statistics, vol 33 (2001).

[45] David J. Pearce, Paul H. J. Kelly, Online algorithms for maintaining the topological order of a directed acyclic graph. Tech. report, Imperial College of Science, Technology, and Medicine, Department of Computing, Jul (2003).

[46] Jie Li, Yi Pan, Hong Shen, More Efficient Topological Sort Using Reconfigurable Optical Buses. The Journal of Supercomputing 24, 251–258 (2003).

[47] David J. Pearce and Paul H. J. Kelly, Online algorithms for maintaining the topological order of a directed acyclic graph. Technical Report, Department of Computing, Imperial College, London (2003).

[48] Wei Pan, A comparative review of statistical methods for discovering differentially expressed genes in replicated microarray experiments. Bioinformatics Vol. 18, no. 4, 546-554 (2002).

[49] B.L. Welch, The generalization of 'students' problem when several different population variance are involved. Biometrika, 34, 28-35 (1954).

[50] Tusher, Tibshirani et Chu, Significance analysis of microarrays applied to the ionizing radiation response. PNAS 98: 5116-5121, (Apr 24) (2001).

[51] Per Broberg, Statistical methods for ranking differentially expressed genes. Genome Biology, 4:R41 (2003).

[52] Holger Schwender, Andreas Krause et Katja Ickstadt, Comparison of empirical bayes and significance analysis of miroarrays. (2003). http://www.sfb475.uni-dortmund.de/berichte/tr44-03.pdf

[53] Bolstad, B.M., Irizarry R. A., Astrand, M., and Speed, T.P., A Comparison of Normalization Methods for High Density Oligonucleotide Array Data Based on Bias and Variance. Bioinformatics 19(2):185-193 (2003).

[54] Cope LM, Irizarry RA, Jaffee HA, Wu Z, Speed T.P., A benchmark for Affymetrix GeneChip expression measures. Bioinformatics Vol 20, No 3, 323-331 (2004).

[55] Rafael. A. Irizarry, Benjamin M. Bolstad, Francois Collin, Leslie M. Cope, Bridget Hobbs and Terence P. Speed, Summaries of Affymetrix GeneChip probe level data. Nucleic Acids Research 31(4):e15 (2003).

[56] Yoseph Barash, Elinor Dehan, Meir Krupsky, Wilbur Franklin, Marc Geraci, Nir Friedman and Naftali Kaminski, Comparative analysis of algorithms for signal quantitation from oligonucleotide microarrays. Bioinformatics 20(6):839-46 (2004).

[57] Irizarry RA, Hobbs B, Collin F, Beazer-Barclay YD, Antonellis KJ, Scherf U, Speed TP, Exploration, Normalization, and Summaries of High Density Oligonucleotide Array Probe Level Data. Biostatistics Apr;4(2):249-64 (2003).

[58] Gentleman RC, Carey VJ, Bates DM, Bolstad B, Dettling M, Dudoit S, Ellis B, Gautier L, Ge Y, Gentry J, Hornik K, Hothorn T, Huber W, Iacus S, Irizarry R, Leisch F, Li C, Maechler M, Rossini AJ, Sawitzki G, Smith C, Smyth G, Tierney L, Yang JY, Zhang J, Bioconductor: Open software development for computational biology and bioinformatics. Genome Biology 5,R80 (2004).

[59] http://www.affymetrix.com/products/arrays/specific/mouse430.affx

[60] Gordon K. Smyth, Linear Models and Empirical Bayes Methods for Assessing Differential Expression in Microarray Experiments. Statistical Applications in Genetics and Molecular Biology 3 No. 1, Article 3 (2004).

[61] HUGO Gene Nomenclature Committee, http://www.gene.ucl.ac.uk/nomenclature/

[62] Wain HM, Lush MJ, Ducluzeau F, Khodiyar VK, Povey S., Genew: the Human Gene Nomenclature. Database, 2004 updates. Nucleic Acids Res. (2004).

[63] Wain HM, Lovering RC, Bruford EA, Lush MJ, Wright MW, Povey S., Guidelines for Human Gene Nomenclature. Genomics Vol. 79, No. 4, 464-470 (2002).

[64] Barrett T, Suzek TO, Troup DB, Wilhite SE, Ngau WC, Ledoux P, Rudnev D, Lash AE, Fujibuchi W, Edgar R, NCBI GEO: mining millions of expression profiles—database and tools. Nucleic Acids Research, Vol. 33, Database issue D562-D566 (2005).

[65] Ball CA, Awad IA, Demeter J, Gollub J, Hebert JM, Hernandez-Boussard T, Jin H, Matese JC, Nitzberg M, Wymore F, Zachariah ZK, Brown PO, Sherlock G., The Stanford Microarray Database accommodates additional microarray platforms and data formats. Nucleic Acids Res Jan 1;33(1):D580-2 (2005).

[66] Homin K. Lee, Amy K. Hsu, Jon Sajdak, Jie Qin et Paul Pavlidis, Coexpression Analysis of Human Genes Across Many Microarray Data Sets. Genome Research 14,1085-1094 (2004).

[67] Liping Ji et Kian-Lee Tan, Mining gene expression data for positive and negative co-regulated gene clusters. Bioinformatics vol 20, no 16, 2711-2718 (2004).

[68] Joshua M. Stuart, Eran Segal, Daphne Koller et Stuart K. Kim, A Gene-Coexpression Network fo Global Discovery of Conserved Genetic Modules. Science vol 302, 249-255 (2003).

[69] http://davidmlane.com/hyperstat/A34739.html

[70] Ashburner M, Ball CA, Blake JA, Botstein D, Butler H, Cherry JM, Davis AP, Dolinski K, Dwight SS, Eppig JT, Harris MA, Hill DP, Issel-Tarver L, Kasarskis A, Lewis S, Matese JC, Richardson JE, Ringwald M, Rubin GM, Sherlock G., The Gene Ontology Consortium. Gene Ontology: tool for the unification of biology. Nat Genet 25: 25-29 (2000).

[71] www.graphviz.org

[72] http://www.math.uah.edu/stat/sample/sample6.xhtml

[73] http://www.mat.ulaval.ca/pages/aqjm/

[74] E. Segal, M. Shapira, A. Regev, D. Pe'er, D. Botstein, D. Koller, and N. Friedman, Module Networks: Identifying Regulatory Modules and their Condition Specific Regulators from Gene Expression Data. Nature Genetics June, 34(2): 166-76 (2003).

[75] Bernard, A. & Hartemink, A., Informative Structure Priors: Joint Learning of Dynamic Regulatory Networks from Multiple Types of Data. In Pacific Symposium on Biocomputing (2005).

[76] A. Hartemink, Principled Computational Methods for the Validation and Discovery of Genetic Regulatory Networks. Massachusetts Institute of Technology Ph. D. dissertation