Adaptation d'un algorithme génétique pour la reconstruction de réseaux de régulation génétique : COGARE.

Julien Briche

THÈSE

présentée à

L'UNIVERSITÉ DU SUD TOULON-VAR

PAR
Julien BRICHE

Pour l'obtention du grade de :
DOCTEUR de l'UNIVERSITÉ DU SUD TOULON-VAR
et de la FACULTÉ DES SCIENCES ET TECHNIQUES, UNIVERSIDAD DE CHILE

SPÉCIALITÉ : Bioinformatique, Biologie structurale et Génomique
OPTION : Outils, méthodes et modèles

Adaptation d'un algorithme génétique pour la reconstruction de réseaux de régulation génétique : COGARE.

Soutenue le 9 Septembre 2009 à l'Université du Sud Toulon-Var.

J.F. BOULICAUT :	PU. Responsable de l'équipe Turing au LIRIS, INSA Lyon	Rapporteur
J.M. CLAVERIE :	PU-PH. Directeur du laboratoire IGS, Marseille	Examinateur
J. DEMONGEOT :	PU. Directeur du laboratoire TIMC-IMAG, Grenoble	Rapporteur
Y. LACROIX :	PU. Directeur du laboratoire SNC, Toulon	Directeur
D. LEANDRI :	PU. Co-directeur du laboratoire SNC, Toulon	Examinateur
A. MAASS :	PU. Directeur du LBMG, Santiago de Chile	Co-Directeur

Laboratoire SNC
ISITV-Bat.X
Avenue POMPIDOU-B.P. 56
83162 LA VALETTE CEDEX-FRANCE
http ://www.labo-snc.com/

Laboratoire CMM
UMI CNRS 2807, Piso 7,
Av. B.ENCALADA 2120
Santiago de Chile-CHILE
http ://www.cmm.uchile.cl/

À toi et moi.

La science a-t-elle promis le bonheur ? Je ne le crois pas. Elle a promis la vérité, et la question est de savoir si l'on fera jamais du bonheur avec de la vérité.

Émile Zola

Remerciements

En premier lieu, je voudrais remercier Yves Lacroix et Alejandro Maass, mes co-directeurs de thèse, pour la confiance qu'ils m'ont témoignée depuis le début de cette thèse, pour m'avoir poussé dans cette voie et pour m'avoir donné les rênes et tous les moyens nécessaires à l'accomplissement de ce projet.
Messieurs les Professeurs Demongeot et Boulicaut ont accepté d'être les rapporteurs de cette thèse, et je les remercie de l'honneur qu'ils me font. Ils ont également contribué par leurs nombreuses remarques et suggestions à améliorer la qualité de ce mémoire, et je leur en suis très reconnaissant.
Messieurs les Professeurs Claverie et Léandri, en tant qu'examinateurs, m'ont fait l'honneur de participer au Jury de soutenance ; je les en remercie. Je tiens à remercier les membres du laboratoire CMM à Santiago du Chili, et tout particulièrement le LBMG (Laboratorio de Bioinformatica et Matematica del Genoma), Andrés Aravena, Angélica Reyes, Rodrigo Assar, Pablo Moreno, Juan Ugalde et Sebastian Gálvez pour leur accueil chaleureux et leur disponibilité.
Je tiens aussi ici à exprimer ma gratitude envers l'ensemble des membres du laboratoire SNC qui m'ont suivi, aidé et accompagné.
L'université du Sud Toulon-Var m'a accueilli et permis de conduire ma thèse dans les meilleures conditions possibles, je tiens à exprimer ma gratitude à l'ensemble de ses services.
Je tiens également à exprimer un grand merci à tous mes amis qui m'ont soutenu et supporté, Jérôme, Rémi, Pierre-Guillaume, Olivier et Emilie, Karin et Pierre.
Enfin, je tiens à remercier ma famille, qui a toujours cru en moi, et en particulier ma compagne, Isabelle Toulgoat, qui a toujours été à mes côtés, m'a soutenu et m'a permis d'aller au bout de cette thèse. J'espère que je pourrais en faire autant pour elle.

Table des matières

Introduction

Depuis sa découverte en 1953, l'ADN n'a cessé de surprendre et d'intriguer les chercheurs. La découverte du fait que l'ADN porte les instructions qui permettent à tout organisme de fonctionner a été un pas gigantesque dans la compréhension du vivant. Ces instructions sont écrites avec les gènes, que l'on peut assimiler à un vocabulaire. Ce vocabulaire est ensuite analysé grâce à des mécanismes de traduction et de transcription pour être compris par l'organisme. Une fois ces mécanismes connus, le problème restant à résoudre est de comprendre le fonctionnement de ce langage, comprendre comment et pourquoi les mots s'agencent et signifient. Ce fonctionnement révèle comment les gènes agissent les uns sur les autres, et donc traduit les interactions existantes entre eux. Le réseau génétique est l'ensemble des interactions existantes entre les gènes, il représente donc la structure fonctionnelle d'un organisme.
La recherche des réseaux génétiques s'est longtemps effectuée à tâtons et ce, à cause du manque de données et de moyens pour les traiter. Les avancées dans les domaines de la biologie et de l'informatique ont entraîné la création d'un nouveau domaine de recherche combinant les deux disciplines : la bioinformatique. La reconstruction de réseau génétique, qui est la recherche de la structure fonctionnelle d'un organisme, s'est développée grâce à l'essor de la bioinformatique. La reconstruction de réseau génétique permet de comprendre comment les molécules constituant un organisme interagissent entre elles, en particulier les gènes et les protéines. Ces interactions permettent à l'organisme de subvenir à tous ses besoins : production d'énergie, protection de l'organisme contre les agressions extérieures, reproduction...

Cette compréhension plus avant des processus faisant fonctionner l'organisme permet la création de nouvelles techniques pour améliorer ces processus (c'est le cas de la biolixiviation), mais aussi le ciblage de gènes ou de protéines, causes de maladies ou de dysfonctionnements. Le développement du génie génétique et de la microbiologie, tant du point de vue pratique que théorique, a amené plus vite que prévu les biologistes à penser à l'utilisation des réseaux génétiques, non plus comme seul outil de savoir brut, mais comme outil potentiellement applicable de façon industrielle. En effet, les réseaux génétiques ne sont plus seulement un résultat qui permet de visualiser le fonctionnement d'un organisme

mais sont un outil qui permet de comprendre et d'optimiser un processus utile à l'organisme. De même, l'apport de la bioinformatique dans le traitement des données a poussé les biologistes à changer de point de vue sur des problème qu'ils pensaient insolubles et qui se sont avérés possibles à résoudre. L'enjeu de la reconstruction de réseaux génétiques pour les biologistes est de pouvoir simuler le comportement d'organismes sans avoir à recourir à des expérimentations coûteuses en temps et en argent.
Le perfectionnement de l'informatique a permis de traiter la multiplication des données disponibles grâce à différentes avancées dans la biologie (découverte de l'ADN, clonage et amplification de l'ADN, puces à ADN...). Ces avancées ont fourni la possibilité aux chercheurs d'utiliser les outils informatiques et mathématiques pour résoudre ces problèmes. En effet, le développement des techniques heuristiques d'optimisation a permis de résoudre beaucoup de problèmes NP-complets tels que celui du voyageur de commerce, et l'augmentation de la puissance des appareils informatiques a autorisé la mise en oeuvre de ces techniques et leur utilisation. La bioinformatique permet d'analyser des quantités de données très importantes, ce que l'analyse "à la main" ne permet pas.
Le but de cette thèse était de trouver une méthode de reconstruction de réseaux génétiques. Pour reconstruire les réseaux génétiques, nous avions à notre disposition les résultats d'expériences menées sur des organismes. Il fallait donc partir des conséquences pour retrouver les causes. Cette méthode de travail oblige à tester des solutions au problème pour voir si elles fournissent des résultats conformes aux résultats donnés par la solution réelle. Le problème que nous cherchions à résoudre était celui de trouver une méthode permettant de construire un réseau qui corresponde aux données fournies, tout en satisfaisant aux conditions imposées par le fait de travailler sur des données biologiques, notamment la taille importante des réseaux et le bruitage des données. Le travail de reconstruction de réseau génétique est complexe. La reconstruction nécessite la compréhension totale du problème posé ainsi que le développement de nouvelles techniques pour retrouver dans l'ensemble des solutions possibles la solution qui va nous permettre de simuler efficacement le réseau recherché.

Afin de faciliter la compréhension de notre travail, ce document de thèse comporte trois parties, la partie permettant de comprendre le contexte du sujet, la partie relative à l'élaboration de la technique de résolution du problème et, enfin, la partie relative aux résultats obtenus.
La première partie regroupe la présentation du problème et des moyens connus jusqu'alors pour le résoudre. Le chapitre un, "Le problème traité", introduit les notions biologiques nécessaires à la compréhension des réseaux génétiques ainsi que les outils informatiques pour les résoudre les problèmes biologiques. Les termes biologiques, tels qu'ADN, gène, protéine, sont définis de telle sorte que, une fois ce vocabulaire assimilé, il sera simple de comprendre les enjeux de la reconstruction de réseaux génétiques. Un exemple de réseau

génétique est proposé ici, celui de l'opéron lactose, développé par Jacob et Monod dans [64]. Ensuite, l'exemple de la biolixiviation est présenté comme cas d'application des problèmes industriels que l'on peut aider à résoudre grâce aux réseaux génétiques. Ensuite, la problématique de la reconstruction est expliquée en introduisant l'idée de l'utilisation de la bioinformatique pour la résolution du problème. Après un rapide historique de la discipline et la présentation de quelques exemples d'applications, ce chapitre montre comment on peut utiliser la bioinformatique pour résoudre des problèmes, en expliquant les tenants et les aboutissants de la reconstruction, comme la comparaison d'expression ou la prédiction de fonctions de gènes. Nous nous penchons ensuite sur les applications des réseaux génétiques et surtout sur leurs spécificités, ce qui nous permet de cerner mieux la question de la reconstruction.
Le chapitre deux présente les techniques générales existantes pour reconstruire les réseaux. Les modèles présentés ici sont très variés. Les modèles booléens, les réseaux de Petri, les graphes, les réseaux bayésiens et les équations différentielles sont autant de formalismes qui nous permettent de modéliser le comportement des gènes et de simuler leur fonctionnement à des niveaux plus ou moins précis. Ces modèles, pour être efficaces, doivent être paramétrés pour être le plus proche possible de l'organisme réel que nous cherchons à modéliser. Ce chapitre s'attarde aussi sur les utilisations de ces modèles par différentes équipes et leurs réalisations.
Le troisième chapitre est consacré à l'étude des moyens de retrouver les paramètres des modèles. Plusieurs techniques sont présentées telles que le partitionnement des groupes de gènes qui permet de grouper les gènes en fonction de leur expression, quelques exemples de méthodes de partitionnement sont expliqués ici et la liste des techniques présentés ici n'est pas exhaustive. L'algorithmie génétique tient une grande place dans ce chapitre. On y retrouve un historique de la technique et de ses développements, une explication de son fonctionnement et de la théorie ainsi que des exemples d'utilisation de l'algorithme génétique pour la biologie.

La deuxième partie de ce document est consacrée à l'explication du travail effectué, la création du logiciel COGARE. Tout d'abord, l'algorithme COGARE est présenté. Cet algorithme a été conçu sur les bases d'un algorithme génétique classique, contenant une partie simulation du modèle, et une partie reconstruction de modèle. La partie simulation est basée sur des considérations biologiques et sur la question : "Comment modéliser l'action d'un gène sur l'expression d'un autre gène ?". La partie reconstruction se base sur la partie simulation pour calculer une solution de réseau. Cette partie reconstruction est la partie innovante de cette thèse car elle prend en compte les spécificités du problème et adapte les fonctions courantes de l'algorithmie génétique pour éviter les écueils de la reconstruction de réseaux. Ces innovations sont au nombre de trois. Le calcul de l'efficacité a été optimisé afin de pouvoir prendre en compte les trois types de données disponibles. La

mutation a été conçue de telle façon que des connaissances préalables du réseau solution étaient disponibles. Enfin, la technique de croisement a été modifiée pour qu'elle prenne en compte les aspects multi-échelles des réseaux biologiques. Cela permet au croisement d'effectuer deux optimisations, une globale et une locale, ce qui améliore l'efficacité de la méthode.

Enfin, la troisième partie du travail a été de compiler les résultats, ce qui a été effectué au chapitre cinq. Ce chapitre montre dans un premier temps comment, dans un registre global, nous pouvions optimiser les paramètres de COGARE pour qu'il donne les meilleurs résultats possibles et comment il se comportait dans le cas de données bruitées. Une étude de robustesse en fonction du bruit sur les données a aussi été effectuée. Dans un deuxième temps, COGARE a été comparé à d'autres logiciels exposés au chapitre cinq. Les algorithmes ont été comparés sur les mêmes types de données et les résultats montrent que COGARE a des résultats meilleurs que les algorithmes concurrents dans certaines configurations, comme la présence de plusieurs types de données ou la grande taille de réseau. Enfin, COGARE a été lancé sur des données d'expression connues afin de voir comment il se comportait en situation réelle. Ces tests ont montré que COGARE pouvait retrouver la composition d'un réseau réel dans des proportions remarquables. COGARE s'est aussi signalé dans la reconnaissances de régulateurs chez une bactérie liée à la biolixiviation. Les travaux effectués ici ont donné lieu à un article qui introduit le concept du logiciel COGARE ainsi que quelques applications du logiciel et des résultats obtenus par COGARE sur des données de biolixiviation [21].

Première partie

État de l'art

Chapitre 1

Le problème traité

1.1 Quelques précisions biologiques

Le vivant est entièrement défini par son génome, en tout cas en ce qui concerne le fonctionnement de ses cellules et les processus biologiques [95] tels que la respiration ou la circulation sanguine. Ce génome est codé sur la double hélice d'ADN (fig.1.1) et correspond à une suite de gènes qui vont définir le comportement macroscopique et microscopique de l'organisme. Les gènes sont une suite de nucléotides présents sur l'ADN qui vont définir

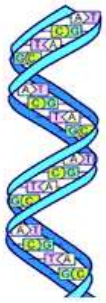


FIG. 1.1 – Représentation de l'ADN et des nucléotides.

le plan de fonctionnement de l'organisme. Pour pouvoir comprendre ce plan de fonctionnement, l'organisme utilise deux mécanismes. En premier lieu, la transcription va créer un négatif de l'ADN afin de permettre ensuite la traduction des nucléotides en acides aminés qui, liés entre eux, vont finalement former les protéines. Ces protéines sont des molécules utilisables par l'organisme. Il faut savoir que l'ADN est constitué de nucléotides de quatre types, l'Adénine (A), la Guanine (G), la Cytosine (C) et la Thymine (T). Ces nucléotides sont complémentaires, Adénine-Thymine et Guanine-Cytosine, c'est-à-dire qu'ils peuvent se lier deux à deux grâce à leur conformation spatiale (voir fig.1.2).Cette complémentarité va permettre de créer un ARN messager par le mécanisme de transcription.

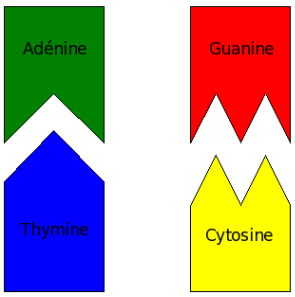


FIG. 1.2 – Schéma de complémentarité des paires de nucléotides.

La transcription *est le processus qui va séparer l'ADN en deux grâce à l'ARN polymérase. Ensuite, les nucléotides complémentaires vont se fixer sur l'ADN l'un à la suite de l'autre pour former de l'ARN messager, parfait complémentaire de l'ADN à ceci près que dans l'ARNm, la Thymine est remplacée par l'Uracile (U). Cet ARN messager porte l'ensemble du code génétique.*

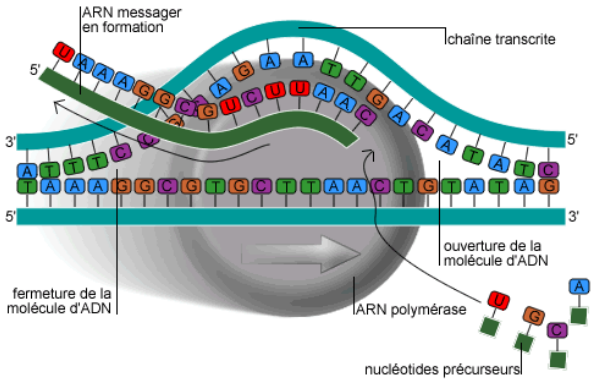


FIG. 1.3 – Schéma de transcription de l'ADN en ARN messager. Source : Wikipédia [131].

La traduction *est le processus qui va créer des protéines à partir de l'ARNm. La protéine est un ensemble d'acides aminés liés ensemble pour former une molécule utilisable par le corps humain. Les protéines remplissent des fonctions très diverses ; catalyse (les enzymes), transport (l'hémoglobine), communication (hormones tels que l'insuline)... L'ARNm va être pris en charge par un ribosome qui traduit la séquence nucléotidique en acides aminés. Les acides aminés sont portés par l'ARN de transfert (ARNt), une molécule dont une extrémité est formé de trois nucléotides et l'autre d'un acide aminé (voir fig.1.4). Un acide aminé est toujours associé au même triplet de nucléotides, que l'on appelle codon lorsque le triplet est sur l'ARNm et anti-codon lorsqu'il est sur l'ARNt. Il existe aussi quatre codons qui indiquent le début (codon initiateur) et la fin (codon stop) du gène. L'ensemble de ces codons forme un langage constitué de mots de trois lettres ayant*

pour correspondance un acide aminé. Le ribosome va récupérer les ARNt complémentaires de l'ARNm dans la cellule et coller bout à bout les acides aminés correspondants (fig.1.5). Le ribosome commencera son travail lorsqu'il rencontrera un codon initiateur, et le terminera lorsqu'il rencontrera un codon stop. Les acides aminés ainsi récupérés forment une protéine en se liant les uns à la suite des autres.

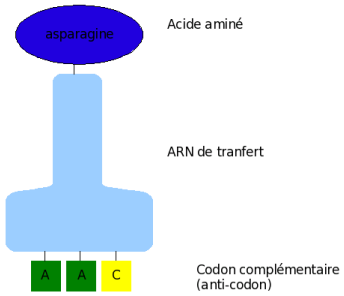


FIG. 1.4 – Schéma d'une molécule d'ARN de transfert.

En résumé, un gène est un ensemble de nucléotides qui vont, grâce à la transcription et à la traduction, être transcrits en protéines fonctionnelles pour l'organisme.

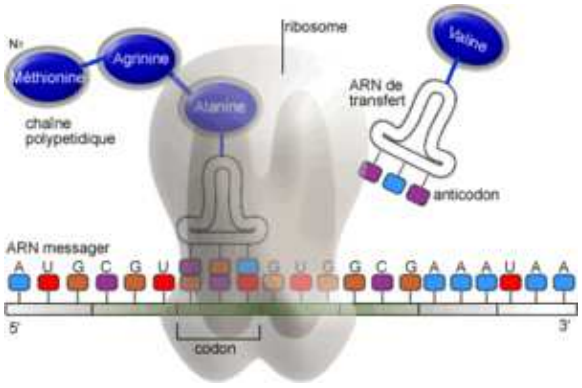


FIG. 1.5 – Schéma de traduction de l'ARNm en protéines. Source : Wikipédia.

Ces protéines vont ensuite se diffuser dans les cellules de l'organisme pour effectuer les tâches qui leur sont attribuées. Cette chaîne de mécanismes biologiques pour aboutir à une protéine est aussi connue sous le nom de dogme de la biologie moléculaire. Il a été énoncé par Crick en 1958 [28] et se résume ainsi :

$$\text{ADN} \xrightarrow{\textit{transcription}} \text{ARN} \xrightarrow{\textit{traduction}} \text{Protéine}$$

1.2 Qu'est ce qu'un réseau génétique ?

Un réseau génétique, l'entité que l'on cherche à modéliser, est l'ensemble des interactions qui existent entre les différents éléments de l'organisme, protéines, gènes, nutriments, oxygène... Le travail de F.Jacob et J.Monod [63] a montré qu'il existait des protéines particulières, spécifiquement dédiées à la régulation des mécanismes de l'organisme. Ces protéines permettent le contrôle des différentes molécules présentes dans l'organisme (protéines, substrats, nutriments...)
Les protéines sont formées par une chaîne d'acides aminés qui leur donne une conformation spatiale particulière définissant leur fonction [44] (par exemple, la forme de l'hémoglobine lui permet de capter les molécules d'O_2). Certaines de ces molécules, les enzymes, ont la capacité de catalyser des réactions biochimiques. Ces protéines jouent donc un rôle en modifiant la cinétique chimique des réactions en interagissant avec d'autres produits. D'autre part, certaines protéines ont la capacité de bloquer la traduction de l'ARNm en acides aminés en se positionnant sur le site initiateur, empêchant le ribosome de se fixer à l'ARNm pour la traduction. Ceci se nomme la régulation et est un processus biologique qui permet de réguler les concentrations de molécules dans les cellules et donc de commander les réactions de l'organisme. Un réseau génétique est l'ensemble des interactions dues à ces protéines dans un organisme. En effet, les protéines étant les produits des gènes, dire qu'une protéine active une réaction chimique est équivalent à dire que le gène active cette réaction.
Pour connaître parfaitement le fonctionnement d'un organisme, il est nécessaire de connaître l'ensemble de ses interactions. Le réseau génétique est donc une modélisation des interactions entre éléments dans l'organisme. Ces interactions peuvent être représentées par un graphe d'interaction dont les sommets représentent le couple gène/protéine, les arêtes indiquent les interactions d'un couple sur un autre et, le cas échéant, le signe de l'arête montre le type d'interaction (activation ou inhibition). Suivant le degré de précision voulu et les informations dont on dispose, on peut assimiler les nœuds à différents éléments (ADN, ARN, protéines, substances nutritives...) et les arêtes à différents types d'interactions (production, utilisation, catalyse, blocage...).

Exemple de réseau génétique : l'opéron Lactose

Prenons l'exemple du métabolisme du lactose chez *Escherichia coli*, qui a été développé par Jacob et Monod [64]. Une bactérie a besoin de sucre pour lui fournir l'énergie nécessaire à ses fonctions métaboliques. Lorsqu'il y a du glucose dans l'environnement, l'organisme va l'utiliser en priorité comme source d'énergie et bloquer les autres voies possibles d'approvisionnement. En cas d'absence de glucose, il faut tout de même subvenir aux besoins en énergie de l'organisme. Pour ce faire, une nouvelle voie est ouverte pour l'approvi-

sionnement : le lactose. Le lactose est constitué de deux molécules, une de galactose et une de glucose, reliées par une liaison osidique. La bactérie doit donc la couper en deux pour pouvoir l'utiliser. Ceci nécessite plus d'énergie et cette voie n'est donc utilisée qu'en dernier recours.
Pour permettre à l'organisme d'utiliser ces sources d'approvisionnement en énergie, un mécanisme de régulation a été mis en place. Ce processus fait intervenir quatre gènes, le lacI, le lacZ, le lacY et le lacA. Les gènes lacA, lacY et lacZ font partie du même opéron.
Un opéron *est un ensemble de gènes qui vont être traduits et transcrits à la suite, car se situant tous après un même promoteur. Le ribosome va lire tous les gènes qui se situent après ce promoteur et tous les gènes de l'opéron seront traduits en protéines à la suite.*
Sur cet opéron, le lacA code pour la thiogalactoside transacétylase, une enzyme peu utile dans notre exemple. Le lacZ code la β-galactosidase qui hydrolyse (sépare) le lactose en glucose et galactose. Le lacY code la Lactose Permease, une enzyme qui permet le passage du lactose de l'extérieur à l'intérieur de la cellule ; sinon la membrane est imperméable au lactose. La lacI code la protéine Lac Repressor qui se fixe sur le promoteur de l'opéron lac et qui empêche la traduction des gènes lacY, lacZ et lacA.
Il est possible de définir deux cas intéressants pour visualiser le fonctionnement de cet opéron : absence de glucose et présence de lactose, et absence de glucose et de lactose.

1. En cas d'absence de glucose et de lactose, le lacI produit du lac Repressor qui va se fixer sur le promoteur de l'opéron lac, ce qui va empêcher la production des protéines de lacZ, lacY et lacA. La voie de récupération d'énergie par le lactose est bloquée.

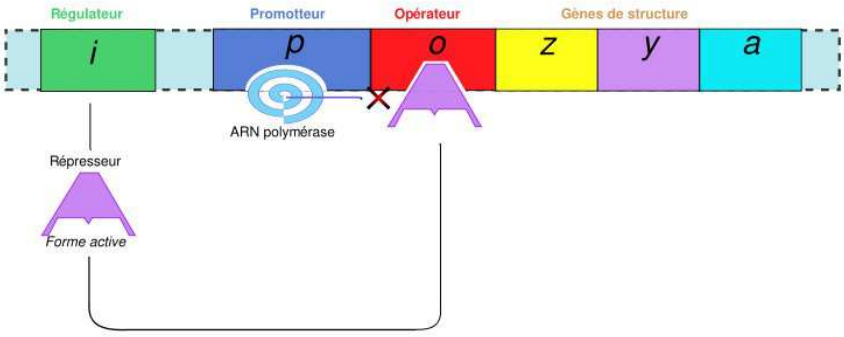


FIG. 1.6 – Comportement de l'opéron lactose en absence de lactose. Source : Wikipédia.

2. En présence de lactose, le lac Repressor se lie avec l' allolactose, ce qui l'empêche d'inhiber l'opéron lac. L'ARN pourra donc se lier au promoteur pour transcrire les protéines, qui vont ensuite servir à traiter le lactose afin d'en extraire le sucre nécessaire à la vie de la cellule.

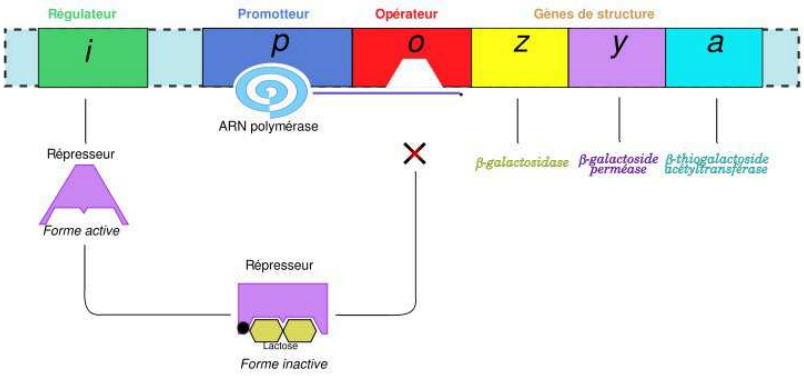


FIG. 1.7 – Comportement de l'opéron lactose en présence de lactose. Source : Wikipédia.

3. En cas de présence de glucose et de lactose, la bactérie va métaboliser préférentiellement le glucose car ce processus est moins gourmand en énergie. Il existe dans la cellule une protéine CAP. Cette protéine a pour but de catalyser la transcription de l'opéron lactose car, lorsqu'elle est liée à la protéine AMPc, elle va se fixer sur le promoteur de l'opéron et augmenter l'affinité de l'ARNp avec le promoteur. La protéine AMPc n'est disponible dans la cellule que lors d'une carence en glucose. Il en résulte que en présence de glucose et de lactose, la traduction de l'opéron est très faible et en absence de glucose, la transcription de l'opéron lactose s'active pour permettre un approvisionnement adéquat de la cellule.

La bactérie a développé des moyens d'inhiber ou d'activer certains processus dans le but de traiter ses besoins naturels en fonction de l'environnement extérieur, dans ce cas la présence ou non de lactose pour la production d'énergie.
Ces processus peuvent se modéliser par un graphe représentant le réseau génétique et ayant pour nœuds les différents éléments présents : L_i pour le lactose interne, L_e pour le lactose externe, Z pour le lacZ et sa protéine associée, Y pour le lacY, A pour le lacA, I pour le répresseur, IL pour le répresseur associé au lactose interne et G pour le glucose.
Le graphe associé à l'opéron lactose correspond à la figure 1.8.

Pour expliquer le fonctionnement de ce graphe, prenons les éléments un par un. Pour des questions de lisibilité, les actions de chaque élément sur lui-même (rétroactions) sont exclues, comme par exemple la dégradation des éléments et la présence d'une source constante (pour le lactose externe notamment). Le glucose est le produit du lactose intérieur et du lacZ, donc plus ces éléments sont présents, plus la concentration de glucose sera grande (Li → G / Z → G sur le graphe). Le lacZ est réprimé par le lacI et est consommé pour produire du glucose (I → Z / Li → Z). Le lactose interne est consommé pour produire du glucose et le lactose externe rentre dans la cellule grâce au lacY et devient ainsi du lactose interne (Le → Li / Y → Li / Z → Li). Le lacY est lui réprimé par le répresseur du lacI qui, comme pour le lacZ, va se fixer sur le promoteur pour empêcher la traduction

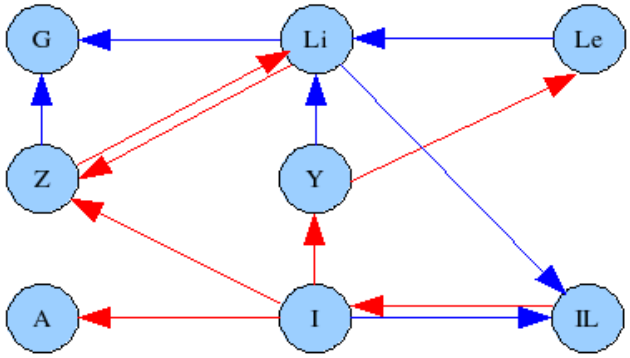


FIG. 1.8 – Graphe représentatif du fonctionnement de l'opéron lactose. Flèche rouge : inhibition. Flèche bleue : activation.

de l'opéron (I → Y), de même pour le lacA (I → A). Le lactose externe va être transporté à l'intérieur de la cellule par le lacY, donc plus il y a de lacY, moins il y aura de lactose à l'extérieur (Y → Le). Dans ce schéma, puisque l'on ne s'intéresse qu'à la partie synthèse du glucose, le gène lacI est juste consommé pour créer la molécule protéine I-Lactose notée IL (IL → I). Finalement, la concentration en IL sera d'autant plus grande que la concentration en lactose interne et en protéine I est importante (I → IL / Li → IL).
Il est donc possible de créer un graphe représentant la dynamique du système et par lequel on modélise que l'augmentation quantitative d'un élément du graphe entraînera la modification des quantités des autres éléments. Par exemple, si l'on augmente la quantité de protéine répresseur I, cela va faire baisser les quantités de lacZ, lacY et lacA, et augmenter la quantité de IL, ce qui va avoir des conséquences sur les autres éléments.

1.3 Problèmes biologiques associés

Les réseaux de régulation génétique permettent de visualiser et de comprendre beaucoup de processus en biologie. En effet ces réseaux permettent de modéliser les actions des enzymes, de voir quelles sont les conditions permettant une meilleure synthèse des protéines. Ils nous offrent donc une vision macroscopique du fonctionnement de l'organisme simplifié. Connaissant les réseaux de régulation génétique, il est plus facile de connaître les fonctions des gènes et donc leur utilité. Il est possible de les utiliser pour visualiser les gènes responsables du mauvais fonctionnement de l'organisme dans le cas de maladies génétiques. Les réseaux génétiques permettent aussi la comparaison des réseaux d'organismes différents pour identifier des processus biologiques communs. Les problèmes biologiques à résoudre sont variés et conduisent à des applications dans des domaines scientifiques et industriels très nombreux. Quelques exemples d'applications peuvent être trouvés dans la littérature, par exemple la découverte de gènes responsables de maladies génétiques [40], ou encore la recherche de gènes cibles pour un traitement [119]...

Le cas de la biolixiviation du cuivre

Le problème de base de ma thèse était de modéliser le processus de biolixiviation dans le cadre de l'extraction de cuivre dans les mines chiliennes. La biolixiviation [76] est une technique minière qui emploie de micro-organismes pour extraire des métaux d'un minerai. L'extraction se fait par lixiviation : cela consiste à séparer une substance soluble d'une structure solide en l'incorporant à une préparation liquide qui facilite son extraction. La lixiviation chimique utilise une solution chimique pour traiter la substance soluble, ce qui a des conséquences environnementales importantes, alors que la biolixiviation utilise des bactéries présentes naturellement dans les mines ce qui diminue drastiquement l'impact environnemental.
Le procédé de biolixiviation est généralement appliqué à des tas de minerai broyés et réduits en poudre pour ensuite être traités dans des installations irriguées. Les amas de poudre sont traités avec un liquide acide renfermant une population bactérienne (qui souvent est déjà en partie présente dans le minerai puisque ces bactéries se nourrissent de métaux). Les bactéries se chargent alors de séparer le cuivre des autres substances inutiles. Le liquide et les métaux extraits sont ensuite pompés et centrifugés pour récupérer les métaux utiles, le cuivre dans notre cas.
La bioprospection minière consiste à utiliser des micro-organismes existant dans la nature pour lixivier et oxyder les métaux. L'étude du génome de ces micro-organismes aide les chercheurs à en apprendre davantage sur leur biologie, ce qui peut amener la création par génie génétique d'organismes à même de maximiser les rendements de la bioprospection minière.

La bioprospection minière fait appel à des organismes et à des méthodes existants dans

FIG. 1.9 – Vue extérieure d'une mine de cuivre, Chuquicamata au Chili.

la nature. L'extraction minière traditionnelle est un procédé particulièrement toxique puisqu'elle se fait au moyen de produits chimiques tels que le cyanure ou le mercure. Bien que le recours à des produits chimiques nocifs ne soit pas complètement éliminé avec l'usage de la bioprospection minière, cette méthode permet d'en diminuer l'utilisation, ce qui réduit les coûts de nettoyage associés aux procédés d'extraction et les dégâts environnementaux. La biolixiviation est un processus qui peut être amélioré grâce à la connaissance du comportement complet des bactéries dans leur milieu. Pour connaître ce comportement, il apparaît opportun d'utiliser la bioinformatique afin de reconstruire le réseau génétique des bactéries, ce qui permettra de comprendre leur fonctionnement.

1.4 Les outils de reconstruction de réseaux génétiques

Reconstruire les réseaux biologiques permet de visualiser les processus biologiques et chimiques à l'oeuvre dans les organismes, souvent complexes. Cette reconstruction permet, outre de modéliser et de simuler les comportements des organismes, une analyse détaillée de ces comportements et l'optimisation des conditions environnementales dans lesquelles ils évoluent afin, par exemple, de renforcer une fonction de l'organisme que l'on souhaite utiliser. La reconstruction de réseau est réalisée par le biais de techniques de bioinformatique.

La bioinformatique

La bioinformatique *se définit comme l'activité scientifique liée à la biologie, fondée sur l'utilisation de logiciels et de programmes informatiques pour organiser, stocker et analyser l'information et les données biologiques. Son objectif est d'améliorer notre compréhension des systèmes biologiques. C'est un domaine scientifique où la biologie, l'informatique et la technologie de l'information convergent en une discipline unique.*

Les chercheurs utilisent des techniques informatiques et statistiques de pointe pour pouvoir traiter et analyser la multitude de données biologiques générées par les biotechnologies modernes, appelées "omiques", comme la génomique et la protéomique. Ces technologies "omiques" portent sur différents aspects des systèmes biologiques :

- la génomique est l'analyse systématique de toute la composition génétique d'un organisme. Elle comprend l'étude de la fonction des gènes dans les cellules, les organes et les organismes.
- la protéomique se penche sur l'ensemble de la composition protéinique d'une cellule à tout moment donné.
- la transcriptomique porte sur l'analyse des ARNm transcrits.

Ces technologies "omiques" génèrent d'énormes quantités de données biologiques dans lesquelles il serait impossible de naviguer sans les systèmes informatiques. Les données comprennent notamment les séquences des acides aminés et des nucléotides qui sont à la base des protéines et des gènes . La bioinformatique vise à traduire les données complexes recueillies en informations utilisables. Cette science s'est développée en suivant les progrès faits par la biologie moléculaire et par l'informatique.

La biologie moléculaire a connu ses grands débuts avec la découverte de l'hérédité par Mendel en 1866 et ses travaux sur les petits pois. On a alors compris qu'il existait des informations dans les organismes qui étaient transmis à la descendance. Cette découverte a marqué le début des recherches pour trouver les supports de cette information. La question de savoir où et comment cette information était stockée fut en partie résolue en 1953 lorsque Watson et Crick ont découvert la double hélice d'ADN. Auparavant, en 1951,

le séquençage des protéines par Sanger et Tupy avait donné quelques indices en indiquant que les protéines étaient un assemblage d'acides aminés. Par la suite, la compréhension des phénomènes génétiques a été approfondie grâce à l'énoncé du dogme de la biologie moléculaire par Crick et l'introduction des termes traduction et transcription.
Les techniques de clonage de l'ADN en 1977 et d'amplification en 1984 ont permis en 1995 le séquençage du premier génome complet, celui de *Haemophilus Influenzae* (un virus de la grippe) qui a coûté 1 million de dollars.
Ces avancées dans le domaine de la biologie moléculaire ont généré une augmentation progressive des données disponibles. En effet, la première base de données publiée par Dayhoff en 1960 contenait 50 protéines. Ensuite les premières utilisations de l'informatique pour traiter ces données sont apparues dans les années 1970, à l'époque pour comparer les génomes. Les bases de données se sont agrandies avec l'essor des techniques (puces à ADN, séquençage automatique...). Les nucléotides ont été progressivement ajoutées aux bases de données, permettant d'arriver à plus de 100 génomes complètement séquencés dans les années 2000. Par exemple, Genbank [130], une des bases de données de référence pour la bioinformatique croît de manière exponentielle [132] en doublant tous les dix mois (voir la figure 1.10). En Février 2008, Genbank contenait plus de 85 millions de bases appartenant à plus de 82 millions de séquences.

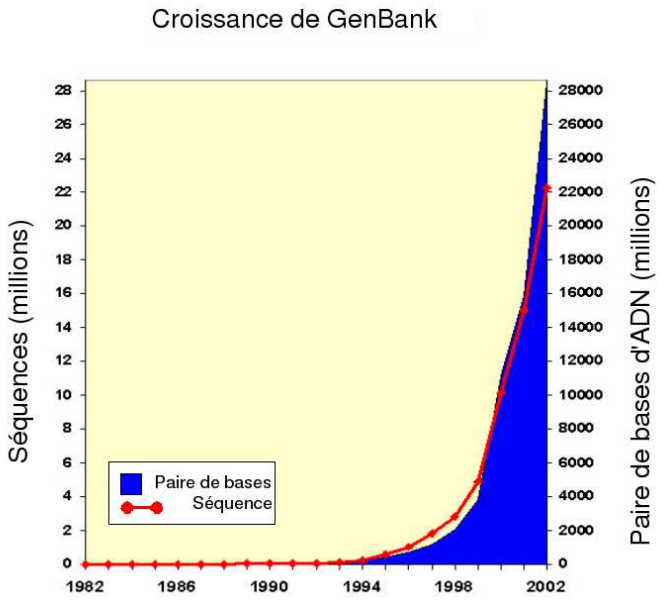


FIG. 1.10 – Evolution de la taille de la base de données Genbank.
Source : http ://esilbac1.esi.umontreal.ca

Les problèmes liés à la bioinformatique

La bioinformatique aide les biologistes dans plusieurs registres souvent en rapport avec les aspects structurels et fonctionnels des gènes et protéines, qui sont les aspects les plus fastidieux car nécessitant beaucoup de traitements informatiques pour comparer, trier et calculer. Ces recherches mènent principalement à la découverte des caractéristiques, des emplacements et des fonctions des gènes. Voici quelques unes des directions actives à l'heure actuelle :

* Comparaison des génomes cartographiés de différentes espèces. Ceci permet de définir les ressemblances et les différences entre les organismes. Par exemple, les génomes d'oiseaux ont été comparés afin de visualiser leur développement et leur différenciation [36].
* Accès à l'information génétique provenant d'autres équipes partout dans le monde. Il existe diverses bases de données consultables, dont plusieurs sur internet, utiles pour définir les séquences d'ADN de gènes particuliers et éviter de refaire des travaux déjà effectués ; par exemple Genbank [130]. Les bases de données donnent également un accès rapide et global aux renseignements sur les protéines. Les chercheurs peuvent retrouver une séquence d'après le nom d'une protéine ou une protéine d'après la séquence d'acides aminés.
* Les bioinformaticiens utilisent les outils de séquençage pour approfondir leur connaissance des caractéristiques du matériel génétique, y compris des protéines. Après avoir analysé les séquences de protéines, les chercheurs peuvent rechercher des séquences similaires dans les bases de données à l'aide de logiciels. D'autres programmes permettent aux chercheurs d'étudier les protéines et les nucléotides en trois dimensions. Comme la forme détermine essentiellement la fonction, ces programmes donnent aux chercheurs des indices sur le rôle des séquences.
* Le dernier point, que je développe dans ma thèse, est l'élaboration de modèles biologiques permettant aux biologistes de comprendre les interactions entre les différentes composantes du vivant (protéines et gènes), ceci afin de pouvoir prédire les réactions des cellules ou bactéries aux stimuli et changements de milieu.

1.5 Quelques applications pratiques

Les applications industrielles de la bioinformatique sont nombreuses et variées. En effet, les biologistes ont de nombreux problèmes d'expérimentations, car elles coûtent très cher en argent, mais aussi en temps. Il leur serait donc très utile de pouvoir récupérer des résultats à partir d'expériences déjà faites. Pour cela, il existe plusieurs approches.

– Séquençage génétique et identification des gènes. Ces procédés sont utilisés pour associer

un gène aux protéines dans lesquelles il est susceptible de se traduire et identifier le rôle de ces protéines.
- Prédiction de fonctions. Cela permet aux biologistes de réduire l'étendue des possibilités à examiner et de faire des recoupements pour des organismes similaires.
- Comparaison de matériels génétiques qui aident à mettre à jour des relations évolutives et fonctionnelles d'un organisme et ainsi de reconstruire des arbres évolutifs qui permettent de connaître où, quand et donc pourquoi sont apparues certaines spécifications génétiques.
- Mesures comparatives. Ces mesures sont utilisées pour le diagnostic. Par exemple la leucémie [103] qui a été diagnostiquée par une approche de classification.
- Modélisation des interactions dans un organisme. Il existe trois types de réseaux.
  - Les réseaux métaboliques modélisent l'ensemble des réactions biochimiques qui fabriquent les composés dont a besoin un organisme (à l'exception de ceux pris dans l'environnement).
  - Les réseaux d'interactions protéine-protéine modélisent l'ensemble des interactions chimiques directes entre protéines : transport de l'information entre intérieur et extérieur de la cellule et modification d'une autre molécule.
  - Les réseaux de régulation génétique sont les réseaux qui modélisent le contrôle de l'expression des gènes par les produits d'autres gènes. Ces réseaux sont ceux qui nous intéressent plus particulièrement.
- Analyse de la topologie (structure). Cette topologie est modélisée par un graphe (ensemble de nœuds reliés par des arêtes) ; on analyse sa connectivité.
- Apprentissage concernant la relation interaction-fonction. Ceci permet la prédiction du rôle fonctionnel d'une protéine en fonction des rôles de ses voisins [108].
- Recherche de motifs (sous-graphes) similaires. Par exemple une voie métabolique (processus biologique) chez un organisme peut être décomposée en deux voies chez une autre espèce. Pour faire le même travail, un organisme va utiliser un processus et l'autre, deux.
- Comparaison, calcul de distance entre sous-graphes, mesure de la distance évolutive, reconstruction de scénarios évolutifs.
- Simulation dynamique. Il existe différentes techniques permettant d'exécuter des simulations : modélisation binaire, équations différentielles, réseaux de pétri... Ces techniques permettent de modéliser le comportement du réseau en fonction des conditions environnementales.

Toutes ces techniques permettent d'économiser des expériences et donc de l'argent et du temps : si l'on connaît les gènes qui sont impliqués dans un processus et, si l'on sait que ces gènes se retrouvent dans un autre processus, alors il y a de grandes chances pour qu'ils codent le même processus.

1.6 Spécificités de la reconstruction de réseaux biologiques

Les réseaux biologiques sont des réseaux ayant des caractéristiques particulières qu'il faut prendre en compte pour pouvoir résoudre les problèmes inhérents à leur reconstruction. Tout d'abord, les données disponibles sont des données extrêmement bruitées. En effet, le moindre élément extérieur à l'expérimentation peut entraîner des changements dans l'équilibre et donc changer les résultats tout en ayant, officiellement, les mêmes conditions expérimentales. De plus, les techniques d'obtention de ces données ne sont pas très précises. Les données obtenues ont donc une grande variabilité.
Les réseaux génétiques sont de plus des réseaux dont les nœuds peuvent s'autoréguler, c'est-à-dire que l'augmentation de la concentration d'un élément peut entraîner ensuite sa baisse par le biais des interactions : nous pouvons observer ce phénomène dans le processus d'homéostasie du cuivre [88] chez *Enterococcus hirae*. Dans cet exemple, le répresseur (appelé copY) est codé à l'intérieur de l'opéron, donc plus l'opéron est traduit et transcrit, plus il y aura de copY. Ce copY réprimera alors la transcription de l'opéron et fera diminuer la concentration en copY. Ces boucles sont souvent difficiles à identifier car elles ne sont clairement visibles que dans le cas ou l'élément est isolé, ce qui est rare dans les expériences sur un organisme.

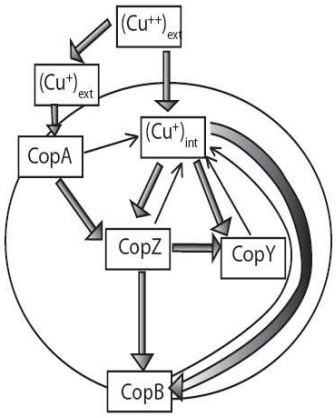


FIG. 1.11 – Trajectoire du cuivre dans un organisme. Les flèches pleines indiquent le transport de cuivre et les flèches minces, la dégradation. Cette figure est extraite de [88].

Les autres spécificités du problème de reconstruction de réseaux biologiques sont que, outre des informations expérimentales, les biologistes ont des connaissances partielles sur le réseau. Il faut intégrer dans les calculs ces informations pour pouvoir en optimiser au mieux l'utilisation. Ces informations contraignent le système et diminuent son degré de liberté. Avec un algorithme classique, il est possible que cette surcontrainte l'empêche de converger. Il faut donc prendre en compte ces informations, mais aussi donner l'oppor-

tunité à l'algorithme de reconstruction de passer outre ces informations dans le cas où il risque de tomber dans une impasse.
Les réseaux génétiques sont des réseaux multi-échelles avec beaucoup de processus imbriqués les uns dans les autres. Par exemple, les protéines permettent de réguler la transcription des gènes et, pour créer la protéine, il faut un processus d'assemblage des acides aminés. Il en résulte qu'il faut faire un choix quant à l'échelle des processus que l'on veut modéliser. Ce choix de modélisation verticale va permettre de régler la précision du modèle que l'on crée. De même, les processus s'imbriquent les uns dans les autres, il est donc nécessaire de faire un choix de modélisation horizontal permettant de choisir quels sont les processus à mettre en avant et quels sont ceux qu'il faut mettre de côté. Un dilemme fréquent est de choisir si on modélise dans le réseau la transcription des protéines avec le ARNm ou si l'on ne considère que le couple gène-protéine dans son ensemble, sachant que modéliser le couple gène-protéine dans son ensemble entraîne l'"oubli" par le modèle du délai nécessaire au processus de traduction/transcription.

Chapitre 2

Les modèles

Il existe de nombreuses manières de modéliser les réseaux de régulation génétique. Plusieurs caractéristiques sont à prendre en compte pour choisir et construire un modèle biologique. Alvarez-Buylla [3] a présenté les différents types de modélisation possibles. Les caractéristiques à prendre en compte dans le choix d'une technique de modélisation sont de plusieurs ordres.
La première caractéristique importante est le niveau de détail que l'on veut obtenir, un exemple de ce niveau de détail est de choisir si on est intéressé par la réaction de traduction et de transcription de l'ADN ou si on reste au niveau gènes-protéines en considérant qu'un gène engendre une protéine.
Les modèles peuvent être discrets, chaque variable pouvant prendre un nombre fini d'état correspondant à des niveaux d'activation, ou continues. Certains modèles sont capables de mixer les deux types de modélisation. Les modélisateurs doivent aussi choisir entre des modèles déterministes et sont régies par des lois connues et définies ou alors stochastiques, ce qui permet d'ajouter au modèle le bruit inhérent aux systèmes naturels.
Une autre caractéristique importante des modèles est le fait qu'ils soient dynamiques ou statiques. Un modèle dynamique va permettre de retrouver ou de visualiser les liens entre les différents gènes et leurs influences les uns sur les autres. Le modèle statique va montrer les groupes de gènes reliés mais sans expliciter leurs relations.
Les modèles sont donc nombreux et utilisent des outils mathématiques et informatiques variés pour résoudre les problèmes et satisfaire les besoins et les caractéristiques des modèles. Nous allons ici en présenter quelques uns afin de donner au lecteur une vue globale du sujet.

2.1 Les graphes

Une des méthodes les plus simples pour modéliser les réseaux biologiques consiste tout simplement à les modéliser avec des graphes. Cette méthode permet de visualiser le réseau

sur un graphe. Cette technique est purement qualitative, c'est-à-dire qu'avec cette modélisation nous n'allons pas simuler le fonctionnement du réseau, mais grâce à la théorie des graphes, il va être possible de savoir quels gènes sont régulés par un certain gène ou quels gènes appartiennent à une fonction métabolique donnée. En substance, un graphe G = (V, E) est composé de 2 entités : un ensemble de points (V, qui peuvent être assimilés aux gènes) et de flèches (E) qui indiquent les interactions entre les gènes. Ces flèches sont signées pour définir l'action inhibitrice ou activatrice du gène. Il existe une bibliothèque très riche de connaissances de graphes particuliers dans laquelle on peut trouver des informations instructives quant à notre expérience. En effet, la théorie des graphes nous dit que si certains motifs sont présents dans un graphe, il existe des attracteurs, des boucles d'équilibre...

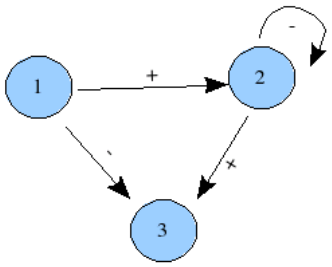


FIG. 2.1 – Exemple de graphe.

De plus, il est possible d'effectuer un certain nombre d'opérations sur les graphes qui vont nous donner des informations intéressantes d'un point de vue biologique. À partir de ces graphes, il est possible d'effectuer un regroupement de gènes, qui permet de visualiser les gènes qui appartiennent au même processus biologique, ou bien qui sont régulés par les mêmes mécanismes.

Une des utilités les plus flagrantes de la théorie des graphes pour l'étude de réseaux biologiques est l'étude et la recherche de sous-graphes fonctionnels dans un graphe complet. En effet, grâce à la théorie des graphes, nous pouvons calculer la connectivité d'un groupe de gènes afin d'isoler les groupes qui sont fortement connectés entre eux et qui forment un groupe de gènes ayant une fonction particulière. Ce processus permet de réduire la complexité d'un réseau : en effet, à la place de travailler sur un graphe entier, nous pouvons travailler sur des morceaux de graphes qui ont une signification biologique, ce qui en facilite l'analyse. La modélisation par graphe est très intéressante car visuelle ; nous voyons très rapidement et facilement où nous voulons en venir, elle permet d'obtenir des résultats qualitatifs, mais malheureusement ne permet pas d'obtenir des résultats quantitatifs, à moins de la coupler avec une autre technique. Nous avons choisi ce modèle comme modèle de base de notre problème car il est simple à mettre en oeuvre et à comprendre. De plus,

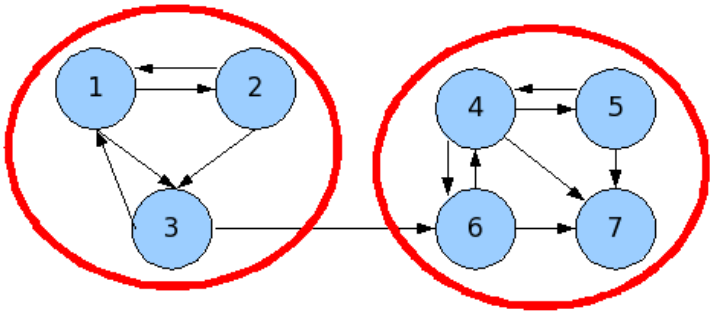


FIG. 2.2 – Visualisation de deux sous-graphes (entourés en rouge) sur un graphe. Les deux sous-graphes ont été identifiés grâce à la forte connectivité entre les éléments du sous-graphe et la faible connectivité avec les éléments hors du sous-graphe.

il peut s'adapter à tous les types de problèmes de reconstruction que l'on peut rencontrer.

Leur utilisation en bioinformatique

Les premiers travaux mixant théorie des graphes et biologie sont l'oeuvre de Benzer [13] dans les années 50 sur les bactériophages T4 (des bactéries tueuses). Les graphes ont été très souvent utilisés dans la biologie grâce à leur ressemblance naturelle avec les réseaux génétiques. Dans les années 90, Lemieux et Khon [72] ont utilisé la théorie des graphes pour identifier des propriétés de régulation dans des réseaux métaboliques. De même, des équipes se sont servies de graphes pour montrer des liens entre des gènes et des tissus biologiques [69]. D'autres travaux sur l'inférence de réseaux ont aussi été conduits par Wagner et Toh dans les années 2000 ([120], [117]).

2.2 La modélisation par équations différentielles

Dans un réseau, il est possible de définir les actions de chaque membre du réseau (gène, protéine, élément...) par leurs actions les uns sur les autres. Les réactions chimiques peuvent, par exemple, être modélisées par des équations bilans :

$$A + B \rightarrow C + D \tag{2.1}$$

À ces équations (voir [100]), nous pouvons associer des équations différentielles agissant sur les concentrations en produits correspondants (on note A la concentration en produit,

ou réactif) :

$$\frac{\partial A}{\partial t} = -k_a AB \tag{2.2}$$

$$\frac{\partial C}{\partial t} = +k_a AB \tag{2.3}$$

où k_a est la vitesse de réaction de la réaction 2.1.
Plusieurs réactions associées à un réseau vont créer un système d'équations différentielles qu'il sera possible de résoudre analytiquement ou numériquement. Grâce à cette procédure, il est assez aisé de comprendre comment une quantité va évoluer au cours du temps. De plus, il est possible pour modéliser plus efficacement encore la réalité, d'ajouter des fonctions qui simulent la réalité, par exemple des fonctions non linéaires ou des fonctions seuils qui permettent de gérer la saturation en certains éléments. Par exemple, lorsqu'une cellule est dans un environnement riche en nutriments, ces nutriments peuvent passer dans la cellule de deux façons, soit par des protéines de transport sur la membrane, soit par pression sur la membrane. La deuxième voie d'accès du nutriment dans la cellule ne s'ouvre donc qu'en cas de forte concentration de ce nutriment. La vitesse de passage du nutriment de l'extérieur de la cellule vers l'intérieur va augmenter en fonction de la concentration de ce nutriment à l'extérieur de la cellule. Pour modéliser cela, on peut utiliser une fonction sigmoïde (cf fig.2.3) appliquée à la concentration extérieure du nutriment. Lorsqu'il existe plusieurs réactions que l'on veut modéliser, nous obtenons des systèmes

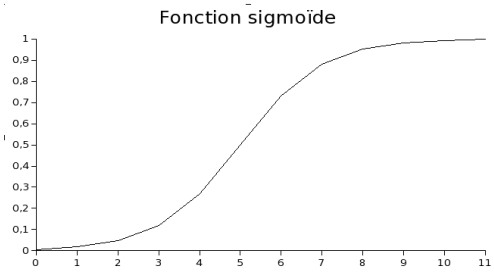


FIG. 2.3 – Fonction sigmoïde. Lorsque la concentration en nutriment à l'extérieur est faible, peu de nutriments passent. Plus cette concentration augmente, plus la vitesse de passage du nutriment de l'extérieur vers l'intérieur augmente aussi.

d'équations de plus en plus grands. Partons de celui-ci :

$$A + B \rightarrow C \tag{2.4}$$

$$2C + B \rightarrow D \tag{2.5}$$

$$D \rightarrow 2E \tag{2.6}$$

Ces équations bilan sont traduites en équations différentielles et le système représentant ce réseau biologique devient :

$$\frac{\partial A}{\partial t} = -k_1 AB \tag{2.7}$$

$$\frac{\partial B}{\partial t} = -k_1 AB - k_2 C^2 B \tag{2.8}$$

$$\frac{\partial C}{\partial t} = k_1 AB - k_2 C^2 B \tag{2.9}$$

$$\frac{\partial D}{\partial t} = k_2 C^2 B - k_3 D \tag{2.10}$$

$$\frac{\partial E}{\partial t} = 2k_3 D \tag{2.11}$$

avec k_1, k_2, k_3 les vitesses de réaction respectives de 2.4, 2.5, 2.6. Si l'on suppose que la réaction 2.6 est du type que nous avons défini plus haut, nous pouvons transformer l'équation 2.11 en :

$$\frac{\partial E}{\partial t} = 2f(D)D \tag{2.12}$$

où la fonction f est la fonction sigmoïde.

Le problème inhérent à ce genre de modélisation est la rapide progression vers des systèmes que l'on ne peut résoudre que numériquement car ils sont trop complexes et non-linéaires. Ces modèles sont utiles du point du vue pratique, mais par contre il faut utiliser d'autres techniques pour pouvoir les analyser et en tirer des informations plus globales sur les réseaux associés.

À partir de ces systèmes, on peut construire d'autres types de modélisations en appliquant des fonctions linéaires par morceaux qui permettent l'analyse qualitative des réseaux en filtrant certains effets (grâce à des filtres, fonctions sigmoïdes, par exemple). Et en compartimentant l'espace des phases, on peut définir des états globaux dans lesquels évolue le réseau, et ainsi donner une valeur qualitative aux phases du réseau : réseau au repos, réseau en activité homéostatique, etc. Toutes ces possibilités font du modèle par équations différentielles un des modèles les plus complets et efficaces pour modéliser les réseaux biologiques, bien que ne prenant pas en compte les phénomènes stochastiques parfois présents dans ces réseaux. Mais le principal problème lié à cette modélisation demeure qu'avec le nombre extrêmement important de variables en jeu, le système se transforme assez vite en système gourmand en temps de résolution et espace mémoire du fait de sa taille.

Une autre manière d'obtenir des résultats à partir des équations différentielles régissant les réactions chimiques est d'utiliser les matrices jacobiennes qui nous permettent de visualiser assez aisément les influences respectives de chaque variable. En effet, les matrices jacobiennes sont les matrices des dérivées partielles de chaque fonction par rapport à chaque variable. Le signe de chaque élément de cette matrice nous permet de déterminer l'influence relative de chaque variable sur les autres.

Pour le système précédent, il est possible de calculer la matrice jacobienne :

$$\begin{pmatrix} \frac{\partial - k_1AB}{\partial A} & \frac{\partial - k_1AB}{\partial B} & \frac{\partial - k_1AB}{\partial C} & \frac{\partial - k_1AB}{\partial D} & \frac{\partial - k_1AB}{\partial E} \\ \frac{\partial - k_1AB - k_2C^2B}{\partial A} & \frac{\partial - k_1AB - k_2C^2B}{\partial B} & \frac{\partial - k_1AB - k_2C^2B}{\partial C} & \frac{\partial - k_1AB - k_2C^2B}{\partial D} & \frac{\partial - k_1AB - k_2C^2B}{\partial E} \\ \frac{\partial k_1AB - k_2C^2B}{\partial A} & \frac{\partial k_1AB - k_2C^2B}{\partial B} & \frac{\partial k_1AB - k_2C^2B}{\partial C} & \frac{\partial k_1AB - k_2C^2B}{\partial D} & \frac{\partial k_1AB - k_2C^2B}{\partial E} \\ \frac{\partial k_2C^2B - k_3D}{\partial A} & \frac{\partial k_2C^2B - k_3D}{\partial B} & \frac{\partial k_2C^2B - k_3D}{\partial C} & \frac{\partial k_2C^2B - k_3D}{\partial D} & \frac{\partial k_2C^2B - k_3D}{\partial E} \\ \frac{\partial 2k_3D}{\partial A} & \frac{\partial 2k_3D}{\partial B} & \frac{\partial 2k_3D}{\partial C} & \frac{\partial 2k_3D}{\partial D} & \frac{\partial 2k_3D}{\partial E} \end{pmatrix}$$

Cette matrice peut se simplifier en la matrice suivante :

$$\begin{pmatrix} -k_1B & -k_1A & 0 & 0 & 0 \\ -k_1B & -k_1A - k_2C^2 & -2k_2BC & 0 & 0 \\ k_1B & k_1A - k_2C^2 & -2k_2BC & 0 & 0 \\ 0 & k_2C^2 & -2k_2BC & -k_3 & 0 \\ 0 & 0 & 0 & 2k_3 & 0 \end{pmatrix}$$

Sachant que les k_i sont positifs car ils sont des vitesses de réaction, et que les concentrations sont toujours positives, il est possible de retrouver une matrice d'incidence partielle :

$$\begin{pmatrix} - & - & 0 & 0 & 0 \\ - & - & - & 0 & 0 \\ + & ? & - & 0 & 0 \\ 0 & + & - & - & 0 \\ 0 & 0 & 0 & + & 0 \end{pmatrix}$$

Cette matrice n'a qu'une inconnue, c'est l'influence de B sur C car elle est la somme d'un nombre positif et d'un nombre négatif. L'utilisation de la matrice Jacobienne peut nous donner de précieux renseignements sur le graphe d'interactions de réactions chimiques et de processus biologiques.

Leur utilisation en bioinformatique

Goodwin [54] a été un de pionniers dans les années 60 dans l'utilisation des équations différentielles pour modéliser les phénomènes biologiques, en décrivant un processus de régulation où le produit d'une réaction inhibe l'expression d'un gène codant une enzyme qui catalyse une partie de cette réaction.
L'utilisation du modèle différentiel s'est depuis enrichi de nouvelles avancées permettant une meilleure modélisation du phénomène biologique : Yagil [125] a mis en lumière le fait que la synthèse d'une protéine régulée par une autre dépend de la concentration de la

protéine régulatrice pouvait être modélisée par une fonction sigmoïde.
Le fait que le comportement des organismes étudiés puisse changer dès lors que certains seuils sont atteints ou certaines conditions activées a été modélisé par des équations différentielles par morceaux. Leur utilisation par Glass dans les années 70 [52] puis par Snoussi [112] vers la fin des années 80 ont permis d'approcher de façon pertinente le comportement des réseaux de régulation génique.
Les années 90 ont vu émerger un formalisme dans les équations différentielles pour la biologie avec les travaux de Savageau sur les lois de puissance ([100], [101]).
Les applications pratiques des modèles différentielles sont donc variées, nous pouvons citer les travaux de Pécou auxquels nous avons collaborés [21] sur l'homéostasie du cuivre ou encore les travaux de Leloup [71] sur la régulation du rythme circadien de la drosophyle.

2.3 Les modèles stochastiques

Lorsque l'on veut utiliser des équations différentielles pour la modélisation, il faut s'assurer que les quantités que l'ont veut modéliser varient de manière continue. Pour un grand nombre de gènes et de réactions, il est possible de le supposer. Dans les problèmes de reconstruction génétique, les gènes ne sont présents qu'en nombre limité et leur évolution ne peut être considéré comme continue. Les modèles stochastiques ont été utilisées par plusieurs auteurs pour modéliser les transitions entre les états ([33], [38], [12]). Bartholomay a par exemple donné une loi probabiliste sur la concentration d'un espèce dans un réaction au cours du temps lors de réactions. Les modèles stochastiques formalisent l'évolution des systèmes par des transitions entre plusieurs états, transitions affectées d'une probabilité. Les équations régissant ces probabilités sont appelées équations maîtresses. L'exemple ci-après montre l'utilisation de modèles stochastiques pour des réactions chimiques.
Soient les réactions :

$$A + B \rightarrow 2B \tag{2.13}$$

$$B \rightarrow D \tag{2.14}$$

de vitesses de réaction respectives f et d. La concentration de l'élément B suit l'équation maîtresse suivante :

$$\frac{dP_x^B}{dt} = (x-1)fP_{x-1}^B + (x+1)dP_{x-1}^B - x(f+d)P_x^B \tag{2.15}$$

où $P_x^B(t) = Prob[B] = x$, [B] étant la concentration en élément B.
Ces équations, lorsqu'elles sont résolues, permettent de suivre la concentration de chaque élément d'une réaction chimique au cours du temps. Ceci nous permet de simuler et comprendre les phénomènes en jeu. La résolution de ces équations peut être effectuée de

diverses manières, notamment grâce à la méthode de Gillepsie [50].

Leur utilisation en bioinformatique

Gillepsie [50] a tout d'abord étudié et proposé une méthode de résolution des équations stochastiques dans les années 1970. Ce n'est que plus récemment que des recherches dans le domaine des modèles stochastiques pour la bioinformatique ont vraiment abouti, avec des recherches sur la régulation de l'expression du bactériophage lambda [6] et la cascade de phosphorylation impliquée dans un processus de chimiotaxie bactérienne [84]. De plus, des travaux sur les oscillations dans la régulation transcriptionnelle [37] ont abouti à des résultats intéressants. L'équipe de Zhu a beaucoup travaillé sur l'analyse de l'expression de gènes grâce à des modèles stochastiques ([127], [128]) et récemment un outil de simulation de modèles stochastiques a vu le jour : SGNsim [94].

2.4 Les réseaux bayésiens

Un réseau bayésien est un modèle d'apprentissage qui se base sur la théorie des probabilités conditionnelles de Bayes [87]. Un réseau bayésien est modélisé par un graphe acyclique dirigé $G = (V, E)$ qui représente, comme dans un graphe classique, les gènes et leurs interactions. À chaque gène i correspond une variable aléatoire et une distribution conditionnelle prenant en compte les parents de i, les parents de i étant les variables qui régulent directement i. Le réseau se définit donc par deux composantes, un graphe et une distribution de probabilités :

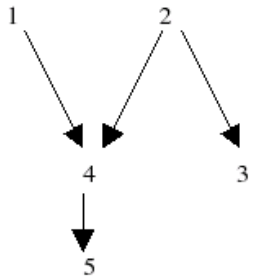


FIG. 2.4 – Graphe d'un modèle bayésien.

p(4|1,2)	(1,2)			
	1,2	1,non(2)	non(1),2	non(1),non(2)
4	0.1	0.6	0.7	0.5
non(4)	0.9	0.4	0.3	0.5

De la même façon, le modèle bayésien fournit le tableau pour les probabilités conditionnelles p(5|4), p(3|2) et non conditionnelles de p(1) et p(2) que nous n'avons pas reproduites ici. La construction du modèle se fait en 2 temps ; la construction du graphe qui se fait de la même façon que pour un graphe classique (soit on connaît déjà les interactions, soit il faut les retrouver à partir des données disponibles). Dans un deuxième temps, il faut construire le tableau de probabilités, qui se construit de la même façon que pour les graphes, soit à partir de connaissances d'experts, soit à partir de données auxquelles notre réseau devra coller au plus près.
La phase d'utilisation permet de réviser les probabilités conditionnelles à la lumière d'observations d'évènements, non plus seulement à partir de probabilités à priori. Par exemple, si l'on sait que 4 est vrai, quelle en est la cause ? Par un jeu de propagation (dans les réseaux bayésien, c'est le fait de réviser les probabilités), il est possible de récupérer cette information.
L'avantage principal de cette modélisation dans le cadre de la modélisation de réseaux de régulation biologique est qu'elle permet de modéliser les aspects stochastiques de l'expression des gènes, ainsi que le bruit inhérent à ces réseaux et aux mesures de données. Un autre avantage de ces modèles est la possibilité de les utiliser sur des réseaux dont nous n'avons qu'une connaissance incomplète (manque de données par exemple). Un désavantage de ces modèles est qu'ils ne modélisent que des états du système et non la dynamique de ce système. Il est donc possible de visualiser les états d'équilibre mais difficile de visualiser comment ils sont atteints. De plus, les réseaux bayésiens ne décrivent que des états stationnaires.

Leur utilisation en bioinformatique

En 1998, Mri et Mian [85] et ensuite Freedman [43] ont utilisé les réseaux bayésiens pour modéliser l'expression des gènes. Il existe des algorithmes pour la recherche de réseaux bayésiens ([113], [60]). Arsenic [57] a démontré grâce aux réseaux bayésiens en 2002 que la protéine TUPI1 a un rôle de répresseur chez la levure.

2.5 Les modèles qualitatifs

Il est souvent très difficile de quantifier dans un réseau les interactions entre deux éléments d'un réseau. De plus, les expériences effectuées ne donnent pas toujours des résultats très précis. Même les outils les plus perfectionnés de la biologie ne permettent pas d'obtenir une fiabilité et une précision adéquats. Pour pallier à ces problèmes, il est possible de considérer les éléments du réseau comme étant présent ou absent du réseau. Dans cette approche booléenne où la présence de l'élément est codée par un 1 et l'absence par un 0, la dynamique du système est définie par des lois logiques qui donnent l'évolution tem-

porelle du système. Nous pouvons distinguer deux types d'approches suivant que l'on traite l'évolution des quantités de manière simultanée ou pas. Ces approches sont appelées "synchrones" et "asynchrones". L'approche asynchrone permet de prendre en compte les délais de réaction lors de l'évolution du système.
Les modèles booléens ne permettent pas de modéliser de façon précise le comportement d'un réseau mais permet de saisir les comportements globaux de ces réseaux, comme les attracteurs ou les cycles de régulation, positifs ou négatifs.
Un exemple de modèle booléen permet de mieux comprendre ces phénomènes. Soit un réseau de deux gènes x et y avec x_t et y_t les valeurs d'activation de ces deux gènes. Les lois d'évolution du réseau sont comme suit :

$$x_{t+1} = non(y_t)$$
$$y_{t+1} = non(x_t)$$

Sans prendre en compte le temps, c'est-à-dire en considérant que les changements s'effectuent de façon synchrone, nous obtenons le tableau suivant :

x_t	y_t	x_{t+1}	y_{t+1}
0	0	1	1
0	1	0	1
1	0	1	0
1	1	0	0

À partir de ce dernier, on peut construire un graphe de dynamique du système 2.5, c'est-à-dire un graphe où tous les états sont présents et où les chemins empruntés par le système sont visibles.

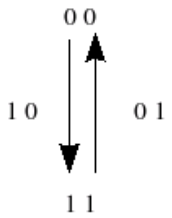


FIG. 2.5 – Dynamique du système synchrone.

Ce graphe (fig. 2.5) montre que le système a trois équilibres possibles, deux points d'équilibre (10 et 01) et un attracteur cyclique, c'est à dire un chemin d'états fermé que le système prend alternativement.
Dans la réalité, ces réactions s'effectuent en un temps donné, et donc les vitesses de réaction peuvent changer la dynamique du processus. Un graphe quelque peu différent est

obtenu en prenant en compte les délais de réaction. Dans le cas asynchrone, le système n'a plus que deux points d'équilibre vers qui il tend en fonction de la rapidité des réactions : 10 et 01.

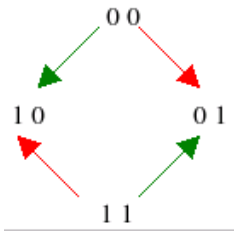


FIG. 2.6 – Dynamique du système asynchrone.

Leur utilisation en bioinformatique

Kaufman ([67], [68]) a été le premier à utiliser les modèles booléens pour les systèmes dynamiques dans les années 1960. Ensuite, Thomas [116] a émis l'idée que la traduction et la transcription des gènes en protéines peuvent être modélisées par par l'asynchronisme dans les modèles booléens. Cette amélioration du modèle a permis la clarification de modèles complexes (par exemple la régulation du bactériophage λ). Ces améliorations ont connu un développement théorique important par van Ham et Lasters [56] qui ont prouvé qu'une cascade d'éléments dans un modèle booléen peut se simplifier en un élément simple dans un modèle.
Dans les années 1990 et 2000, l'approche booléenne a été développée en ajoutant des variables intermédiaires (l'activation des gènes pouvant être modélisée par un nombre d'états supérieurs à deux) qui ont permis un rapprochement entre les modèles différentiels et qualitatifs. Ces développements ont permis d'établir des résultats intéressants, notamment sur le réseau contrôlant la différentiation des organes floraux chez la plante modèle *Arabidopsis thaliana* [80], ou sur le développement embryonnaire de la drosophile ([97], [98]). De même l'équipe de Thieffry a utilisé et développé ce modèle dans le cas de recherches sur la différenciation, la maturation et l'activation de lignées lymphocytaires T [114].
Récemment, le réseau de régulation de *Escherichia Coli* a été modélisé par un système dynamique booléen et a montré un comportement d'homéostasie [99].

Chapitre 3

L'inférence de réseaux génétiques

3.1 Le partitionnement

Les techniques de partitionnement permettent, suivant la valeur d'activation des gènes, de savoir s'ils sont corégulés.
La corégulation de deux gènes signifie que ces deux gènes ont le même profil d'activation. Cela veut dire que leur expression, c'est à dire leur degré d'activité, mesurée par colorimétrie sur des puces à ADN, sont similaires ou évoluent de la même façon. Biologiquement, cela veut dire qu'ils partagent les mêmes régulateurs. Le fait qu'ils partagent les mêmes régulateurs va les faire réagir de la même façon aux mêmes stimuli et donc va leur faire s'activer de façon similaire.

Le partitionnement hiérarchique

La technique la plus simple de regroupement consiste à grouper deux à deux les gènes qui ont les niveaux d'expression proches. Pour savoir quels sont les gènes dont les niveaux d'expression sont les plus proches, il faut d'abord définir une distance entre les gènes qui permette de quantifier cette proximité. Cette distance peut être définie simplement, comme la somme des différences entre les niveaux d'expression par exemple. Mais, suivant le type de données que l'on traite, d'autres distances peuvent être intéressantes, le choix de distance se fait en fonction des données disponibles et de la nature de l'information recherchée.
Dans cette technique, le premier groupe se crée avec les deux gènes les plus proches selon la distance choisie. Ensuite, il faut considérer le nouveau groupe comme une entité à part qui a sa propre distance aux autres gènes, et de nouveau grouper les deux entités (gène ou groupe de gènes) les plus proches. La nouvelle distance du groupe de gènes aux autres groupes peut se calculer de différentes manières. Par exemple on peut prendre le maxi-

mum des distances entre chaque entité du nouveau groupe et les anciens groupes comme nouvelle distance : si la distance entre le gène 1 et le gène 3 est de 1, et que la distance entre le gène 2 et le gène 3 est de 2, alors la distance du groupe (gène 1-gène 2) au gène 3 sera le maximum des distances, soit 2. De la même manière, on peut définir la nouvelle distance en prenant le minimum, la moyenne classique ou pondérée des distances...
Le choix de la distance joue un rôle important dans les résultats, les arbres résultants pouvant être radicalement différents, il faut donc choisir soigneusement la distance.

Les cartes auto-organisatrices.

Les cartes auto-organisatrices se basent sur la théorie des cartes de Kohonen [70] : ce sont des cartes spatiales en dimension N sur lesquelles il existe deux entités, les gènes et les clusters, toutes deux représentées par des points dans cet espace. Un cluster est représenté par un point, un centre qui regroupe un ensemble de gènes. Ce cluster est mobile dans l'espace jusqu'à trouver la position qui lui permette de définir au mieux les ensembles de gènes. Pour cela, les clusters sont attirés par les gènes de façon inversement proportionnelle à leur distance.
L'algorithme de cette technique peut être décrit de la façon suivante :

Initialisation : assimiler les gènes à des points dans un espace à N dimensions. Définir K clusters à des emplacements choisis, ou au hasard. Choisir un nombre d'itérations ou une condition d'arrêt particulière (taille de cluster, distance maximale, nombre de clusters...).

Itération :

1. Choisir un gène gagnant selon une règle pré-établie. Le choix de la règle se fait en fonction des buts recherchés.

2. Rapprocher les clusters du gène gagnant de façon inversement proportionnelle à leur distance.

Les endroits de la carte où les gènes seront les plus nombreux auront normalement tendance à attirer plus les clusters que les endroits vides. Nous obtiendront une carte avec des clusters regroupant les gènes d'expression similaire. La procédure standard veut que les règles d'attraction suivent une Gaussienne. Les figures 3.1, 3.2, 3.3, 3.4 montrent un exemple de définition des clusters dans un espace à deux dimensions.

Après plusieurs itérations au cours desquelles les clusters sont attirés par des gènes différents, nous obtenons la figure 3.4. Cette figure montre les clusters, qui se situent au centre des amas de gènes. Ils forment un assemblage de gènes ayant des expressions similaires. Cette technique, utilisée par Wang [121] pour la recherche de partitions dans le cas de l'étude de lympocytes B ainsi que pour le bipartitionnement par Busygin [22].

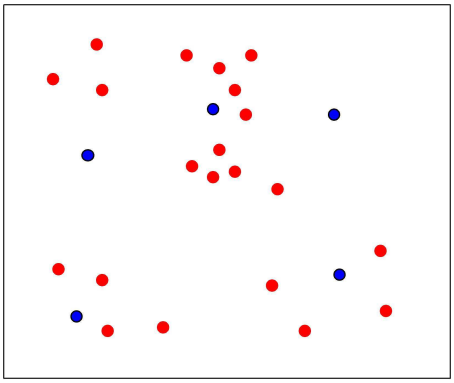


FIG. 3.1 – Position de départ. En rouge, les gènes. En bleu, les cluster.

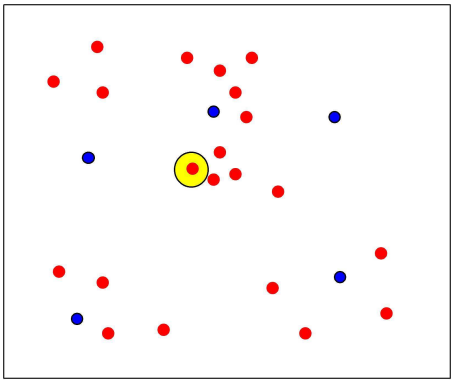


FIG. 3.2 – Gène gagnant, ici on a choisi un gène au hasard.

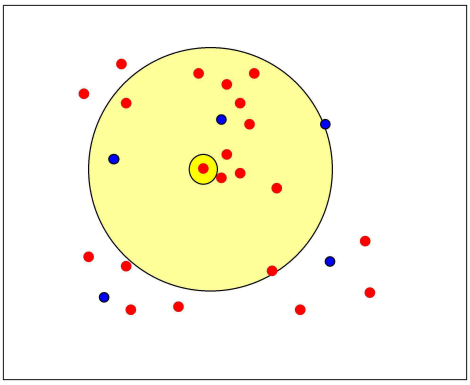


FIG. 3.3 – Voisinage dans lequel les clusters sont attirés.

Le bipartitionnement

Les matrices d'expression des gènes tirées des puces à ADN sont créées en faisant correspondre un gène sur ligne et une condition expérimentale sur chaque colonne. Leur analyse s'est souvent effectuée en deux dimensions, soit en comparant les expressions par ligne

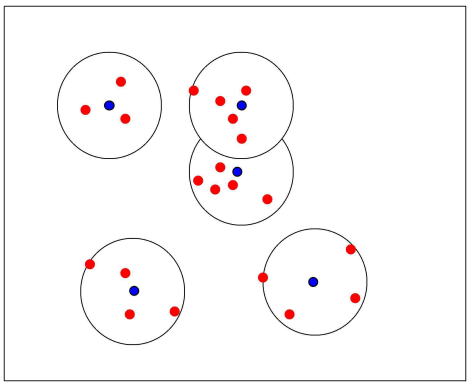


FIG. 3.4 – Position finale.

soit par colonne.
Le bipartitionnement analyse les résultats sur les deux dimensions de la matrice d'expression simultanément. Les groupes de gènes formés par le bipartitionnement permet d'identifier les gènes qui montrent des comportements similaires sous des conditions expérimentales similaires. Cette technique s'applique donc dans des situations précises :
– Un petit groupe de gènes participe à un processus biologique particulier.
– Un processus cellulaire est actif sous certaine conditions.
– Un gène participe à plusieurs processus qui ne s'activent pas sous les mêmes conditions.
Le but du bipartitionnement est d'extraire de la matrice d'expression des sous-matrices montrant de l'homogénéité dans l'expression.
Il existe plusieurs types de bipartitions, en fonction des caractéristiques que l'on prend en compte pour regrouper les gènes (voir la figure 3.5).

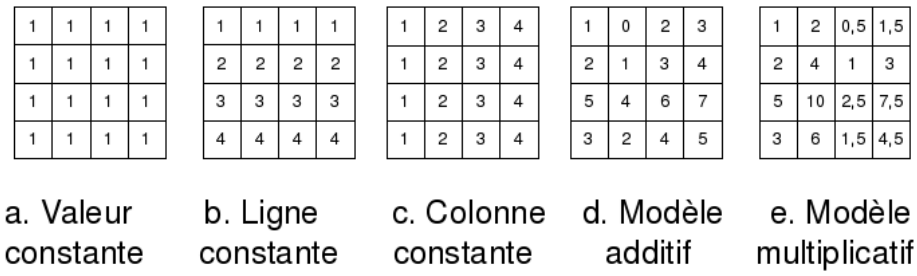


FIG. 3.5 – Exemple de bipartition selon la méthode d'association.

Hartigan [58] a développé un algorithme basé sur les variances pour pouvoir calculer les bipartitions à valeur constante, ce qui permet de prendre en compte le bruit inhérent aux données des puces à ADN. Cet algorithme permet de diviser les matrices en un nombre de bipartitions spécifié à l'avance. D'autres travaux par Tibshirani [115] ont permis de calculer le nombre optimal de clusters grâce à l'utilisation de permutations.

Pour trouver les bipartitions de type b et c de la figure 3.5, Califano [24] a cherché les lignes ou colonnes de telle sorte que la différence entre le minimum et le maximum de la ligne soit inférieure à un certain seuil. D'autres groupes, notamment Sheng [110], ont poussé leur approche vers des structures bayésiennes qui permet de représenter les schémas de bipartition par des modèles probabilistes.
Une approche pour trouver les bipartitions de modèle additif ou multiplicatif est d'analyser la covariance de deux sous-groupes de gènes et de conditions dans la matrice d'origine. Cheng [25] introduisent un score appelé résidu minimum carré (Mean Squared Residue, MSR) qui permet de discriminer les sous-matrices formant des bipartitions.
Les différentes techniques de bipartitionnement ont pour but soit de trouver une seule bipartition soit un ensemble de bipartitions. Dans le cas de la recherche d'un nombre de partitions, connaître leur structure permet de faciliter leur recherche. En effet, les biclusters peuvent se structurer de différentes façons, voir figure 3.6.

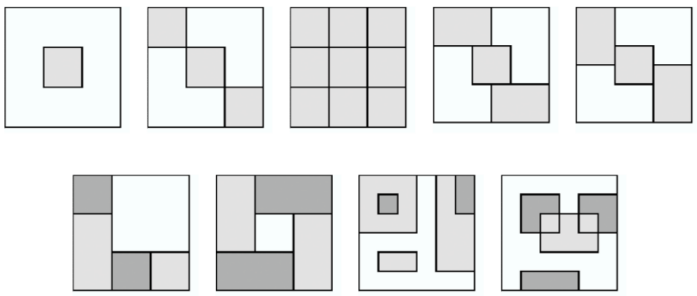


FIG. 3.6 – Quelques exemples de structures de partitions. Tiré de [77].

Le nombre d'applications du bipartitionnement est assez important. De nombreux groupes, Cheng [25] et Wang ([122], [123]) ont appliqués des techniques de bipartitionnement sur la levure de bière *Saccharomyces cerevisiae* comprenant 4026 gènes et 96 conditions expérimentales.
Segal [109] a identifié des groupes de gènes dans des organismes de levure qui participent à des métabolismes connus de même que de nouveaux groupes de gènes. De la même façon, ils ont réussi à assigner des fonctions à des gènes dont la fonction était inconnue.
L'équipe de Getz [49] a identifié un diagnostic possible à la leucémie en identifiant la réponse de l'ARN aux différents traitements proposés. Leur recherche a porté sur l'expression de 6817 gènes humains.
Busygin [22] a utilisé les cartes auto-organisatrices de Kohonen pour trouver des bipartitions. Les conditions et les gènes sont tout d'abord partitionnées grâce aux cartes et ensuite, les deux tables de partitions sont reliées les unes aux autres par une bijection afin d'obtenir un réel bipartitionnement.

Le partitionnement sous contraintes

Le partitionnement de groupes de gènes en partitions a progressé de façon très importante ces dernières années, amenant la possibilité de rechercher des groupes de gènes répondant à certains motifs dans leur expression. La connaissance préalable de ces motifs par les biologistes permet de contraindre le partitionnement.
Une des contraintes les plus simples est d'imposer qu'un cluster contienne certains gènes. Pour ces contraintes simples, il suffit de partitionner les gènes avec une technique classique (hiérarchique, K-means...), et de ne garder comme cluster potentiels que ceux dont les contraintes sont satisfaites, par exemple uniquement ceux dont la taille est assez grande ou ceux qui contiennent les groupes de gènes voulus. Il s'agit donc ici de faire un tri *a posteriori*.
Il est aussi possible de ne tester comme clusters potentiels que ceux qui satisfassent déjà aux contraintes voulues. Dans ce cas, nous parlons de partitionnement *a priori*.
Beaucoup de groupes de chercheurs ont utilisé la théorie des ensembles fréquents comme contraintes pour le partitionnement. Cette théorie a été utilisée pour la première fois par l'équipe d'Agrawal dans les années 90 ([1], [2]). Un ensemble de gènes est dit fréquent si l'ensemble de ses composants ont une expression similaire simultanément. Cette méthode a été utilisée par plusieurs équipes, notamment les équipes de Pfaltz [91], et de Boulicaut ([18], [65]).
D'une manière pratique, la technique nécessite d'utiliser des données booléennes qui vont associer à un gène la valeur 1 lorsqu'il est considéré comme actif ou surexprimé et 0 sinon. La fréquence absolue d'un groupe de gènes est le nombre de conditions dans la matrice où l'ensemble de gènes est surexprimé et la fréquence relative est cette fréquence absolue sur le nombre total de conditions.
Trouver les sous-ensembles contraints nécessite de faire passer les clusters possibles par des filtres qui vont discriminer les clusters qui satisfont les contraintes des autres. Identifier des groupes de gènes fréquents de fréquence assez élevée signifie trouver des ensembles de gènes qui sont corégulés dans un nombre assez important de situations. En comparant la fréquence de deux groupes de gènes dans des conditions expérimentales différentes (souches saines et malades), il est possible de tirer des conclusions intéressantes sur l'organisme étudié. En effet, si un ensemble de gènes est fréquent dans un organisme sain et que dans l'organisme malade le groupe de gènes fréquents est réduit, amputé d'un gène par exemple, il est naturel alors de penser que ce gène a un impact important sur le changement de situation de sain à malade.
Li et Wong [74] ont trouvé des résultats intéressants sur des données de cancer de la Prostate, notamment sur le diagnostic de ce cancer et sur la classification de types de leucémie.

Le partitionnement superparamagnétique

Le partitionnement superparamagnétique utilise la théorie des modèles de Potts. Un modèle de Potts consiste en une série de spins qui sont positionnés sur une matrice. Les clusters apparaissent naturellement dans les modèles de Potts comme des spins alignés. L'algorithme de partitionnement de Domany [15] utilise cette théorie pour partitionner des gènes selon des données de puces à ADN. Cet algorithme attribue un spin S_i à chaque donnée i. Ces spins sont positionnés sur une matrice de dimension q (ici q=20), la position du spin dépendant d'une variable température T. La matrice des distances entre les gènes est créée : $D_{ij} = |X_i - X_j|$ où X_i sont les valeurs des données du gène i. À chaque spin est associé un voisinage de 12 spins qui vont interagir en fonction de la distance, cette interaction est calculée grâce à un couplage ferromagnétique [16] : $J_{ij} = f(D_{ij})$ où f est une fonction décroissante. Pour compléter la fonction, l'interaction entre des spins non-voisins est poussé à 0.
Chaque spin porte une énergie et l'énergie totale de la configuration est donnée par la fonction :

$$\mathcal{H}[\{S\}] = -\sum_{<ij>} J_{ij}[1 - \delta(S_i, S_j)]$$

Une simulation par une méthode de Monte-Carlo est utilisée pour des séries de température différentes. À chaque température est mesurée une corrélation entre deux spins voisins :

$$G_{ij} =< [\delta(S_i, S_j) - 1/q]/[1 - 1/q] >$$

Si i et j appartiennent au même cluster, G_{ij} sera égal à 1 alors que s'ils ne sont pas corégulés, $G_{ij} = 0$. Les <.> indiquent une moyenne sur la température. On considère que si $G_{ij} > 0.5$ les points i et j sont connectés par une arête. À T=0, tous les S_i ont la même valeur et l'algorithme génère un cluster unique composé de tous les gènes. À T=∞, tous les gènes sont positionnés dans des clusters différents. Lorsque T grandit, l'algorithme génère des solutions de clusters dont la taille diminue.
Cet algorithme a comme avantage qu'il détermine par lui-même le nombre de clusters, qu'il est stable par rapport au bruit et qu'il peut identifier des clusters de haute densité et de forme irrégulière. Il peut de même donner une mesure de la stabilité et de la robustesse d'un cluster en identifiant à quelle température un cluster se forme et disparaît. Plus la plage de température à laquelle le cluster existe est grande, plus ce cluster est stable. SPC a été utilisé pour l'analyse de la dépendance en température de l'expression des gènes dans une levure [49]. Cette technique a aussi été utilisée pour identifier la cible d'un répresseur de tumeur et les facteurs de transcription [66] de cette tumeur, responsable de cancer chez l'être humain.

3.2 Une approche dynamique

Trouver des cycles d'attraction ou des points fixes d'un réseau de régulation génétique est très utile dans la reconstruction de réseau car il est possible d'en déduire les sous-graphes du réseau et donc de simplifier le problème ou d'apporter de nouvelles informations utiles sur celui-là.
Aracena et Demongeot [5] ont développé un outil mathématique permettant de retrouver ces sous-graphes. Cet outil part du principe que si un réseau est composé de deux sous réseaux, alors les points fixes du réseau sont la combinaison des points fixes des deux sous-réseaux. Le travail effectué ici consistait donc à prendre le problème inverse, c'est-à-dire à essayer de trouver comment les points fixes peuvent déterminer les sous-graphes.
Dans un premier temps, l'algorithme qui détermine les sous-graphes calcule la matrice de différence des points fixes, cette matrice M est définie comme suit : $m_{xy} = \{j < s : x_j \neq y_j\}$ où s est le nombre d'éléments dans le réseau et x_i est la valeur d'activation du gène i au point fixe x. Par exemple, si un réseau a pour points fixes 001 et 000 (0 codant pour un gène inactif et 1 pour un gène actif), la valeur de la matrice associée pour ces deux points fixes sera : $m_{12} = \{3\}$.
L'algorithme crée ensuite des sous-graphes associés à chaque ligne de la matrice en prenant pour chaque sommet du graphe un gène ou une protéine, et comme arêtes les m_{ij} dont le cardinal est le plus petit jusqu'à ce que tous les sommets soient reliés. Un exemple complet est montré ci-dessous, tiré de [3].
Soit un réseau de 3 gènes dont les points fixes sont les suivants : {000, 001, 100, 010, 011, 110}. La matrice M est la suivante :

$$\begin{pmatrix} - & \{3\} & \{1\} & \{2\} & \{2,3\} & \{1,2\} \\ \{3\} & - & \{1,3\} & \{2,3\} & \{2\} & \{1,2,3\} \\ \{1\} & \{1,3\} & - & \{1,2\} & \{1,2,3\} & \{2\} \\ \{2\} & \{2,3\} & \{1,2\} & - & \{3\} & \{1\} \\ \{2,3\} & \{2\} & \{1,2,3\} & \{3\} & - & \{1,3\} \\ \{1,2\} & \{1,2,3\} & \{2\} & \{1\} & \{1,3\} & - \end{pmatrix}$$

Ensuite, la construction du premier graphe s'effectue en regardant la première ligne puis les colonnes de cardinal le plus petit. Ici c'est la colonne 1 avec {3}. Il y aura une flèche partant de 3 pour aller à 3. Cette opération est effectuée jusqu'à ce que chaque sommet ait été tiré une fois au moins.
Les sous graphes associés sont montrés à la figure 3.7. Cette technique a été utilisée efficacement pour retrouver le réseau génétique d'*Arabidopsis thaliana*.

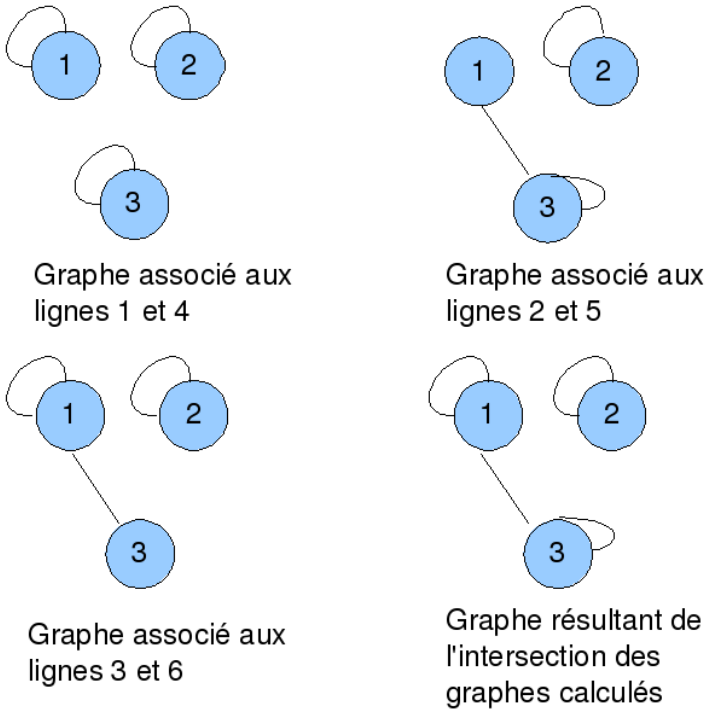


FIG. 3.7 – Graphes associés à chaque ligne et graphe résultant de l'intersection des graphes. Plusieurs lignes peuvent donner naissance à un même graphe.

3.3 Une technique utilisant les forces de régulations

La technique utilisant les forces de régulation est basée sur le travail de de la Fuente et al, dans [31], cette technique est appelée Metabolic Control Analysis (MCA).
MCA amène un formalisme quantitatif qui permet d'extraire les trajectoires des états du système à partir des réactions biologiques. Cette technique permet aussi de mesurer la sensibilité de variables lorsqu'elles sont perturbées. Elle se base sur les états d'équilibre du système lorsque ceux-ci sont modifiés par la perturbation de certaines variables (ici des gènes) connues. MCA se base sur la calcul du niveau d'expression relatif, qui est le ratio de la concentration d'ARNm du gène dans l'état stimulé par rapport à la concentration en état normal.

$$FR = \frac{[ARNm_i]'}{[ARNm_i]} \tag{3.1}$$

où $[ARNm_i]'$ est la concentration de ARNm du gène i modifié,
et $[ARNm_i]$ est la concentration de ARNm du gène i en conditions normales,
À partir de cette quantité, nous pouvons définir deux quantités qui sont le coefficient de co-contrôle et la force de régulation. Le coefficient de co-contrôle mesure comment deux variables changent quand un paramètre est altéré. Le paramètre qu'il faut altérer pour la méthode MCA doit être la vitesse de transcription du gène, notée v_j pour le gène j. Le

coefficient de co-contrôle $O_{i,j}$ du gène i sur le gène j se calcule alors ainsi :

$$O_{i,j} = \frac{\frac{\partial[ARNm_i]}{[ARNm_i]}}{\frac{\partial[ARNm_j]}{[ARNm_j]}} \tag{3.2}$$

La force de régulation exprime comment la perturbation de la vitesse de transcription d'un gène modifie les concentrations de ARNm des autres gènes, elle se calcule ainsi :

$$R_{i,j} = \frac{\frac{\partial v_j}{v_j}}{\frac{\partial[ARNm_i]}{[ARNm_i]}} \cdot \frac{\frac{\partial[ARNm_j]}{[ARNm_j]}}{\frac{\partial v_j}{v_j}} \tag{3.3}$$

Une force de régulation négative indique une inhibition, une force de régulation positive indique une activation, et une force nulle indique qu'il n'y a pas d'interaction. Grâce à cette force, il est possible de créer un graphe associé au réseau et aux forces de régulation dont les sommets vont être les gènes et les arêtes seront la représentation de la force de régulation. Il n'est pas toujours aisé de calculer la force de régulation car il n'est

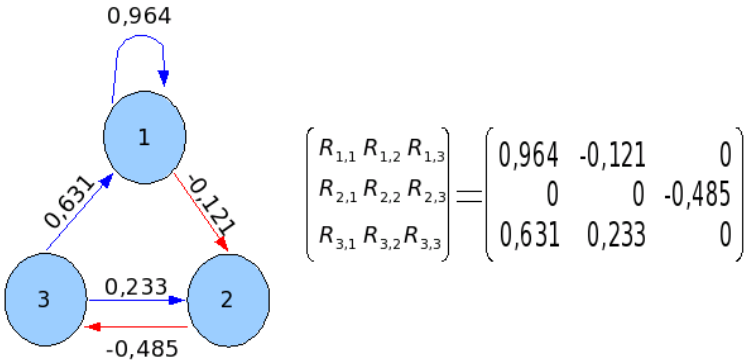


FIG. 3.8 – Exemple de graphe associé à une matrice de force de régulation. À droite est montrée la matrice de régulation, et à gauche le graphe tiré de cette matrice. En bleu sont indiquées les activations et en rouge, les inhibitions.

pas simple d'obtenir l'information sur les vitesses de transcription. Mais, Hofmeyer et Cornish-Bowden [62] ont démontré que la matrice de régulation est égale à l'inverse de la matrice de co-contrôle contenant les coefficients de co-contrôle :$R_{i,j} = O_{i,j}^{-1}$, et le calcul des co-contrôles est beaucoup plus simple à effectuer.
Dans la réalité, nous pouvons uniquement effectuer des perturbations finies. Or le calcul des forces de régulation et des coefficients de co-contrôle se basent sur des calculs infinitésimaux. Il faut donc reformuler les calculs en utilisant des approximations numériques. En utilisant un schéma aux différences finies centré, nous obtenons pour un calcul du type :

$$\frac{\Delta C}{C} = \frac{(C - C_0)}{(C + C_0)/2} \tag{3.4}$$

où C_0 est la concentration référence,
et C est la concentration après perturbation.
Cette fraction peut s'écrire, en divisant le numérateur et le dénominateur par C_0 :

$$\frac{\Delta C}{C} = \frac{2(\frac{C}{C_0} - 1)}{\frac{C}{C_0} + 1} = \frac{2(FR - 1)}{FR + 1} \tag{3.5}$$

En prenant cette écriture, nous pouvons modifier le calcul de $O_{i,j}$:

$$O_{i,j} = \frac{\frac{\Delta[ARNm_i]}{[ARNm_i]}}{\frac{\Delta[ARNm_j]}{[ARNm_j]}} = \frac{(FR_i - 1)(FR_j + 1)}{(FR_j - 1)(FR_i + 1)} \tag{3.6}$$

Une fois $O_{i,j}$ calculé, il nous suffit de l'inverser pour trouver la matrice des forces de régulation. En résumé, la méthode MCA est consignée dans l'algorithme suivant :

1. On choisit les gènes à identifier.
2. On perturbe la vitesse de transcription d'un gène.
3. On mesure les différences d'expression des gènes entre le milieu normal et le milieu perturbé.
4. On calcule les coefficients de co-contrôle de ce gène.
5. On effectue les pas 3-4 jusqu'à ce que la matrice de co-contrôle soit entièrement remplie.
6. On inverse la matrice de co-contrôle pour obtenir la matrice de régulation.
7. On construit le réseau de régulation à partir de la matrice de régulation.

L'avantage de cette technique est que les résultats obtenus sont quantitatifs, car le graphe obtenu est pondéré. De plus, les calculs à effectuer sont assez simples. Une fois les données obtenues, il suffit d'effectuer N^2 calculs pour remplir la matrice de co-contrôle puis d'effectuer une inversion de matrice, chose qu'il est assez aisé de faire avec les techniques classiques d'inversion et les puissances de calcul disponibles. Le problème principal de cette technique est qu'elle est basée sur des données très spécifiques et qu'il est difficile de les obtenir. Ces données ne sont pas forcément présentes pour tous les réseaux que nous voudrions reconstruire grâce à cette méthode. De plus, le fait d'utiliser les différences finies peut entraîner une perte de précision très importante en fonction des données disponibles.

3.4 La technique utilisée dans l'algorithme Reveal

La technique, utilisée dans l'algorithme Reveal de Roland Somogyi, présentée dans [111], se fonde sur la théorie de l'information et permet de déterminer pour chaque gène les gènes qui le régulent.

L'entropie de Shannon d'une variable aléatoire discrète X, avec N états possibles, [1..N] dans un univers Ω, est définie comme suit :

$$H(X) = -\sum_{i=1}^{N} p_i log(p_i) \tag{3.7}$$

où p_i la probabilité d'apparition de l'évènement i pour la variable aléatoire X, p(X=i). voir Shannon & Weaver (1963) [124]. Dans le cas de deux variables aléatoires X, prenant des états entre 1 et N, et Y, prenant des états entre 1 et P, nous pouvons définir l'entropie de Shannon :

$$H(X,Y) = -\sum_{i=1}^{N}\sum_{j=1}^{P} p_{i,j} log(p_{i,j}) \tag{3.8}$$

avec $p_{i,j}$ la probabilité d'apparition de l'évènement $\{X = i, Y = j\}$, p(X=i,Y=j). Nous pouvons aussi définir les entropies conditionnelles d'une variable par rapport à l'autre : H(X|Y) et H(Y|X) qui sont reliées par la relation suivante :

$$H(X,Y) = H(Y|X) + H(X) = H(X|Y) + H(Y) \tag{3.9}$$

H(X|Y) est l'information contenue dans X qui n'est pas partagée avec Y. L'algorithme Reveal se base sur la propriété suivante : si H(X|Y)=0, il n'y a pas d'information dans X qui ne soit partagée avec Y. En d'autres termes, connaître Y suffit à connaître toute l'information contenue dans X. Cela signifie que Y détermine complètement X. En généralisant à plusieurs variables, il se peut qu'une variable dépende complètement d'un groupe entier de variables, ce qui est souvent le cas dans le domaine biologique. Nous avons dans ce cas retrouvé les régulateurs du gène X.
Le nombre de variables nécessaires pour déterminer une variable se nomme la connectivité de la variable déterminée.
Les entrées nécessaires à l'algorithme sont :
- le nombre d'éléments dans le réseau à tester,
- le nombre d'expériences,
- les résultats des expériences,
- la connectivité maximale K, qui est le nombre maximal de variables nécessaires pour déterminer les autres variables.

Premier pas : pour chaque élément X à tester, initialiser la connectivité (appelée C) à 1 et faire le calcul de la quantité H(Y)-H(X,Y) pour tout élément Y différent de X. Si cette quantité est nulle, alors l'élément Y détermine complètement X. En effet, d'après 3.9

$$H(Y) - H(X,Y) = 0 \Rightarrow H(X|Y) = 0$$

Dans l'idéal, le programme devrait s'arrêter dès qu'il trouve un élément (ou dans les cas suivants un couple ou groupe d'éléments) qui satisfasse H(Y)-H(X,Y)=0. Les éléments

qui ne satisfont pas à cette règle sont les éléments non encore déterminés.
Deuxième pas : on augmente la connectivité des éléments X non détérminés. $C = C + 1$. On calcule la quantité H(X)-H(X,Z) avec Z étant un ensemble de C éléments que l'on va tester comme régulateurs de X. Pour cela, on va calculer H(Z)-H(X,Z). Si H(Z)-H(X,Z)=0, alors l'ensemble des éléments contenus dans Z vont réguler X. On teste pour Z toutes les combinaisons de C éléments parmi l'ensemble des éléments, afin de balayer l'ensemble des groupes possibles de régulateurs.
Quatrième pas : répéter l'opération 2 avec des connectivités de plus en plus élevées jusqu'à détermination de tous les éléments ou jusqu'à ce que $C = K$.

Cette méthode nous donne des résultats intéressants car, avec suffisamment d'expériences, nous sommes sûrs d'obtenir un résultat optimal. Malheureusement, la quantité d'informations disponibles n'est que très rarement suffisante et, de plus, avec des grands réseaux, le nombre de tests à effectuer devient vite extrêmement important et devient donc coûteux en temps.

3.5 Les algorithmes génétiques

3.5.1 Leur émergence

L'idée à l'origine des algorithmes génétiques remonte aux années 50. Elle a émergé au travers des premières tentatives de modifier des programmes ou des paramètres avec du (pseudo)hasard, et d'évaluer le résultat, par Nils Aall Barricelli ([10], [11]) et Alex Fraser [41]).
Le problème d'alors était la limitation en puissance des machines pour l'implémenter. En effet, pour pouvoir modifier ces paramètres et chercher une solution à ce problème, il fallait un espace mémoire important pour stocker les informations et une certaine rapidité, du fait des itérations successives nécessaires à la recherche d'une solution. Les ordinateurs de l'époque, encore principalement articulés autour de tubes à vide, n'étaient pas assez puissants pour résoudre ces problèmes. Dans les années 60, les biologistes se penchent sur la simulation informatique de l'évolution jusqu'à ce que cette méthode soit décrite par Fraser et Burnel [42] et Crosby [29]. Les simulations de Fraser incluaient les éléments essentiels des algorithmes génétiques modernes : la sélection, le croisement et la mutation. En parallèle, Hans Bremermann [19], [20], publia dans les années 60 une série d'articles qui utilisaient une population de solutions qui évoluaient pour résoudre des problèmes d'optimisation, qui sous-entendait la mutation, le croisement et la sélection. Cette méthode permettait de choisir dans la population de solutions les solutions les plus acceptables pour ensuite essayer de les modifier légèrement pour les améliorer.
L'évolution artificielle, qui consiste à faire évoluer vers plus de complexité des solutions dans le but d'optimiser des problèmes, devint une méthode d'optimisation reconnue

lorsque les travaux de Ingo Rechenberg et Hans-Paul Schwefel ont été publiés vers la fin des années 60 et le début des années 70 - son groupe était capable de résoudre des problèmes complexes d'ingénierie au travers de stratégies d'évolution ([105], [106], [107]). Une autre approche de l'algorithmie génétique est due à Lawrence J. Fogel [39] et sa technique de programmation évolutive. La programmation évolutive utilisait originellement des machines à états finis pour prédire l'environnement. Cette programmation utilisait la sélection et les variations (assimilables à la mutation et au croisement) pour optimiser le problème. L'algorithmie génétique devint populaire avec le travail de John H. Holland dans le début des années 70 [61]. Les travaux de Holland et de ses étudiants à l'université du Michigan sur les automates cellulaires leur permirent d'approfondir l'idée des algorithmes se rapprochant des processus biologiques. Holland formalisa une méthode pour prédire la qualité des générations futures, connue comme le Théorème du schéma de Holland. La recherche en algorithmie génétique est restée longtemps théorique, par manque de puissance des machines, jusqu'au milieu des années 80 lorsque la première conférence internationale sur les algorithmes génétiques a eu lieu à Pittsburgh en Pennsylvanie.
L'intérêt grandissant pour ces techniques et la multiplication de l'accès à l'outil informatique ont permis des percées dans ce domaine, jusqu'à la première mise sur le marché d'un produit à base d'algorithmes génétiques par General Electric en 1980, une boîte à outils basée sur un serveur central utilisé pour des questions industrielles. Depuis lors, les algorithmes génétiques sont sortis de leur confidentialité académique et sont utilisés pour résoudre des problèmes divers et variés.
Certaines recherches sur le thème ont abouti à quelques résultats théoriques intéressants, notamment les travaux de Cerf [26] qui a démontré que la convergence des algorithmes génétiques vers le maximum d'efficacité nécessitait une taille de population supérieure à un seuil critique dépendant très fortement de la taille du problème. Cerf a aussi prouvé que ce seuil est plus petit qu'une fonction linéaire du nombre de gènes de l'individu. Ces résultats ont permis d'être sûrs que le problème est solvable en temps machine raisonnable.

3.5.2 Leur formalisation

Les algorithmes génétiques sont des algorithmes d'apprentissage basés sur des processus évolutionnaires et guidés par des fonctions d'efficacité. Un algorithme génétique étant un outil d'optimisation, il faut tout d'abord définir le problème que nous voulons optimiser. Pour cela, il nous faut choisir le modèle que nous voulons reconstruire et ses paramètres. Les algorithmes génétiques se basent sur le principe d'évolution comme énoncé par Charles Darwin dans "De l'origine des espèces" (1859) : *"In the struggle for survival, the fittest win out at the expense of their rivals because they succeed in adapting themselves best to their environment"*. Le but des algorithmes génétiques est de combiner les points forts de chaque individu pour en créer de nouveaux dont l'efficacité est meilleure. Pour illustrer

cela, si nous voulons un cochon gras et résistant aux maladies, nous allons croiser ensemble un parent gras et un autre résistant aux maladies, le but étant bien sûr qu'ils transmettent la bonne partie de leur génome à leur enfant.
L'algorithmie génétique utilise un langage spécifique que nous allons développer.
- Un individu est un jeu de valeurs des paramètres du modèle.
- Le génome d'un individu est l'ensemble de ces paramètres.
- Un gène est un paramètre particulier d'un individu.
- Une population est un ensemble d'individus.
- Une génération est une population a un moment donné.
- Un parent est un individu choisi comme base pour en créer un autre de la génération suivante.
- Un simulateur est un outil qui permet au modèle de fournir des résultats en fonction de l'individu.
- L'efficacité d'un individu est sa capacité à rapprocher les résultats du modèle des informations disponibles sur le problème. Pour cela, on compare les informations disponibles aux informations données par le simulateur.

Le schéma classique d'un algorithme génétique est de choisir une population (composée d'individus) et, à partir de cette population, de créer une nouvelle génération. Comme pour les êtres vivants, pour engendrer un membre de la génération suivante, il faut deux parents dont on va croiser le génome. Les parents seront choisis en fonction de leur efficacité. Pour ajouter un brassage génétique, ce membre va connaître des mutations qui s'effectueront sur les gènes de l'individu, stockés dans son génome, qui est conçu comme l'ensemble des paramètres d'un individu. Un certain nombre de ces nouveaux membres va constituer la seconde génération de la population. Si l'on choisit des parents efficaces, il est très probable que les enfants aient une efficacité au moins aussi importante que leurs parents ; c'est le principe de sélection et croisement des espèces animales ou végétales pour un meilleur rendement.

3.5.3 Les applications de l'algorithmie génétique en bioinformatique

L'utilisation des algorithmes génétiques pour l'inférence de réseaux génétiques vient du fait qu'ils procèdent à une recherche globale de solutions en évitant les minima locaux et qu'ils peuvent estimer de nombreux paramètres variant dans des plages de valeurs importantes [53]. Pridgeon a expliqué en 2004 [92] que les algorithmes évolutionnaires dont font partie les algorithmes génétiques étaient particulièrement efficaces dans la reconstruction de réseaux génétiques. Les algorithmes génétiques ont d'ailleurs été utilisés pour l'estimation de paramètres de systèmes biologiques, notamment dans des modèles métaboliques

de régulation allostérique [51], ou encore pour l'estimation de paramètres de modèles dynamiques de réactions biologiques [86].
Les applications étant nombreuses, nous allons en détailler deux de façon plus précises, l'un portant sur un modèle bayésien [30], et l'autre autre sur un modèle différentiel [32].

Le modèle bayésien. L'équipe de d'Alché Buc a beaucoup travaillé sur l'inférence de réseaux génétiques utilisant les algorithmes évolutionnaires ([7], [48], [89]). En particulier, ils ont développé une méthode permettant d'inférer les paramètres d'un modèle bayésien de réseau génétique à partir de données statiques.
Le modèle à inférer est un modèle bayésien classique qui est défini par un couple (G,P) avec G un graphe acyclique et P la probabilité de passer d'un sommet à un autre. Les paramètres à identifier dans ce modèle sont donc ces G et P. Afin d'identifier ces paramètres, il est tout d'abord nécessaire de définir la fonction d'évaluation d'une configuration précise des paramètres. Cette fonction a été choisie comme la composée d'un terme relatif aux données et d'un terme relatif à la complexité du modèle. La fonction choisie dans ces travaux est la BIC (Bayesian Information Criterion), développée par Shwarz en 1978 [104] qui ajoute une pénalité en fonction du nombre de termes dans le modèle.

$$BIC(G) = \sum_i \sum_k \sum_l -2N_{jk}^l log(\widehat{\theta_{ik}^l}) + K^i log(s) \tag{3.10}$$

où s est la taille du réseau à retrouver, $\widehat{\theta_{ik}^l}$ est l'estimation du maximum de vraisemblance de θ_{ik}^l, $\theta_{ik}^l = P(X_i = k|P_{a_i} = l)$, P_a sont les parents du gène X_i, k et l sont des états possibles des gènes et de ses parents, N_{jk}^l les occurrences des états $X_i = k$ et $P_{a_i} = l$ dans les données et K^i le nombre de paramètres.
Un algorithme génétique classique est appliqué aux réseaux bayésiens potentiels, choisissant deux parents dans une population initiale, les croisant pour produire deux enfants, mutant les enfants, puis renouvelant la population initiale en enlevant les deux individus avec la moins bonne efficacité.
Le modèle a été testé sur des données d'homéostasie du glucose et a fourni des résultats comparables, voire meilleurs, par rapport à d'autres algorithmes connus tels que les algorithmes gourmands, notamment lorsque peu de données sont disponibles.

Le modèle différentiel. Deng et son équipe ont développé un algorithme génétique qui permet de découvrir les paramètres d'un modèle différentiel de réseau génétique qui se nomme EXAMINE.
Le modèle différentiel proposé ici modélise la concentration en ARN messager d'un gène en fonction des régulateurs de ce gène. Nous posons v_i comme étant la concentration d'ARNm du gène i, m_i le facteur d'échelle du gène i, w_{pi} la force de la régulation entre le

gène p et i, r_i le taux de dégradation de l'ARNm du gène i, b_i est une source de produit i, et S une fonction sigmoïde. La concentration en ARNm du gène i est définie par la fonction suivante :

$$\frac{\partial v_i}{\partial t} = m_i S(w_{pi} v_p + ... + w_{qi} v_q + b_i) - r_i v_i \tag{3.11}$$

m_i et S permettent d'éviter aux concentrations de tendre vers l'infini ou d'obtenir des valeurs de concentration négative. b_i permet de modéliser des effets qui sont en-dehors du problème auquel on se restreint.
Les données auxquelles nous avons accès sont la concentration d'ARNm des gènes considérés, il est donc aisé de comparer les valeurs des concentrations que l'on obtient avec l'algorithme $(v_i(t))$ avec celles des données $(u_i(t))$. Dans cet algorithme, les données utilisés sont nécessairement temporels car l'algorithme utilise l'évolution des concentrations dans le temps. La fonction d'efficacité associée au modèle différentiel nous est donnée par l'équation suivante :

$$efficacite = \frac{1}{1 + \sum_{i=0}^{N} \sum_{t=0}^{T} (u_i(t) - v_i(t))^2} \tag{3.12}$$

Cette fonction est comprise entre 0 et 1 et est d'autant plus grande que les données sont proches.
Alors que dans l'exemple précédent la connectivité était un élément de la fonction d'efficacité grâce au critère BIC, ici la connectivité sera un des paramètres de l'algorithme. En effet, Deng a développé un algorithme génétique qui boucle sur la connectivité. Tous les gènes sont initialisés avec une connectivité de deux. Un algorithme génétique est lancé afin de retrouver gène par gène les paramètres définis plus haut. Si la solution a une efficacité suffisante (par rapport à un seuil défini par l'utilisateur), alors le gène est retiré de la liste des gènes à définir. Sinon, l'algorithme boucle en augmentant la connectivité. Cet algorithme permet de s'assurer que toutes les solutions trouvées sont celles dont la connectivité est minimale.
Des tests ont été effectués avec cet algorithme sur un ensemble de réseaux et de données créés artificiellement. Cela a permis de calculer l'efficacité de l'algorithme en fonction de plusieurs facteurs, en particulier la taille du réseau à retrouver et la taille des données disponibles. Les résultats sur des petits réseaux sont très bons avec plus de 90% de relations retrouvées par rapport aux réseaux artificiels créés.
Cet algorithme a aussi été testé sur le système nerveux du rat, et en particulier sur 112 gènes appartenant à ce système. EXAMINE a retrouvé les mêmes résultats que des études antérieures sur ces mêmes gènes notamment sur le partitionnement de ces gènes en cinq clusters pertinents.

3.6 Algorithmes de comparaison

Comme nous l'avons vu précédemment, il existe plusieurs méthodes algorithmiques pour la reconstruction de réseaux. J'ai choisi de présenter trois algorithmes de reconstruction que nous allons utiliser au chapitre 5 afin de comparer les scores de ces méthodes aux scores donnés par notre méthode.

3.6.1 Le logiciel Banjo

Banjo est un algorithme de reconstruction de réseaux développé par le groupe de Hartemink [126]. Banjo est basé sur des méthodes bayésiennes qui ont démontré des capacités intéressantes dans le domaine de la reconstruction de réseaux. En effet, ces méthodes permettent de traiter des données bruitées et de retrouver les cycles (rétroactions) et des indications sur le sens des interactions. Les limitations de l'algorithme Banjo sont dues au fait qu'il requiert un nombre très important de données et, de plus, le problème de détermination du signe et de l'intensité des interactions peut être difficile à résoudre avec les réseaux bayésiens dynamiques. Banjo propose une technique de simulation de réseaux biologiques pour pouvoir déterminer la précision des résultats donnés par l'algorithme en comparant les résultats donnés par le simulateur et par le logiciel. Cette comparaison se résume au calcul d'un score d'influence qui permet d'estimer l'efficacité de la solution proposée. L'algorithme comporte deux parties, une partie simulation et une partie reconstruction de réseaux, que nous présentons ci-après.

Le simulateur de réseaux biologiques de Banjo

Le simulateur de réseaux biologiques de Banjo, nommé GeneSim, produit les expressions de gènes à des temps discrets. Le calcul des expressions se fait par le biais du processus stochastique suivant :

$$Y(t+1) - Y(t) = f(Y(t)) = A(Y(t) - T) + \epsilon \tag{3.13}$$

où Y(t) est l'expression des gènes au temps t,
A est la matrice du réseau,
T est le seuil au-delà duquel le gène est dit actif et participera à la régulation des autres gènes,
et enfin ϵ est le bruit des données biologiques, tiré aléatoirement entre -10 et 10.
Les données initiales sont prises au hasard. Ensuite, les expressions sont discrétisées entre 0 et 100, ce qui permet d'éviter les expressions négatives des gènes ou leur croissance sans limite, mais aussi de générer des effets non-linéaires. Le simulateur GeneSim permet aussi de récupérer des données créées à partir d'un réseau connu.

La partie reconstruction de réseau de Banjo

La partie reconstruction de réseaux est conçue pour la recherche de réseaux dont les résultats des simulations, grâce à GeneSim, sont le plus proche possible des données disponibles. Cette similitude entre les données réelles et la simulation est définie par un score. La méthode Banjo va choisir les réseaux succeptibles d'être les solutions du modèle en fonction de ce score.
Dans cette approche, les réseaux génétiques sont représentés par des réseaux bayésiens dynamiques, de type Markov du premier ordre. Un réseau bayésien est un graphe dans lequel les noeuds représentent des variables aléatoires, et les arcs (le graphe est donc orienté) reliant ces dernières sont rattachées à des probabilités conditionnelles. Un réseau bayésien est dit dynamique lorsque on rajoute un lien causal d'un pas de temps à l'autre. Dans ce réseau, chaque nœud représente une variable, ici un gène et chaque liaison une interaction. De plus, l'état de chaque nœud est influencé par l'état des nœuds au temps précédent. Un score est calculé pour chaque réseau, ce score permet d'évaluer la probabilité que le réseau a d'expliquer correctement les données observées. Deux fonctions de score sont utilisées, la "Bayesian Dirichlet equivalence" (BDe) et le "Bayesian Information Criterion" (BIC). Le score BIC est une approximation asymptotique du score BDe. La BDe est basée sur le calcul de probabilité a posteriori qui se calcule par [59] :

$$log(p(D, S^h)) = log(p(S^h)) + log(p(D|S^h)) \tag{3.14}$$

où D désigne les données disponibles et S^h, le réseau, p étant la distribution des évènements D et S^h. Ce score a deux composantes, la composante à posteriori ($p(S^h)$) et la ressemblance marginale ($p(D|S^h)$). La composante a posteriori peut se définir comme la probabilité d'avoir le réseau S^h, indépendemment des données connues. La ressemblance marginale peut se comprendre comme la probabilité d'obtenir les données D en ayant le réseau bayésien S^h.
Une fois calculé le score de chaque réseau, il faut utiliser un algorithme de recherche pour permettre de récupérer le réseau qui a le score le plus important. Pour cela, Banjo peut utiliser trois techniques différentes :
– les algorithmes gourmands,
– le recuit simulé,
– les algorithmes génétiques.
Un algorithme gourmand recherche une solution meilleure a chaque phase du problème (moment, lieu, variable quelconque). Cet algorithme se base sur l'espérance qu'une succession d'optimisations locales vont aboutir à une optimisation globale [8]. Par exemple, le problème de rendre 6 €31 avec le moins de pièces possible parmi 5, 2, 1, 0.5, 0.2, 0.1, et 0.01 € se résoudrait de la manière suivante avec un algorithme gourmand :
– Chercher la pièce la plus grande qui soit inférieure à 6 €31. Il s'agit de 5 €.
– Réduire le problème à chercher à rendre 1 €31.

– Choisir 1 €. Le problème se réduit à 31 cents.
– Choisir 20 cents. Le problème se réduit à 11 cents.
– Choisir 10 cents. Le problème se réduit à 1 cents.
– Choisir 1 cent. Le problème se réduit à 0 cent.

Le nombre de pièce serait donc de 5.

Le recuit simulé s'appuie sur l'algorithme de Metropolis [82], qui permet de décrire l'évolution d'un système thermodynamique. Par analogie avec le processus physique, la fonction à minimiser deviendra l'énergie E du système. On introduit également un paramètre fictif, la température T du système. Partant d'une solution donnée, en la modifiant, on en obtient une seconde. Soit celle-ci améliore le critère que l'on cherche à optimiser, on dit alors qu'on a fait baisser l'énergie du système, soit celle-ci le dégrade. Si on accepte une solution améliorant le critère, on tend ainsi à chercher l'optimum dans le voisinage de la solution de départ. L'acceptation d'une mauvaise solution permet alors d'explorer une plus grande partie de l'espace de solution et tend à éviter de s'enfermer trop vite dans la recherche d'un optimum local.

Banjo propose les trois approches, qui donnent des résultats similaires, mais laisse le choix à l'utilisateur d'utiliser l'un ou l'autre suivant ses besoins.

Afin de pouvoir reconstruire le réseau avec les signes des interactions et leur intensité, Banjo utilise un score d'influence. Ce score d'influence,

$$\theta_i^{j,k} = P(X_i = k | pa(X_i) = j),$$

permet de calculer la probabilité pour que le nœud X_i soit dans l'état k lorsque son parent ($pa(X_i)$) est à l'état j. Basiquement, s'il y a une forte probabilité que l'enfant soit à un état haut quand le parent est dans un état haut puis que l'état de enfant soit bas lorsque celui du parent est bas, alors le parent active sûrement l'enfant.

En combinant ces éléments, Banjo permet de reconstruire des réseaux génétiques, il crée un graphe signé indiquant comment les gènes sont régulés.

3.6.2 Le logiciel Aracne

Aracne [78] est un algorithme qui utilise l'information mutuelle comme base de départ (cf. paragraphe 3.4). En entrée il prend des informations sur chaque gène du réseau à reconstruire. L'algorithme calcule alors l'information mutuelle de chaque couple de gènes et garde les couples dont l'information mutuelle est inférieure à un certain seuil. Cela permet de ne garder que les candidats possibles aux couples (régulateur-régulé). Mais il arrive dans certaines conditions que deux gènes n'ayant pas d'interactions communes aient quand même une information mutuelle basse, ceci étant dû à l'existence d'un gène intermédiaire qui modifie peu l'action du premier gène sur le deuxième. Ce phénomène va générer un certain nombre de faux positifs. Pour remédier à ce problème, Aracne filtre les

interactions indirectes grâce à la propriété de la théorie de l'information appelée inégalité de traitement de l'information (data processing inequality ou DPI [27]) qui stipule que si un gène g_3 est activé par un gène g_1 par l'intermédiaire d'un gène g_2 alors :

$$M(g_3, g_1) \leq min[M(g_2, g_1), M(g_3, g_2)] \tag{3.15}$$

L'algorithme Aracne compare donc tous les triplets pour s'assurer que les interactions sont directes. Aracne permet donc de récupérer des interactions non signées. Aracne a la particularité d'être un algorithme qui permet d'être sûr que le réseau récupéré est juste si les informations données en entrée sont suffisantes, ce qui est un avantage appréciable. Mais, comme la plupart des informations disponibles sont partielles, l'algorithme retombe dans les mêmes travers que tous ses congénères, à savoir qu'il ne procure qu'une relative sécurité au niveau de la réponse obtenue.

3.6.3 Le logiciel NIR

Le logiciel NIR (Network Identification by multiple Regression) [45] utilise la théorie des équations différentielles pour modéliser le réseau génétique. Il considère que le système d'équations est linéaire lorsque l'on se rapproche du point d'équilibre. Ensuite, il estime les paramètres du modèle en observant le système après l'avoir perturbé, en changeant les conditions expérimentales.
Le modèle est, pour chaque gène :

$$\frac{dx}{dt} = Ax + u$$

où x est la concentration des gènes, u est la perturbation extérieure, A est la matrice de régulation. Les variables x et u sont estimées par l'expérimentation, par contre, la matrice A reste à déterminer.
Pour cela, on considère le système à l'équilibre car, sinon, $\frac{dx}{dt}$ est plus compliqué à calculer et il est plus difficile de contrôler les expériences. Dans le cas de l'équilibre, nous avons :

$$\frac{dx}{dt} = 0 = Ax + u \text{ et donc } Ax = -u$$

Il nous reste maintenant à déterminer la matrice A. Comme nous avons plusieurs expériences avec plusieurs stimuli, nous pouvons considérer X et U comme des matrices ayant N lignes et M colonnes, avec N le nombre de gènes du système et M le nombre d'expériences prises en compte. L'équation devient alors : $AX = -U$.
Pour chaque expérience nous pouvons scinder cette équation matricielle en :
$a_i^t X = -U$ où a_i est la i^{eme} colonne de A.
Cela nous donne un systèmes d'équation qui n'admet pas une solution unique. En effet, à cause du caractère bruité des données, des erreurs de mesure et des imperfections du

modèle, le système d'équations n'est pas parfait car les paramètres des équations changent en fonction de ces bruits. Pour calculer cette matrice A, NIR utilise une méthode des moindres carrés qui lui permet de minimiser les erreurs [47]. On retombe alors dans des techniques connues d'inversion de matrice.

Deuxième partie

COGARE

Chapitre 4

L'algorithme COGARE

4.1 Les principes de construction de l'algorithme

L'algorithme de reconstruction de réseaux est appelé COGARE (COnstrainted Genetic Algorithm for Reverse Engineering), traduction latine de "rassembler", parce que l'algorithme rassemble les informations venant de sources diverses pour trouver le réseau. Il est basé sur deux actions : la collecte d'informations et leur utilisation pour trouver une solution.
L'algorithme est basé sur le principe de recherche de fonctionnement d'un mécanisme biologique à partir des résultats biologiques de ce processus, en particulier les données d'expression des gènes. Pour ce faire, il est nécessaire de savoir ce que l'on cherche, le modèle, et ce dont on dispose, à savoir les données disponibles. Dans notre cas, nous avons des données sur l'état des gènes sous certaines conditions et nous cherchons à comprendre comment ces résultats ont pu être obtenus. De par les spécificités de notre problème (cf. paragraphe 1.6), il paraît difficile d'appliquer à la reconstruction de réseaux génétiques des méthodes déterministes classiques telles qu'une descente du gradient. Ces techniques nécessitent des données fiables qui n'ont qu'une variabilité restreinte. Les problèmes biologiques ne permettent pas d'obtenir de telles données. De plus la taille du problème entraînerait des temps de calcul beaucoup trop longs pour résoudre le problème en un temps raisonnable. Enfin, les fonctions que l'on peut extraire de ces problèmes sont pour la plupart non linéaires, non convexes, et empêchent donc l'utilisation de techniques classiques.
Pour toutes ces raisons, le choix de la méthode de reconstruction pour COGARE s'est porté sur les algorithmes génétiques.
L'algorithme génétique a besoin d'une fonction d'efficacité qui va permettre de noter les solutions possibles. Cette efficacité ne peut se faire que par comparaison des résultats extraits du candidat-solution aux données expérimentales. Pour extraire ces résultats il faut un simulateur qui permette de modéliser le fonctionnement d'une solution. L'al-

gorithme génétique modifie ensuite les solutions pour tenter de retrouver les meilleures solutions au problème, pas à pas. Nous allons voir à présent les deux sections qui composent l'algorithme, la section simulation et la section reconstruction qui permet d'utiliser les simulations pour améliorer les solutions au problème.

4.2 Les données utilisées par COGARE

Plusieurs techniques ont été développées afin de calculer les expressions de gènes lors d'expériences. En particulier les puces à ADN [23] sont devenues un outil classique pour la compréhension des fonctions des gènes, de leur régulation et de leurs interactions. Une puce à ADN se présente sous la forme d'une puce sur laquelle ont été collés plusieurs dizaines de milliers de brins d'ADN. Les puces à ADN se servent des propriétés physiques

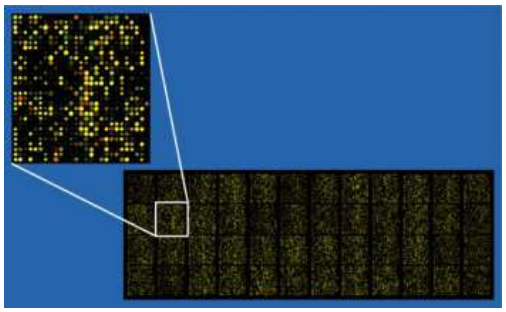


FIG. 4.1 – Exemple de puce à ADN. Source : Wikipédia.

et chimiques des molécules d'ADN. Chaque brin d'ADN se lie avec des molécules suivant sa conformation topologique. Les molécules s'attachent au brin d'ADN et l'activent, cette activation peut être retrouvée par colorimétrie, l'intensité de la fluorescence donnant l'intensité de l'activation du gène. Le processus de fabrication d'une puce à ADN commence par la fabrication d'une lame de verre sur laquelle on dépose les brins d'ADN dont on veut suivre les réactions. Deux réactifs de couleur différentes Cyanine 3 (vert) et Cyanine 5 (rouge) sont ajoutés afin qu'ils se lient aux brins d'ADN. Ils vont permettre de suivre l'évolution de ces brins d'ADN grâce à la colorimétrie.
Avec un scanner, l'intensité de la réaction est mesurée et, grâce à des procédés de reconnaissance informatique, il est possible de récupérer des données brutes. L'intensité de l'expression d'un gène en fonction d'un stimulus donné est alors récupérée.

Les types de données

Les données expérimentales sont utilisées dans le programme dans le but de retrouver le réseau génétique. Suivant le type de données que l'on utilise, l'information apportée n'est pas la même. Trois grands types de données sont disponibles ; les données dynamiques, les données statiques et les données complémentaires. Ces trois types de données sont traitées de manière différente, grâce à la fonction d'efficacité qui varie d'un type de données à l'autre. Ceci permet de disposer de plus de sources d'informations qui vont apporter au logiciel des informations différentes qui lui permettront de retrouver plus efficacement le réseau.

Les données dynamiques

Les données dynamiques sont des données qui ont été obtenues sur la même expérience mais à des temps différents et qui apportent donc des informations sur l'évolution temporelle du système. Pour obtenir ces données, il suffit de reproduire n fois la même expérience et de récupérer les données à des instants différents. Ces données sont très utiles car elles renseignent sur l'évolution de l'organisme, il est donc plus facile d'extraire de ces données des indices sur la dynamique du système et sur les interactions entre les gènes. Ces données sont souvent utilisées dans les logiciels pour tirer profit de cette caractéristique. Par exemple, les logiciels basés sur l'information mutuelle utilisent exclusivement ce type de données. Toutes les réactions internes à l'organisme se faisant simultanément, de manière continue, l'intérêt de ces données est principalement qualitatif, et l'échelle de temps n'a qu'une importance relative.

Les données statiques

Les données statiques sont des données obtenues pour des expériences différentes et donnent l'état du système sous des conditions particulières. Chaque expérience donne un jeu de données qu'il est impossible de lier (en tout cas de façon simple) aux autres expériences. Ces données donnent moins d'indices sur les évolutions et les interactions des gènes entre eux, mais donnent des indices sur la corégulation des gènes (le fait que deux gènes aient les mêmes régulateurs). Les données statiques sont donc des données ponctuelles et sans lien avec les autres données. Peu de logiciels peuvent traiter ces données autrement que par partitionnement, afin de regrouper les gènes corégulés, par exemple les logiciels à base d'équations différentielles nécessitent des données dynamiques car ils cherchent à évaluer la dérivée partielle de chaque quantité en fonction du temps. COGARE permet de calculer l'efficacité d'un partitionnement et donc d'utiliser ces données pour l'algorithme génétique.

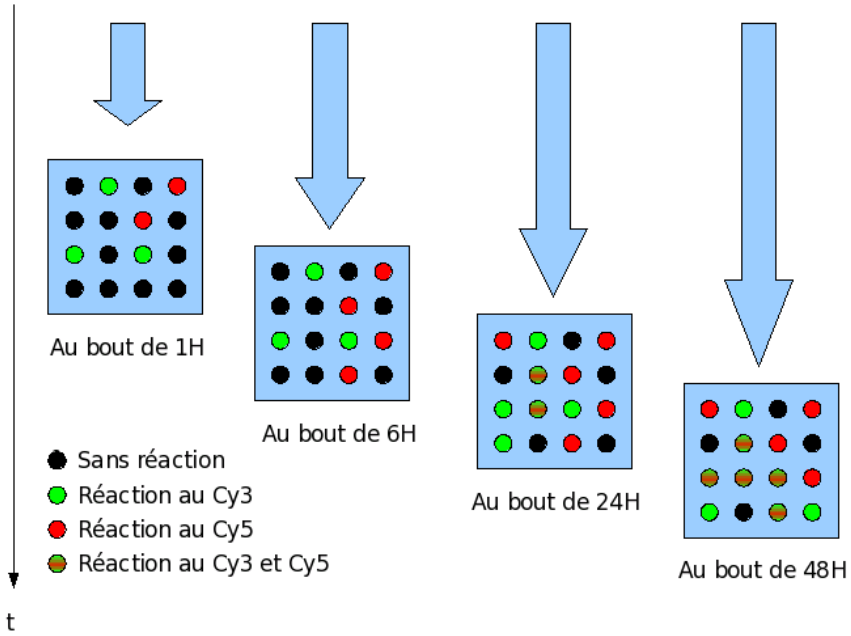
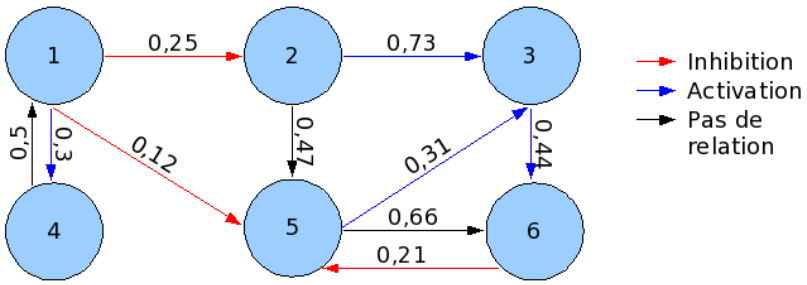


FIG. 4.2 – Exemple d'acquisition de données dynamiques.

Les données complémentaires

Les données complémentaires sont des données fournies par des biologistes. Elles correspondent à des informations que les biologistes ont du mécanisme biologique étudié, une interaction entre deux gènes ou une absence d'interaction. Elles sont formatées sous forme d'une matrice. À chaque élément de la matrice correspond la fiabilité de l'information (de 0,01 à 0,99) plus un entier donnant le type d'interaction ;
0 pour l'information : "pas d'information sur cette relation" ;
1 pour l'information : "pas d'interaction" ;
2 pour l'information : "activation" ;
3 pour l'information : "inhibition".
Le manque d'information sur une interaction sera codé par un 0. Il est très facile de récupérer l'information réelle à partir de cette matrice ; par exemple, un 2.21 à l'élément (2,3) de la matrice d'information sera interprété comme une probabilité de 21% d'avoir une activation du gène 2 sur le gène 3. Cette matrice sera interprétée par le programme afin de favoriser sa convergence vers une solution optimale. En effet, notre logiciel va utiliser les informations complémentaires pour optimiser sa recherche de solution. Un exemple de passage d'informations connues par les biologistes en matrice utilisable par le logiciel est montré à la figure 4.3.

Ces données peuvent être récupérées de plusieurs manières. La première est la récupération d'informations sur le réseau grâce aux biologistes. La seconde est de récupérer ces



0	3,25	0	2,3	3,12	0
0	0	2,73	0	1,47	0
0	0	0	0	0	2,44
1,5	0	0	0	0	0
0	0	2,31	0	0	1,66
0	0	0	0	3,21	0

FIG. 4.3 – Réseau connu par les biologistes et matrice associée. Les flèches bleues signalent une activation, les flèches rouges une inhibition. Les flèches noires signalent que les biologistes pensent qu'il n'y a pas d'interaction entre les deux gènes. Les nombres sur les flèches indiquent la fiabilité de l'information.

informations grâce à une analyse des données au préalable. Cette analyse peut se faire grâce à d'autres techniques de reconstruction qui peuvent faire ressortir certaines caractéristiques du réseau que l'on réinjectera dans le programme.
De même, il est possible de boucler COGARE pour retrouver certaines relations qui paraissent sûre afin de remettre à jour à chaque itération les informations complémentaires.

4.3 La section simulation

La section simulation permet d'obtenir des points de comparaison avec les données réelles disponibles. En effet, cette section permet de simuler le fonctionnement d'un réseau afin d'en extraire les états du système dans une configuration donnée. Ensuite, ces états peuvent être comparés aux états du système fournis par les données expérimentales. Cette comparaison permet de calculer l'efficacité d'une solution. Pour connaître l'état du système au temps suivant il faut prendre en compte l'état initial et les différentes actions des gènes sur le gène considéré. Les gènes régulateurs agissent sur leurs gènes-cibles si et seulement si les régulateurs sont actifs, on peut écrire la règle régissant cette action : l'état d'un gène à l'instant t+1 est égal à son état à l'instant t plus la somme du signe de toutes les flèches pointant vers ce gène multiplié par l'état du gène à l'autre bout de la flèche :

$$E_g^{t+1} = E_g^t + \sum_{fl} sgn(fl) * E_{gm}^t \tag{4.1}$$

où E est l'état du gène à un temps donné,
g est le gène considéré,
t est le temps,
fl désigne les flèches pointant vers g,
et gm sont les gènes activant g.
Ceci nous donne un degré d'activation qui peut être négatif ou supérieur à 1. Or nous voulons des données binaires, il faut donc les traiter pour retrouver des données au format voulu. Les résultats passent donc par une fonction seuil qui pousse à 0 les gènes dont l'activation est négative ou nulle et à 1 pour les gènes dont l'activation est supérieure à 1. Ces règles sont visibles sur les exemples des figures 4.4 et 4.5 ci-dessous.

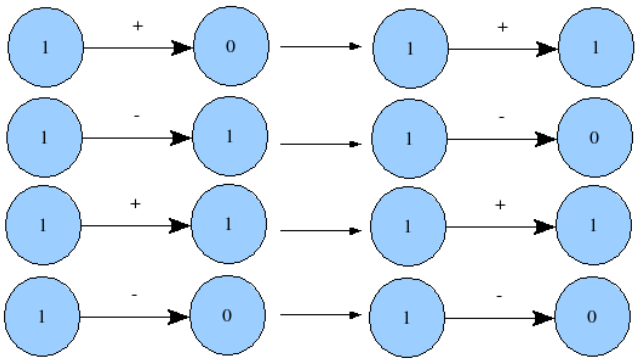


FIG. 4.4 – Évolution de réseaux après simulation : le cas de mono-actions.

Cette approche booléenne de la simulation n'est pas parfaite car elle ne rend pas compte de la force de la régulation, c'est à dire de l'intensité de l'interaction entre deux gènes.

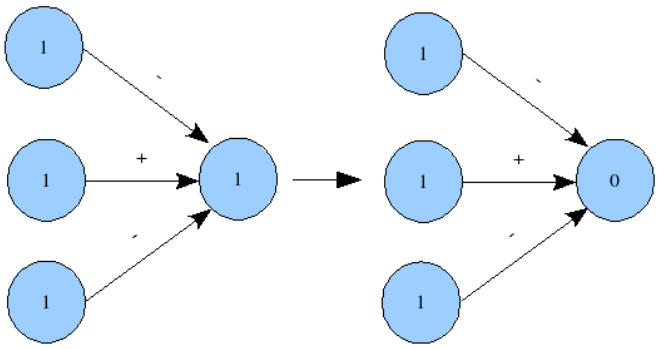


FIG. 4.5 – Évolution de réseaux après simulation : le cas de trois gènes agissant sur un gène. Le gène d'état 1 subit deux inhibitions et une activation, son état passe alors à 0, donc le gène est inhibé.

Cette simulation permet de récupérer l'état du système au cours du temps. Cet état va nous permettre, une fois comparé aux données disponibles, de reconstruire au mieux le réseau.

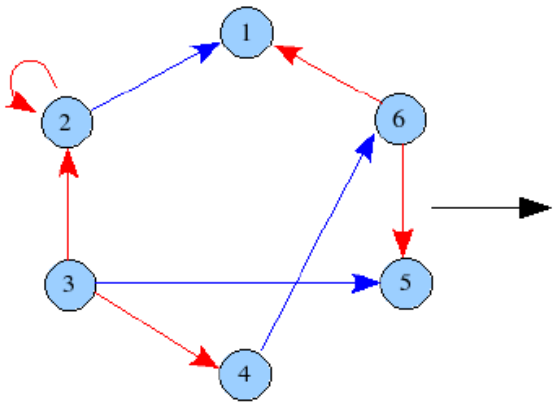
4.4 La section reconstruction

La section reconstruction est la section de calcul de l'algorithme, celle qui va fournir les réseaux potentiels qui vont correspondre au mieux aux données disponibles. Cette section est basée sur un algorithme génétique dont les paramètres sont optimisés par rapport au problème. Les différents paramètres à optimiser sont le taux de mutation, la fonction d'efficacité, le type de sélection et le type de croisement à utiliser.

La matrice d'incidence

Tout d'abord, il est nécessaire de comprendre comment est représentée une solution possible de réseau. Il est facile de représenter un réseau par une matrice selon la règle suivante : si i active j alors $M_{(i,j)} = 1$, si i inhibe j, $M_{(i,j)} = -1$ sinon, $M_{(i,j)} = 0$. Tout le travail de reconstruction se fera à partir de cette matrice (fig.4.6) appelée matrice d'incidence.

Cette représentation permet de faciliter les calculs pour la simulation d'un réseau ou pour les fonctions de l'algorithme génétique. En effet, les calculs matriciels sont plus simples à effectuer et le stockage des informations par matrice facilite grandement la portabilité du logiciel et des données.



	1	2	3	4	5	6
1	0	0	0	0	0	0
2	-1	1	0	0	0	0
3	0	1	0	1	-1	0
4	0	0	0	0	0	-1
5	0	0	0	0	0	0
6	1	0	0	0	1	0

FIG. 4.6 – Exemple de graphe de réseau transposé en matrice.

Le calcul de l'efficacité

La fonction d'efficacité est essentielle au bon fonctionnement de l'algorithme génétique, c'est elle qui va décider du chemin à emprunter pour optimiser la solution. Pour calculer l'efficacité d'une solution – d'un réseau génétique – il faut comparer l'efficacité de la solution par rapport à chaque type de données. En effet, nous avons trois types de données présentes dans l'algorithme, les données statiques, les données dynamiques, et les données complémentaires, fournies par les biologistes. Ces trois types de données vont pouvoir être comparés aux résultats fournis par l'algorithme, et ce de façons différentes, car ces données n'ont ni les mêmes formats, ni les mêmes spécificités. Pour les informations complémentaires, l'efficacité peut se calculer simplement. L'information complémentaire est la connaissance d'un chiffre de la matrice de réseau, de façon plus ou moins sûre. L'efficacité d'une stratégie particulière pour les informations complémentaires, notée $\mathcal{E}_1$, se calcule de la manière suivante :

$$\mathcal{E}_1 = \frac{\sum_{i \in D} R_i * \delta_{I_i,G}}{\sum_i R_i} \tag{4.2}$$

où i est le numéro de l'information connue,
D est l'ensemble des informations connues,
R_i est le degré de fiabilité de l'information i,
I_i est l'information,
G est le résultat donné par l'algorithme,
et $\delta_{i,j}$ est le symbole de Kronecker qui vaut 1 si i=j et 0 sinon.
Cette efficacité est comprise entre 0 et 1 et est d'autant plus grande que les résultats de l'algorithme restreint à D sont proches des informations complémentaires.
Pour les données dynamiques, la simulation sera mise à contribution. Les données dynamiques comportent deux parties, l'état des gènes à l'instant t et l'état des gènes à

l'instant t+1. Grâce à la simulation, nous allons simuler le fonctionnement du réseau à partir de l'instant t. La simulation va permettre d'obtenir l'état des gènes à l'instant t+1. Si le réseau testé est le réseau réel, l'état des gènes simulé doit correspondre aux données disponibles. Nous pouvons comparer dans un contexte dynamique les données et ainsi calculer l'efficacité d'un réseau par rapport aux données dynamiques. Pour cela, il faut comparer une à une toutes les données et comptabiliser celles qui sont identiques :

$$\mathcal{E}_2 = 1 - \frac{\sum_{i=1}^{M}\sum_{j=1}^{N} \|Q_{i,j} - G_{i,j}\|}{M * N} \tag{4.3}$$

où M est le nombre d'expériences,
N est le nombre de gènes,
$Q_{i,j}$ est l'état du gène j simulé depuis l'expérience j,
et $G_{i,j}$ est l'état du gène j dans l'expérience j.
Cette équation montre que si deux résultats sont identiques, la valeur de l'efficacité est augmentée de 1/(M*N). De même que pour $\mathcal{E}_1$, cette efficacité est comprise entre 0 et 1 et est d'autant plus grande que les résultats de l'algorithme s'accordent avec les données expérimentales.
Pour les données statiques, le principe est le même, mais n'ayant pas de point de départ, d'instant initial, il est nécessaire de développer un nouveau moyen de comparer les résultats pour trouver l'efficacité des solutions par rapport à ces données. Pour trouver une fonction d'efficacité performante, il faut en trouver une avec une caractéristique particulière : elle doit être insensible aux modifications temporelles dans les données. J'ai choisi d'utiliser le partitionnement pour calculer cette efficacité. En effet, le partitionnement est insensible aux variations des données ; il dépend exclusivement du réseau. Le partitionnement groupe les gènes par fonction, par ressemblance. Quelque soit l'instant que l'on choisi pour les données, le partitionnement va retrouver les fonctions des gènes. L'idée est de comparer les partitionnements entre les données temporelles disponibles et des données issues du réseau que l'on veut tester. Pour créer des données à partir du réseau, nous allons utiliser le simulateur : on choisit aléatoirement plusieurs états du système, on fait tourner le simulateur sur ces états plusieurs fois et on réalise un partitionnement à partir des données ainsi récupérées. Le partitionnement de ces données est enfin comparé au partitionnement des données statiques fournies.
Dans COGARE, le partitionnement utilisé est particulier : pour chaque gène on calcule sa distance aux autres gènes et on ordonne ces gènes par distance croissante.
On récupère donc pour chaque gène une liste de gènes ordonnés par ordre croissant des distances. Ensuite, on compare les deux listes pour les expériences statiques et pour les données recréées, cette comparaison normalisée donne l'efficacité de la solution par rap-

port aux données statiques.

$$\mathcal{E}_3 = 1 - \frac{\sum_{i=1}^{N}\sum_{j=1}^{N} 1 - \delta_{O_{i,j},P_{i,j}}}{N * N} \tag{4.4}$$

où $O_{i,j}$ est la place du gène j dans la liste des distances du gène i pour les données expérimentales,
$P_{i,j}$ est la place du gène j dans la liste des distances du gène i pour les données reconstruites
et $\delta_{i,j}$ est le symbole de Kronecker.
Si les ordres de partitionnement sont les mêmes, alors $\mathcal{E}_3$ est nul et l'efficacité est maximale, égale à 1. Sinon, pour chaque différence dans l'ordre de partitionnement, l'efficacité diminue.
Une fois ces efficacités calculées, pour trouver l'efficacité totale, on fait une moyenne pondérée par l'importance relative que l'on accorde à chacune de ces efficacités (cette pondération est l'objet d'une recherche au chapitre 5.1 et fait partie des paramétrables de COGARE). Soient α, β et γ ces poids (α, β, $\gamma \geq 0$), l'efficacité totale d'une solution est définie par :

$$\mathcal{E} = \frac{\alpha * \mathcal{E}_1 + \beta * \mathcal{E}_2 + \gamma * \mathcal{E}_3}{\alpha + \beta + \gamma} \tag{4.5}$$

Cette efficacité va nous aider à définir un croisement et une sélection qui vont favoriser la convergence de l'algorithme vers une solution acceptable.

La technique de sélection

La sélection des individus en vue de la reproduction se fait par tirage aléatoire selon les distributions données par l'efficacité ; plus l'efficacité d'une solution est bonne (plus elle est proche de 1), plus la chance de la tirer au hasard est grande. Le tirage au hasard se fait suivant la distribution empirique de l'efficacité relative des individus. Voici la présentation de l'algorithme de sélection :

1. On calcule l'efficacité de chaque candidat à la sélection.
2. Pour tout candidat i, on lui associe la valeur :

$$d(i) = \frac{\sum_{j=1}^{i} eff(j)}{\sum_{j=1}^{N} eff(j)} \tag{4.6}$$

où eff(j) est l'efficacité du candidat j ; N est le nombre de candidats. La quantité d(i) est comprise entre 0 et 1.

3. On tire un nombre nd aléatoirement entre 0 et 1.
4. On choisit le candidat i tel que

$$d(i-1) \leq nd < d(i)$$

.

La sélection devant servir au croisement, il faut effectuer deux sélections, et la deuxième se fait en retirant le premier candidat choisi du groupe de candidats, pour éviter d'itérer sur un candidat fixe.

La technique de croisement

Une autre des spécificités des réseaux biologiques est leur interconnectivité, c'est-à-dire qu'il existe en général plusieurs processus biologiques qui interagissent entre eux, créant des hiérarchies de réseaux, avec un processus agissant sur un autre et, à l'intérieur de ce processus, des gènes interagissant entre eux, etc. Il faut donc pouvoir optimiser le réseau non seulement d'un point de vue global mais aussi d'un point de vue local.
Pour ce faire, le croisement a été optimisé de façon à ce que soit conservé, lors du croisement, l'ensemble des gènes qui interagissent entre eux et que le croisement transmette aux générations futures les meilleures combinaisons de gènes de chaque parent, plutôt que de prendre des gènes au hasard. Les groupes de gènes qui modélisent bien une partie du réseau forment en général des processus biologiques que l'on a identifié dans notre réseau. Le croisement s'effectue donc en favorisant la préservation des processus internes à l'organisme les plus efficaces.
Cette optimisation permet de gérer les processus biologiques et de ne pas couper aléatoirement les génomes des parents, comme le fait un croisement classique en algorithmie génétique. Ce nouveau type de croisement permet d'éviter qu'un processus modélisé efficacement par un parent ne soit transmis qu'à moitié à son enfant.
Un processus local est un groupe de gènes qui sont fortement liés entre eux, et peu avec l'extérieur. On prend en compte cette spécificité afin de retrouver ces processus. De même, il faut penser à l'efficacité du processus dans la modélisation de notre réseau. Le choix de découpe du génome se fait en considérant la performance de groupes de gènes. COGARE va créer des groupes de gènes dont l'efficacité, lorsqu'ils sont isolés des autres gènes, est la plus grande possible. Cette performance s'évalue en combinant l'efficacité de groupes de gènes avec un calcul de connectivité des groupes. L'efficacité d'un groupe de gènes se calcule en séparant le groupe de gènes des autres et en appliquant les mêmes formules que pour l'efficacité globale.
L'efficacité par rapport aux données connues se calcule en prenant uniquement les données complémentaires qui appartiennent au groupe de gènes dont on veut calculer l'efficacité locale, et en appliquant la formule pour l'efficacité globale. On compare donc le sous-réseau

calculé au sous réseau dont les connexions sont connues par les biologistes.

$$\mathcal{E}_{\{Gr\},1} = \frac{\sum_{i \in \{Gr\}} R_i * \delta_{I_i,G}}{\sum_i R_i} \tag{4.7}$$

L'efficacité par rapport aux données temporelles se calcule en réduisant les données disponibles et le réseau au groupe de gène local que l'on veut tester. Ensuite, l'efficacité locale se calcule en prenant la formule de l'efficacité globale appliquée à ce groupe de gènes. Pour garder une efficacité comprise entre 0 et 1, on divise par le nombre de gènes dans le groupe (dans la formule Card(Gr)) par le nombre d'expériences qui n'est pas modifié par le calcul local.

$$\mathcal{E}_{\{Gr\},2} = 1 - \frac{\sum_{i \in \{Gr\}} \sum_{j=1}^{N} \|Q_{i,j} - G_{i,j}\|}{Card(Gr) * N} \tag{4.8}$$

L'efficacité locale par rapport aux données statiques se calcule de la même façon que l'efficacité globale en restreignant le réseau au groupe de gènes à tester. De la même façon que pour $\mathcal{E}_{\{Gr\},2}$, M est remplacé par Card(Gr).

$$\mathcal{E}_{\{Gr\},3} = 1 - \frac{\sum_{i \in \{Gr\}} \sum_{j=1}^{N} 1 - \delta_{O_{i,j},P_{i,j}}}{Card(Gr) * N} \tag{4.9}$$

L'efficacité totale d'un groupe est calculée par la somme pondérée des mêmes coefficients α, β et γ que pour l'efficacité globale.

$$\mathcal{E}_{\{Gr\}} = \frac{\alpha * \mathcal{E}_{\{Gr\},1} + \beta * \mathcal{E}_{\{Gr\},2} + \gamma * \mathcal{E}_{\{Gr\},3}}{\alpha + \beta + \gamma} \tag{4.10}$$

où Gr est le groupe de gènes dont on veut calculer l'efficacité.
La connectivité d'un groupe se calcule en sommant les liens à l'intérieur du groupe et en retranchant les liens des gènes vers des gènes extérieurs au groupe :

$$\mathcal{C}_{\{Gr\}} = \sum_{i \in Gr} \sum_{j \in Gr} \delta_{m(i,j),1} + \delta_{m(i,j),-1} - \sum_{i \in Gr} \sum_{j \notin Gr} \delta_{m(i,j),1} + \delta_{m(i,j),-1} \tag{4.11}$$

Cette équation fonctionne de la manière suivante : si dans la matrice d'incidence (m(i,j)), il y a 1 ou -1 entre deux membres du groupe, alors on ajoute un à la connectivité ; et s'il y a 1 ou -1 entre un membre du groupe et un gène non-membre, on retranche un. La performance d'un processus est donc la paire (efficacité locale, connectivité) dont il faut maximiser les valeurs pour obtenir les meilleurs processus. Cette performance va permettre de définir les groupes de gènes.
Pour identifier les groupes de gènes que l'on veut transmettre en entier, groupes qui formeront un processus, il existe trois approches possibles :

– ou bien on connaît les processus grâce aux biologistes,
– ou bien on teste tous les groupes d'un certain nombre de gènes pour voir s'ils forment un processus,
– ou bien on recherche des groupes d'un certain nombre de gènes grâce au partitionnement.

La taille minimale et maximale du groupe de gènes qui peuvent former des processus est une variable paramétrable dans le logiciel. Nous avons choisi pour ces paramètres des valeurs de 5 pour la taille minimale, et de 15 pour la taille maximale. Lorsque les processus sont connus par les biologistes, il suffit juste de calculer l'efficacité locale du processus pour savoir si ce processus est intéressant à propager tel quel. En effet, la connectivité du groupe a déjà été utilisée par les biologistes et il n'est pas nécessaire de la réutiliser. Nous faisons donc confiance aux biologistes pour la définition du groupe de gènes formant un processus biologique, il ne nous reste qu'à déterminer si ce groupe est intéressant à propager.
Le test de tous les groupes de gènes se fait assez aisément mais utilise un temps de calcul très long (plusieurs années) car il nécessite de tester un nombre très important de combinaisons. Pour une taille de groupe allant de 5 à 15 gènes, il est de $\sum_{i=5}^{15} C_i^N$, ce qui pour un réseau de 300 gènes donne $1.5.10^{26}$ groupes dont il faut calculer l'efficacité, ce qui est impossible à effectuer en temps raisonnable. Il faut donc, tant que les performances des outils informatiques ne sont pas suffisantes, utiliser une autre méthode.
La troisième méthode pour retrouver un processus utilise le partitionnement afin d'éviter de faire des calculs trop longs. La méthode consiste à choisir le gène ayant l'efficacité locale maximale (pour le calcul, le groupe se résume au gène seul) pour un réseau donné. Ce gène sera le pilier du groupe. À partir de ce gène, on ajoute au groupe les gènes ayant les expressions les plus proches de celles du gène pilier par le partitionnement utilisé précédemment au paragraphe 4.4. Il en résulte plusieurs groupes comportant un certain nombre de gènes. Ces groupes sont ensuite départagés par l'indice de performance : tout d'abord on filtre les meilleurs processus par l'efficacité locale, et ensuite on cherche parmi ces groupes, celui dont la connectivité est maximale.

Pour utiliser à bon escient le principe de propagation des processus biologiques bien modélisés, nous allons utiliser la technique de croisement. Voici les étapes effectuées pour le croisement par COGARE :

1. Définir les processus biologiques des réseaux parents.
   – Initialiser le groupe de gènes dans lequel choisir les processus, appelés gènes-candidats : c'est l'ensemble des gènes du réseau.
   – Choisir le premier parent.
   – Définir un processus biologique soit en utilisant les processus donnés par les biol-

ogistes soit, s'il n'y en a pas, déterminer un processus par partitionnement.
   - Si l'efficacité de ce processus est supérieure à un seuil fixé, paramètre de COGARE, valider le processus. Enlever les gènes inclus dans le processus du groupe de gènes-candidats.
   - Définir un processus biologique pour le deuxième parent dans le groupe de gènes candidats. Le valider si son efficacité est supérieure au seuil.
2. Pour chaque parent : effectuer une sélection de la partie du génôme à transmettre en fonction de l'efficacité locale du processus, plus elle est grande, plus le processus (ou le gène) a de chances d'être transmis. On associe aux processus biologiques des parents une distribution basée sur l'efficacité, grâce à laquelle on tirera aléatoirement un processus à transmettre.
3. Enlever la partie du génome transmise du groupe de gènes à transmettre. Si le gène faisait partie d'un processus de l'autre parent, casser ce processus, c'est à dire isoler chaque gène du processus et les considérer indépendamment les uns des autres.
4. Revenir au 2 jusqu'à ce que l'enfant ait un génome complet.

Pour expliquer le processus, prenons un exemple : deux parents ayant le premier un processus, les gènes 1 et 3 et le second, deux processus, gènes 2 et 4, et gènes 5 et 6, qui sont bien modélisés et que l'on veut transmettre. En transposant à la matrice représentant le graphe, ce que chaque parent a à transmettre est l'ensemble de ses coefficients. Le schéma ci-dessous (fig.4.7) montre les différentes parties à transmettre, chaque partie est codée par une couleur différente, et l'on considère que ce réseau a deux processus à transmettre, l'un comprenant les gènes 1 et 3 et l'autre les gènes 4 et 5.

Pour des raisons de clarté, le croisement de deux parents est expliqué sur des schémas. Chaque matrice représente un parent comportant 6 gènes. Chaque processus (groupe de gènes ou gène indépendant) est indiqué par des chiffres chez le parent 1 et des lettres chez le parent 2. Pour plus de lisibilité, les processus comportant plus d'un gène sont indiqués en couleur.

Un premier tirage aléatoire portant sur le parent 1 donne le processus comportant les gènes (1,3) à transmettre. Les coefficients marqués par un 1 dans la matrice sont transmis à l'enfant.

Ensuite le tirage aléatoire porte sur le parent 1. Le coefficient 7 a été tiré. Il est donc transmis à l'enfant. Ce coefficient faisant partie d'un processus chez le parent 2 et étant déjà transmis, il n'est plus possible de transmettre le processus marqué par la lettre H du parent 2. Ce processus est donc cassé en quatre coefficients que l'on peut dorénavant transmettre indépendemment les uns des autres.

Ainsi de suite, nous allons tirer aléatoirement un coefficient de chaque parent à tour de rôle. Il en résultera un enfant dont les coefficients auront été tirés pour que les gènes qui

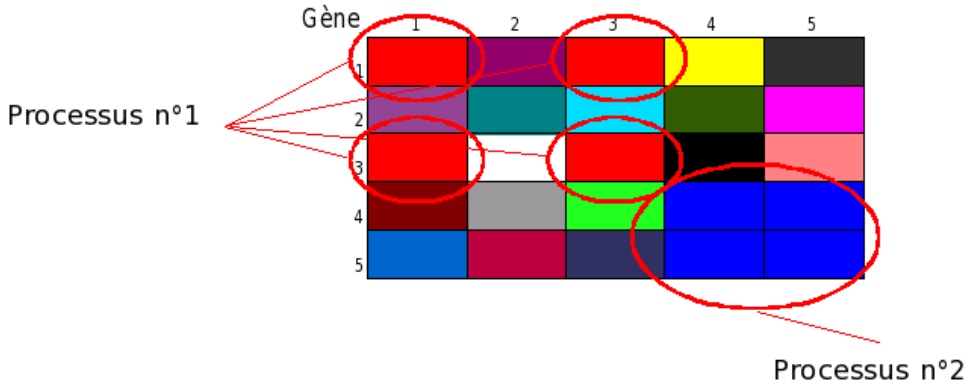


FIG. 4.7 – Représentation des processus sur une matrice d'incidence. Les couleurs définissent chaque groupe de gènes chez un parent. Ce parent à un processus (1,3) (en rouge) et un processus (4,5) (en bleu) et le reste sont des gènes indépendants (de différentes couleurs).

Parent 1

1	2	1	3	4	5
6	7	8	9	10	11
1	12	1	13	14	15
16	17	18	19	20	21
22	23	24	25	26	27
28	29	30	31	32	33

Parent 2

A	B	C	D	E	F
G	H	I	H	J	K
L	M	N	O	P	Q
R	H	S	H	T	U
V	W	X	Y	Z	Z
AA	AB	AC	AD	Z	Z

FIG. 4.8 – Représentation des parents. Les couleurs définissent chaque groupe de gènes chez les parents. Le parent 1 a 1 processus (1,3) et les autres sont des gènes indépendants. Le parent 2 a deux processus (2,4) et (5,6) et les autres sont des gènes indépendants.

modélisent le mieux notre problème soient transmis à l'enfant.

Cet exemple montre comment sont transmis les gènes des parents aux enfants en gardant en tête que certains gènes peuvent être liés à d'autres par des processus biologiques.

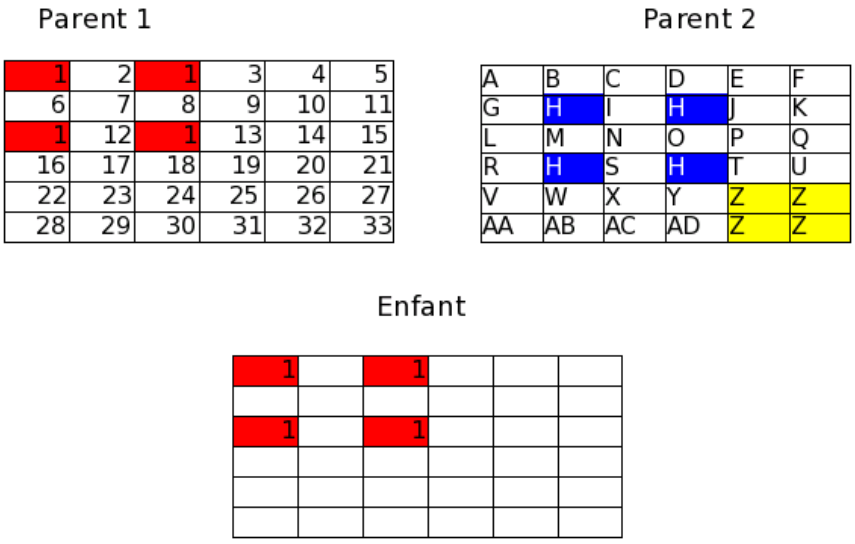


FIG. 4.9 – L'enfant hérite du processus (1,3) du parent 1.

La technique de mutation

Les biologistes connaissent déjà beaucoup de choses sur les réseaux biologiques étudiés, de plus ils ont une certaine sensibilité due à leur expérience sur les méthodes de régulation de certains processus biologiques. Ces informations sont très importantes car elles proviennent de l'expertise. Mais il ne faut pas non plus leur donner trop d'importance car les réseaux étant extrêmement complexes, il se peut que certaines relations ne soient pas du tout telles que les ont prédites les biologistes. Il faut donc une technique permettant de comptabiliser ces informations comme informations extérieures, mais capable de passer outre si le programme se trouve dans une impasse.
Dans un algorithme génétique, il n'existe pas à priori de processus permettant d'ajouter des informations extérieures autres que celles des expériences. Mais il est possible d'implémenter une méthode permettant de les prendre en compte. Par exemple, si l'on sait qu'un gène A active un gène B, nous pouvons prendre comme population initiale une population ayant exclusivement cette liaison. Il en résultera que, hormis les mutations, cette liaison se transmettra toujours aux populations suivantes. Il est aussi possible de considérer cela comme une contrainte de l'algorithme qui ne peut trouver que des solutions ayant ce trait de caractère, cette liaison. Par contre, il faut pouvoir passer outre cette contrainte pour permettre à COGARE de ne pas se concentrer sur des solutions ayant cette liaison en oubliant les autres possibilités. Pour cela, COGARE utilise un taux de mutation variable : les interactions ont un taux de mutation qui évolue en sens inverse par rapport à la fiabil-

Le deuxième tirage porte sur le parent 2 et donne le coefficient portant la lettre D à transmettre. Lors de ce tirage ont été exclus les coefficients A, C, L et N car ils ont déjà été transmis par le parent 1. Ils sont notés en noir.

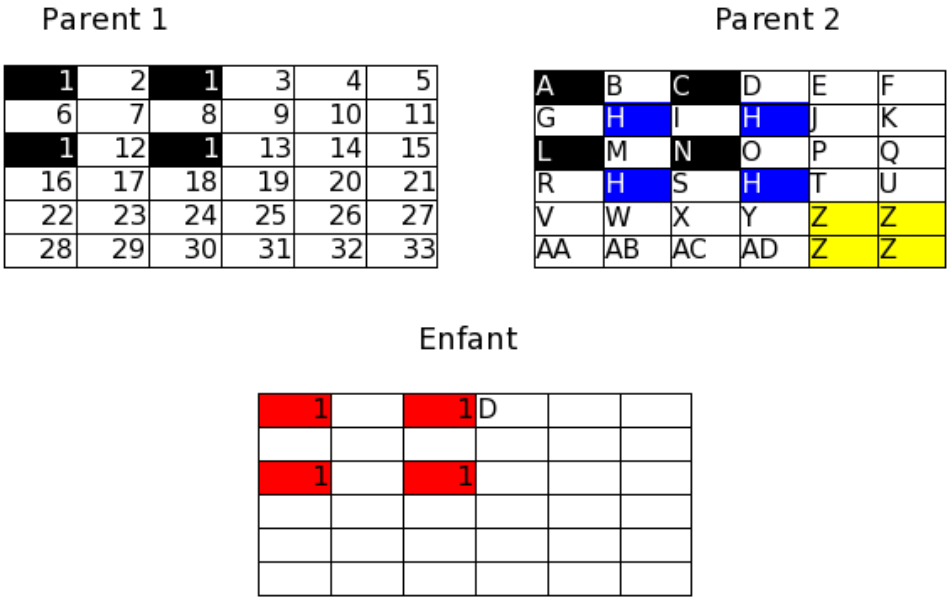


FIG. 4.10 – L'enfant hérite du coefficient D du parent 2.

ité des informations préalables. Plus une information est fiable, moins elle aura de chance de muter et cela d'après les règles suivantes : si l'interaction donnée par l'algorithme est la même que celle de l'information, le taux de mutation est égal à $tx - fi * 0.9 * tx$ avec tx le taux de mutation normal et fi le degré de fiabilité de l'information. Si l'information est différente, ce taux de mutation devient : $tx + fi * 0.9 * tx$. Cela signifie que si l'information est en adéquation avec le réseau trouvé, le taux de mutation de ce point du réseau sera diminué, et inversement. Ce taux permet, comme le montre la figure 4.13, d'améliorer les résultats de l'algorithme.

Ici la ressemblance est calculée comme le nombre de relations correctement retrouvées sur le nombre total de relations possibles (incluant la relation vide). Elle se calcule comme suit :

$$R = \sum_{i=1}^{N} \sum_{j=1}^{N} \frac{\delta_{M_{i,j},E_{i,j}}}{N * N} \tag{4.12}$$

où $M_{i,j}$ est le coefficient (i,j) de la matrice connue, $E_{i,j}$ est le coefficient (i,j) de la matrice reconstruite, et N est le nombre de gènes dans le réseau.

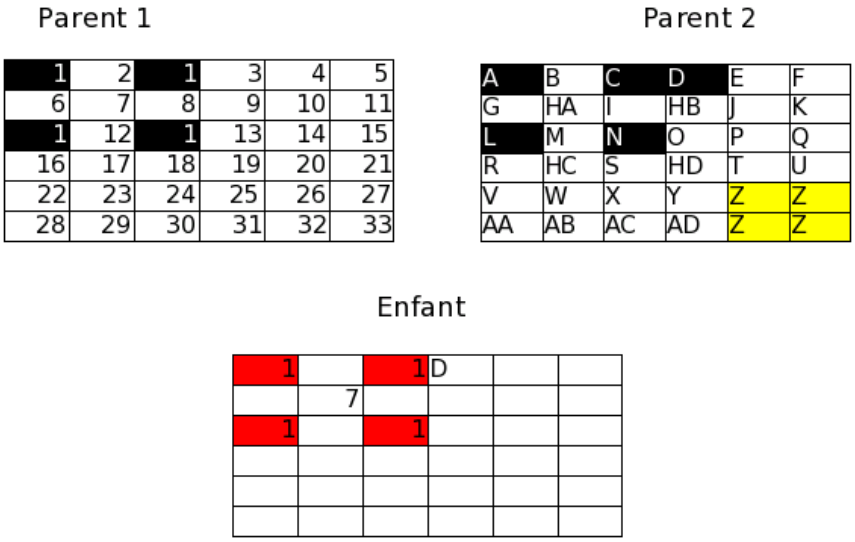


FIG. 4.11 – L'enfant hérite du coefficient 7 de la matrice du parent 1.

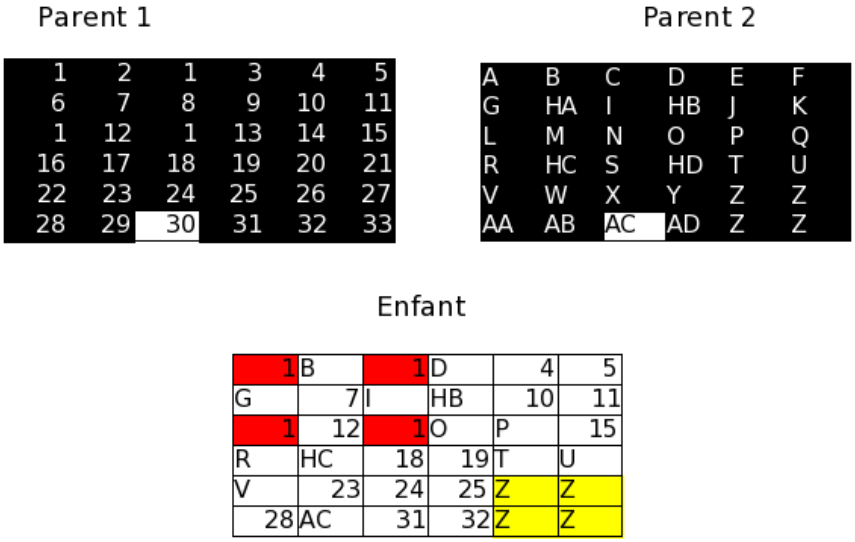


FIG. 4.12 – L'enfant hérite du dernier gène non encore transmis du parent 2 et obtient un génome complet.

Algorithme général

L'algorithme commence par créer une population initiale constituée de réseaux. Ces réseaux sont ensuite simulés pour obtenir différentes données. Ces données vont être par-

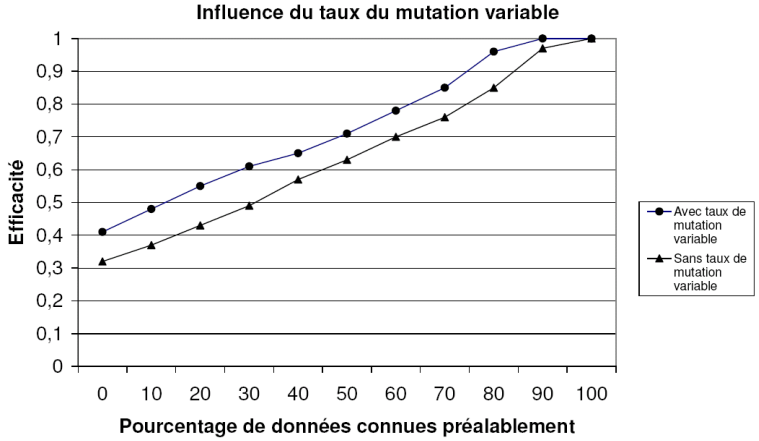


FIG. 4.13 – Comparaison du pourcentage de ressemblance du réseau solution avec le réseau réel connu, avec ou sans taux de mutation variable. Cette ressemblance est la moyenne de la ressemblance observée sur plusieurs jeux de données (taille du réseau et nombre d'expériences disponibles). Les jeux de données et les réseaux ont été créés artificiellement.

titionnées, de même que les données statiques et dynamiques disponibles, afin de calculer l'efficacité de ce réseau en y ajoutant les données complémentaires sur les interactions et les simulations directes des données dynamiques. Une fois l'efficacité calculée, la section de reconstruction de réseau va commencer. Cette section va créer une nouvelle population à partir de la précédente. Elle va tout d'abord sélectionner deux individus de la génération précédente en fonction de leur efficacité. Ensuite ces deux individus vont être croisés de manière intelligente en utilisant l'efficacité locale et la connaissance préalable de processus. Enfin, l'individu nouvellement créé par croisement va être muté en prenant en compte les informations sur les interactions.
Ce processus de sélection, croisement, mutation est effectué autant de fois que nécessaire pour reconstruire une nouvelle population. L'algorithme se termine soit par la découverte d'une solution ayant une efficacité suffisante par rapport au besoin de l'utilisateur, soit il se termine quand un certain nombre de générations ont été créées. L'algorithme général est présenté ici :

1. Création d'une population initiale.
2. Création de données issues de chaque individu de la population.
3. Calcul de l'efficacité globale des individus.
4. Sélection des parents à croiser.
5. Calcul de l'efficacité locale de chaque parent et identification des groupes de gènes à croiser.
6. Croisement et création d'un individu enfant.

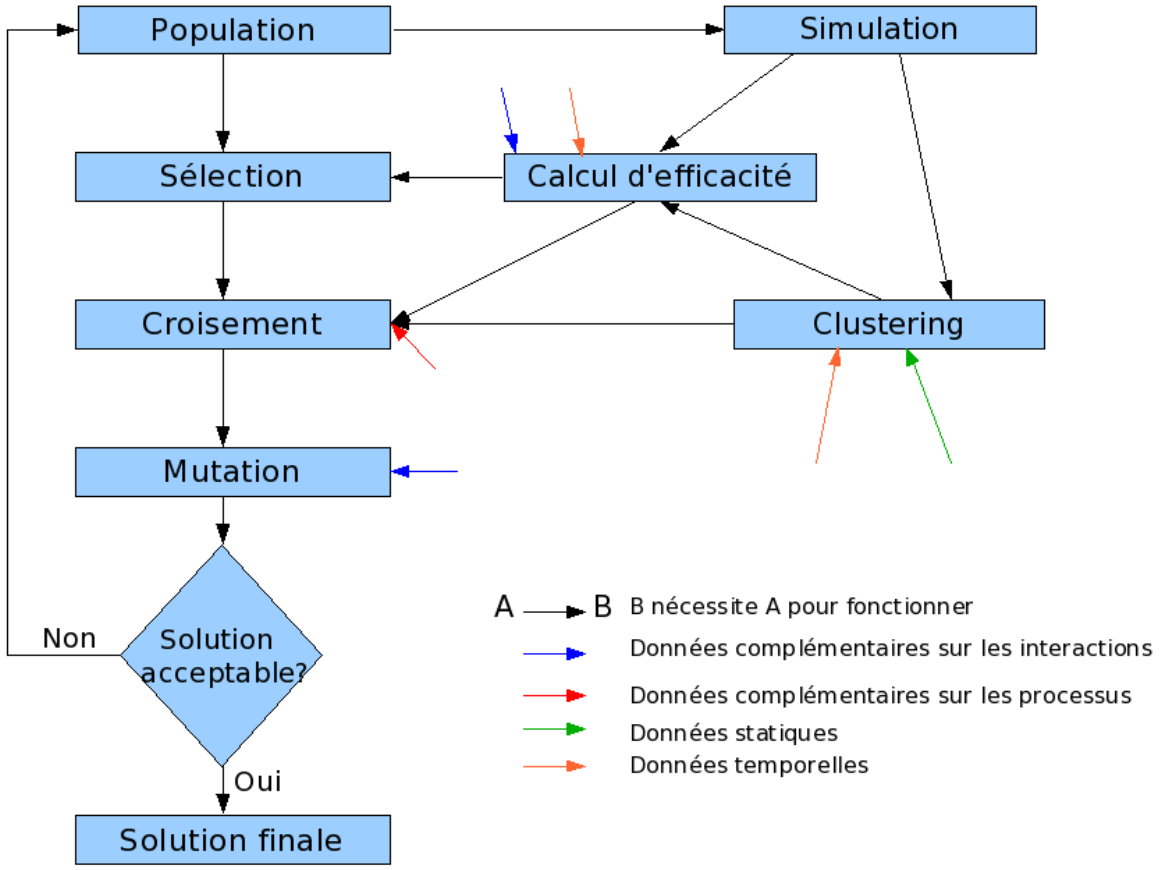


FIG. 4.14 – Schéma de fonctionnement de l'algorithme COGARE. Les flèches noires montrent l'ordre d'utilisation des fonctions de l'algorithme. Les flèches de couleur montrent les fonctions où sont utilisées les différentes données disponibles.

7. Mutation de cet enfant.
8. Étapes 4-8 jusqu'à obtention d'un nouvelle population.
9. Choix d'un individu gagnant dans la population.
   - Si l'efficacité de l'individu est suffisante (supérieure à un seuil prédéfini), l'algorithme s'arrête.
   - Sinon, refaire les étapes 2-8 pour créer une nouvelle population.

Cet algorithme est à la base du logiciel COGARE qui a été programmé et testé sur différents exemples qui sont présentés à la partie suivante.

Chapitre 5

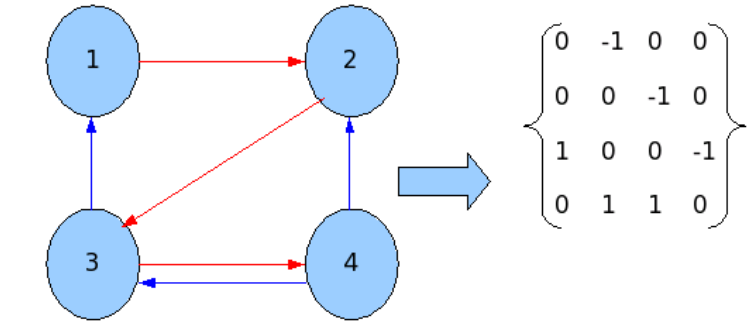
Présentation des résultats obtenus avec COGARE

Pour tester notre algorithme, nous l'avons fait fonctionner sur plusieurs sortes de données, des données réelles et des données construites. Les données réelles sont des données que nous ont procurés les biologistes sur des expériences réelles, elles sont donc bruitées et souvent incomplètes, ces données sont en relation avec la biolixiviation. Les données construites sont les résultats d'expériences artificielles que l'on a effectuées à partir d'un réseau que l'on connaissait et sur lequel on a lancé des simulations pour récupérer des données. Elles sont créées comme suit. Tout d'abord un réseau est créé au hasard, c'est à dire que l'on tire au hasard la matrice d'incidence du réseau, avec comme probabilité : 0.5 de tirer un 0 (pas d'interaction), 0.25 de tirer 1 (activation) ou -1 (inhibition). Cette distribution a été choisie car les matrices d'incidence d'un réseau génétique sont généralement creuses, elles contiennent beaucoup de zéros. Ensuite, on initialise cette matrice au hasard et on fait tourner le module de simulation dessus plusieurs fois pour obtenir des données de départ. Les entrées statiques sont créées directement avec ces données. Les entrées dynamiques sont créées en prenant comme temps initial l'état courant, puis en faisant tourner le module de simulation pour obtenir les états aux temps suivants (cf. fig.5.1).
De plus, les différents paramètres de l'algorithme ont dû être optimisés, et pour cela certains tests ont été faits dans le but de découvrir quelles étaient les valeurs optimales de ces paramètres. Pour visualiser simplement le résultat et la performance de l'algorithme, nous avons choisi comme fonction d'efficacité la fonction de ressemblance définie en 4.4 :

$$R = \sum_{i=1}^{N} \sum_{j=1}^{N} \frac{\delta_{M_{i,j},E_{i,j}}}{N * N}; \tag{5.1}$$

où $M_{i,j}$ est le coefficient (i,j) de la matrice connue, $E_{i,j}$ est le coefficient (i,j) de la matrice reconstruite, et N est le nombre de gènes dans le réseau.
Rappelons que cette ressemblance représente le pourcentage de coefficients de la matrice

calculée qui sont identiques à ceux de la matrice connue. Cette ressemblance ne peut se calculer que si l'on a à disposition la matrice originale. Dans ce cas, l'efficacité est le pourcentage de relations retrouvées. Si la matrice originale n'est pas disponible, l'efficacité prise en compte sera l'efficacité calculée en 4.4.



Gène	Etat initial créé au hasard	État à t=1	État à t=2
1	0	1	1
2	1	1	0
3	1	0	0
4	0	0	0

FIG. 5.1 – Processus de création de données artificielles.

5.1 L'optimisation des paramètres de l'algorithme

Les paramètres qu'il faut optimiser pour faire fonctionner au mieux l'algorithme sont de plusieurs sortes :
- le nombre de générations,
- la taille d'une population,
- la valeur du taux de mutation,
- les poids pour l'efficacité (α, β et γ, voir le paragraphe 4.4).

Tous ces paramètres ont été testés sur des données artificielles de différentes tailles, en calculant les réseaux pour différentes valeurs de ces paramètres, pour ensuite comparer l'efficacité de ces paramètres en comparant l'efficacité des solutions proposées. Comme la matrice originale est connue, l'efficacité calculée ici est la ressemblance entre la matrice originale et la matrice calculée.

Le nombre de générations et la taille d'une génération sont assez simples à optimiser : plus il y a de générations et plus une génération est grande, plus l'espace des solutions sera balayé, et donc plus on aura de chance de tomber sur une meilleure solution. Le problème inhérent à ces variables à optimiser est que plus elles sont grandes, plus le temps de calcul

sera important (fig.5.2). Les simulations donnent des tailles de population optimales de 25, car au-delà le gain devient plus négligeable. Pour le nombre de générations, le nombre optimal est de 15 (fig.5.3, fig.5.4).

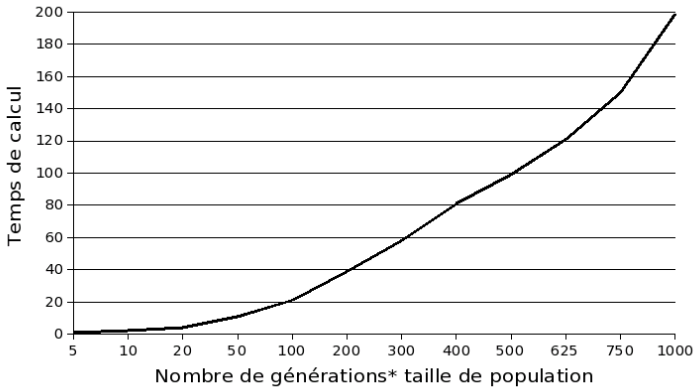


FIG. 5.2 – Temps de calcul de l'algorithme en unité arbitraire en fonction de la taille d'une population et du nombre de générations. L'unité est prise arbitrairement car les tests ont été effectués sur plusieurs tailles de réseaux, ce qui modifie fortement le temps de calcul. Le temps de calcul augmente exponentiellement, il faut donc trouver une taille de population et un nombre de générations qui permettent de trouver de bons résultats tout en ne générant pas un temps de calcul trop long.

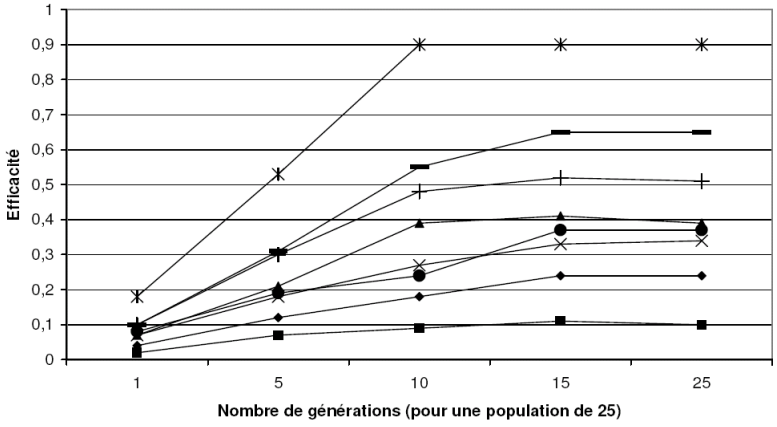


FIG. 5.3 – Efficacité de l'algorithme pour divers nombres de générations.

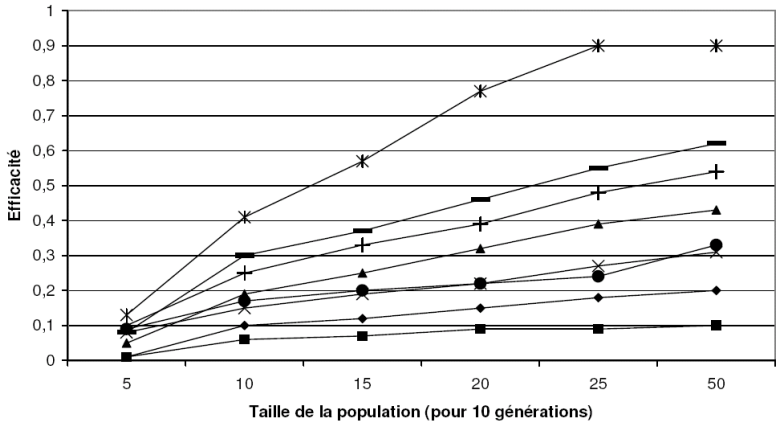


FIG. 5.4 – Efficacité de l'algorithme pour diverses tailles de populations.

L'optimisation du taux de mutation pour les informations non connues se fait de la même manière, la figure 5.5 nous montre les résultats, indiquant le taux de mutation optimal aux alentours de 0.05.

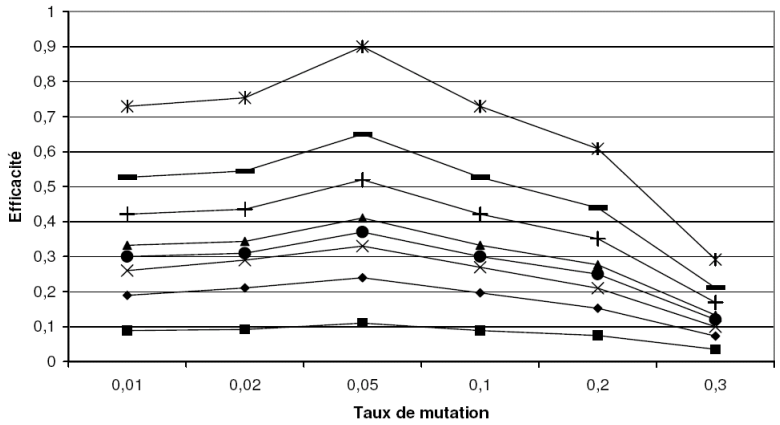


FIG. 5.5 – Efficacité de l'algorithme pour divers taux de mutation.

Pour calculer l'efficacité d'une solution il est nécessaire de connaître l'apport de chaque type d'efficacité. Leurs poids respectifs, notés α, β, γ, sont des paramètres plus difficiles à optimiser car ils dépendent beaucoup des expériences disponibles. Grâce aux nombreux tests effectués pour optimiser les paramètres, nous avons pu en tirer une règle empirique, qui serait de donner un poids deux fois plus important aux données dynamiques par rapport aux données statiques, ceci à moduler suivant leur nombre ; si le nombre de données statiques est plus important que le nombre de données dynamiques, il faut augmenter le poids des données statiques car elles apportent plus d'informations.
Ensuite, les informations doivent avoir un poids en relation avec leur nombre et leur fiabil-

ité. Le tableau suivant donne des ordres de grandeur de ces poids pour diverses conditions expérimentales. Donner plus de poids aux données dynamiques permet d'accentuer l'importance des informations sur l'évolution du réseau, tandis que les données statiques ne donnent d'information que sur un état particulier du système. Les valeurs des poids ci-dessous ont été définies comme les plus performantes lors des différentes expérimentations effectuées avec COGARE sur différents problèmes de reconstruction, sur données réelles et données construites. Les autres jeux de poids nous ont généralement amené à des résultats moins performants. Il n'a pas été possible de tester l'intégralité des jeux de valeur pour les poids, mais affiner les poids en prenant des valeurs décimales est toujours possible, c'est pour cela que ce tableau ne représente qu'une indication basée sur la connaissance empirique issue de l'utilisation du logiciel.

Nb d'exp statiques	Nb d'exp dynamiques	Nb d'informations	Fiabilité	α	β	γ
100	50	10	0.8	2	2	2
100	100	10	0.8	2	4	2
50	100	10	0.8	1	4	2
100	50	10	0.4	2	2	1
100	50	20	0.8	2	2	3

Ce tableau donne un ordre de grandeur des poids à utiliser pour la plupart des cas que nous avons rencontrés. Dans certains cas, les poids optimaux pour deux expériences de même taille étaient différents. En effet, suivant les données disponibles, il est possible qu'il faille augmenter ou diminuer la valeur des poids pour obtenir un résultat optimal. Le choix des poids fait partie de la partie initialisation de COGARE qui peut prendre énormément de temps et qui dépend fortement du cas expérimental que nous voulons traiter. Trouver les poids optimaux peut faire l'objet d'une étude à part ou bien l'utilisateur peut, grâce à son expérience et à sa sensibilité, choisir les paramètres les plus adaptés.
Un autre paramètre qui joue sur la précision des résultats est le nombre de données initiales disponibles, ce nombre permet de contraindre l'algorithme à évoluer dans certaines parties de l'espace de solution, et plus il est important, plus la précision sera grande (fig.5.6).

Il est normal de voir l'algorithme converger vers un meilleure efficacité lorsque le nombre de données supplémentaires augmente, car ces données sont des informations directes sur le réseau.

5.2 Contrôle de robustesse

Un algorithme sur des données expérimentales se doit de pouvoir réagir aux spécificités du problème recherché. Une des spécificités du problème biologique qui nous intéresse ici réside dans la grande variabilité des données résultant d'une même expérience. En effet,

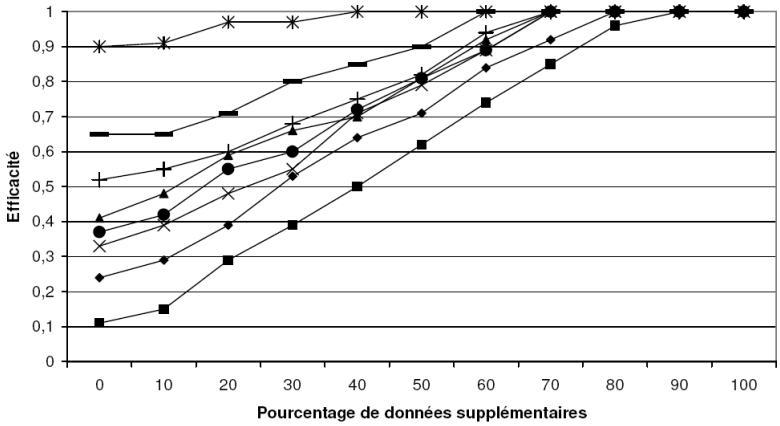


FIG. 5.6 – Efficacité de l'algorithme pour différents pourcentages de données supplémentaires.

pour une même expérience, les données peuvent varier de façon très importante, car les conditions expérimentales peuvent varier très facilement et sont difficilement contrôlables. De plus, chaque variation infime des conditions expérimentales peut entraîner de grandes différences dans les réponses du réseau, ceci étant dû à la grande réactivité des organismes biologiques et à la connectivité importante des réseaux qui entraînent des modifications en chaîne des états du système lorsqu'un seul des éléments du réseau est perturbé. Les données obtenues peuvent donc être très facilement bruitées. Pour vérifier que le logiciel fonctionne correctement même en cas de bruit sur les données, j'ai testé le logiciel sur des données classiques, puis modifié les données aléatoirement avec de plus en plus de bruit pour modéliser les changements dans les données expérimentales. Ceci a été fait de la façon suivante :

$$nvdon = ancdon + amp * alea * bruit \tag{5.2}$$

où nvdon est la nouvelle valeur des données,
ancdon est l'ancienne valeur des données,
amp est l'amplitude de bruit généré, initialisée à max(données)-min(données),
alea est un entier dont la valeur est tirée aléatoirement dans {-1,1},
et bruit est le pourcentage de bruit désiré, entre 0% et 100%.
Enfin, j'ai comparé les résultats des données bruitées (nvdon) avec les données normales (ancdon). COGARE a été exécuté sur ces deux types de données dans plusieurs cas afin d'obtenir des résultats en fonction des données. La figure 5.7 nous montre le pourcentage de ressemblance des résultats avec les résultats des données normales en fonction du pourcentage de bruit. Le pourcentage de ressemblance est le pourcentage de liaisons identiques (inhibition, activation ou pas de liaison) entre le réseau bruité et le réseau initial (sans données bruitées) par rapport aux nombre de liaisons total (égal à N^2 où N est le nombre

de gènes dans le réseau) :

$$R = \frac{\sum_{i=1}^{N}\sum_{j=1}^{N} \delta_{A_{i,j},N_{i,j}}}{N^2} \tag{5.3}$$

où $A_{i,j}$ est le coefficient (i,j) de la matrice résultat calculée pour les données normales (sans bruit), et $N_{i,j}$ est le coefficient (i,j) de la matrice résultat calculée pour les données bruitées. Cette robustesse est relativement bonne car à 30% de bruit, l'efficacité est encore

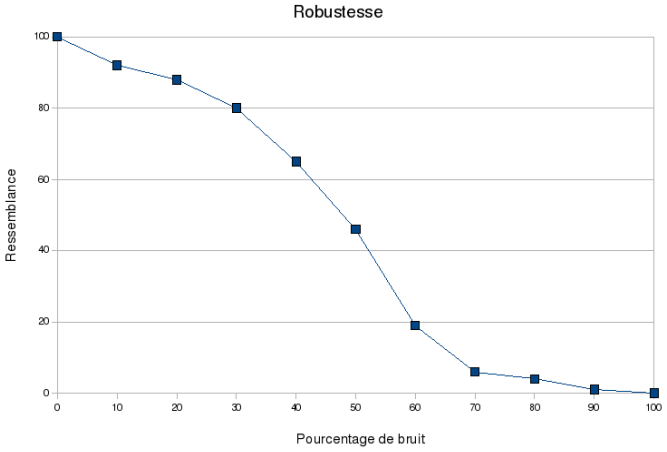


FIG. 5.7 – Robustesse de l'algorithme.

de 0.75. C'est à dire que si les données sont bruitées à 30%, le réseau est retrouvé à 75%. Par contre à partir d'un pourcentage de bruit trop important, ici 35%, l'efficacité chute de façon très rapide, et passe de 0.75 à 0.5 en augmentant le bruit de 35% à 45%, soit en l'augmentant de 10 points. Cette chute brutale s'explique par le fait que, à partir de 30% de bruit, les données n'ont plus rien à voir avec les données créées sans bruit, COGARE a donc beaucoup plus de mal à retrouver le réseau d'origine. La robustesse de l'algorithme permet de s'assurer que les résultats ne changent pas trop si les données changent raisonnablement. Ceci est un avantage de COGARE qui permet de prendre en compte une des spécificités importante de la reconstruction de réseau génétique.

5.3 Données construites

COGARE a été comparé avec d'autres algorithmes comme cela a été fait dans [9]. Pour comparer les résultats des différentes méthodes, Banjo (cf. paragraphe 3.6.1), NIR (cf. paragraphe 3.6.3), Aracne (cf. paragraphe 3.6.2), les programmes ont été testés sur des données construites. Pour bien pointer les avantages et les inconvénients de toutes ces méthodes, les tests ont été effectués sur différentes tailles de données et de réseaux.
Dans le tableau récapitulatif ci-dessous, les tailles (N) des réseaux testés vont de 10 éléments jusqu'à 1000 éléments et le nombre de données (M) varie lui aussi.

Les résultats. La comparaison des résultats des algorithmes s'est effectuée sur deux valeurs qui permettent de définir les points forts et faibles de chaque algorithme. Ces deux valeurs sont la Sensibilité (SE), et la Valeur Prédictive Positive (VPP). La VPP est le nombre de bonnes relations retrouvées divisé par le nombre total d'interactions retrouvées. La SE est le quotient du nombre de bonnes relations par le nombre total de relations (cf. figure 5.8). La VPP permet de voir si l'algorithme n'oublie pas des relations, et permet de visualiser sa précision par rapport aux réseaux réels. La SE permet elle d'apprécier si l'algorithme ne fait pas trop d'erreurs sur les relations retrouvées, et donc permet de savoir si l'on peut se fier aux relations retrouvées. Deux types de données ont été testées, les données dynamiques et les données statiques. Comme il existe plusieurs niveaux de reconstruction génétique et que les techniques de reconstruction sont spécialisées dans certains types de reconstruction (graphes non-orientés, orientés ou signés), toutes les techniques ne donnent pas de résultat pour tous les types d'essais. Trois types de résultats sont visibles dans le tableau, suivant ce qui était recherché :
– les graphes non-orientés (^{u}),
– les graphes orientés (^{d}) ,
– les graphes signés (^{s}).
Les résultats figurant en bleu dans les tableaux ci-après correspondent aux cas où COGARE obtient la meilleure performance. Les résultats en grisé correspondent aux cas où COGARE obtient des performances supérieures à 90% des performances des meilleurs algorithmes.

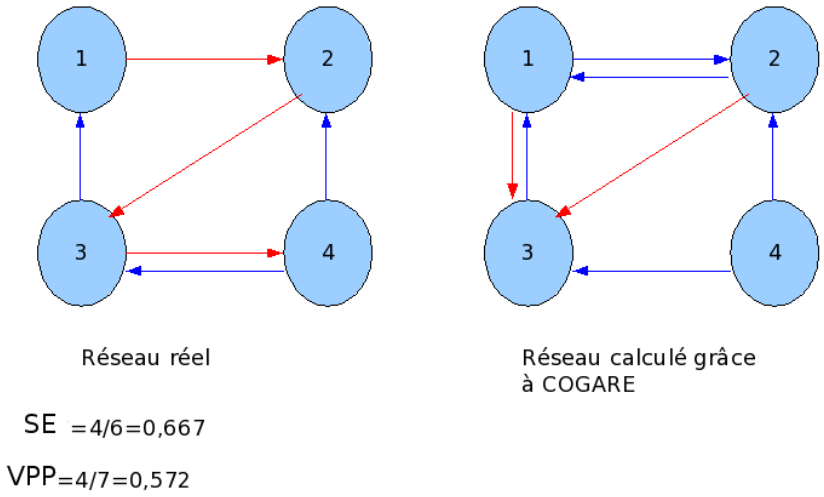


FIG. 5.8 – Calcul de VPP et de SE pour un cas test. À droite, le réseau réel. À gauche, le réseau retrouvé par COGARE. COGARE a trouvé 4 bonnes relations. Il y a en tout 6 interactions dans le réseau et COGARE en a retrouvé 7, d'où le calcul de la VPP (VPP=4/6) et de la SE (SE=4/7).

Quelques résultats sur des données statiques. Le tableau ci-dessous montre les résultats des algorithmes pour des données statiques.

	Aracne		Banjo		NIR		COGARE	
N*M	VPP	SE	VPP	SE	VPP	SE	VPP	SE
$10*10^u$	0.37	0.40	0.41	0.49	0.34	0.71	0.12	0.12
d			0.25	0.17	0.18	0.45	0.11	0.08
s			0.16	0.05	0.09	0.22	0.09	0.01
$10*100^u$	0.37	0.44	0.96	0.11	0.36	0.70	0.15	0.17
d			0.79	0.05	0.20	0.46	0.12	0.11
s			0.84	0.05	0.09	0.21	0.08	0.04
$100*10^u$	0.19	0.11	0.19	0.04	0.18	0.09	0.05	0.01
d			0.10	0.02	0.10	0.05	0.01	0.00
s			0.71	0.06	0.00	0.05	0.00	0.00
$100*100^u$	0.19	0.17	0.70	0.00	0.19	0.19	0.08	0.11
d			0.47	0.00	0.10	0.10	0.06	0.09
s			0.71	0.00	0.05	0.05	0.04	0.04
$100*1000^u$	0.19	0.26	0.99	0.05	0.20	0.19	0.13	0.12
d			0.68	0.03	0.10	0.09	0.09	**0.09**
s			0.68	0.03	0.05	0.05	0.04	**0.05**
$1000*$	0.02	0.10	-	-	-	-	**0.02**	0.04
1000^u								

COGARE ne donne pas de résultats très intéressants sur les données statiques. En effet, COGARE récupère beaucoup plus d'informations provenant de données dynamiques, les données statiques servant à peaufiner le réseau pour départager les endroits du réseau où il y aurait un doute sur l'interaction à retrouver. Les données statiques sont mieux utilisées par Banjo car le logiciel utilise des informations sur le type d'expérimentations effectuées, en l'occurrence sur l'élément du réseau perturbé pour créer les données statiques. J'ai fait le choix de ne pas prendre en compte cette information car elle n'est pas toujours accessible suivant les données auxquelles nous avons accès. En effet, si les perturbations s'effectuent sur des conditions environnementales, aucune perturbation d'un gène n'est opérée, et il n'est pas possible de récupérer l'information. Un projet pour le développement de COGARE inclut d'intégrer ce type de données parmi les données que COGARE peut traiter. Cela permettrait de récupérer plus d'informations des données statiques et d'améliorer la puissance du logiciel en augmentant le nombre de données accessibles et en augmentant leur rendement en information.

COGARE fournit des résultats moyens sur les petits réseaux, cela à cause du manque d'informations précises que le logiciel peut extraire des données disponibles. L'algorithme

ne dispose pas d'assez de points de comparaisons pour que le calcul d'efficacité soit réellement utile à la recherche d'un solution au problème. Plus le réseau recherché et le nombre de données disponibles augmentent, plus les résultats de COGARE se rapprochent des résultats fournis par les logiciels concurrents. En effet, COGARE est basé sur l'algorithmie génétique qui permet d'explorer des espaces de solutions très grands, et il n'a pas été conçu pour la recherche dans des petits espaces, espaces dans lesquels il existe des algorithmes plus appropriés.

Quelques résultats sur des données dynamiques. Le tableau suivant montre les résultats pour des expérimentations sur des données dynamiques (NIR n'utilisant pas les données dynamiques, il a été retiré du tableau).

Données	Aracne		Banjo		COGARE	
N*M	VPP	SE	VPP	SE	VPP	SE
$10*10^u$	0.00	0.39	0.36	0.35	0.25	0.30
d			0.22	0.21	0.20	0.20
s			0.00	0.00	**0.10**	**0.11**
$10*100^u$	0.35	0.43	0.36	0.29	0.31	0.22
d			0.21	0.16	**0.28**	**0.20**
s			0.25	0.00	0.20	**0.11**
$100*10^u$	0.19	0.10	0.18	0.08	0.10	0.05
d			0.10	0.04	0.05	0.01
s			0.06	0.00	0	0
$100*100^u$	0.19	0.15	0.19	0.05	0.15	**0.15**
d			0.10	0.02	**0.12**	**0.05**
s			0.04	0.00	**0.09**	**0.07**
$100*1000^u$	0.19	0.19	0.19	0.04	0.18	0.18
d			0.10	0.02	**0.12**	**0.08**
s			0.05	0.00	**0.06**	**0.02**
$1000*1000^u$	0.02	0.10	-	-	**0.02**	**0.10**

Pour la reconstruction de réseaux à partir de données dynamiques, les résultats donnés par COGARE sont à des niveaux comparables à ceux des autres algorithmes, et largement supérieurs aux résultats issus de données purement statiques. Ceci est dû au fait que les données dynamiques apportent plus d'informations, notamment sur l'évolution temporelle de l'état d'activation des gènes des réseaux. COGARE est le seul logiciel avec Banjo à pouvoir calculer à la fois des graphes signés, orientés et non-orientés et ce sur des données statiques ou dynamiques, mais il est le seul à pouvoir fonctionner sur des tailles de réseaux supérieures ou égales à 1000. Banjo sera donc plutôt utilisé pour effectuer une

comparaison plus en profondeur des résultats, car il est le logiciel qui a le plus de points de comparaison possibles.
Les résultats sur des données dynamiques seules permettent de le mettre en compétition directe avec Banjo. Par exemple sur les réseaux de taille 10 et avec 10 expériences, COGARE a des résultats très proches de ceux de Banjo. La moyenne de l'efficacité de COGARE sur ces types d'expériences est de 0.13 alors que Banjo a une moyenne de 0.12. COGARE a donc en moyenne une meilleure efficacité que Banjo. De plus, sur les 30 tests en commun pour les données dynamiques, COGARE obtient 15 fois une meilleure efficacité, et Banjo 13 fois. Pour les deux tests restants (pour la VPP en 10*10 orienté et pour la SE en 100*10 signé), les résultats des deux algorithmes sont identiques.
Par contre, COGARE a comme problème qu'il n'est pas souvent le meilleur algorithme de reconstruction. COGARE n'est l'algorithme qui renvoie l'efficacité maximale qu'à quatre reprises dont trois fois en compagnie d'une autre méthode :
– 0.22 pour la VPP sur des réseaux orientés de 10*10, avec Banjo.
– 0.02 pour la VPP sur des réseaux non-orientés de 1000*1000, avec Aracne.
– 0.10 pour la SE sur des réseaux non-orientés de 1000*1000, avec Aracne.
– 0.12 pour la VPP sur des réseaux orientés de 100*100, seul.

Résultats sur données mixtes. L'innovation dans COGARE réside dans le fait d'utiliser les deux types de données de façon simultanée. En effet, les logiciels présents sur le marché actuellement ne peuvent manipuler que certains types de données et sans jamais les croiser. COGARE, lui, permet non seulement d'utiliser tous les types de données disponibles mais en plus utilise leur complémentarité pour permettre d'obtenir des résultats meilleurs. Cette spécificité nous est montrée dans le tableau suivant, qui donne les résultats pour un nombre d'expériences statiques (S) variant de 5 à 500 et un nombre d'expériences dynamiques (D) allant de 5 à 900, et qui récapitule les temps de calcul pour chaque configuration. La dernière colonne permet de visualiser le meilleur résultat des algorithmes concurrents. Dans cette configuration, nous pouvons améliorer les résultats pour différentes tailles (T) de réseau.
Ce tableau nous montre une augmentation très importante de la précision des résultats, qui permet de rivaliser avec les résultats des meilleurs algorithmes. Ceci est dû au fait que l'algorithme récupère les informations provenant de deux types de données différentes et les combine pour obtenir une meilleure approximation. Nous pouvons remarquer que la précision est plus importante lorsque les données dynamiques sont plus nombreuses que les statiques. En effet, les données dynamiques portant en elles les informations sur l'évolution du système, elles apportent une information plus importante que les données statiques, qui elles, mêmes si elles sont moins porteuses d'information, sont utiles par leur nombre et leur capacité à reconnaître les corégulations.

Données	COGARE			Meilleur résultat			
T*(D+S)	VPP	SE	Hrs	VPP	Algo	SE	Algo
10*(5+5) (^{u})	0.37	0.35	0.5	0.41	Banjo	0.71	NIR
^{d}	0.20	0.19	0.5	0.25	Banjo	0.45	NIR
^{s}	0.11	0.09	0.5	0.16	Banjo	0.22	NIR
10*(50+50)(^{u})	0.88	0.65	2.5	0.96	Banjo	0.70	NIR
^{d}	0.43	0.21	2.5	0.79	Banjo	0.46	NIR
^{s}	0.11	0.11	2.5	0.84	Banjo	0.21	NIR
10*(90+10)(^{u})	0.90	0.68	1.5	0.96	Banjo	0.70	NIR
^{d}	0.54	**0.51**	1.5	0.79	Banjo	0.46	NIR
^{s}	0.43	**0.37**	1.5	0.84	Banjo	0.21	NIR
10*(10+90)(^{u})	0.17	0.15	3	0.96	Banjo	0.70	NIR
^{d}	0.10	0.08	3	0.79	Banjo	0.46	NIR
^{s}	0.01	0.00	3	0.84	Banjo	0.21	NIR
100*(50+50)(^{u})	0.58	0.35	5.5	0.70	Banjo	0.19	NIR
^{d}	0.27	**0.17**	5.5	0.47	Banjo	0.10	NIR
^{s}	0.11	**0.08**	5.5	0.71	Banjo	0.06	Banjo
100*(90+10)(^{u})	0.65	**0.52**	5	0.70	Banjo	0.19	NIR
^{d}	0.33	**0.27**	5	0.47	Banjo	0.10	NIR
^{s}	0.24	**0.18**	5	0.71	Banjo	0.06	Banjo
100*(900+100)(^{u})	0.91	**0.55**	14	0.99	Banjo	0.19	Aracne
^{d}	0.46	**0.34**	14	0.68	Banjo	0.09	NIR
^{s}	0.33	**0.22**	14	0.68	Banjo	0.05	NIR
100*(500+500)(^{u})	0.77	0.12	17	0.99	Banjo	0.26	Aracne
^{d}	0.38	**0.29**	17	0.68	Banjo	0.09	NIR
^{s}	0.27	**0.19**	17	0.68	Banjo	0.05	NIR
1000*(900+100)(^{u})	**0.15**	**0.15**	24	0.02	Aracne	0.10	Aracne
^{d}	**0.10**	**0.12**	24	-	-	-	-
^{s}	**0.06**	**0.03**	24	-	-	-	-

Les résultats de COGARE sur des données mixtes nous donnent deux informations intéressantes. La première concerne la similitude des résultats entre la réalité et la modélisation. La seconde concerne la fiabilité des résultats que COGARE donne ce trait étant caractérisé par la sensibilité.

L'efficacité moyenne donnée par COGARE pour l'ensemble des tailles de réseaux et de base de données est de 0.39. Pour l'ensemble des techniques, c'est à dire en prenant la meilleure efficacité pour chaque test et pour toutes les techniques, cette efficacité est de 0.33. Pour de petits réseaux (10-100 gènes) avec peu de données (à peu près 1 donnée par gène du réseau) COGARE est surpassé par les autres techniques. Lorsque le nombre de données par gène augmente (10 données par gènes pour des tailles de réseaux de 100 par

exemple), COGARE retrouve des réseaux de manière aussi bonne (voir dans le tableau les lignes 10*100 ^{u} et ^{s}) et meilleure lorsque le réseau grandit encore (1000 gènes).
Si on ne prend en compte que la sensibilité (SE), COGARE donne des résultats très intéressants avec la meilleure sensibilité dans presque tous les cas (10 cas sur 13) et une moyenne sur les cas traités de 0.34, alors que cette moyenne pour l'ensemble des meilleures modélisations par les logiciels concurrents n'est que de 0.28. Cela signifie que COGARE ne retrouve pas forcément toutes les relations dans un réseau, mais que la plupart de celles qu'il retrouve sont des bonnes relations ; il ne fournit que peu de faux-positifs. Cela permet de donner des bases de solution solides sur lesquelles s'appuyer lors de la reconstruction de réseau. Il est possible de supposer que les relations retrouvées peuvent former des informations complémentaires que l'on pourra intégrer au pool d'informations du logiciel pour refaire tourner COGARE avec une base plus importante.
Le type de données disponibles influe fortement sur les résultats. En effet, nous pouvons voir que les résultats avec moins ou autant de données statiques que dynamiques donnent des résultats moins bons que ceux avec un nombre de données dynamiques plus important. Nous pouvons prendre par exemple le 10x100 le résultat varie de 1 à 5.
Le temps de calcul augmente rapidement avec la taille et le nombre de données à traiter, mais reste un temps raisonnable sur des machines puissantes (quadricorps cadencé à 2,4Ghz). Le temps de calcul ne prend en compte que le temps de calcul effectif, et ne prend pas en compte le temps passé à configurer le logiciel ni les données. Ce temps (par exemple le calcul des poids optimaux, cf. le paragraphe 5.1) peut être considérable si il faut en plus formater les données à un format lisible par le logiciel.
L'intérêt de COGARE est de pouvoir localiser les efficacités à un niveau plus faible, c'est à dire que même si l'efficacité globale n'est pas très importante, il est possible d'extraire des informations précises sur certaines parties du réseau et de les réintroduire dans le logiciel comme informations complémentaires avec une certaine fiabilité, ce qui peut permettre à l'algorithme d'augmenter sa précision en disposant de plus d'information.
Afin de mieux visualiser les différences entre les logiciels concurrents et le nôtre, nous pouvons créer un diagramme qui récapitule les forces et faiblesses des différents logiciels. Ce diagramme positionnera les algorithmes en fonction des paramètres déjà connus que sont la sensibilité et la VPP. Dans un premier temps, calculons la VPP et la SE moyenne pour chaque algorithme :

	COGARE	NIR	Aracne	Banjo	Partitionnement
VPP	0.19	0.37	0.15	0.22	0.23
Se	0.24	0.08	0.24	0.17	0.17

Sur le diagramme 5.9, COGARE se comporte bien, n'étant jamais le meilleur en SE ou VPP, mais ayant un positionnement plutôt intéressant, assez loin derrière Banjo en VPP, mais tout proche des meilleurs en SE (NIR et Aracne) en Se.

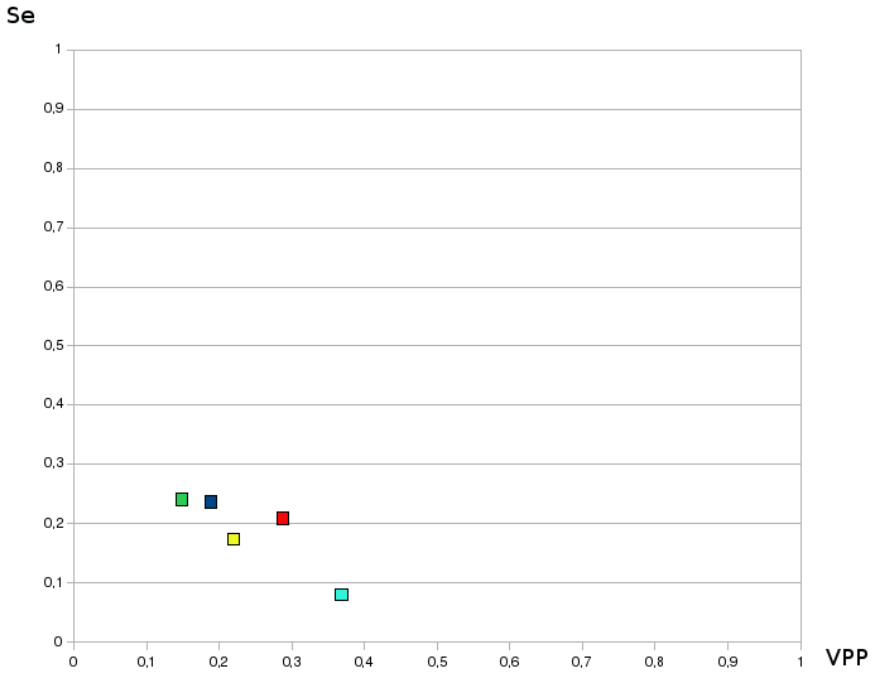


FIG. 5.9 – Diagramme de positionnement des algorithmes en fonction des leurs résultats. En vert : NIR. En bleu : Aracne. En jaune : partitionnement. En rouge : COGARE. En bleu ciel : Banjo.

5.4 Les tests sur données réelles

Le processus de SOS chez Escherichia coli

Escherichia coli est une des bactéries les plus étudiées au monde. Cette bactérie intestinale présente chez les mammifères est très commune chez l'être humain. J'ai appliqué COGARE à un processus de coli nommé le SOS [83]. Ce processus biologique permet à coli d'envoyer un message d'alerte et d'erreur lorsque l'ADN est endommagé, c'est à dire lorsque deux nucléotides ne sont plus en concordance à cause de génotoxines –substances toxiques agissant sur les gènes–, telles que le rayonnement ultra-violet ou des substances cancérigènes. Ce processus comporte plus de 100 gènes et COGARE a été appliqué à un ensemble de neuf gènes au coeur du réseau sur lesquels les résultats étaient connus et les données facilement accessibles. Les données pour faire les simulations ont été récupérées dans les bases de données accessibles sur internet, GENBANK [130] et dans les travaux de Gardner et al [46]. Le nombre d'expériences disponibles sur lesquelles nous avons travaillé

a été de 9 expériences statiques et 6 dynamiques.
Les paramètres de COGARE pour cette expérience sont résumés dans le tableau suivant :

Nombre de générations	15
Taille d'une population	10
α	2
β	4
γ	2
Taux de mutation	0.01

Les données dynamiques et statiques étant à peu près en nombres égaux, le coefficient β est plus important que le coefficient γ car les données dynamiques ont un poids plus important dans le logiciel pour la résolution du problème.
Le réseau initial connu comporte 25 relations, montrées à la figure 5.10.

De ces relations connues, j'ai tiré aléatoirement 5 relations qui ont formé les données préalablement connues et dont j'ai fixé la fiabilité à 0.6.
Une fois les paramètres mis en place, COGARE a été lancé sur les données. Les résultats peuvent être visualisés à la figure 5.11. Les résultats montrent que COGARE a retrouvé 23 des 25 interactions déjà connues du réseau. Les deux interactions restantes ont été retrouvées mais avec des signes opposés, activation à la place d'inhibition pour l'une (ssb → ssb) et inhibition à la place d'activation (umuDC → lexA). De plus, d'autres interactions ont été trouvées :
– recF → recA
– recF → recF
– rpoH → dinl
– rpoS → rpoD
Ces interactions peuvent être de deux natures : soit elles sont des faux positifs, des erreurs du programme lors de la reconstruction du réseau, soit elles sont des nouvelles influences, non encore signalées dans la littérature. Le but du programme a donc été atteint lors de ce test. COGARE a bien reconstruit un réseau qui est très proche de la réalité et en plus a donné des pistes aux biologistes en direction desquelles chercher de nouveaux leviers de régulation pour ce processus. Le tableau suivant montre les différences entre le réseau reconstruit et le réseau connu :

1	**1**	**1**	0	**1**	0	0	0	0
-1	**-1**	**-1**	0	0	**-1**	0	0	0
-1	**-1**	-1	0	0	**-1**	**-1**	0	0
-1	0	0	1	0	0	0	0	0
1	**1**	**1**	0	0	**1**	**1**	0	0
0	1	0	0	0	**-1**	0	0	0
0	0	**1**	0	0	0	0	0	**1**
0	0	0	0	0	1	**1**	**1**	0
0	0	0	0	0	-1	0	0	0

En gras sont indiquées les interactions bien retrouvées, en vert les interactions nouvelles non décrites dans la littérature, et en bleu les interactions mal retrouvées.

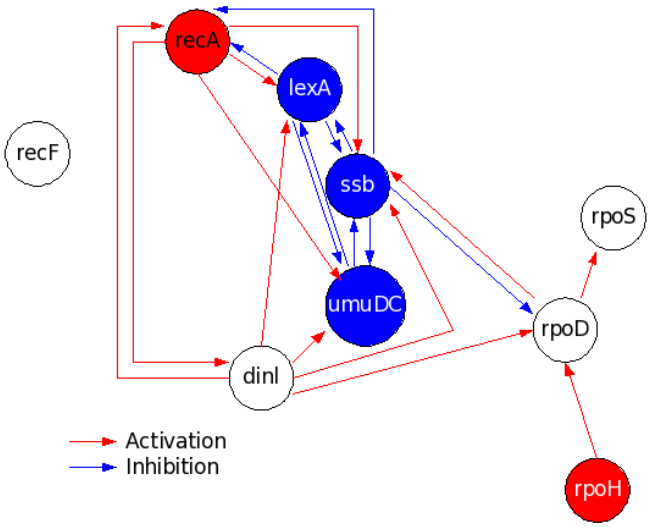


FIG. 5.10 – Graphe du processus de SOS de Escherichia coli. Les flèches rouges représentent une activation, les bleues une inhibition. Pour plus de visibilité, les auto-régulations ont été signalées par la couleur rouge (activation) ou bleue (inhibition) du sommet.

Pour visualiser la différence de résultats lorsque l'on exécute COGARE avec différents types de variables, j'ai aussi testé le réseau sous trois autres configurations de données :
- seulement les données statiques (au nombre de 9), numéro d'expérience 1 ;
- seulement les données dynamiques (au nombre de 6), numéro d'expérience 2 ;
- les données statiques et les données dynamiques (sans les données complémentaires), numéro d'expérience 3.

Les résultats de ces expériences sont résumés dans le tableau suivant, l'expérience 0 est l'expérience de base avec tous les types de données. IR est le nombre d'interactions cor-

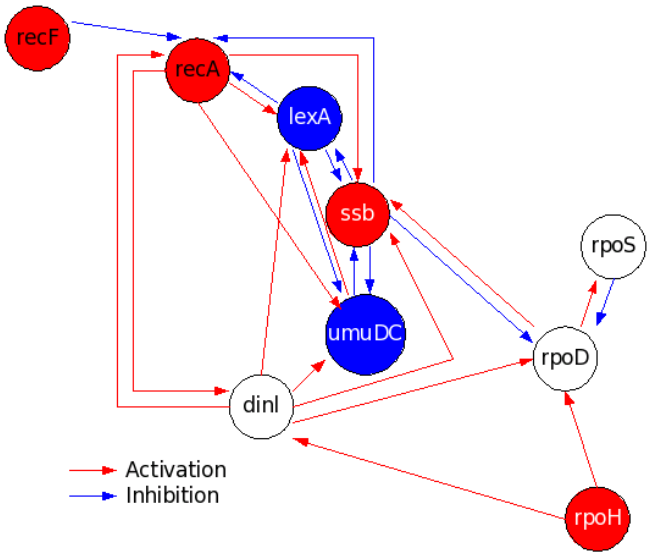


FIG. 5.11 – Graphe du processus de SOS de Escherichia coli retrouvé par COGARE. Les flèches rouges représentent une activation, les bleues une inhibition. Pour plus de visibilité, les auto-régulations ont été signalées par la couleur rouge (activation) ou bleue (inhibition) du sommet.

rectement retrouvées. IS est le nombre d'interactions retrouvées de signe contraire. IO est le nombre d'interactions oubliées. IN est le nombre de nouvelles interactions retrouvées.

Numéro d'expérience	IR	IS	IO	IN
0	23	2	0	4
1	12	3	10	8
2	15	3	7	7
3	18	4	3	5

Ce tableau montre principalement l'intérêt pour COGARE d'utiliser toutes les données à disposition. En effet, chaque donnée apporte une information et COGARE a pour but de rassembler ces informations pour optimiser la reconstruction du réseau. Le gain d'utilisation de l'ensemble des données est de 50% entre les expériences n'utilisant que les données statiques et celles utilisant les données statiques et dynamiques. Il est de 91% si l'on compare avec les expériences utilisant aussi des informations supplémentaires.

Données sur la biolixiviation

Les données suivantes ont été récupérées sur des expériences faites sur Thiobacillus ferrooxidans, une bactérie qui peut lixivier le cuivre [73]. Elles ont étés créées en déposant sur une puce à ADN, des brins d'ADN et en soumettant cette puce à plusieurs stimuli,

environnement de chalcopyrite (CuFeS2), environnement acide...
Ces données étant confidentielles, il est impossible d'aller plus loin dans l'explication du réseau testé et sur les données obtenues. Parmi les expériences, nous avons pu détacher 65 expériences statiques et 8 expériences dynamiques. Le réseau que nous voulons reconstruire comporte 354 gènes. Nous n'avons de plus aucune information complémentaire sur le réseau que nous pourrions fournir à COGARE. Les paramètres de COGARE pour cette expérience sont résumés dans le tableau suivant :

Nombre de générations	15
Taille d'une population	10
α	0
β	1
γ	4
Taux de mutation	0.01

Le coefficient β pour les données dynamiques est plus faible que le coefficient γ pour les données statiques car les données statiques sont beaucoup plus nombreuses (plus de 8 fois plus nombreuses) que les données dynamiques. Une vingtaine d'heures de calcul a été nécessaire à COGARE pour pouvoir fournir les résultats.
Le réseau réel de cet organisme n'est pas connu, il est donc impossible de comparer les résultats obtenus avec la réalité. Bien que le réseau ne soit pas connu, il existe certaines informations sur les gènes que nous avons testés. En effet, il est possible de récupérer des informations sur les gènes et protéines présents dans le réseau grâce au National Center for Biotechnology Information (site internet : *http* : *//www.ncbi.nlm.nih.gov/*). Cet organisme compile les informations sur les gènes et protéines qui ont été publiées. Les informations récupérées portent sur les gènes régulateurs. Il a été possible d'isoler certains des gènes régulateurs tout en ignorant quels gènes ils régulaient. Cette information a permis de tester si les résultats obtenus étaient cohérents ou non.
Le réseau, d'après la littérature, comporte une centaine de gènes régulateurs identifiés. Cela veut dire que l'on est sûrs que ces gènes sont des régulateurs, mais, par contre, rien ne dit qu'il n'en existe pas d'autres. COGARE a retrouvé en tout 113 gènes régulateurs dans le réseau. Sur les 113 régulateurs retrouvés, 85 faisaient partie des 100 gènes régulateurs connus. Le logiciel a donc trouvé 28 régulateurs qui ne sont pas présents dans la littérature. Ces nouveaux régulateurs peuvent être des régulateurs non encore identifiés par les biologistes ou alors des erreurs de reconstruction de COGARE. Connaissant les résultats de COGARE, les biologistes peuvent tester directement ces régulateurs pour infirmer leur statut de régulateur ou, au contraire, le confirmer.
Parmi la liste de ces 28 gènes régulateurs "nouveaux", nous avons identifié certains gènes dont la présence parmi les régulateurs peut s'expliquer.
Par exemple, un gène que nous savons être impliqué dans la production d'ATPase est

présent sur cette liste. L'ATPase est une protéine permettant de casser les molécules d'ATP. L'ATP servant à fournir de l'énergie aux cellules, réduire la quantité de cette molécule présente dans un organisme a des chances d'aboutir au ralentissement de certains processus biologiques et de la production d'autres protéines à cause d'un manque d'énergie.
Deux autres gènes codant pour une ATPase font partie de la liste et leur présence peut s'expliquer de la même manière que pour le gène précédent.
Un gène codant une protéine de transfert de phosphoglycerol membranaire est aussi présent sur cette liste. Ceci peut s'expliquer par le fait que la hausse ou la baisse de concentration en phosphoglycerol par le passage par la membrane va entraîner des modifications dans l'équilibre du système.
Parmi les nouveaux gènes régulateurs nous avons identifié un autre gène de transfert dont la présence sur la liste des régulateurs peut s'expliquer de la même façon.
L'absence de ces gènes sur la liste de régulateurs connus s'explique par le fait que l'action des protéines produites par ces gènes peut s'effectuer sur des gènes qui ne sont pas présents dans le réseau recherché. De plus, l'action de ces protéines peut s'effectuer par des moyens qui ne sont pas quantifiables par les expériences dont nous disposons (catalyse non-chimique, action sur des éléments autres que les gènes). COGARE parvient à les identifier par le biais des actions successives, qui font apparaître les modifications d'états par un enchaînement non visible directement.
D'autres gènes présents sur cette liste peuvent paraître plus délicats à justifier comme par exemple le gène codant pour une protéine permettant la division cellulaire ou un gène codant pour une protéine qui condense des chromosomes. En effet, la condensation des chromosomes agit sur les chromosomes et donc ne régule pas spécifiquement la production d'un gène ou d'un groupe de gènes. La division cellulaire agit sur les aspects macroscopiques de l'organisme et les effets sur les gènes ne peuvent donc pas être détectés. Ces gènes peuvent donc être considérés comme des faux-positifs, c'est à dire des erreurs de l'algorithme.
En résumé, sur ce qui était connu du réseau, COGARE a retrouvé 85% des régulateurs (85/100) et nous avons pu justifier la présence de 5 nouveaux gènes régulateurs, ce qui nous donne 79% (90/113) des régulateurs retrouvés potentiellement justes.
Le réseau retrouvé par COGARE comporte trois grandes parties (clusters de premier plan), dont une complètement isolée du reste des autres. Les deux autres parties forment des parties fortement connectées entre elles et peu avec les membres de l'autre partie. Il existe aussi des gènes que COGARE n'a connecté à aucun autre. Ces gènes, au nombre de 8, peuvent être considérés comme soit ayant peu d'incidence sur le réseau global, soit faisant partie d'un autre réseau. Ils sont alors présents sur les expérimentations à cause de la proximité spatiale de ces gènes au sein de l'ADN avec ceux formant le réseau. Il est alors possible, lors de la création des puces à ADN pour les expériences, que ces gènes

aient été ajoutés aux expériences à cause de cette proximité.
La partie isolée semble être une erreur du logiciel car elle comporte trois gènes qui codent pour des protéines n'ayant peu de rapport les unes avec les autres. Les trois gènes codent pour des protéines de fusion membranaire, de division cellulaire et de déshydratation du glucose. Leur agencement est montré à la figure 5.12.

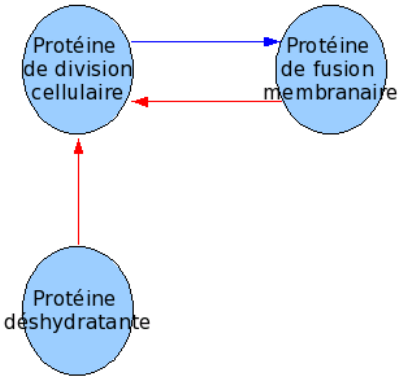


FIG. 5.12 – Groupe de gènes isolés par COGARE. La fonction des protéines associées aux gènes est indiquée sur la figure. Les flèches bleues indiquent les inhibitions et les flèches rouges, les activations.

Les deux autres parties de réseau retrouvées forment des réseaux de respectivement 229 et 114 gènes. Le premier réseau comporte 300 relations entre les éléments de ce groupe. Le deuxième réseau en comporte 120. Les relations entre ces deux groupes sont au nombre de 12, nous pouvons donc considérer que les deux groupes forment des sous-réseaux car leur connectivité interne (entre les membres du sous-réseau) est beaucoup plus importante que leur connectivité externe (entre les membres de sous-réseaux différents).
La figure 5.13 montre le schéma général du réseau que COGARE a retrouvé.
Sur le schéma général du réseau, les deux sous-ensembles sont représentés par des arbres dont chaque sommet représente un gène et chaque branche une relation. Cette représentation hiérarchique des gènes n'est pas seulement représentative ; c'est effectivement la forme globale du réseau retrouvé, avec quelques gènes qui régulent d'autres gènes qui à leur tour en régulent d'autres et ce, avec peu de régulations vers un gène de la couche du dessus. Cette configuration de réseau se retrouve en effet souvent dans les réseaux génétiques. Par exemple, les levures ont des réseaux fortement hiérarchisés comme le montre la figure 5.15, extraite de [17].
Les résultats donnés par COGARE sur les données de la biolixiviation sont assez intéressants car ils sont en adéquation avec ce que l'on sait de ces réseaux (85 des 100 gènes régulateurs ont été retrouvés, et une conformation proche des réseaux d'organismes proches). L'intérêt de ces résultats réside dans le fait que de nouveaux mécanismes ont

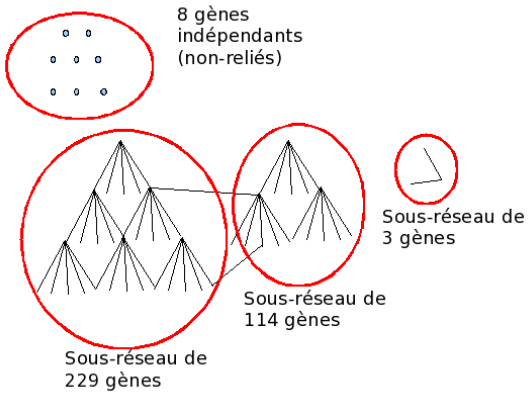


FIG. 5.13 – Schéma général du réseau retrouvé par COGARE. 4 ensembles sont visibles : le premier ensemble des 8 gènes qui ne sont reliés à aucun autre, l'ensemble formé par le sous réseau de trois gènes et les deux ensembles fortement connectés entre eux.

été retrouvés, mécanismes qui peuvent apporter aux biologistes des pistes pour l'étude de ces réseaux, par exemple, les deux sous-réseaux identifiés.

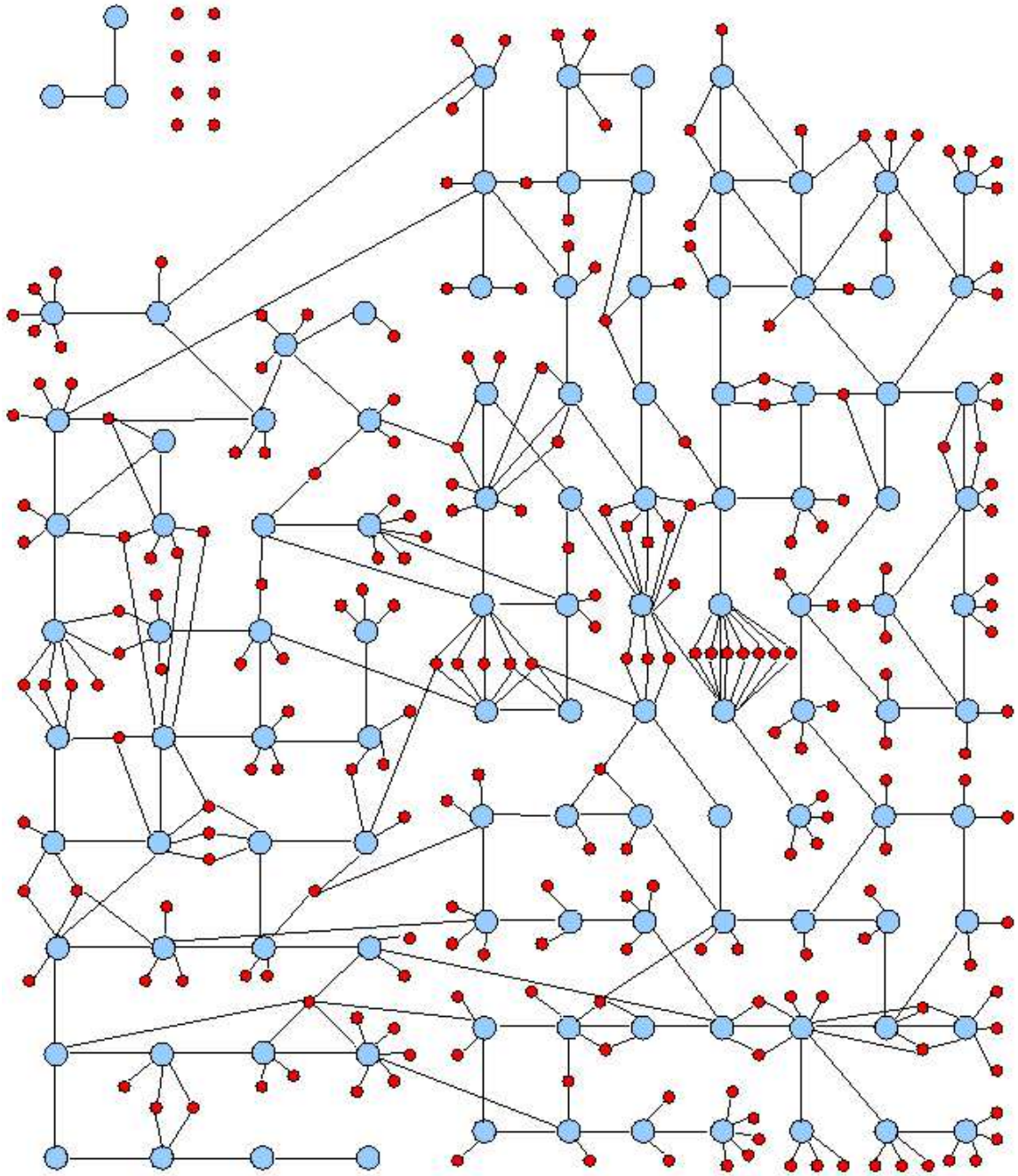


FIG. 5.14 – Schéma général du réseau de *Thiobacillus Ferooxydans*. Les points bleus désignent les régulateurs et les points rouges les gènes régulés.

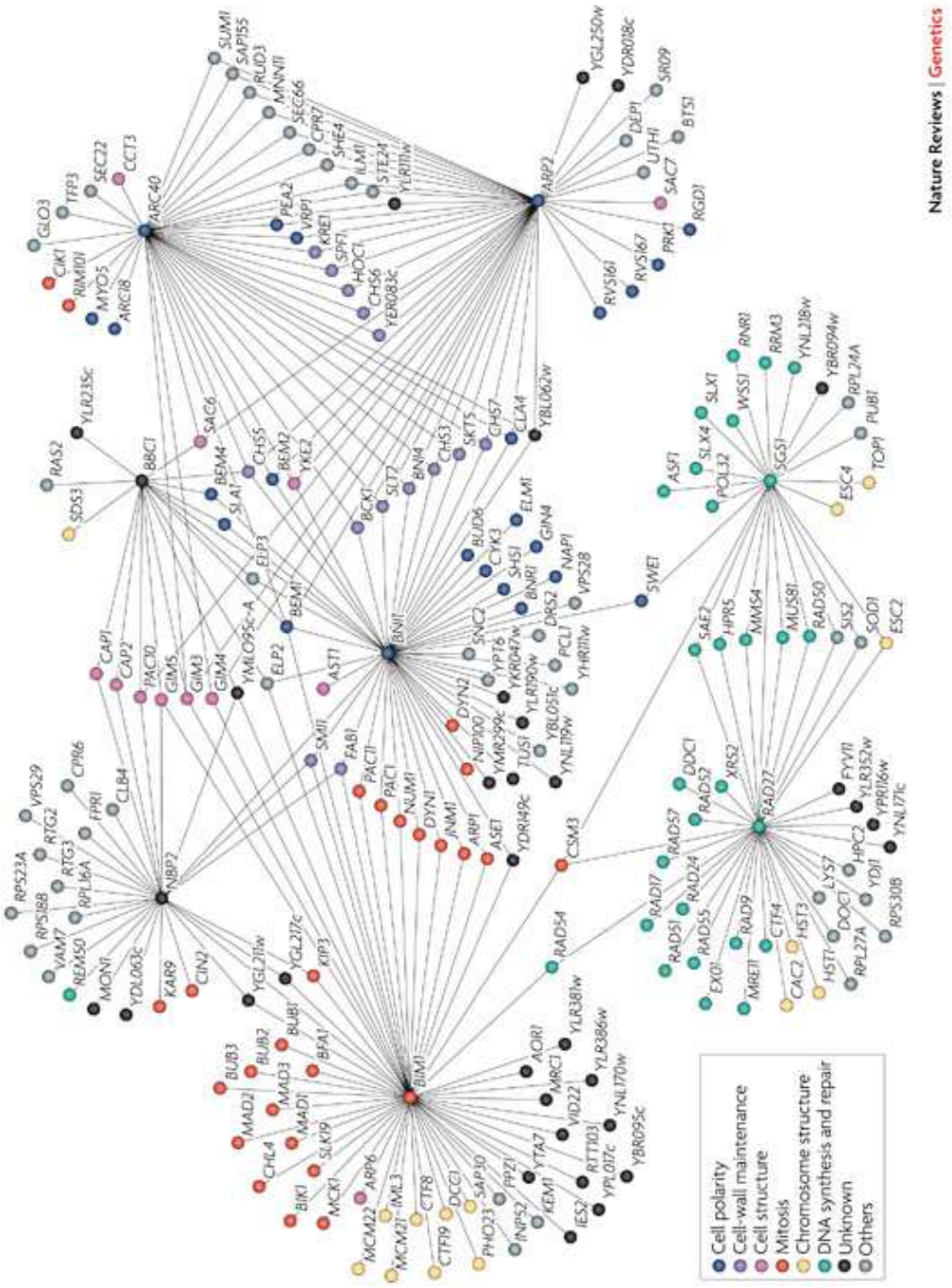


FIG. 5.15 – Schéma général du réseau de levure. Le réseau est formé de plusieurs sous-ensembles qui forment des arbres. Le réseau est hiérarchisé. Réseau obtenu par l'équipe de Charles Boone [17] sur des levures.

Conclusion et perspectives

COGARE est un logiciel utilisant l'algorithmie génétique pour optimiser un problème de reconstruction de réseau génétique. Les techniques d'acquisition de données évoluant jour après jour, il est nécessaire de mettre à jour le logiciel afin de lui permettre d'utiliser toutes les données à sa disposition pour optimiser le mieux possible les données disponibles.

Améliorations à apporter

Les différentes améliorations à apporter couvrent principalement deux aspects du logiciel, le premier porte sur les données à traiter ; le second sur l'ajout d'un module de quantification des relations.
Les données que nous pouvons traiter sont des données classiques, dynamiques et statiques. De plus en plus, il existe des nouvelles données, telles les données mutantes qui donnent de nouveaux renseignements sur les réseaux. Les données mutantes sont des données qui proviennent d'une souche modifiée génétiquement, les données obtenues peuvent donc être comparées aux données récupérées à partir des souches classiques, non-mutantes, et cela afin de mieux visualiser l'impact de la partie génétiquement modifiée. En effet, si dans les mêmes conditions expérimentales, les expressions des gènes sont différentes entre celles résultantes des données mutantes et celles résultantes des données non-mutantes, alors les modifications aperçues sont dues à la mutation, puisque le gène mutant est le seul élément de départ qui a été modifié. Il est facile d'extraire de ces données les gènes qui ont réagi et ceux qui n'ont pas été modifiés. Cela donne une information sur les liaisons existantes entre le gène muté et les autres gènes. Il est facile alors de récupérer des informations sur les cibles des gènes mutés. En recoupant l'ensemble des informations sur un grand nombre de gènes mutants, il est possible de récupérer une somme d'information importante sur le réseau. COGARE ne prend pas encore en compte ces données. Pour lui permettre de le faire, il faudrait rajouter une classe C++ qui définit les données mutantes en mettant en attributs le tableau de données et le gène muté. Ensuite il faudrait rajouter au module de calcul de l'efficacité, une partie prenant en compte les informations mutantes.
Une deuxième amélioration importante à apporter pourrait être d'augmenter la précision

du réseau en ajoutant un module de quantification des relations. En effet, cela permettrait de se sortir du carcan des données binaires. Cette quantification des relations peut se faire de différentes façons ; nous pouvons soit discrétiser de façon plus précise que le binaire pour récupérer des niveaux d'activation, soit nous pouvons essayer de calculer de façon continue l'apport de chaque relation dans le système. Le temps a manqué pour l'inclure dans COGARE mais cette partie fait partie des axes de travail que nous avons sur le logiciel.
La discrétisation plus fine des données n'entraîne pas de changement global dans l'algorithme du logiciel mais change la section simulation et la nature des matrices que l'on cherche à retrouver. Par exemple, si on veut trois niveaux d'expression à la place de deux actuellement (ce qui correspond aux états "pas exprimé", "exprimé moyennement", "fortement exprimé"), la matrice d'incidence comportera des nombres appartenant à {2,1,0,-1,-2}. Pour la simulation, il faudra multiplier chaque interaction par le poids de son arête.

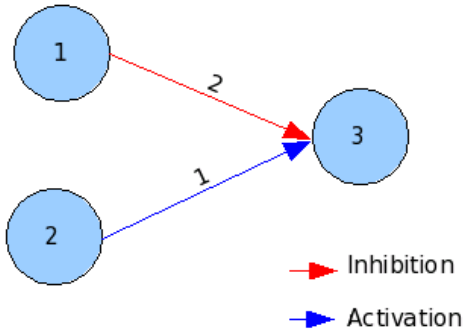


FIG. 5.16 – Exemple de réseau avec arêtes pondérées.

Gène 1	Gène 2	Gène 3	Etat final
0	0	0	0
0	2	0	2
1	1	0	-1
2	0	1	-2
0	1	2	2
1	1	1	0
2	2	2	0
1	2	1	1

Les exemples ci-dessus nous montrent les lois de calcul pour la simulation avec l'exemple de la figure 5.16.

Si l'on veut un réseau avec contribution continue, c'est à dire avec arêtes pondérées par un réel, une piste à suivre serait de calculer ces pondérations une fois le réseau classique calculé. Le fonctionnement d'un réseau avec pondération réelles est illustré à la fig.5.17. Le modèle mathématique associé est le suivant :

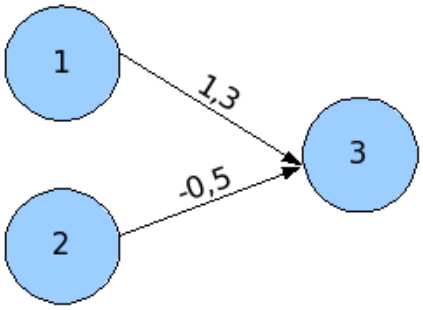


FIG. 5.17 – Exemple de réseau avec arêtes pondérées réelles.

$$G_3 = 1.3 * G_1 - 0.5 * G_2 \tag{5.4}$$

ou, plus généralement :

$$G_i = \sum_j P_{j \to i} * G_j \tag{5.5}$$

où G_i est l'expression du gène i, et $P_{j \to i}$ est la pondération de l'interaction du gène j sur le gène i.

Pour retrouver les valeurs des pondérations, dans le cas optimal, il suffit d'avoir un système de N expériences qui nous donnent les valeurs des expressions de G_i à plusieurs moments, N étant le nombre de flèches pointant vers le gène i. Malheureusement, les données sont bruitées et l'ensemble des données disponibles ne permettent généralement pas de trouver une solution unique qui satisfasse cette équation pour chaque expérience. Une piste possible serait de faire calculer ces pondérations par un réseau de neurones simple en prenant comme base d'apprentissage l'ensemble des expériences dont on dispose. Les neurones du réseau seraient les gènes, et les liaisons seraient les arêtes.

Une autre amélioration possible de COGARE serait comme expliqué dans le corps du document de créer des itérations sur le logiciel qui lui permettent d'ajouter des informations complémentaires à chaque itération. En effet, il est possible, grâce à l'efficacité locale définie de choisir les relations les plus "sûres" afin de les ajouter au pool des informations complémentaires. Ceci nous permettrait d'augmenter le nombre d'informations complémentaires et ainsi, comme le montre la figure 5.6, d'améliorer l'efficacité des solutions proposées par COGARE.

Utilisation de COGARE

Le logiciel COGARE est capable de reconstruire un réseau génétique à partir de données expérimentales avec une précision assez intéressante. Cela permettra aux biologistes de récupérer des indices sur le rôle d'un gène et/ou d'une protéine. Cet indice peut faire économiser beaucoup de temps aux chercheurs car à la place de tester un grand nombre de possibilités pour trouver le gène responsable d'une fonction biologique ou d'un phénotype, il pourra se consacrer aux candidats que le logiciel lui aura fourni.
Ensuite, une fois que le réseau est créé, il est très facile –grâce aux equations aux dérivées partielles par exemple– de simuler plus complètement le comportement de l'organisme, afin de comprendre ses évolutions et ses réactions à certains stimuli.

Perspectives sur les techniques algorithmiques

COGARE est un logiciel utilisant deux échelles d'optimisation, le local et le global. L'utilisation d'algorithmes génétiques multi-échelle est une piste à suivre pour développer les techniques d'optimisation. En effet, il est possible d'obtenir une optimisation plus fine de mécanismes de toutes sortes en combinant plusieurs algorithmes génétiques calculés à partir d'algorithmes génétiques différents, puis de faire une mise en commun des résultats grâce à un algorithme génétique chapeautant tout ça. L'algorithme génétique du dessus permettrait alors de calculer les combinaisons de solutions les plus efficaces en optimisant les pondérations de chaque efficacité.
Une autre possibilité pour les algorithmes génétiques multi-échelle serait d'utiliser les algorithmes génétiques pour optimiser des sous-processus dont l'arrangement serait à nouveau optimisé grâce à un nouvel algorithme génétique. Ceci pourrait être effectuer sur un nombre quelconque d'échelles, selon le nombre de couches de sous-processus à optimiser et le temps de calcul désiré. En effet, cette technique entraînerait des temps de calcul très longs, à cause de l'imbrication d'algorithmes génétiques déjà gourmands en temps.

Conclusion

Reconstruire un réseau génétique implique de pouvoir traiter un grand nombre de données et de pouvoir naviguer dans un espace de solutions extrêmement grand. Pour optimiser cette navigation, il est utile de pouvoir utiliser toutes les données auxquelles nous ayons accès. L'algorithme de COGARE permet d'utiliser plusieurs types de données en plus des données connues sur le réseau. Ceci lui permet d'obtenir de bons résultats dans le domaine de la reconstruction car ces données servent à contraindre l'algorithme à rester dans un espace proche de la solution.

De plus, en prenant en compte les caractéristiques des processus biologiques, COGARE permet une double optimisation, au point de vue local et au point de vue global, ce que d'autres algorithmes, même ceux utilisant l'algorithmie génétique, ne sont pas capables de faire. Ce nouveau type d'algorithme génétique permet de résoudre toutes sortes de problèmes qui ont plusieurs niveaux d'analyse. En effet, la double optimisation permet de trouver des morceaux de réseaux intéressants qui peuvent être le point de départ d'une nouvelle optimisation avec plus de contraintes.
L'analyse locale permet de découvrir les gènes responsables de processus biologiques, ce qui est un des buts de la bioinformatique. En effet, des gènes formant un processus biologique particulier seront fortement connectés entre eux. L'analyse globale quant à elle va permettre de découvrir réellement comment l'organisme réagit à son environnement et comment il évolue dans certaines circonstances, et l'optimisation globale permet justement d'optimiser le comportement de l'organisme dans sa globalité. La contrainte de l'algorithme par ajout de données fournies par l'expertise des biologistes permet d'ajouter des données ayant une fiabilité plus importante que les données classiques et donne à l'algorithme un moyen supplémentaire de converger. Cette surcontrainte de l'algorithme s'apparente aux croisements que l'on effectue depuis des siècles dans l'agriculture afin d'obtenir des plantes aux caractéristiques intéressantes.
Cet algorithme génétique donne des résultats comparables aux autres méthodes connues (cf. chapitre 5) pour la reconstruction de réseaux génétiques, tels que Banjo, NIR ou Aracne. Mais il peut être utilisé pour résoudre des problèmes autres que ceux de réseaux génétiques. En effet, n'importe quel problème de rétro-ingénierie peut se résoudre par cet algorithme, il ne suffit que de définir le modèle, les stratégies et l'efficacité, le seul inconvénient que l'on puisse rencontrer est le temps de calcul, qui peut être très long. L'avantage apporté par cet algorithme est que l'on peut ajouter toutes sortes de données connues sur le problème en modifiant la fonction d'efficacité et la fonction de mutation.
Un exemple où l'on pourrait utiliser COGARE est, toujours sur les réseaux, celui des réseaux de communications sociales et de partage de l'information. L'utilisation de COGARE permettrait de connaître les nœuds du réseau surchargés, les chemins les plus fréquentés... Dans le même registre, optimiser un temps de parcours ou un réseau routier sont des applications possibles de COGARE.

Les techniques de reconstruction génétiques se développent et deviennent de plus en plus précises, en utilisant notamment les techniques heuristiques d'optimisation. Ces techniques heuristiques permettent de résoudre des problèmes très compliqués et pas toujours résolubles en temps raisonnable. Ces problèmes se multiplient dans toutes les branches de la recherche actuelle en même temps que les avancées technologiques, scientifiques et théoriques. COGARE est un exemple de ce que l'on peut faire sur les réseaux génétiques grâce à un paradigme biologique. De plus, l'algorithme utilisé ouvre une nouvelle voie pour les algorithmes génétiques, l'approche multi-échelle, qui permet de faciliter la résolution

des problèmes de plus en plus complexes que l'industrie rencontre.

Table des figures

Annexes

Annexe A

Glossaire

Les définitions de ce glossaire sont pour la plupart tirées de l'encyclopédie en ligne Wikipédia [131].

A

Acide aminé Un acide aminé est une molécule organique possédant un squelette carboné et deux fonctions : une amine (-NH2) et un acide carboxylique (-COOH). Les acides aminés sont les unités structurales de base des protéines. Ils mesurent environ 100 picomètres (pm).

ADN L'acide désoxyribonucléique (souvent abrégé en ADN) est une molécule que l'on retrouve dans toutes les cellules vivantes. On dit que l'ADN est le support de l'hérédité ou de l'information génétique, car il constitue le génome des êtres vivants et se transmet en totalité ou en partie lors des processus de reproduction. L'ADN détermine la synthèse des protéines.

ARNm ARN synthétisé par transcription de l'ADN et codant pour une protéine. L'agencement des quatre bases (AGCU) en codons (triplets de bases) détermine la séquence en acides aminés de la protéine.

B

Bactérie Les bactéries sont des organismes vivants unicellulaires procaryotes, caractérisées par une absence de noyau et d'organites. La plupart des bactéries possèdent une paroi cellulaire glucidique, le peptidoglycane. Les bactéries mesurent quelques micromètres de long et peuvent présenter différentes formes : des formes sphériques, des formes allongées ou en bâtonnets, des formes plus ou moins spiralées. L'étude des bactéries est la bactériologie, une branche de la microbiologie.

C

Classe d'ambigüité Classe de Markov dans lesquels on ne peut séparer les éléments.

Codon Ensemble de trois nucléotides codant pour un acide aminé.

Corégulation Deux gènes sont corégulés lorsqu'ils partagent les mêmes régulateurs.

Croisement Dans un algorithme génétique, le croisement est la fonction permettant de croiser les génomes de deux parents afin d'obtenir un enfant ayant un génome nouveau.

G

Gène Un gène est une séquence d'acide désoxyribonucléique (ADN) qui spécifie la synthèse d'une chaîne de polypeptide ou d'un acide ribonucléique (ARN) fonctionnel.

Gène-cible Un gène-cible est un gène qui est régulé par un autre gène.

Gène-moteur Un gène-moteur est un gène régulateur d'autres gènes.

Génome Le génome est l'ensemble du matériel génétique d'un individu ou d'une espèce encodé dans son ADN (à l'exception de certains virus dont le génome est porté par des molécules d'ARN). Il contient en particulier toutes les séquences codantes (traduites en protéines) et non-codantes (transcrites en ARN, non traduites).

Génotoxine Les génotoxines sont des éléments qui altèrent les bases nucléotidiques de l'ADN, nous pouvons citer comme exemple les rayonnements ultraviolets, l'ozone et des toxines tels que la colibactine.

Graphe d'interaction Un graphe d'interactions est un ensemble de noeuds et d'arêtes, dirigées et signées qui va définir les différentes actions (positives ou négatives) des gènes les uns sur les autres.

H

Heuristique Une heuristique est un algorithme qui fournit rapidement (en temps polynomial) une solution réalisable, pas nécessairement optimale, pour un problème d'optimisation NP-difficile.

J

Jacobienne La Jacobienne, ou matrice jacobienne est la matrice des dérivées partielles du premier ordre d'une fonction vectorielle.

M

Matrice creuse Une matrice creuse est une matrice contenant beaucoup de zéros.

Matrice d'incidence La matrice d'incidence sommets-arcs d'un graphe G = (X,U) est une matrice A = $(a_{i,j})$ à coefficients entiers appartenant à l'ensemble 0,+1,-1 telle que $a_{i,j}$=1 si les sommets i et j sont reliés par une arête positive, -1 si l'arête est négative et 0 si il n'y a pas d'arête entre eux.

Méthode de Monte-Carlo Une méthode de Monte-Carlo est une méthode visant à calculer une valeur numérique, en utilisant des procédés aléatoires, c'est-à-dire des techniques probabilistes.

Molécule Une molécule est un assemblage d'atomes dont la composition est donnée par sa formule chimique. Le mot molécule vient du latin *molecula / moles* désignant une petite masse de matière, ou un grain de matière.

Mutation Dans un algorithme génétique, technique permettant de changer un bit de la stratégie en cours en le bit opposé.

N

NP-complet Problème dont le temps de résolution par un algorithme augmente plus vite que la taille du problème.

Nucléotide Unité de construction des acides nucléiques, résultant de l'addition d'un sucre (ribose pour l'ARN et désoxyribose pour l'ADN), d'un groupement phosphate et d'une base azotée à l'origine de l'information. Il existe quatre nucléotides différents pour l'ADN : adénine (A), thymine (T), guanine (G), cytosine (C) et quatre nucléotides différents pour l'ARN : uracile (U), guanine (G), cytosine (C), adénine (A).

O

Opéron Un opéron est un groupe de gènes comprenant un opérateur (accepteur de l'enzyme de transcription), un promoteur commun, et un ou plusieurs gènes structuraux qui sont contrôlés pour produire des ARN messagers (ARNm).

P

Phénotype Le phénotype est l'ensemble des traits observables (caractères anatomiques, morphologiques, moléculaires, physiologiques, éthologiques) caractérisant un être vivant donné (ex : couleur des yeux, des cheveux...).

Processus biologique Un processus biologique est une fonction de l'organisme correspondant à un phénotype ou à une fonction interne dont le gènes codant sont liés entre eux.

Promoteur Une séquence promotrice est une région située à proximité d'un gène et indispensable à la transcription, sur laquelle se fixe l'ARN polymérase. Les séquences promotrices peuvent être situées en aval du site d'initiation de la transcription, et même dans le gène transcrit.

Protéine Une protéine est une macromolécule composée par une ou plusieurs chaîne(s) d'acides aminés liés entre eux par des liaisons peptidiques. En général, on parle de protéine lorsque la chaîne contient plus de 100 acides aminés.

R

Réseau Un réseau est un graphe connecté constitué d'un ensemble de noeuds et d'arêtes. Biologiquement parlant, ce sont toutes les interactions des gènes et protéines d'un organisme entre eux.

Réseau Bayesien dynamique de Markov Réseau bayésien qui suit la propriété de Markov : "Il n'y a pas de dépendances directes dans le système qui ne soit directement montré par les arcs."

Rétroaction Action d'un gène sur lui-même.

Rétro-ingénierie La rétro-ingénierie (traduction littérale de l'anglais reverse engineering), également appelée rétroconception, ingénierie inversée ou ingénierie inverse, est l'activité qui consiste à étudier un objet pour en déterminer le fonctionnement interne ou sa méthode de fabrication.

Ribosome Les ribosomes sont de petites structures (ou organites) présentes dans le cytoplasme des cellules . Ce sont eux qui assemblent les acides aminés pour former les protéines . Ils suivent pour cela le plan de montage contenu dans l'ADN .

S

Sélection Méthode en algorithmie génétique permettant de choisir dans une population un individu en fonction de son adaptation au milieu.

Spottage Méthode permettant de déposer sur les puces à ADN les fragments d'ADN voulus.

T

Traduction La traduction est l'interprétation des codons de l'ARNm en acides aminés. Le code génétique est le système de correspondances (code) permettant à l'ARN d'être traduit en protéine par une cellule.

Transcription La transcription est un processus biologique ubiquitaire qui consiste, au niveau de la cellule, en la copie des régions dites codantes de l'ADN en molécules d'ARN.

Transcriptome Le transcriptome est l'ensemble des ARN messager (molécules servant de matrice pour la synthèse des protéines) issu de l'expression d'une partie du génome d'un tissu cellulaire ou d'un type de cellule.

Annexe B

COGARE

Voici quelques images du logiciel COGARE pour permettre de visualiser son fonctionnement.
Benjamin Surowiec et Thomas Szientsoff ont participé à la programmation d'une partie de l'algorithme génétique, en particulier la classe "individu", dans le cadre d'un projet de fin d'études en troisième année option Calcul Scientifique de l'école d'ingénieurs ISITV.
Ici nous pouvons voir un diagramme UML du logiciel. Chaque classe est représentée dans un rectangle avec son nom. Les attributs sont identifiés dans le rectangle du dessous et les méthodes dans celui du bas. COGARE est articulé autour de six classes qui ont chacune une fonction précise. La classe Donnees stocke les données complémentaires fournies par les biologistes, il nécessite de connaître la taille du réseau (ng) et la valeur des connaissances et leur fiabilité. Cette classe ne peut que lire et charger des données fournies par l'utilisateur sous forme de fichier.
Les classes Seriexpt et Serieexps ont pour fonction de stocker respectivement les données dynamiques et statiques. Ces données peuvent être binarisés en fonction d'un seuil. Serieexps stocke les données normales et les données binarisées. Serieexpt stocke les données normales et binarisées sous forme de données de départ (seriedep) et données d'arrivée (seriearr).
Individu est une classe qui permet de définir un individu, c'est à dire une solution possible de réseau. Elle permet de simuler le comportement de cette individu en fonction de ses caractéristiques, et possède une efficacité.
La classe Clustering permet de calculer la liste des gènes classés par ordre croissant des distances à un gène donné, et ceci à partir des données statiques (exps) et temporelles (expt) à disposition. Il est possible de choisir (choix) sur quels types de données effectuer cela (1=statique ; 2=temporelle ; 3=les deux).
AG est la classe au coeur de COGARE. Elle permet de calculer les populations d'individu en fonction du calcul d'efficacité des solutions calculé à partir des données complémentaires, des données statiques et des données temporelles. Tous les paramètres nécessaires

sont dans les attributs de cette classe (α, β, γ, taille de population (pop), nombre de générations (gen) et taux de mutation (mut).

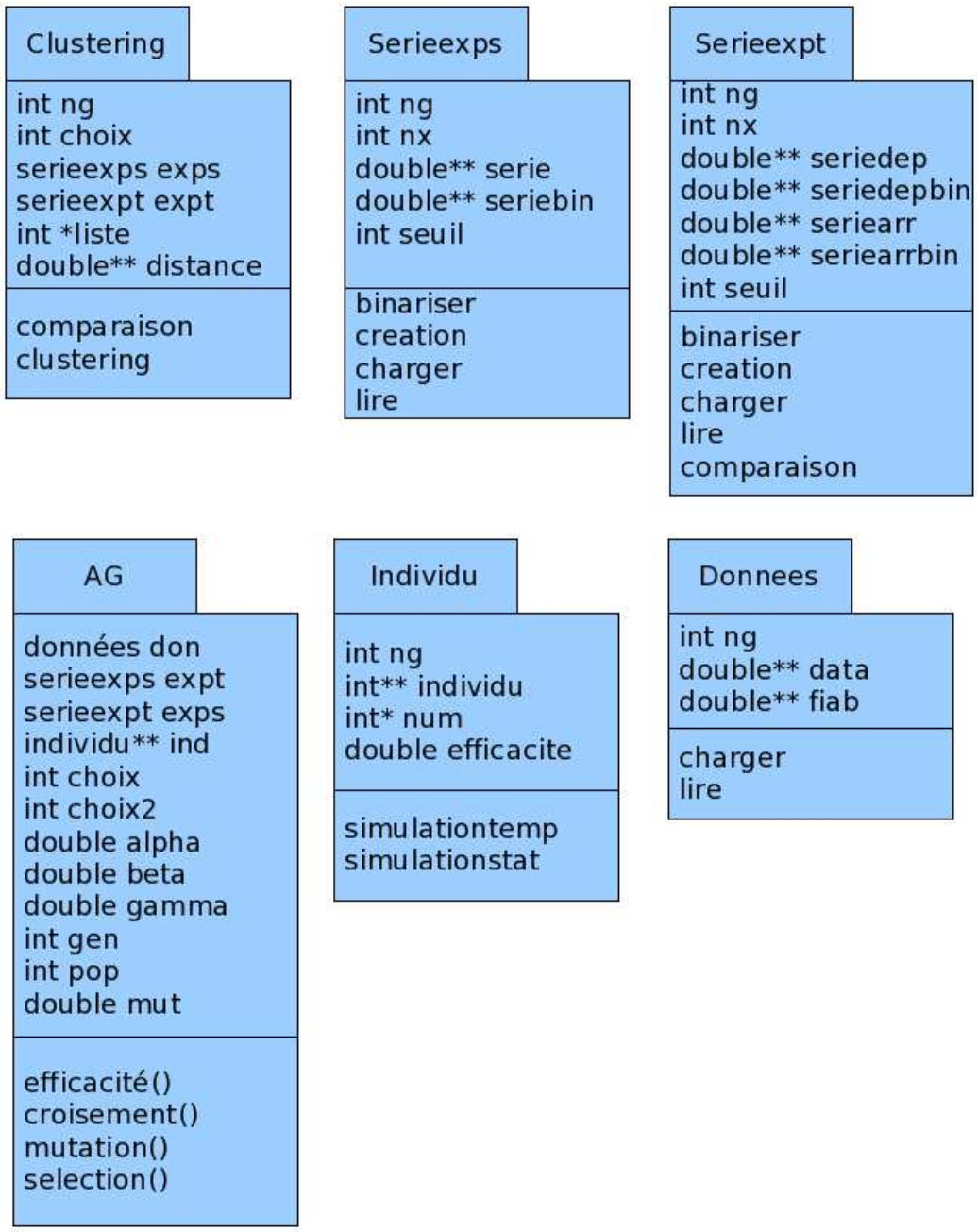


FIG. B.1 – Diagramme des classes de COGARE.

Le fonctionnement effectif de COGARE est montré dans les figures suivantes. Tout d'abord COGARE demande à l'utilisateur s'il veut utiliser des données complémentaires sur son problème, ce choix est stocké dans la variable choix de la classe AG.

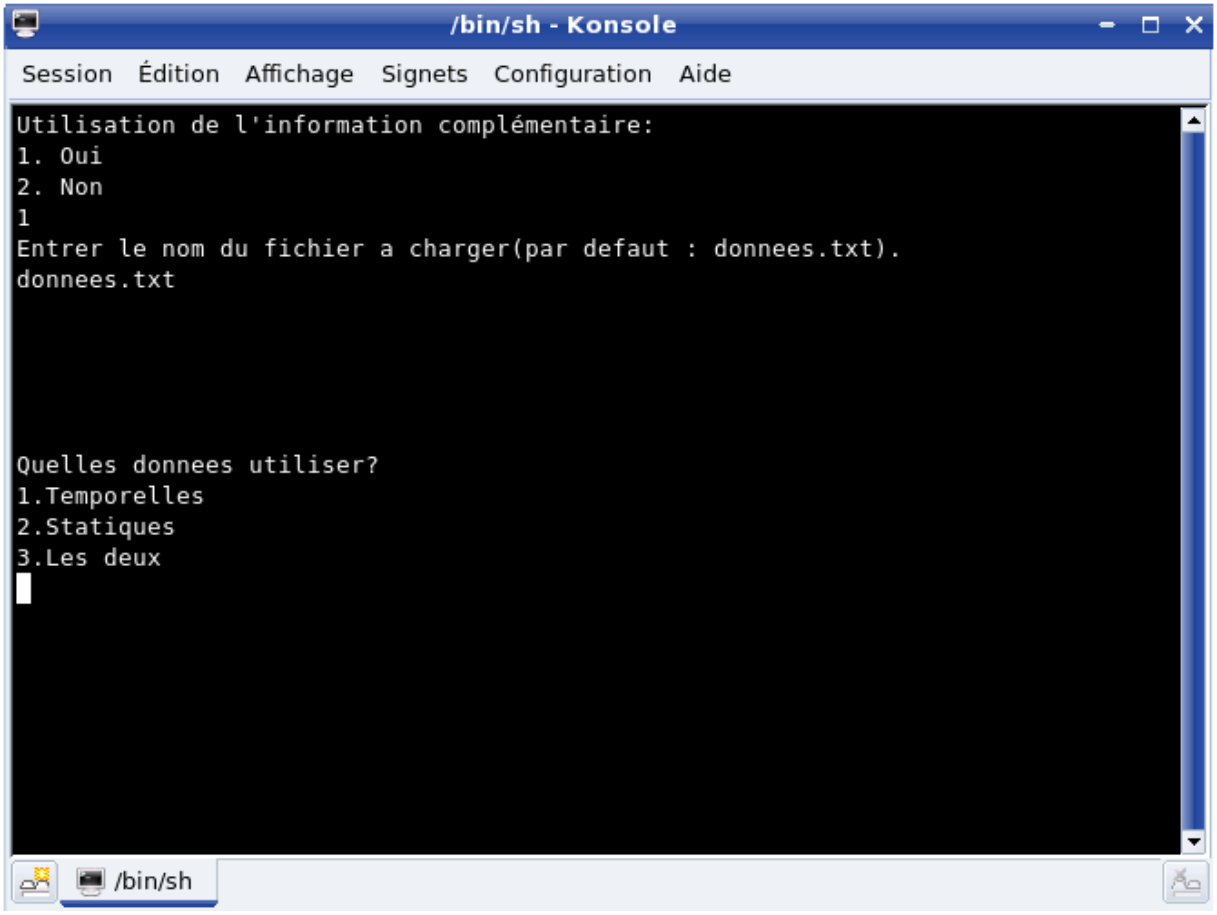


FIG. B.2 – Fenêtre de lancement de COGARE.

Dans le cas où des données complémentaires sont présentes, COGARE demande le nom du fichier dans lequel charger ces données. Les fichiers de données complémentaires doivent être au format lisible par COGARE. Ce format est visible à la figure B.3.
Ensuite, il faut choisir le type de données disponibles pour le problème. Nous pouvons choisir soit qu'il n y ait que des données statiques, soit que des données dynamiques, soit qu'il y ait les deux. Ce choix est stocké dans la variable choix2 de la classe AG.
Lorsqu'on a choisi le type de données voulues, il faut charger les données en entrant le nom du fichier à charger et en entrant le nombre d'expériences et le nombre de gènes du fichier à charger.

Les fichiers de données statiques et dynamiques ont des formats différents. Leur format peut être consulté aux figures B.5 et B.6.

Une fois les données chargées, l'utilisateur peut changer les paramètres. Après avoir rappelé les paramètres par défaut, COGARE demande à l'utilisateur s'il veut les changer. Le cas échéant, il demande les valeurs de tous les paramètres (voir fig. B.7).

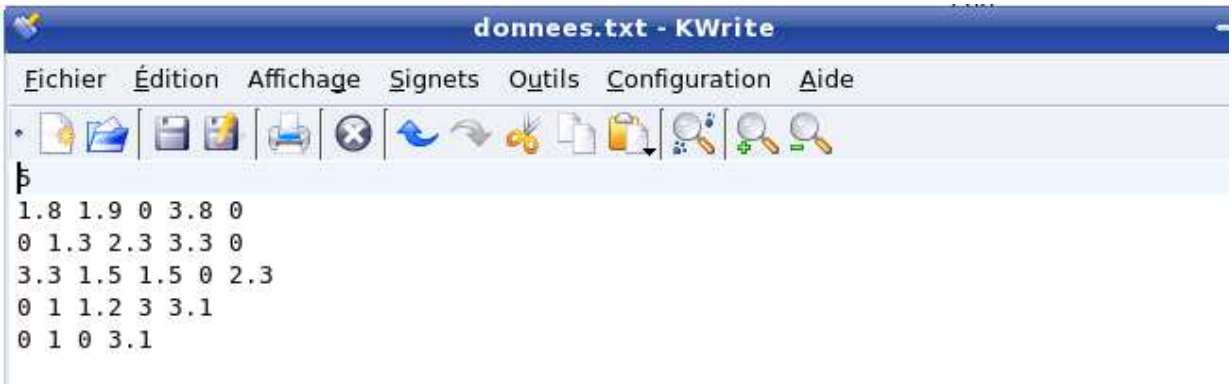


FIG. B.3 – Exemple de fichier de données complémentaires. Ce fichier comporte le nombre de gènes du réseau sur la première ligne, et ensuite les données sous le format expliqué au paragraphe 4.2.

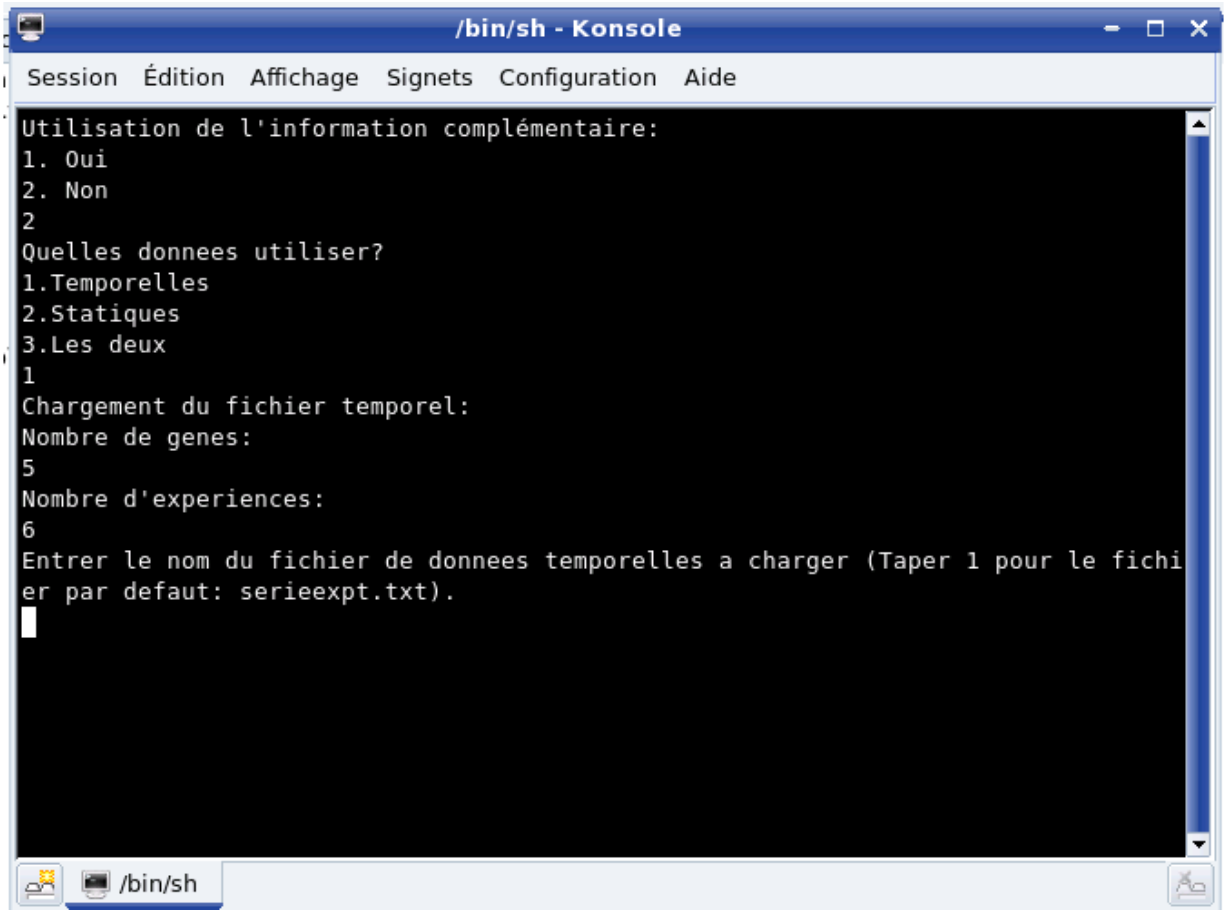


FIG. B.4 – Choix du type de données.

Une fois les données chargées et les paramètres choisis, le logiciel commence sa phase calcul. Cette phase peut durer plus ou moins longtemps suivant la taille des données et

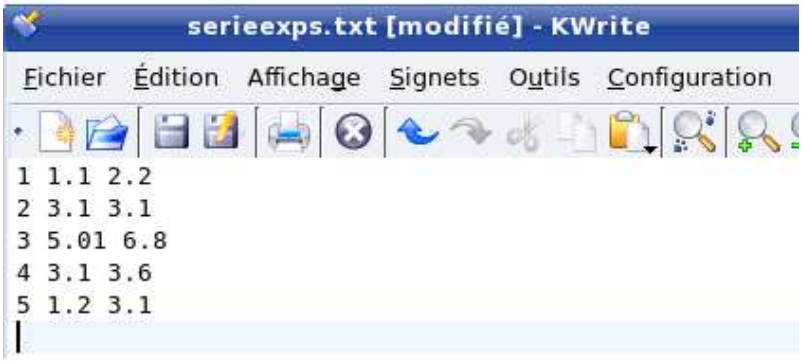


FIG. B.5 – Format des données statiques. Sur la première colonne est indiquée le numéro du gène qui nous permet de toujours garder en mémoire, ce qui permet la traçabilité des gènes tout au long du programme. Les autres colonnes sont les expériences avec sur chaque ligne, le résultat de l'expérience pour le gène indiqué par le numéro de la première colonne. Dans cet exemple, il y a 5 gènes et 2 expériences.

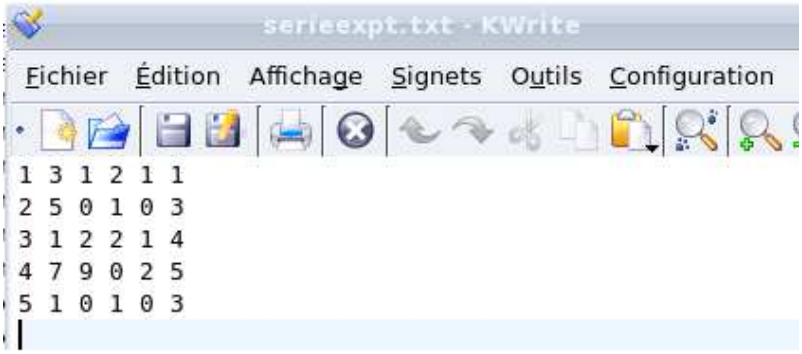


FIG. B.6 – Format des données dynamiques. Sur la première colonne est indiquée le numéro du gène qui nous permet de toujours garder en mémoire, ce qui permet la traçabilité des gènes tout au long du programme. Les autres colonnes codent les résultats des expériences, Les colonnes vont par deux avec, sur la première colonne, l'expression de départ des gènes, et sur la seconde le résultat de l'expérience. Dans cet exemple, il y a 5 gènes et 2 expériences.

les paramètres (principalement la taille d'une population et le nombre de générations). Durant ce temps de calcul, le logiciel affiche à l'écran les informations sur l'efficacité de chaque individu pour chaque génération, sous la forme 10000*efficacité, ce qui donne un nombre entre 0 et 10000, avec, comme séparateur, un zéro.
La figure B.8 montre le résultat affiché par COGARE.

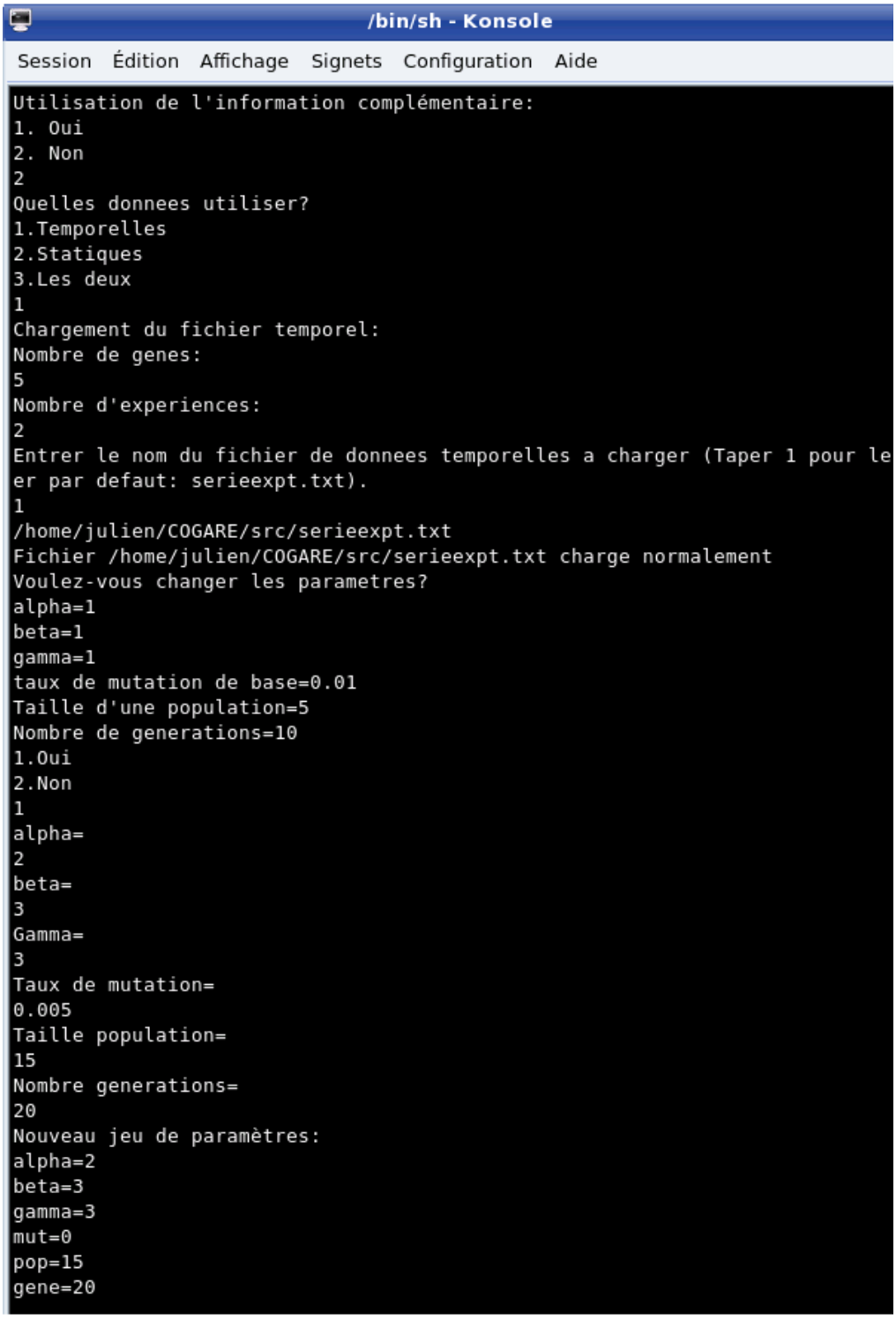


FIG. B.7 – Fenêtre de demande de changement des paramètres de COGARE.

Sous la même format que l'affichage, COGARE crée un fichier contenant l'individu résultat.

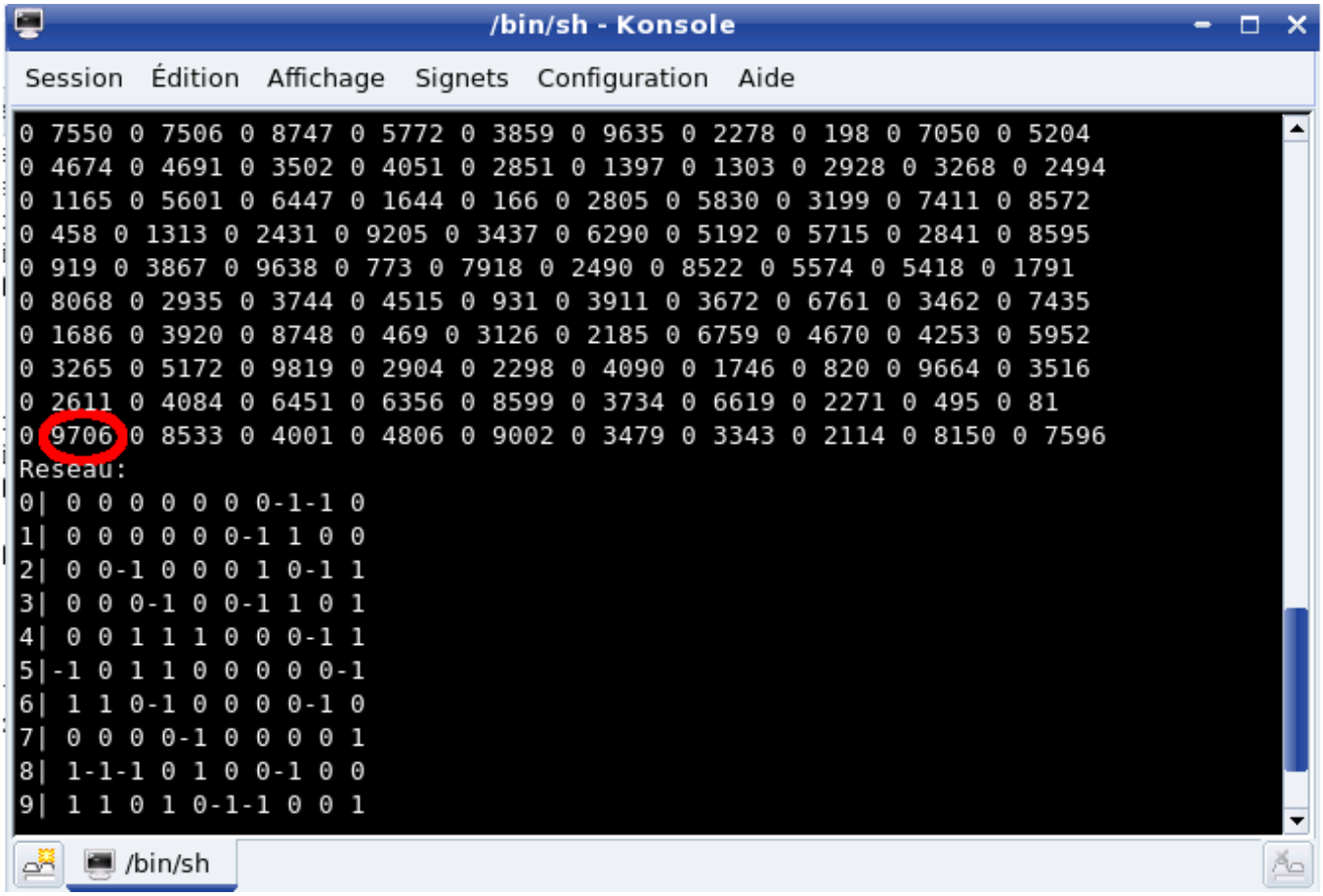


FIG. B.8 – Résultat de COGARE. Les 10 premières lignes montrent l'efficacité de chaque individu pour chaque génération (une génération par ligne). Dans ce cas, il y avait 10 générations de 10 individus. L'efficacité maximale est entourée en rouge et COGARE affiche sous forme matricielle l'individu correspondant à cette efficacité maximale.

Index

T

Bibliographie

[1] Agrawal R, Imielinski T, Swami A. (1993) Mining association rules between sets of items in large databases. Proceedings of ACM SIGMOD Conference on Management of Data. ACM Press, p. 207-216.

[2] Agrawal R., H. Mannila, R. Srikant, H. Toivonen and A. I. Verkamo (1995), Fast Discovery of Association Rules. Advances in Knowledge Discovery and Data Mining, Chapter 12, AAAI/MIT Press.

[3] Alvarez-Buylla1 E., Benítez M., Balleza Dávila E., Chaos A., Espinosa-Soto C., Padilla-Longoria P. (2007). Gene regulatory network models for plant development. Current Opinion in Plant Biology 10 (1) : 83-91.

[4] Amato R, Ciaramella A, Deniskina N, Del Mondo C, di Bernardo D, Donalek C, Longo G, Mangano G, Miele G, Raiconi G, Staiano A, Tagliaferri R (2006) A multi-step approach to time series analysis and gene expression clustering. Bioinformatics 22 : 589-596

[5] Aracena J., Demongeot J. (2004). Mathematical methods for inferring regulatory networks interactions : application to genetic regulation. Acta Biotheoretica 52 : 391-400.

[6] Arkin A, Ross J, McAdams HH (1998). Stochastic kinetic analysis of developmental pathway bifurcation in phage l-infected Escherichia coli cells. Genetics ; 149 : 1633-48.

[7] Auliac C, Frouin V, Gidrol X, d'Alché-Buc F (2008), Evolutionary Approaches for the Reverse-Engineering of Gene Regulatory Networks : a study on a biologically realistic Dataset, BMC Bioinformatics 2008, 9 :91.

[8] Baïna K., Programmation avancée. Leçon n. 22. Résolution de Problèmes : Algorithmes avides, support de cours. *http* : *//cru.um5s.ac.ma/moodle/mod/resource/view.php?id* = 551

[9] Bansal M., Belcastro V., Ambesi-Impiombato A., di Bernardo D., (2007). How to infer gene networks expression profiles. Molecular System Biology.

[10] Barricelli, Nils Aall (1954). Esempi numerici di processi di evoluzione. Methodos : 45-68.

[11] Barricelli, Nils Aall (1957). Symbiogenetic evolution processes realized by artificial methods. Methodos : 143-182.

[12] Bartholomay A. (1958). Stochastic models for chemical reactions : I. Theory of the unimolecular reaction process. Bulletin of Mathematical Biology Volume 20, Number 3.

[13] Benzer S. (1955). Fine structure of a genetic region in bacteriophage. PNAS USA 41, pp. 344-354

[14] Billingsley P., Probability and measure, Wiley, 3rd edition 1995.

[15] Blatt M, Wiseman S, Domany E, (1996). Superparamagnetic Clustering of Data. Physical Review Letters, Vol 76 N18.

[16] Blatt M, Non-ferromagnetic Potts models can be obtained from maximum likelihood and maximum entropy principles ; Ph.D. thesis, Weizmann Inst. Of Science (1997). Phys. Rev. E 63 :1101 (2001).

[17] Boone C., Bussey H., Andrews,B (2007). Exploring genetic interactions and networks with yeast. Nature Reviews Genetics 8, 437-449.

[18] Boulicaut JF, Pensa R, (2008). Constrained Co-clustering of Gene Expression Data. Proceedings SIAM International Conference on Data Mining SDM'08, Atlanta, USA, April 24-26.

[19] Bremermann H.J.,(1968) Numerical Optimization Procedures Derived from Biological Evolution Processes. In : Proc. Bionics Conf.Dayton, OH, Gordon and Breach pp. 543-561 (entitled : Cybernetic problems in Bionics) .

[20] Bremermann H.J., Rogson M. and Salaff S., (1966). Global Properties of Evolution Processes. Natural Automata and Useful Simulations, Spartan Books, Washington, D.C. (1966), pp. 3-41.

[21] Briche J., Lacroix Y., Maass A. (2009). COGARE : recherche de réseaux de régulation génétique par algorithmie génétique. Revue d'intelligence artificielle. En cours de soumission.

[22] Busygin S, Jacobsen G, Kramer E (2002). Double conjugated clustering applied to leukemia microarray data. SIAM ICDM Workshop on clustering high dimensional data.

[23] Butte A.J.(2002). The Use and Analysis of Microarray Data, Nature Reviews : Drug Discovery, 1, 12, 951-960.

[24] Califano A, Stolovitzky G, and Tu Y (2000). Analysis of gene expression microarays for phenotype classi ?cation. In Proceedings of the International Conference on Computacional Molecular Biology, pages 75-85.

[25] Cheng Y, Church G (2000). Biclustering of expression data. In Proceedings of the 8th International Conference on Intelligent Systems for Molecular Biology (ISMB ?00), pages 93-103.

[26] Cerf R. (1996). An asymptotic theory for genetic algorithms. Lecture Notes in Computer Science Volume 1063.

[27] Cover TM, Thomas JA, (1991). Elements of Information Theory. New York : John Wiley & Sons.

[28] Crick, F.H.C. (1958) : On Protein Synthesis. Symp. Soc. Exp. Biol. XII, 139-163.

[29] Crosby, Jack L. (1973). Computer Simulation in Genetics. London : John Wiley & Sons.

[30] d'Alché-Buc F, Wehenkel L (2007). Selected proceedings of Machine Learning in Systems Biology , BMC proceedings, dec 2008.

[31] de la Fuente, A., Brazhnik, P. & Mendes, P. (2001). A quantitative method for reverse engineering gene networks from microarray experiments using regulatory strengths. Proceedings of the 2nd International Conference on Systems Biology, California Institute of Technology, Pasadena, CA.

[32] Deng X, Geng H, Ali H (2005).EXAMINE : A computational approach to reconstructing gene regulatory networks. Biosystems, vol.81, p.125-136.

[33] Delbrück M., Luria S.E. (1949). Les processus stochastiques de la génétique. Coll. intern. du C.N.R S., t.13, p.121-126.

[34] D'haeseleer P., Liang S. , and Somogyi R..Genetic Network Inference : From Co-Expression Clustering to Reverse Engineering, Bioinformatics 16(8) :707-26.

[35] Dong Xu, Victor Olman, Li Wang and Ying Xu. EXCAVATOR : a computer program for efficiently mining gene expression data in Nucleic Acids Research, 2003, Vol. 31, No. 19.

[36] Edwards, S., Fertil, B., Giron, A. and Deschavanne, P.J. (2002) A genomic schism in birds revealed by phylogenetic analysis of DNA strings. Syst Biol, 51, 599-613.

[37] Elowitz, M. B., and Leibler, S., (2000). A synthetic oscillatory network of transcriptional regulators. Nature, 403, 335-338.

[38] Feller, W. (1951) Two Singular Diffusion Problems. Annals of Mathematics, 2. Ser. 54 :173-182.

[39] Fogel (1966), Artificial Intelligence Through Simulated Evolution. with A.J. Owens, and M.J. Walsh. Wiley. New York.

[40] Franke L., Harm van Bakel, Like Fokkens, Edwin D. de Jong, Michael Egmont-Petersen, Cisca Wijmenga, (2006). Reconstruction of a functional human gene network, with an application for prioritizing positional candidate genes. Am J Hum Genet 2006 Jun ;78(6) :1011-25

[41] Fraser, Alex (1957). Simulation of genetic systems by automatic digital computers. I. Introduction. Aust. J. Biol. Sci. 10 : 484-491.

[42] Fraser, Alex, Donald Burnell (1970). Computer Models in Genetics. New York : McGraw-Hill.

[43] Friedman, N., Lineal, M., Nachman, I., and Pe'er, D. (2000) Using Bayesian Networks to Analyze Expression Data. Journal of Computational Biology.

[44] Fruton JS. (1979). Early theories of protein structure, Ann. N.Y. Acad. Sci., 325, 1-18.

[45] Gardner T, di Bernardo D, Lorenz D, Collins J (2003) Inferring genetic networks and identifying compound mode of action via expression profiling. Science 301 : 102-105.

[46] Gardner,T., Faith,J. (2005). Reverse-engineering transcription control networks. Physics of Life Reviews 2 65-88.

[47] Gauss C.F. (1809). Theoria Motus Corporum Coelestium in sectionibus conicis solem ambientium.

[48] Geurts P, Wehenkel L, d'Alché-Buc F (2006), Kernelizing the ouputs of tree-based methods, to appear in the proceedings of International Conf. On Machine Learning, (ICML06).

[49] Getz G, Levine E, Domany E (2000) Coupled two-way clustering analysis of gene microarray data. PNAS USA, 97, 12079 ?12084.

[50] Gillespie D. (1976). A General Method for Numerically Simulating the Stochastic Time Evolution of Coupled Chemical Reactions. Journal of Computational Physics 22 (4) : 403-434.

[51] Gilman,A. and Ross,J. (1995) Genetic-algorithm selection of a regulatory structure that directs flux in a simple metabolic model. Biophys. J., 69, 1321-1333.

[52] Glass L, Kauffman S (1973). The logical analysis of continuous non linear biochemical control networks. Journal of theoretical Biology vol.39, p.103-129.

[53] Goldberg,D.E. (1989) Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley, Reading, MA.

[54] Goodwin BC (1965). Oscillatory behavior in enzymatic control processes. In Weber G, ed. Advances in enzyme regulation. Oxford Pergamon Press, 425-38.

[55] Halász A., Julius A., Sakar S., Kumar V., Pappas G. (2007) Controlling biological systems : the lactose regulation system of E.coli, IEEE Transactions on Automatic Control Special Issue on Systems Biology, 51-64.

[56] van Ham P., Lasters I. (1978). Reduction methods for logical control networks. J. theor Biol, 72, 269-281.

[57] Hartemink, A. J., Giford, D. K., Jaakkola, T. S. and Young, R. A. (2002). Combining Location and Expression Data for Principled Discovery of Genetic Regulatory Network Models. In Pacifc Symposium on Biocomputing 7 (2002) 437-449.

[58] Hartigan, J (1972). Direct clustering of a data matrix. Journal of the American Statistical Association (JASA), 67(337) :123-129.

[59] Heckerman, D. (1996) A tutorial on learning with Bayesian networks. Technical Report MSR-TR-95-06, Microsoft Research (March 1995 ; revised November 1996).

[60] Heckerman, D., Meek, C., and Cooper, G. (1999) A Bayesian approach to causal discovery. in Computation, Causation and Discovery. Cambridge, MA, MIT Press.

[61] Holland, John H (1975), Adaptation in Natural and Artificial Systems, University of Michigan Press, Ann Arbor.

[62] Hofmeyr, J.H. and Cornish-Bowden, A. (1994). Co-response analysis : a new experimental strategy for metabolic control analysis. Journal of Theoretical Biology, 182 (3). 371-380.

[63] Jacob F., Monod J. (1961). Genetic regulatory mechanisms in the synthesis of proteins. Journal of molecular biology, 3, 318-356. 1961.

[64] Jacob F., Monod J. et al.(1960). L'opéron : groupe de gènes à expression coordonnée par un opérateur, C.R. Acad. Sci., 250, pp. 1727-1729.

[65] Jeudy B. (2002). Optimisation de requètes inductives : application à l'extraction sous contraintes de règles d'association. Thèse à l'INSA Lyon.

[66] Kannan K, Amariglio N, Rechavi G, Jakob-Hirsch J, Kela I, Kaminski N, Getz G, Domany E, Givol D (2001). DNA microarrays identification of primary and secondary target genes regulated by p53. Oncogene 20, 2225.

[67] Kauffman S.A. (1969), Metabolic stability and epigenesis in randomly constructed genetic nets, J. Theor. Biol. 22, pp. 437-467

[68] Kauffman S. A. (1993). The origins of order. Oxford university press.

[69] Kishino H., Waddell, P. (2000). Correspondence analysis of genes and tissue types and finding genetic links from microarray data. Proceedings of the 11th Workshop on Genome Informatics (GIW '00), pp. 83-95.

[70] Kohonen T. (1982), Self-Organized Formation of Topologically Correct Feature Maps, Biological Cybernetics, vol. 46, pp. 59-69.

[71] Leloup JC, Goldbeter A.(2000). Modeling the molecular regulatory mechanism of circadian rhythms in Drosophila. Bioessays ; 22 : 84-93.

[72] Lemieux D., Kohn M. (1991). Identification of regulatory properties of metabolic networks by graph theoretical modeling. Journal of theoretical biology 1991, vol. 150, p. 3-25.

[73] Lennox, J, Biaha, T. (1991). Leaching of Copper Ore by Thiobacillus Ferrooxidans. American Biology Teacher, v53 n6 p361-68.

[74] Li D, Hao L, Dong Z (2005).On the complexity of finding emerging patterns Source. Theoretical Computer Science archive Volume 335 Pages : 15-27.

[75] Lippman R. P. (1987), An Introduction to Computing with Neural Nets, IEEE ASSP Magazine, avril 1987, p. 4-22.

[76] Lizama H.M., Fairweather M.J., Dai Z., Allegretto T.D., (2003). How does bioleaching start ? in Hydrometallurgy 69 :109-116.

[77] Madeira S, Oliveira A (2004). Biclustering Algorithms for Biological Data Analysis : A Survey . INESC-ID Technical report 1/2004.

[78] Margolin A, Nemenman I, Basso K, Wiggins C, Stolovitzky G, Della Favera R, Califano A (2006) Aracne : an algorithm for the reconstruction of gene regulatory networks in a mammalian cellular context. BMC Bioinformatics S1.

[79] Mary Huard T., Picar F. & Robin S., (2004). Introduction to statistical method for microarray data Analysis.

[80] Mendoza L, Thieffry D, Alvarez-Buylla ER. (1999). Genetic control of flower morphogenesis in Arabidopsis thaliana : a logical analysis. Bioinformatics ; 15 : 593-606.

[81] Metropolis N., Ulam S.(1949). The Monte Carlo Method. Journal of the American Statistical Association, Vol. 44, No. 247, pp. 335-341.

[82] Metropolis, N., Rosenbluth, A.W., Rosenbluth, M.N., Teller, A.H. & Teller, E. (1953). Equation of State Calculations by Fast Computing Machines. J. Chem. Phys. 21, 1087-1092.

[83] Morel,P., Reverdy, C., Michel,B,Ehrlich,S.D., & Cassuto,E, (1998). The role of SOS and flap processing in microsatellite instability in Escherichia coli. Proc. Natl. Acad. Sci. USA 95 :10003-10008.

[84] Morton-Firth CJ, Shimizu TS, Bray D (1999). A free-energy-based stochastic simulation of the Tar receptor complex. J Mol Biol ; 286 : 1059-74.

[85] Murphy, K. and Mian, S. (1999). Modeling gene expression data using dynamic bayesian networks. Technical Report, University of California at Berkeley, Department of Computer Science.

[86] Park,L.J., Park,C.H., Park,C. and Lee,T. (1997) Application of genetic algorithms to parameter estimation of bioprocesses. Med. Biol. Eng. Comput, 35, 47-49.

[87] Pearl Judea (1985). Bayesian Networks : A Model of Self-Activated Memory for Evidential Reasoning. In Proceedings of the 7th Conference of the Cognitive Science Society, University of California, Irvine, CA, pp. 329-334, August 15-17.

[88] Pécou E., Maass A., Remenik D., Briche J. and Gonzalez M., (2006). A Mathematical Model for Copper Homeostasis in Enterococcus hirae. Mathematical Biosciences, Volume 203, Issue 2, October 2006, Pages 222-239.

[89] Perrin B, Ralaivola L, d'Alché-Buc F, Bottani S, A. Mazurie (2003), Gene networks inference using dynamic bayesian networks. Journal of Bioinformatics

[90] Petri, CA., (1962). Fundamentals of a Theory of Asynchronous Information Flow. 1st IFIP World Computer Congress. Munich 1962, North Holland, pp 386-390,

[91] Pfaltz J, Jamison R. (2000). Closure Systems and their Structure, 5th Intern'l Seminar on Relational Methods in Computer Science, RelMiCS 2000, Valcartier, Quebec, Jan. 2000, 121-123.

[92] Pridgeon C, Corne D (2004). Genetic network reverse-engineering and network size ; can we identify large GRNs ? Computational Intelligence in Bioinformatics and Computational Biology. CIBCB '04. Proceedings of the 2004 IEEE Symposium.

[93] Reddy, V.N., Liebman, MN & Mavrovouniotos, ML (1996) Qualitative analysis of biochemical reaction systems. Comput. Biol. Med 26(1), 9-24.

[94] Ribeiro A., Lloyd-Price J. (2007). SGN Sim, a Stochastic Genetic Networks Simulator, Bioinformatics, 23(6) :777-779.

[95] Robert D., Vian B. (1994), Eléments de biologie cellulaire. Doin.

[96] Rose K, Gurewitz E, Fox G (1990). Statistical mechanics and phase transitions in clustering. Phys. Rev. Lett. 65 :945.

[97] Sánchez L, van Helden J, Thieffry D. (1997). Establishment of the dorso-ventral pattern during the embryonic development of Drosophila melagonaster : a logical analysis. J Theor Biol ; 189 : 377-89.

[98] Sánchez L, Thieffry D. (2001). A logical analysis of the gap gene system. J Theor Biol ; 211 : 115-41.

[99] Samal A., Jain S. (2008). The regulatory network of E. coli metabolism as a Boolean dynamical system exhibits both homeostasis and flexibility of response. BMC Syst Biol. ; 2 : 21.

[100] Savageau, M. A.(2001)Chaos 11(1), 142-159).

[101] Savageau,M.A. (1976) Biochemical System Analysis : a Study of Function and Design in Molecular Biology. Addison-Wesley, Reading, MA.

[102] Schneider J (1998). First-order phase transitions in clustering. Phys. Rev. E 57 :2449.

[103] Schoch, C et al (2002). Acute myeloid leukemias with reciprocal rearrangements can be distinguished by specific gene expression profiles, PNAS July 23, 2002 vol. 99 no. 15 10008-10013.

[104] Schwarz, G. (1978). Estimating the dimension of a model. Annals of Statistics 6 (2) : 461-464

[105] Schwefel, Hans-Paul (1974). Numerische Optimierung von Computer-Modellen (PhD thesis).

[106] Schwefel, Hans-Paul (1977). Numerische Optimierung von Computor-Modellen mittels der Evolutionsstrategie : mit einer vergleichenden Einführung in die Hill-Climbing- und Zufallsstrategie. Birkhäuser.

[107] Schwefel, Hans-Paul (1981). Numerical optimization of computer models Translation of 1977 'Numerische Optimierung von Computor-Modellen mittels der Evolutionsstrategie'. Wiley. ISBN 0471099880.

[108] Schwikowski B, Uetz P, Fields S, (2000). A network of protein-protein interactions in yeast in Nat Biotechnol. 2000 Dec ;18(12) :1242-3.

[109] Segal E, Taskar B, Gasch A, Friedman N, Koller D (2001). Rich probabilistic models for gene expression. Bioinformatics, volume 17 , pages 243-252.

[110] Sheng Q, Moreau Y, De Moor B (2003). Biclustering micrarray data by gibbs sampling. In Bioinformatics, volume 19 (Suppl2).

[111] Shoudan L., Fuhrman Stefanie, Somogyi Roland. Reveal, A General Reverse Engineering Algorithm For Inference Of Genetic Network Architectures. Pacific symposium on Biocomputing 3 :18-29 (1998).

[112] Snoussi E (1989). Qualitative dynamics of piecewise-linear differential equations : 1 discrete mapping approach. Dynamics and Stability of Systems vol.4 p.189-207.

[113] Spirtes P, Glymour C, Scheines R, Kauffman R, Aimale V, Wimberly F. Constructing Bayesian Network Models of Gene Expression Networks from Microarray Data (2000).

[114] Projet LumImDynNet financé par l'ACI IMPBIO. http ://pari-stic.labri.fr/IMPBio/LumImDynNet_2004.pdf

[115] Tibshirani R, Hastie T, Eisen M, Ross D, Botstein D, and Brown P (1999). Clustering methods for the analysis of DNA microarray data. Technical report, Department of Health Research and Policy, Department of Genetics and Department of Biochemestry, Stanford University.

[116] Thomas R., 1973, Boolean formalization of genetic control circuits, J. Theor Biol 42, pp. 563 ?585.

[117] Toh H., Horimoto K. (2002).Inference of a genetic network by a combined approach of cluster analysis and graphical Gaussian modeling. Bioinformatics, vol. 18, p. 287-297.

[118] Törönen P., Kolehmainen M., Wong G., Castréna E.. Analysis of gene expression data using self-organizing maps, FEBS Letters 451 (1999) 142-146.

[119] Ucar D., Neuhaus I., Ross-MacDonald P., Tilford C., Parthasarathy S., Siemers N., and Ji R.-R. (2007). Construction of a reference gene association network from multiple profiling data : application to data analysis. Bioinformatics, October 15, 2007 ; 23(20) : 2716 - 2724.

[120] Wagner A. (2004). Reconstructing Pathways in Large Genetic Networks from Genetic Perturbations Andreas Wagner. Journal of Computational Biology, 11(1) : 53-60.

[121] Wang J, Delabie J, Aasheim H, Smeland E, Myklebost O (2002). Clustering of the SOM easily reveals distinct gene expression patterns : results of a reanalysis of lymphoma study. BMC Bioinformatics 2002 3-36.

[122] Wang H, Wang W, Yang J, Yu P (2002). Clustering by pattern similarity in large data sets. In Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data, pages 394-405.

[123] Wang H, Wang W, Yang J, Yu P (2002) S-clusters : Capturing subspace correlation in a large data set. In Proceedings of the 18th IEEE International Conference on Data Engineering, pages 517-528.

[124] Weaver W. & Shannon C. (1963). The Mathematical Theory of Communication. Univ. of Illinois Press. ISBN 0252725484.

[125] Yagil E, Yagil G (1971). On the relation between effector concentration and the rate of induced enzyme synthesis. Biophysical Journal vol 11 p11-27.

[126] Yu J, Smith VA, Wang PP, Hartemink AJ, Jarvis ED (2004) Advances to bayesian network inference for generating causal networks from observational biological data. Bioinformatics 20 : 3594-3603.

[127] Zhu R., Roussel, M.R. (2006). Validation of an algorithm for delay stochastic simulation of transcription and translation in prokaryotic gene expression. Phys. Biol. 3, 274-284

[128] Zhu R., Ribeiro A., Kauffman, S.A. (2006). A General Modeling Strategy for Gene Regulatory Networks with Stochastic Dynamics. Journal of Computational Biology, 13(9), 1630-1639.

[129] http ://www.kdevelop.org

[130] http ://www.ncbi.nlm.nih.gov/Genbank/

[131] http ://fr.wikipedia.org

[132] http ://perl.developpez.com/tutoriels/bioinformatique/bio-db-genbank/

Résumé.

Les réseaux génétiques permettent aux biologistes de simuler le fonctionnement d'un organisme, et facilitent ainsi la compréhension des fonctions des gènes. Nous présentons une approche pour la reconstruction de réseaux génomiques de grande taille, basée sur l'utilisation d'algorithmes génétiques. L'innovation réside en ce que notre algorithme propose une combinaison d'évolutions génétiques à une échelle locale d'abord, globale ensuite.
Il intègre des contraintes, données du problème. Il a la capacité de traiter indifféremment des données statiques, dynamiques ou les données complémentaires. Comparé aux algorithmes existants sur des données numériques, il se révèle plus performant dans certains cas de figures, équivalent sinon.
Il a été également testé avec succès dans le cadre de la régulation génomique appliquée au processus de biolixiviation du cuivre, et permet d'identifier de nouvelles fonctions de la bactérie impliquée.

Mots-clefs : Réseau génétique, ADN, Puces à ADN, Algorithme génétique, Ingénierie reverse, multi-échelle.

Abstract

Genetic networks allow biologists to simulate the behaviour of an organism, therefor ease the understanding of genes functions. We present an algorithm for the reconstruction of genetic networks based on a new method combining local and global evolutionary rules. It has the ability to integrate constraints avoiding searching non relevant parts of the solution space. It has also the capacity to deal with either static or dynamic data, and was applied succesfully in the case of gene regulation for copper biolixiviation process. Compared to existing algorithms on numerical data, it reveals, in some circumstances, better, and, in a large panel of other cases, equivalent to the best.

Keywords : Genetic network, DNA, Microarrays, Genetic Network, Reverse-engineering, multi-scale.