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

Julien Briche^{*,} – Yves Lacroix^{*} – Alejandro Maass^{**}.**

** Systèmes Navals Complexes,
Avenue Georges Pompidou 83160 La Valette-du-var, France*

*** Centro de Modelamiento Matemático, UMI CNRS 2806
Blanco Encalada 2120 Piso 7 Santiago, Chile*

briche@univ-tln.fr

RÉSUMÉ. Nous proposons une approche “algorithme génétique” pour la reconstruction génomique. Notre approche introduit le concept d’algorithmie génétique multi-échelle : l’optimisation est conduite simultanément à une échelle locale et à une échelle globale. La fonction d’efficacité est donc hybride. Notre approche prend également en compte plusieurs types de données, dynamiques, statiques, ou imposées. Il en résulte un nouveau logiciel de reconstruction génomique, COGARE. Il est étalonné sur données simulées et comparé aux algorithmes existants. Il est utilisé sur deux cas réels, sur lesquels il révèle des capacités à renvoyer des informations pertinentes au biologiste.

ABSTRACT. We propose a Genetic Algorithm for genetic reconstruction. Our approach introduces a multi-scale genetic algorithm : the optimization takes place at two levels, one local on single genes, and one global on groups of genes. Hence the efficiency function is hybrid. We also take account of three kinds of data, dynamical, static and supplementary. This is set up in a software: COGARE. It has been tested on simulated data and compared to existing softwares. It has also been used on two real cases, where it proved able to return interesting hints for biologists.

MOTS-CLÉS : Algorithme génétique, rétro-ingénierie, réseaux génétiques, reconstruction de réseaux, optimisation, puces à ADN.

KEYWORDS: Genetic algorithm, reverse engineering, genetic network, network reconstruction, optimisation, microarrays.

1. Introduction

Dans cet article, nous utilisons les réseaux génétiques (Schwefel, 1977) pour la reconstruction génomique. Notre travail porte sur la reconstruction des réseaux de grande taille. Pour traiter ce cas particulier, nous proposons un algorithme génétique modifié qui développe une approche multi-échelle et multi-donnée.

Les données nécessaires pour la reconstruction de réseau génétique proviennent de l'analyse de biopuces, qui donnent des renseignements sur l'activation de gènes durant différentes expériences. Les biopuces sont ensuite analysées et normalisées afin de pouvoir être utilisées pour la reconstruction génétique. Le chapitre 1.6 du *Handbook of Statistical Genetics* (Huber *et al.*, 2004) explique de manière précise l'ensemble des procédés nécessaires pour la création et l'utilisation des puces à ADN.

Les interactions génétiques sont représentées par différents modèles : citons les réseaux booléens (Kauffman, 1969) et les réseaux bayésiens (Pearl, 1985). Nous avons choisi comme modèle celui des graphes signés (Berge, 2003), qui associent à chaque réseau des sommets et des arêtes, les premiers représentant les éléments (le couple gène-protéine), et les seconds les interactions (comment un gène agit sur un autre). Ce modèle a déjà été utilisé sur les problèmes de reconstruction génétique (Wagner, 2001).

La recherche des paramètres du modèle constitue la seconde étape importante de la reconstruction génomique. Il existe plusieurs techniques, par exemple celle des équations aux dérivées partielles (EDP), ou celle des méthodes de classification telles que le partitionnement. Nous avons choisi d'utiliser un algorithme génétique pour retrouver les paramètres de notre modèle (Holland, 1975). Cette méthode permet de trouver des solutions acceptables à des problèmes complexes et en un temps raisonnable (Cerf, 1996).

L'étude de la reconstruction de réseaux génétiques a conduit au développement de plusieurs logiciels, chacun se basant sur différentes théories (tant modèles que recherche des paramètres). Notre approche est concrétisée par la production d'un nouveau logiciel, COGARE. Il a été comparé avec trois logiciels, de façon avantageuse dans le cas des grands réseaux à données mixtes. Il a également été testé de façon concluante sur des données réelles.

Le logiciel NIR (Gardner *et al.*, 2003) utilise la théorie des EDP. 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.

Aracne (Margolin *et al.*, 2006), quant à lui, est un algorithme qui utilise l'information mutuelle comme base de départ. Il compare l'information au sens de Shannon apportée par un gène par rapport à un groupe de gènes. Si cette information est nulle il considère que le gène est défini par le groupe.

Enfin, Banjo (Yu *et al.*, 2004) est basé sur des méthodes bayésiennes qui ont démontré des capacités intéressantes dans le domaine de la reconstruction de réseaux. Dans cette approche, les réseaux génétiques sont représentés par des réseaux bayésiens dynamiques, de type Markov du premier ordre. Le réseau solution optimise un score qui dépend du réseau et des données disponibles.

Nous avons testé COGARE sur plusieurs types de données afin de valider le logiciel et l’algorithme d’optimisation dont il est issu. Les tests ont d’abord porté sur des données simulées, ce qui nous a permis de comparer COGARE avec les autres logiciels du marché. La seconde série de tests a été effectuée sur des données réelles, bruitées ou incomplètes. Les données choisies portaient sur un organisme en lien avec la biolixiviation et sur *Escherichia Coli*, une bactérie bien connue et largement documentée. Ces deux séries de tests ont donné des résultats prometteurs qui sont en adéquation avec les informations connues sur les organismes étudiés. COGARE a donc de bonnes capacités de reconstruction sur des données réelles.

L’article est organisé comme suit : nous présentons dans un premier temps le principe de fonctionnement du logiciel COGARE, et notamment nous expliquons l’approche multi-échelle, qui constitue l’innovation principale de notre démarche. Ensuite, les deux types de résultats sont présentés : les résultats sur des données simulées, puis ceux sur des données réelles. En guise de conclusion, nous proposons quelques pistes pour l’évolution du logiciel et de la technique.

2. Présentation de COGARE

COGARE signifie COnstrained GEnetical Algorithm for Reverse Engineering, et se traduit par “rassembler” en latin. Le code est écrit en C++ sur kdevelop et comporte 5 000 lignes. Le code est conçu de façon modulaire. Il est actuellement l’objet d’un dépôt auprès de l’agence pour la protection des programmes (APP).

COGARE accepte en entrée trois types de données :

- les données statiques (des données sur un même organisme suivant des expériences différentes),
- les données dynamiques (des données sur une même expérience à des temps différents),
- les données complémentaires (des informations que les biologistes connaissent sur l’organisme).

Les algorithmes génétiques (AG) permettent de recombinaison des jeux de données (stratégies par la suite). Un AG recompose une population (de stratégies) et génère une descendance (de nouvelles stratégies). On sélectionne deux parents, que l’on croise, puis on fait muter le descendant. Les individus d’une population donnée sont évalués à travers une fonction d’efficacité. Si l’on choisit des parents efficaces, on espère que les enfants hériteront d’une efficacité meilleure que celle de leurs parents. COGARE est composé de deux sections, “simulation” d’une part, et “reconstruction” d’autre part.

COGARE va générer des réseaux à partir des données d’entrée de façon itérative. La génération des réseaux relève de la section “reconstruction”, celle des données, de la section “simulation”.

2.1. “Simulation”

Les données générées par la simulation sont destinées à être comparées aux données d’entrée, et cette comparaison permettra de calculer l’efficacité du réseau utilisé (la stratégie considérée, ou “candidat” réseau).

La section simulation se contente de faire évoluer sur un pas de temps le réseau utilisé. Pour connaître l’état du système au temps suivant il faut prendre en compte l’état initial du système, qui est, avec le réseau utilisé, la seconde donnée d’entrée de la section simulation.

Le réseau utilisé transforme l’état initial du système en suivant certaines règles. Les gènes agissent sur leurs gènes-cibles si et seulement si ils sont actifs, ces actions étant régies par la règle générique suivante :

$$E_g^{t+1} = E_g^t + \sum_{fl} sgn(fl) * E_{gm}^t \quad [1]$$

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. Nous traitons ces données pour les rendre simplement binaires. Pour cela, nous faisons passer les résultats par une fonction seuil, qui pousse à 0 les gènes dont l’activation est inférieure à un certain seuil, et à 1 les gènes dont l’activation est supérieure à ce seuil.

2.2. “Reconstruction”

La section reconstruction est la section de calcul de l’algorithme qui va fournir les stratégies. Cette section est basée sur un AG dont les caractéristiques sont optimisées. Les différentes caractéristiques à optimiser sont :

- le taux de mutation,
- la fonction d’efficacité,
- le type de croisement à utiliser,
- le type de sélection.

2.2.1. Taux de mutation

Le taux de mutation est optimisé en fonction des données complémentaires : si l'interaction donnée par l'algorithme est la même que celle de l'information complémentaire, le taux de mutation est égal à $tx - fi * 0.9 * tx$ où tx est le taux de mutation normal et fi le degré de fiabilité de l'information. Si l'information est différente, ce taux de mutation devient : $tx + fi * 0.9 * tx$.

Ce taux permet, comme le montre la figure 1, d'améliorer la convergence de l'algorithme, en le contraignant à rester dans des zones de l'espace des solutions où l'on sait a priori qu'il doit se cantonner.

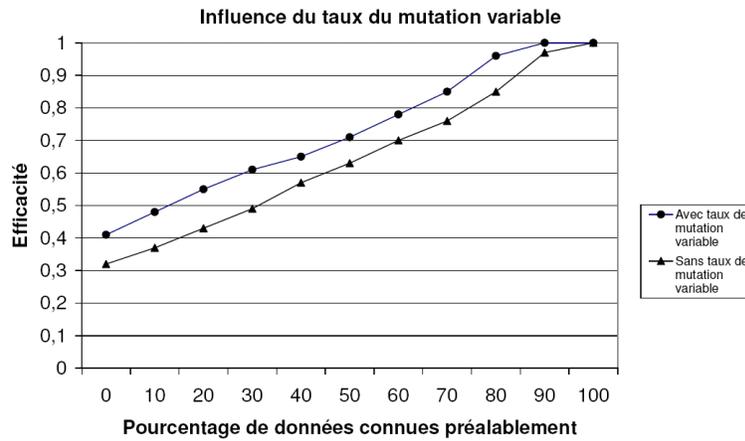


Figure 1. Comparaison du pourcentage de ressemblance du réseau solution avec le réseau réel connu, avec ou sans taux de mutation variable

2.2.2. La fonction d'efficacité

La deuxième caractéristique importante à optimiser est la fonction d'efficacité. Pour pouvoir calculer l'efficacité, les trois types de données acceptées par COGARE (statiques, dynamiques et complémentaires) vont être comparées aux résultats donnés par l'algorithme, de façon différenciée, car ces données ne sont pas au même format.

Les informations complémentaires sont la collection des connaissances que l'utilisateur a d'une interaction du réseau (un chiffre de la matrice d'incidence du graphe), assorties d'un degré de fiabilité. L'efficacité d'une stratégie particulière pour les informations complémentaires, notée \mathcal{E}_1 , est calculée de la manière suivante :

$$\mathcal{E}_1 = \frac{\sum_{i \in D} R_i * \delta_{I_i, G}}{\sum_i R_i} \quad [2]$$

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 sont proches des informations complémentaires.

Pour les données dynamiques, la simulation est mise à contribution. Les données dynamiques donnent l'état des gènes à un instant t et à l'instant $t + 1$. La simulation renvoie l'état des gènes simulé à l'instant $t + 1$ à partir de leur état à l'instant t . Les deux états disponibles à l'instant $t + 1$ sont comparés, et cette comparaison est résumée au travers de la fonction d'efficacité :

$$\mathcal{E}_2 = 1 - \frac{\sum_{i=1}^M \sum_{j=1}^N \|Q_{i,j} - G_{i,j}\|}{M * N} \quad [3]$$

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 i et $G_{i,j}$ est l'état du gène j dans l'expérience i . 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, pour palier l'absence d'instant initial, nous avons choisi d'utiliser le clustering pour calculer une efficacité associée à ces données. L'idée est de comparer le clustering des données temporelles disponibles avec celui des données issues du réseau à tester. Pour créer des données à partir de la stratégie testée, nous utilisons le simulateur : on choisit aléatoirement plusieurs états du système et on fait tourner le simulateur sur ces états plusieurs fois. Le clustering de ces données est ensuite comparé au clustering des données statiques fournies.

Dans COGARE, le clustering utilisé est particulier : on calcule la matrice des distances entre les gènes et on ordonne chacune de ses colonnes par distance croissante. Ensuite, on compare les ordres des colonnes de chaque matrice pour les expériences statiques et pour les données reconstruites. Cette comparaison normalisée conduit à la définition de l'efficacité associée de la solution :

$$\mathcal{E}_3 = 1 - \frac{\sum_{i=1}^N \sum_{j=1}^N 1 - \delta_{O_{i,j}, P_{i,j}}}{N * N} \quad [4]$$

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.

Ensuite, pour définir l'efficacité, nous recombinaisons les efficacités \mathcal{E}_1 , \mathcal{E}_2 , \mathcal{E}_3 par un procédé de pondération : soient α , β et γ positifs, alors l'efficacité s'écrit :

$$\mathcal{E} = \frac{\alpha * \mathcal{E}_1 + \beta * \mathcal{E}_2 + \gamma * \mathcal{E}_3}{\alpha + \beta + \gamma} \quad [5]$$

2.2.3. Le croisement

Le croisement que nous proposons diffère de celui que l'on rencontre dans les méthodes dites "classiques". Le croisement s'effectue généralement en coupant aléatoirement les génomes et en accolant l'un à l'autre deux des morceaux sélectionnés.

Ici le choix de découpe du génome se fait en considérant l'efficacité locale d'une solution. Cette efficacité locale se calcule comme l'efficacité globale, à ceci près qu'elle ne prend en compte que la colonne de la matrice d'incidence associée au gène considéré (ici, numéro i) :

$$\mathcal{E}_{\{Gr\},1} = \frac{\sum_{i \in \{Gr\}} R_i * \delta_{I_i,G}}{\sum_i R_i} \quad [6]$$

$$\mathcal{E}_{\{Gr\},2} = 1 - \frac{\sum_{i \in \{Gr\}} \sum_{j=1}^N \|Q_{i,j} - G_{i,j}\|}{Card(Gr) * N} \quad [7]$$

$$\mathcal{E}_{\{Gr\},3} = 1 - \frac{\sum_{i \in \{Gr\}} \sum_{j=1}^N 1 - \delta_{O_{i,j},P_{i,j}}}{Card(Gr) * N} \quad [8]$$

L'efficacité locale \mathcal{E}_i nous guide pour choisir les morceaux d'une stratégie que nous allons sélectionner, dans chacune des stratégies parentes, pour construire la stratégie descendante. La méthode de sélection de ces "morceaux" est ici en fait plus complexe que ce que nous nous restreignons à exposer dans cet article (Briche, 2009). En effet, pour définir localement une efficacité, il faut enrichir un peu les formules ci-dessus, pour prendre en compte les nœuds voisins du gène en question, afin de déterminer dans chacune des stratégies parentes les processus les plus efficaces. Ici on entend par processus un groupe de gènes fortement connectés.

Tester toutes les combinaisons de gènes pour trouver l'efficacité mutuelle maximale demande un temps très long. Pour garder le gain dû aux interactions et à l'optimisation de processus, le choix des morceaux de génomes à croiser doit se faire non seulement sur le gène qui a l'efficacité maximale, mais aussi sur les gènes qui lui sont proches. Pour cela, le choix des morceaux de génomes à croiser se fait selon l'algorithme suivant :

- 1) Tirage d'un gène aléatoirement suivant une distribution en fonction de l'efficacité locale chez le premier parent.
- 2) Récupération des gènes proches dans le clustering (le nombre de gènes dans le voisinage est fonction de la distance ou d'une taille de voisinage).
- 3) Retrait des gènes choisis chez les parents.

- 4) Choix d'un gène chez le deuxième parent selon l'efficacité.
- 5) Retour au 1 jusqu'à extinction du pool de gènes.

2.2.4. La sélection

La sélection des stratégies en vue de la reproduction se fait par tirage aléatoire selon les distributions données par l'efficacité; plus l'efficacité d'une solution est grande, plus la chance de la tirer est forte. Une fois une solution tirée, il n'est pas possible de la retirer pour le partenaire, ce qui évite l'auto-reproduction.

3. Résultats et comparatifs sur les données simulées

3.1. Calcul des résultats

COGARE a été testé sur des échantillons de données créées afin de comparer les résultats de COGARE avec les résultats des logiciels concurrents, NIR, Aracne et Banjo. 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).

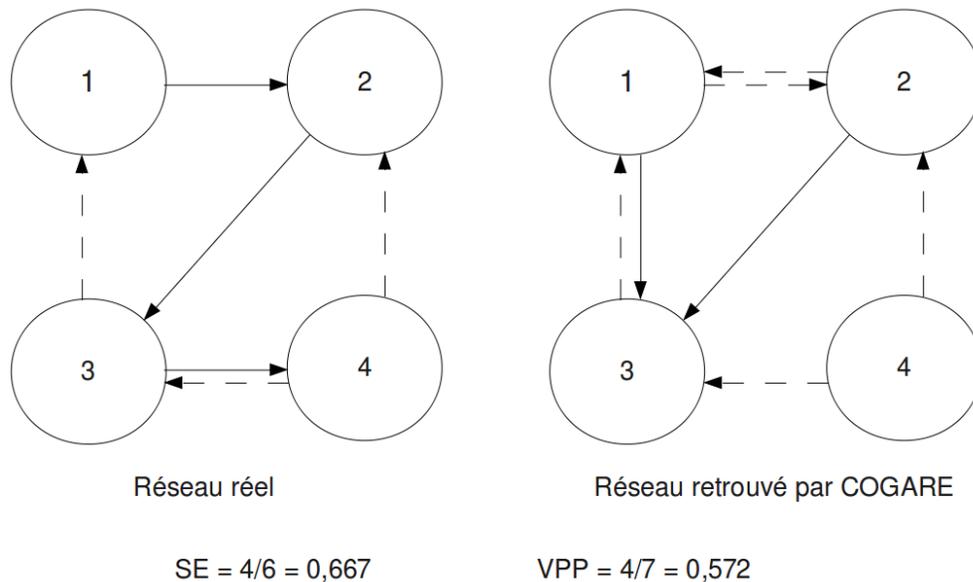


Figure 2. Calcul de VPP et de SE pour un cas test

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 2). La VPP permet de voir si l'algorithme n'oublie pas des relations. Elle permet de visualiser sa précision par rapport aux réseaux réels. La SE permet, elle, d'apprécier la fiabilité relative de l'algorithme.

3.2. La création des données

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. 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. Ensuite, on initialise l'état du réseau, au hasard, et on fait tourner le module de simulation plusieurs fois, pour obtenir des données de départ. Les entrées statiques sont créées directement avec ces données. Les entrées dynamiques sont créées en prenant en temps initial l'état courant, puis en faisant tourner le module de simulation sur ces données de départ, pour obtenir les états aux temps suivants.

3.3. Contrôle de robustesse

Un algorithme sur des données expérimentales doit pouvoir réagir aux spécificités du problème recherché. Ici, cette spécificité tient à la grande variabilité des données résultant d'une même expérience. En effet, chaque variation infime des conditions expérimentales peut entraîner de grandes différences dans les réponses du réseau. Les données obtenues peuvent donc être très fortement bruitées. Pour vérifier que le logiciel fonctionne correctement sur données bruitées, nous avons testé le logiciel sur des données classiques, puis modifié les données aléatoirement, le bruit étant de plus en plus fort, pour modéliser les changements dans les données expérimentales :

$$nvdon = ancdon + amp * alea * bruit \quad [9]$$

où $nvdon$ est la nouvelle valeur des données, $ancdon$ est l'ancienne valeur des données, $amp = \max(donnes) - \min(donnes)$ est l'amplitude de bruit généré, $alea$ 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, les résultats des données bruitées ($nvdon$) ont été comparés avec les données normales ($ancdon$). La figure 3 nous montre le pourcentage de ressemblance des stratégies calculées sur les données bruitées à celles calculées sur les 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 au nombre de liaisons total. Cette robustesse est relativement bonne car à 30 % de bruit, la

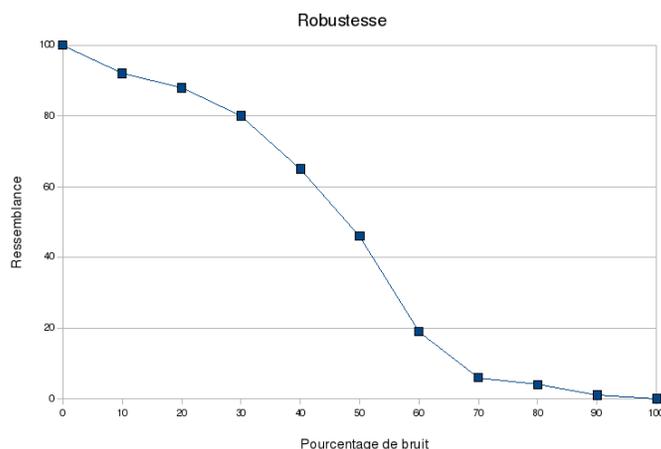


Figure 3. *Robustesse de l'algorithme*

ressemblance est encore de 0,75. Par contre, à partir d'un pourcentage de bruit trop important, ici 35 %, la ressemblance chute rapidement. COGARE permet donc de prendre en compte une des spécificités importantes de la reconstruction de réseaux génétiques.

3.4. Les résultats

COGARE a été comparé avec d'autres algorithmes comme cela a été fait dans (Bansal *et al.*, 2007). Les programmes Banjo, NIR et Aracne 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 différents types de données et tailles de réseaux.

Dans les tableaux 1 2 et 3, la taille (N) des réseaux testés va de 10 éléments jusqu'à 1 000 éléments et le nombre de données (M) varie lui aussi. 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 en gras dans les tableaux correspondent aux cas où COGARE obtient la meilleure performance. Les résultats en italique correspondent aux cas où COGARE obtient des performances supérieures à 90 % des performances des meilleurs algorithmes.

3.4.1. Résultats sur données statiques

Le tableau 1 montre les résultats des algorithmes pour des données statiques. COGARE ne donne pas de résultats très intéressants sur les données statiques. On peut supposer que la dimension locale de l'AG vient perturber les performances de COGARE sur les petits réseaux. C'est par ailleurs un défaut connu des AG sur les petits espaces, pour lesquels des algorithmes plus efficaces sont disponibles. 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.

N*M	Aracne		Banjo		NIR		COGARE		
	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	
			<i>d</i>	0.25	0.17	0.18	0.45	0.11	0.08
			<i>s</i>	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	
			<i>d</i>	0.79	0.05	0.20	0.46	0.12	0.11
			<i>s</i>	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	
			<i>d</i>	0.10	0.02	0.10	0.05	0.01	0.00
			<i>s</i>	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	
			<i>d</i>	0.47	0.00	0.10	0.10	0.06	0.09
			<i>s</i>	0.71	0.00	0.05	0.05	0.04	0.04
100*1 000 ^u	0.19	0.26	0.99	0.05	0.20	0.19	0.13	0.12	
			<i>d</i>	0.68	0.03	0.10	0.09	0.09	0.09
			<i>s</i>	0.68	0.03	0.05	0.05	0.04	0.05
1 000*	0.02	0.10	-	-	-	-	0.02	0.04	
1 000 ^u									

Tableau 1. Résultats sur données statiques

3.4.2. Résultats sur données dynamiques

Le tableau 2 montre les résultats obtenus pour des expérimentations sur des données dynamiques (NIR n'utilisant pas les données dynamiques, il a été retiré du tableau). Pour les données dynamiques, les résultats de 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. 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 1 000. Les résultats sur des données dynamiques seules permettent de le mettre en compétition directe avec Banjo. Par exemple sur les réseaux

Données N*M	Aracne		Banjo		COGARE	
	VPP	SE	VPP	SE	VPP	SE
10*10 ^u	0.00	0.39	0.36	0.35	0.25	0.30
<i>d</i>			0.22	0.21	0.20	0.20
<i>s</i>			0.00	0.00	0.10	0.11
10*100 ^u	0.35	0.43	0.36	0.29	0.31	0.22
<i>d</i>			0.21	0.16	0.28	0.20
<i>s</i>			0.25	0.00	0.20	0.11
100*10 ^u	0.19	0.10	0.18	0.08	0.10	0.05
<i>d</i>			0.10	0.04	0.05	0.01
<i>s</i>			0.06	0.00	0	0
100*100 ^u	0.19	0.15	0.19	0.05	0.15	0.15
<i>d</i>			0.10	0.02	0.12	0.05
<i>s</i>			0.04	0.00	0.09	0.07
100*1 000 ^u	0.19	0.19	0.19	0.04	0.18	0.18
<i>d</i>			0.10	0.02	0.12	0.08
<i>s</i>			0.05	0.00	0.06	0.02
1 000*1 000 ^u	0.02	0.10	-	-	0.02	0.10

Tableau 2. Résultats sur données dynamiques

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. De plus, sur les 30 tests en commun pour les données dynamiques, COGARE obtient 15 fois une meilleure efficacité, Banjo 13. 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 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 1 000*1 000, avec Aracne.
- 0,10 pour la SE sur des réseaux non orientés de 1 000*1 000, avec Aracne.
- 0,12 pour la VPP sur des réseaux orientés de 100*100, seul.

3.4.3. 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 de meilleurs résultats. Le tableau 3 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 récapitule les temps de calcul pour chaque configuration. La dernière colonne permet de visualiser le meilleur résultat des algorithmes concurrents.

Données	COGARE			Meilleur résultat			
	VPP	SE	Hrs	VPP	Algo	SE	Algo
T*(D+S)							
10*(5+5) (^u)	0.37	0.35	0.5	0.41	Banjo	0.71	NIR
<i>d</i>	0.20	0.19	0.5	0.25	Banjo	0.45	NIR
<i>s</i>	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
<i>d</i>	0.43	0.21	2.5	0.79	Banjo	0.46	NIR
<i>s</i>	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
<i>d</i>	0.54	0.51	1.5	0.79	Banjo	0.46	NIR
<i>s</i>	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
<i>d</i>	0.10	0.08	3	0.79	Banjo	0.46	NIR
<i>s</i>	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
<i>d</i>	0.27	0.17	5.5	0.47	Banjo	0.10	NIR
<i>s</i>	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
<i>d</i>	0.33	0.27	5	0.47	Banjo	0.10	NIR
<i>s</i>	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
<i>d</i>	0.46	0.34	14	0.68	Banjo	0.09	NIR
<i>s</i>	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
<i>d</i>	0.38	0.29	17	0.68	Banjo	0.09	NIR
<i>s</i>	0.27	0.19	17	0.68	Banjo	0.05	NIR
1 000*(900+100)(^u)	0.15	0.15	24	0.02	Aracne	0.10	Aracne
<i>d</i>	0.10	0.12	24	-	-	-	-
<i>s</i>	0.06	0.03	24	-	-	-	-

Tableau 3. Résultats sur données mixtes

Le tableau 3 nous montre une augmentation très importante de la précision des résultats, qui permet de rivaliser avec les résultats des meilleurs algorithmes. De plus, la quatrième colonne du tableau montre le temps de calcul moyen de l'algorithme en fonction du nombre de données utilisées.

La moyenne des meilleures efficacités pour l'ensemble des techniques est de 0,43 alors que pour COGARE, elle est de 0,39. De plus, COGARE n'a pas souvent les

meilleurs résultats. Par contre, 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. Il est intéressant de noter que pour un ratio de 10 entre le nombre de données et la taille du réseau, COGARE arrive à reconstruire le réseau à plus de 90 %.

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. Les VPP et SE moyennes pour chaque algorithme sont montrées au tableau 4.

	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

Tableau 4. VPP et SE moyennes des différents algorithmes

4. Résultats sur données réelles

4.1. Le processus de SOS chez *Escherichia Coli*

Escherichia coli est une bactérie intestinale présente chez les mammifères et très commune chez l'être humain. Nous avons appliqué COGARE à un processus de coli nommé le SOS (Morel *et al.*, 1998). Ce processus biologique permet à coli d'envoyer un message d'alerte et d'erreur lorsque l'ADN est endommagé à cause de génotoxines par exemple. Ce processus comporte plus de 100 gènes et COGARE a été appliqué à un ensemble de neuf gènes au cœur du réseau sur lesquels les résultats étaient connus et les données facilement accessibles. Les données pour faire les simulations ont été récupérées sur GENBANK (NCBI, 2000) et dans les travaux de Gardner (Gardner *et al.*, 2005). Le nombre d'expériences disponibles sur lesquelles nous avons travaillé a été de 9 expériences statiques et 6 dynamiques. Le réseau initial connu comporte 25 relations, montrées à la figure 4. De ces relations connues, 5 relations ont été tirées aléatoirement et forment les données préalablement connues de fiabilité 0.6. Les résultats peuvent être visualisés à la figure 5.

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 :

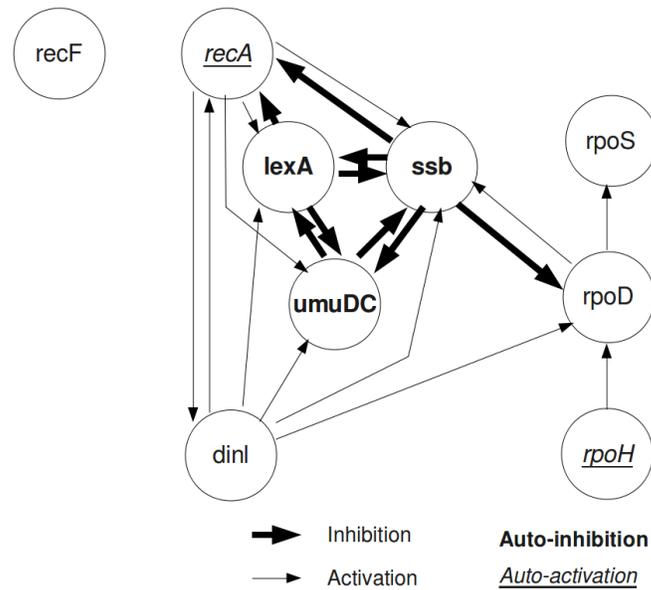


Figure 4. *Graphe du processus de SOS de Escherichia coli*

- $\text{recF} \Rightarrow \text{recA}$
- $\text{recF} \rightarrow \text{recF}$
- $\text{rpoH} \rightarrow \text{dinI}$
- $\text{rpoS} \Rightarrow \text{rpoD}$

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.

Pour visualiser la différence de résultats lorsque l'on exécute COGARE avec différents types de variables, nous avons aussi testé le réseau sous trois autres configurations de données :

- seulement les données statiques (au nombre de 9), numéro 1 ;
- seulement les données dynamiques (au nombre de 6), numéro 2 ;
- données statiques et dynamiques (sans les complémentaires), numéro 3.

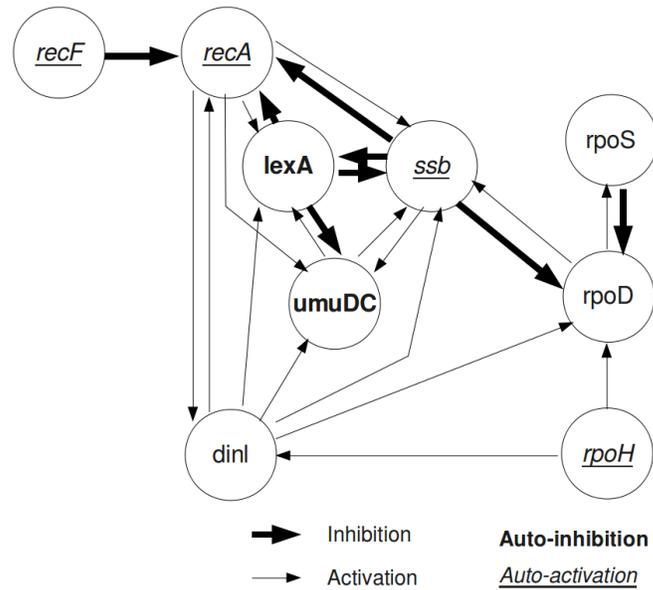


Figure 5. *Graph du processus de SOS de Escherichia coli retrouvé par COGARE*

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

Tableau 5. *Résultats de COGARE sur le processus de SOS d'Escherichia Coli*

Les résultats de ces expériences sont résumés dans le tableau 5, l'expérience 0 est l'expérience de base avec tous les types de données. IR est le nombre d'interactions correctement 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.

4.2. Données sur la biolixiviation

4.2.1. La biolixiviation

La biolixiviation (Lizama *et al.*, 2003) est une technique minière qui emploie des bactéries pour extraire des métaux d'un minerai. L'extraction se fait en séparant une substance soluble d'une structure solide en l'incorporant à une préparation liquide qui facilite son extraction. La biolixiviation utilise des bactéries présentes naturellement dans les mines pour séparer les substances, ce qui diminue drastiquement l'impact environnemental. La biolixiviation est appliquée à des tas de minerai, broyé, dont le résultat est traité dans des installations irriguées. Les amas de poudre sont traités avec un liquide acide renfermant une population bactérienne. Les bactéries se chargent alors de séparer le métal cible du reste. Le liquide et le métal extrait sont ensuite pompés et centrifugés pour récupérer le métal, du cuivre. L'étude du génome des bactéries 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, ou encore la détermination de conditions physico-chimiques optimales pour l'extraction.

4.2.2. L'exemple traité

Les données suivantes ont été récupérées sur des expériences faites sur *Thiobacillus ferrooxidans*, une bactérie qui peut lixivier le cuivre (Lennox *et al.*, 1991). Elles ont été créées en déposant sur une puce à ADN, des brins d'ADN et en soumettant cette puce à plusieurs stimuli, environnement de chalcopyrite ($CuFeS_2$), 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.

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 au National Center for Biotechnology Information (NCBI, 2000). 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, ce qui a permis de tester la cohérence des résultats. Un exemple de régulateur est le gène de l'opéron lactose qui code pour une protéine qui peut se fixer sur l'ADN, empêchant la transcription des gènes en protéines.

4.2.3. Les résultats

Le réseau, d'après la littérature, comporte une centaine de gènes régulateurs identifiés. COGARE a retrouvé en tout 113 gènes régulateurs dans le réseau. Sur les 113

régulateurs retrouvés, 85 faisaient partie des 100 gènes régulateurs connus. Des 28 gènes régulateurs “nouveaux”, nous en avons identifié certains dont la présence parmi les régulateurs peut s’expliquer.

– Un gène que nous savons être impliqué dans la production d’ATPase est présent sur cette liste. L’ATP 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.

– Deux autres gènes codants 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 manière.

L’absence de ces gènes sur la liste de régulateurs connus s’explique par le fait que l’action des protéines issues de ces gènes peut s’effectuer sur des gènes qui ne sont pas présents dans le réseau recherché. De plus, leur action 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 des actions indirectes.

D’autres gènes présents sur cette liste apparaissent 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 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 pas être détectés. Ces gènes peuvent être considérés comme des faux-positifs, c’est à dire des erreurs de l’algorithme.

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 se décompose en trois grandes parties (clusters de premier plan), dont une complètement isolée du reste des autres. Les deux autres parties forment des parties fortement connectées en interne et peu en externe. Il existe aussi des gènes que COGARE n’a connectés à aucun autre. Ces gènes, au nombre de 8, peuvent être considérés soit comme 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.

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 uns avec les autres. Les trois gènes codent pour des protéines de fusion membranaire, de division cellulaire et de déshydratation du glucose.

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 220. 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).

5. Conclusion et perspectives

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 aurons accès.

Par ailleurs, 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 faire de la rétro-ingénierie sur des problèmes qui ont plusieurs niveaux d'analyse. Par exemple, 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 comment l'organisme réagit à son environnement et comment il évolue dans certaines circonstances. 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.

COGARE a une marge de progression importante. En effet, le logiciel n'est pour l'instant pas capable de quantifier les relations retrouvées. Ces relations sont des réactions chimiques et donc ont une vitesse de réaction qui varie en fonction des éléments en jeu dans les réactions. Quantifier ces relations permettrait de faire des simulations du comportement des organismes beaucoup plus précises. De plus, COGARE utilise trois types de données pour fonctionner, mais d'autres types de données sont disponibles et apportent aussi des informations intéressantes, par exemple les données

mutantes qui permettent d'obtenir des informations sur un gène qui est muté afin de visualiser les différences entre l'organisme avec le gène normal et l'organisme avec le gène muté.

6. Bibliographie

- Bansal M., Belcastro V., Ambesi-Impiombato A., di Bernardo D., « How to infer gene networks expression profiles. », *Molecular System Biology*, 2007.
- Berge C., *The theory of Graph*, Dover, 2003.
- Briche J., Adaptation d'un algorithme génétique pour la reconstruction de réseaux de régulation génétique : COGARE., PhD thesis, Université du Sud Toulon var, 2009.
- Cerf R., « An asymptotic theory for genetic algorithms. », *Lectures Notes in Computer Science*, 1996.
- Gardner T., di Bernardo D., Lorenz D., Collins J., « Inferring genetic networks and identifying compound mode of action via expression profiling. », *Science 301 : 102-105*, 2003.
- Gardner T., Faith J., « Reverse-engineering transcription control networks. », *Physics of Life Reviews 2 65-88.*, 2005.
- Holland J., *Adaptation in Natural and Artificial Systems*, University of Michigan Press, 1975.
- Huber W., von Heydebreck A., Vingron M., *Handbook of statistical genetics*, Wiley and sons, 2004.
- Kauffman S., « Metabolic Stability and epigenesis in randomly constructed nets. », *Journal of Theoretical Biology*, 1969.
- Lennox J., Biaha T., « Leaching of Copper Ore by Thiobacillus Ferrooxidans. », *American Biology Teacher*, v53 n6 p361-68., 1991.
- Lizama H., Fairweather M., Dai Z., Allegretto T., « How does bioleaching start ? », *Hydrometallurgy*, 2003.
- Margolin A., Nemenman I., Basso K., Wiggins C., Stolovitzky G., Favera R. D., Califano A., « Aracne : an algorithm for the reconstruction of gene regulatory networks in a mammalian cellular context. », *BMC Bioinformatics 51*, 2006.
- Morel P., Reverdy C., Michel B., Ehrlich S., Cassuto E., « The role of SOS and flap processing in microsatellite instability in Escherichia coli. », *Proc. Natl. Acad. Sci. USA 95 :10003-10008.*, 1998.
- NCBI, « GENBANK », <http://www.ncbi.nlm.nih.gov/Genbank/>, 2000.
- Pearl J., « Bayesian Networks : A Model of Self-Activated Memory for Evidential Reasoning. », *Proceedings of the 7th Conference of the Cognitive Science Society, University of California, Irvine, CA, pp. 329-334, August 15-17.*, 1985.
- Schwefel H., *Numerical optimization of computer models*, Birkhuser, 1977.
- Wagner A., « How to reconstruct a large genetic network from n gene perturbations in fewer than n^2 easy steps. », *Bioinformatics*, 2001.
- Yu J., Smith V., Wang P., Hartemink A., Jarvis E., « Advances to bayesian network inference for generating causal networks from observational biological data. », *Bioinformatics 20 : 3594-3603*, 2004.