EGNER
Simulation d'un réseau de régulation de gènes

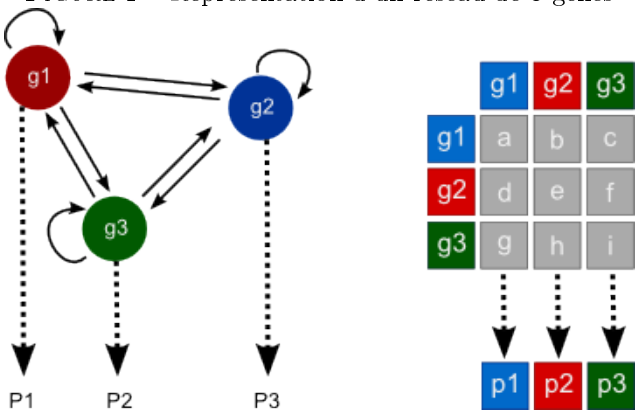
Sacha Schutz

23 mars 2015

1 Introduction

Cette étude porte sur la simulation d'un réseaux de régulation de gènes reprenant le modèle introduit par A. Wagner. Il s'agit de représenter l'Evolution d'un population possédant N gène ayant tous une activité promotrice sur ces même N gènes. L'expression de ce réseau génétique, est défini par un phénotype viable ou non, subissant une pression de sélection au cours du temps. Afin de réaliser ces simulations, le logiciel Egner sous licence GPL3, a été crée permettant de définir plusieurs paramètres de simulation et d'observer des phénomènes émergents au modèle.

FIGURE 1 – Représentation d'un réseau de 3 gènes



2 Matériels et Méthodes

La modélisation de ce réseau génétique est illustré par la figure 1. L'activité répresseur ou activateur des gènes est défini par une matrice de taille N x N ou N est le nombre de gène du réseau. Le phénotype, est quand à lui défini par un

vecteur de taille N.
La simulation a été écrite entièrement en C++11 afin de bénéficier de la programmation orienté objet et de profiter d'une vitesse de calcul convenable. Le framework Qt [1] a quand à lui permis la réalisation d'une interface graphique permettant d'interagir et de faire des observations sur la population. L'ensemble du code source est disponible sur *https ://github.com/dridk/Egner*

2.1 Architecture

Le code source est divisé en 2 partie. Une partie appelé **Core** , contenant la logique pure et les classes de base nécessaires à la simulation. L'autre partie, appelé **Gui** contient les différents composants de l'interface graphique.

2.1.1 Core

Le réseau de gènes a été modelisé sous forme d'une classe appelé *GenotypeNetwork* contenant une matrice de dimension NxN. Cette classe possède des méthodes d'écriture et de lecture des données matricielles ainsi que d'autre méthodes pour générer des mutations aléatoires et des knockout de gènes. Le Phénotype est représenté par la classe *Phenotype*, contenant un vecteur de dimension N et ses méthodes d'accès. La surcharge d'opérateur, permise grâce au C++, a été appliqué a chacune de ces classes afin d'avoir des règles combinatoires. La code du listing 1 montre comment combiner 2 génotype avec éventuellement un crossing over. Celui du listing 2, comment effectuer le produit matricielle entre une génotype et un phénotype

Listing 1 – Addition de genotype

```
GenotypeNetwork mother("0,0,2,4,1,7,5,0,4");
GenotypeNetwork father("0,1,0,3,1,0,5,4,4");
GenotypeNetwork child = mother + father
//donnera par exemple : 0,0,2,3,1,0,5,4,4
```

Listing 2 – produit d'un genotype et d'un phenotype

```
GenotypeNetwork mother("0,0,2,4,1,7,5,0,4");
Phenotype initial("1,1,1");
initial = initial * mother;
//donnera : -1,-1,1
```

Tous les individus défini par un *GenotypeNetwork* peuvent être inséré dans un classe *Population*. Cette classe contient les méthodes d'ajouts et suppression des individus. Mais ce sont les méthodes *init(size, mean, sd, geneCount)* et *int next()* qui reste les plus importante. La première permet l'initiation d'une population aléatoire de taille *size*, où chaque individu contient un nombre de gènes

1. Pronnoncé Cute

défini par *geneCount*, dont les coefficients suivent une loi normale de moyenne *mean* et d'écart type *sd*. Le calcul de cette loi normale a été permise par l'utilisation du C++11 et de sa nouvelle librarie standard .
La methdode *int next()* permet de basculer de l'ancienne à la nouvelle population et retourne le nombre d'individu non viable généré. Ce basculement d'une population à l'autre a été optimisé par la méthode *swap(std : :vector)* de la std. Voir l'example du listing 3

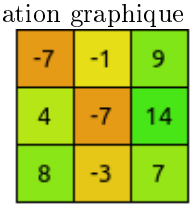
Listing 3 – Generation d'une population et basculement vers la génération suivante

```
Population pop;
pop.init(300, mean=0, sd=1, geneCount=3);
int deadNumber = pop.next();
```

2.1.2 Gui

L'interface graphique a été réalisé grâce au Framework Qt [2]. Disponible sous double licence LGPL/Commercial. Qt permet la création d'interface graphique de grande qualité et s'exécute sous toutes les platformes (windows, mac,linux, mobile) . Les différents graphiques ont été réalisé avec la librairie QCustomPlot [3] également sous licence GPL.
Chaque *GenotypeNetwork* a été représenté visuellement sous forme d'une matrice (figure 2) ou chaque coefficient est coloré suivant un gradient rouge vert. Ces deux couleurs correspondent respectivement à la valeur minimal et maximal des coefficients sur l'ensemble de la population.

FIGURE 2 – Représentation graphique d'un GenotypeNetwork



2. http ://www.qt.io
3. http ://www.qcustomplot.com

L' interface comprend 2 vues de la population courante sur le panneau central et plusieurs outils sur le panneau de gauche pour manipuler la population. La figure 9 résume toutes les actions possible du logiciel.

2.1.3 La Vue Grid

Cette vue affiche la population courante en listant l'ensemble des *GenotypeNetwork* par leurs représentations graphiques. C'est une vision idéal pour observer les différentes valeurs des coefficients dans leurs globalités.

2.1.4 La vue Stat

Cette vue représente les fréquences des coefficients pour chaque gène. Ainsi, si la population est constitué d'individu avec 3 gènes, il y a y aura 9 coefficients, soit 9 histogrammes. Cette représentation est intéressante, pour visualiser la répartition des coefficient dans la population.

2.1.5 L'outil Initialisation

Permet de générer une population initiale en spécifiant les propriétés de la loi normale ainsi que le nombre de gène par individus.

2.1.6 L'outil Run

Permet de lancer la simulation et possède les actions pour générer les graphique *lineplot* et *histogramplot*. *LinePlot* affiche la courbe de viabilité globale de la population pour chaque itération. tandis que *HistogramPlot* affiche la viabilité global depuis le dernier run.

2.1.7 L'outil Knockout

Permet de désactiver des gènes sur la population courante . Pour chaque individu, les coefficients du gène sélectionné sont mis à zéro.

2.1.8 L'outil mutation

Permet quand à lui de générer des mutations aléatoires sur la population en cours. En fonction d'une probabilité défini, un nombre d'individu subira la mutation de ses coefficients par une addition ou soustraction de la valeur *step*.

3 Résultat

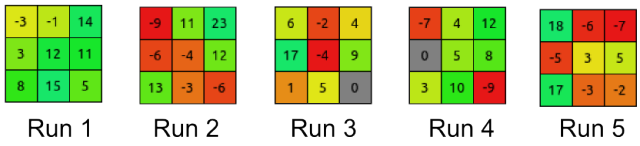
Plusieurs expériences ont été réalisé avec le logiciel Egner. Ci dessous seront décris chaque expérience séparément avec leurs résultats .

3.1 Convergence ou Contingence

3.1.1 Homogénéisation de la population

Lors de plusieurs essaies, a été remarqué, que sans aucune mutation, les populations convergent vers une homogénéisation des génotypes. C'est à dire qu'a partir d'une population initiale variable, on observe au bout de plusieurs itérations, une population de clone avec le même génotype. La question était de savoir si ce dernier clone était le résultat d'une finalité optimale depuis la population initiale, ou le résultat d'une contingence.
Une population initiale aléatoire de 200 individus viables de 3 gènes a été généré, puis sauvegardé dans un fichier. A partir de celui ci, 50 run de 2000 itérations [4] ont été réalisé. Pour chaque run, le clone final a été sauvegardé. Comme vous pouvez le voir sur la figure 3, les clones finaux sont différents et ne convergent pas à priori vers un même génotype.

FIGURE 3 – 5 premiers Génotypes finaux obtenu depuis la même population initiale



A partir des ces 50 clones obtenus, la figure 4 a alors été réalisé représentant la proportion des coefficients des différents génotypes.

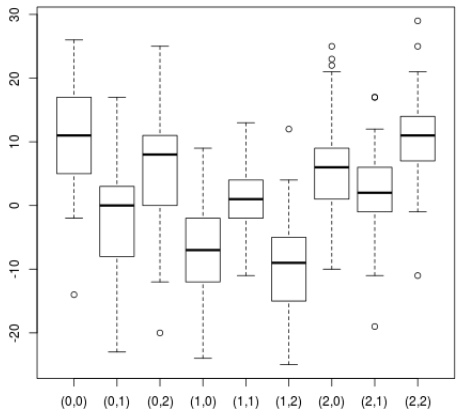
On observe ainsi une grande répartition des coefficients. Par exemple le coefficient en position [0,1] varie de -20 à 20 et celui en position [1,1] varie de -10 à +12. Cependant, les coefficients ne prennent pas toutes les valeurs. Le coefficient en position (2,2) par exemple, ne prend jamais une valeur inférieur à 10. Ceci aurait pu s'expliquer par l'absence de valeur inférieur à 10 dans la population initiale à cette coordonnée précise. Hors, on en retrouve bien dans la population initiale, de plus est qu'elle suit une loi normale.
Il est donc supposable, qu'il existe un certain déterminisme dans l'évolution d'un réseau génétique. Connaissant une population initial, on pourrait prédire des probabilités d'apparition de certains coefficients.

3.1.2 Temps d'homogénéisation de la population

A partir de la première expérience, une question s'est posé. Quels sont les facteurs influençant le vitesse d'homogénéisation de la population. C'est à dire à partir de quelle itération la population devient homogène. 2 facteurs ont été testé. La taille de la population et le nombre de gène. 3 tailles de population initiale ont été testé 100, 200 et 300 avec un nombre de gènes variant de 3 à 5. Chaque combinaison a été testé 10 fois. La figure 5 a alors été réalisé pour

4. Empiriquement, l'homogénéisation se passe avant 2000 itération

FIGURE 4 – Repartition des coefficients des genotypes finaux sur 50 Runs depuis la même population initiale



chaque population. L'ordonnée représente le nombre d'itération pour atteindre la clonalité.

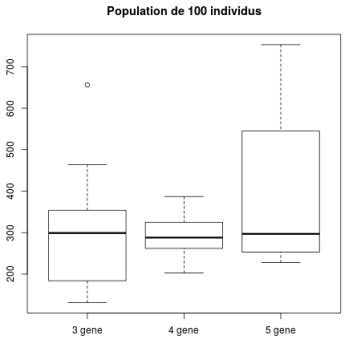
Au vu de ces résultats, il s'avère clairement que le nombre d'individu influence plus fortement le temps d' homogénéisation que le nombre de gène. Dans la population à 300 individus le temps d'homogénéisation tourne autour de 600, alors que pour la population de 100 individus elle est de 300. Ceci reste cohérent avec la simulation. Plus la population est grande, plus les gènes vont mettre un certain temps à se diluer dans l'ensemble de la population.

3.2 Évolution de l'Evolvabilité

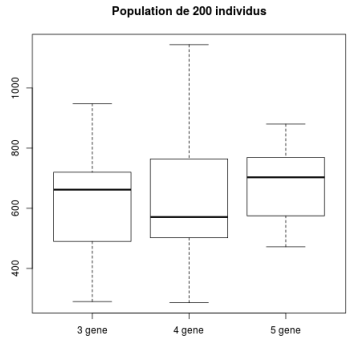
L'Evolvabilité ou adaptabilité est la capacité des organismes vivants à s'adapter. celle ci est en étroite relation avec la variabilité de l'organisme. Plus la variation est grande, et plus une population sera apte à répondre à la sélection naturelle. En contrepartie, cette variation peut entraîner des mutations préjudiciables à l'organisme. On appelle la robustesse, la capacité d'un organisme à résister aux mutations tout en maintenant un niveau de variabilité suffisant en vu d'un stress futur. C'est dans cette optique que les expériences suivantes ont été réalisé sur le réseau génétique.

3.2.1 Mesure de la robustesse

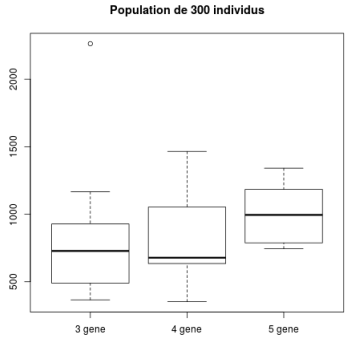
Une population de 300 individus de 5 gènes a été crée. Pour chaque enfant crée, une probabilité de mutation 0.1 a été défini. La simulation a été lancé sur 200 run.
Sur la figure 6 est représenté la proportion d'individu non viable par itération,



(a)



(b)

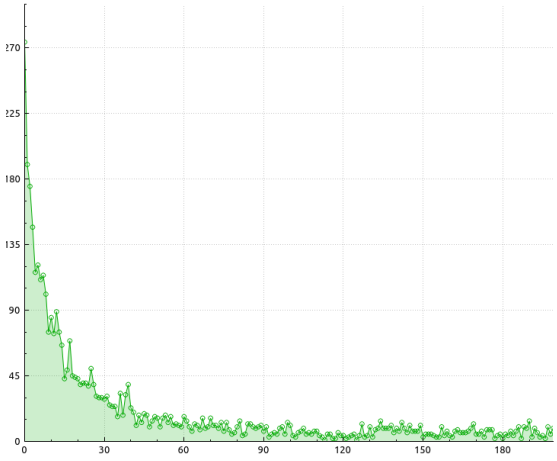


(c)

FIGURE 5 – Temps d'homogénéisation pour 3 populations différentes et 3 gènes différents

assimilable à l'inverse de la robustesse. On observe ainsi qu'au cours des itérations, la population se stabilise est devient plus robuste au mutation.

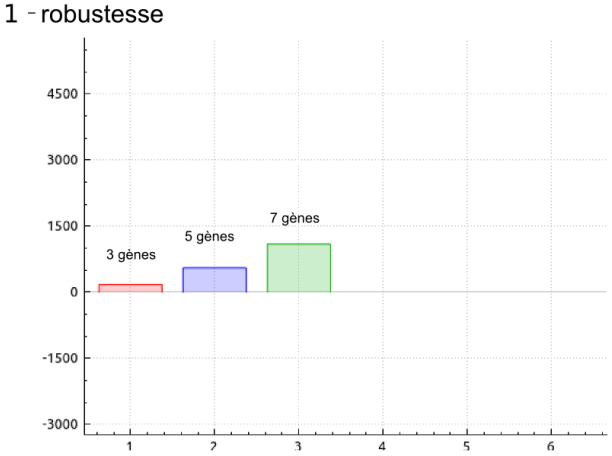
FIGURE 6 – Évolution de la robustesse au cours des itérations



3.2.2 Influence du nombre de gène sur la robustesse

3 population avec 3 , 4 et 5 gènes ont été généré , puis ont évolué vers un état stable avec les mêmes paramètres que précédemment. Ont été ensuite mesuré la robustesse pour chaque population respective comme l'illustre la figure 7

FIGURE 7 – Proportion d'individu non viable provenant d'une population évolué

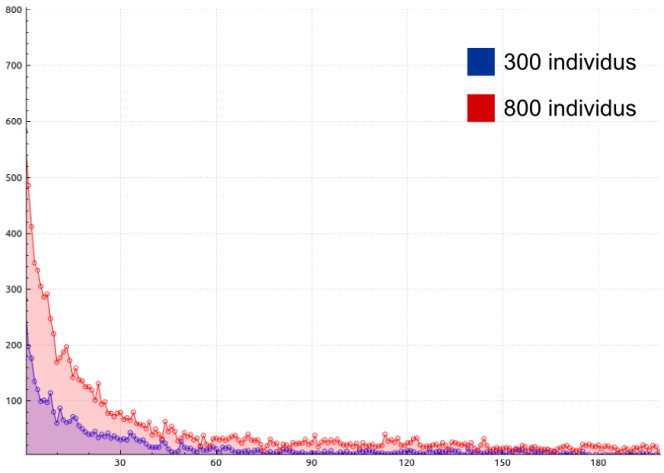


Dans cette figure, plus le nombre de gène est faible, plus le système devient résistant au mutation. D'un point de vu biologique, cette expérience invite à penser un compromis entre la complexité d'un réseau génétique et sa capacité d'adaptation. Un réseau trop complexe présenterait des difficultés à s'adapter.

3.2.3 Influence de la taille de la population

1 population de 300 individu et une autre de 800 ont été généré avec 5 gènes. 200 itérations ont été lancé sur chacune d'elle. Puis à partir de ces deux populations évoluées, la robustesse a été mesuré.
On observe sur la figure 8 que la population de 300 individus attends plus rapidement la stabilité. En revanche, la robustesse diffère de très peu entre les deux populations.

FIGURE 8 – Proportion d'individu non viable provenant d'une population évoluée



3.3 Autre experience non décrite par manque de temps

Adaptation au Knock-out
Les populations deviennent égallement résistente au knockout ?

Duplication de gène
Quel est l'influence des duplications de gènes ?

Influence du taux de mutation
Comment varie la robustesse en fonction du taux de mutation ? ...

4 Conclusion

L'Evolvabilité d'un réseau génétique, repose sur sa capacité à tamponner l'effet délétère des mutations. On observe dans ces simulations, que cette capacité est elle même un caractère visible au regard de la sélection naturelle. Ainsi, un réseau génétique possède la capacité de maintenir un certain taux de mutation, tout en absorbant les dommages des effets délétères.
Les premières expériences nous ont révélé quand elles que la composition d'une

population initiale peut influencer le devenir d'une population. D'autres expériences pourrait peut être nous éclairé sur les mécanismes régulateurs de l'ontogenèse et du principe de canalisation. C'est à dire, comment les réseaux régulateurs se différencient dans les différents tissues alors qu'elle provienne tous de la même cellule œuf.

A Problèmes encourus

J'ai été confronté à un gros problème qui m'a fait perdre beaucoup de temps. L'initialisation de ma population générait des individus viables avec le même phénotype. Mais je précisais que ce phénotype devait être 1,1,1,1,1. De plus j'utilisais des coefficients entiers par soucis de clarté dans mes représentations graphiques. La conséquence était que tous les enfants de parents viables de phénotype 1,1,1,1,1 étaient viables. Il n'y avait donc aucune létalité, la population était toujours stable. J'ai donc changé mes conditions d'initialisation et ajouté une option pour utiliser des nombres à virgule. Cependant, il reste toujours un risque d'erreur. En Générant une population de 3 gènes avec des nombres entiers, il est possible d'avoir une population avec un phénotype 1,1,1 et se retrouver à nouveau confronté au problème.

B Compilation du logiciel

Télécharger de Qt Créator Télécharger la version opensource de Qt creator depuis le site http ://www.qt.io/download/

Récupérer le code source Télécharger le code source depuis github sur la branche master. https ://github.com/dridk/Egner

Compilation Depuis Qt creator, ouvrir simplement le fichier Egner.pro et appuyer sur le gros boutton vert pour compiler et executer.

FIGURE 9 – Interface graphique d'Egner