EGNER
Simulation d'un reseau de régulation de gène

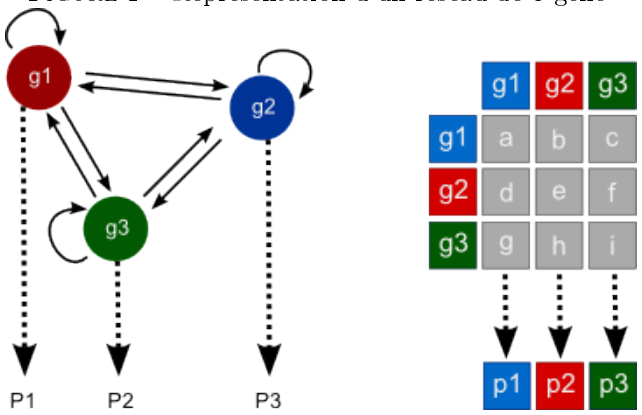
Sacha Schutz

22 mars 2015

1 Introduction

Cette étude porte sur la simulation d'un réseaux de régulation de gène 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 license GPL, a été créer permettant de définir plusieurs paramètre de simuation et d'observer des phénomènes emergents au modèle.

FIGURE 1 – Représentation d'un réseau de 3 gène



2 Materiels et Méthodes

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

vecteur de taille [N].
La simulation a été écris entièrement en C++11 afin de béneficier de la programmation orienté objets 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 observartions 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 differents composants de l'interface graphique.

2.1.1 Core

Le réseau de gène a été modelisé sous forme d'une classe appelé *GenotypeNetwork* contenant une matrice de dimension [NxN]. Cette classe possède des methodes d'écriture et de lecture des donnée matricielles ainsi que d'autre methodes pour générer des mutations aléatoire et des knockout de gène. Le Phenotype est représenté par la classe *Phenotype*, contenant un vecteur de dimension [N] et ses methodes d'acces. La surcharge d'operateur, permise grâce au C++, a été appliqué a chacune de ces classes afin d'avoir des regles combinatoires. La code du listing 1 montre comment combiner 2 genotype avec eventuellement un crossing over. Celui du listing 2, comment effectuer le produit matricielle entre une genotype et un phenotype

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 : 0,0,2,3,1,0,5,8,4
```

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 indivdus défini par un *GenotypeNetwork* sont contenu dans un classe *Population*. Cette classe contient les méthodes d'ajouts et de suppression des individus. Mais ce sont les methodes *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*, oû chaques individus contient un nombre de gène défini

1. Pronnoncé Cute http ://www.qt.io

par *geneCount*, dont les coefficients suivent une loi normal de moyenne *mean* et d'ecart type *sd*. Le calcul de cette loi normal a été permise par l'utilisation du C++11 et de sa nouvelle librarie standard .
La deuxième permet de basculer de l'ancienne à la nouvelle population et retourne le nombre d'individu non viable qui a été crée. Ce basculement d'une popluation à l'autre a été optimisé par la methode 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. Disponible sous double license LGPL, Qt permet la création d'interface graphique de grande qualité et s'execute sous toutes les platformes windows, mac et linux. Les differents graphiques ont été réalisé avec la librairie QCustomPlot [2] egalement sous license GPL.
Chaques GenotypeNetwork a été representé visuellement sous forme d'une matrice (figure 2.1.2) ou chaques coefficient est coloré suivant un gradient rouge vert.Ces deux couleurs representant respectivement la valeur minimal et maximal des coefficient sur l'ensemble de la population.

FIGURE 2 – Representation graphique d'un GenotypeNetwork

2. www.qcustomplot.com

Le code source comprend 2 vues de la population courrante sur le panneau central et plusieurs outils sur le panneau de gauche pour manipuler la population. La figure 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 representations graphiques. C'est une vision idéal pour observer les differentes valeurs des coefficients dans leurs globalité.

2.1.4 La vue Stat

Cette vue represente les frequences des coeffients pour chaques gènes. Ainsi, si la population est constitué d'indivdu avec 3 gènes, il y a yaura 9 coefficient, soit 9 histogramme. Cette representation est interessante, pour visualiser une population généré par une loi normal.

2.1.5 L'outil Initialisation

Permet de générer une population initial en specifiant les propriétés de la loi normal 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 histogram plot. LinePlot affiuche la courbe de viabilité global de la population pour chaques itération. tandis que Histogram Plot affiche la viabilité global depuis le dernier run.

2.1.7 L'outil Knockout

Permet de desactiver des gènes sur la population courante . Pour chaques individus, les coefficients du gènes selectionné sont mis à zero.

2.1.8 L'outil mutation

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

3 Résultat

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

3.1 Convergence ou Contingence

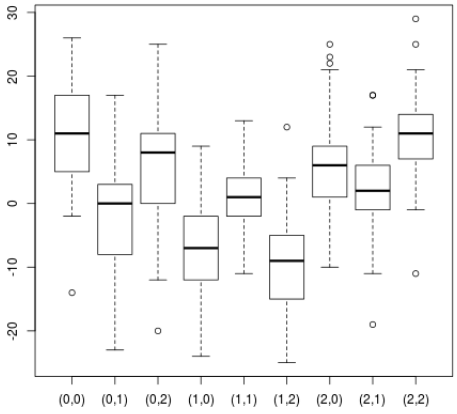
3.1.1 Homogénisation de la population

Lors de plusieurs essaie, 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 initial variable, on observe au bout de plusieurs itération, une population de clone avec le même génotype. La question était de savoir si ce dernier clone était le résultat d'une finalité optimal depuis la population initial, ou le résultat d'une contingence.
Une population initial aléatoire de 200 individus viable de 3 gène a été généré, puis sauvegarder dans un fichier xp1reference.txt. A partir de celui ci, plusieurs run de 2000 itération [3] ont été réalisé. Pour chaque run, le clone final a été sauvegardé. Comme vous pouvez le voir sur la figure 3.1.1, les clones finaux sont différents et ne convergent pas à priori vers un même génotype.

FIGURE 3 – Genotypes finaux à partir de la même population initial

A partir des ces 50 clones obtenus, un box plot 3.1.1 a alors été réalisé représentant la proportion des coefficients des différents génotype.

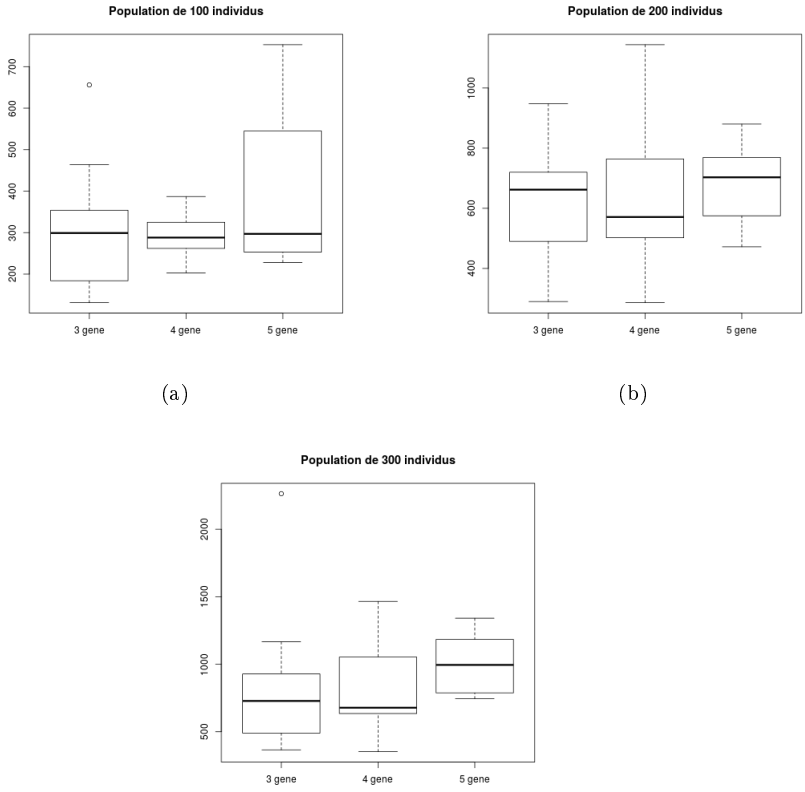
FIGURE 4 – Repartition des coefficients des genotypes finaux sur 50 Runs depuis la même population initiale



3. Empiriquement, l'homogénisation se passe avant 2000 itération

3.1.2 Temps d'homogénisation de la population

A partir de la première expérience, une question s'est posé. Quels sont les facteurs influençant le vitesse d'homogenisation de la population. C'est à dire à partir de quel itération la population devient homogène. 2 facteurs ont été testé. La taille de la population et le nombre de gène. 3 tailles de population initial ont été testé 100, 200 et 300 avec un nombre de gène variant de 3 à 5. Chaque combinaison a été testé 10 fois. Un boxplot 3.1.2 a alors été réalisé pour chaque population. L'ordonnée représente le nombre d'iteration pour atteindre la clonalité.



(a) (b)

(c)

FIGURE 5 – Add your own figures before compiling

Au vu de ces résultats, il s'avère clairement que le nombre d'individu influence plus fortement le temps d' homogénisation que le nombre de gène. Ceci reste cohérent avec la simulation. Plus la population est grande, plus les gènes vont mettre un certain temps à se dilluer dans l'ensemble de la population.

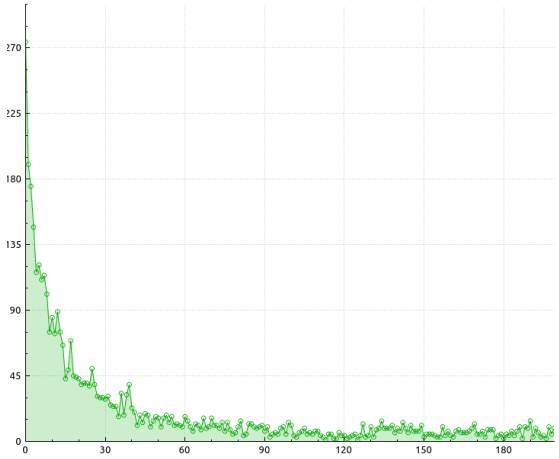
3.2 Evolution de l'Evolvabilité

L'Evolvabilité ou adaptabilité est la capacité des organismes vivant 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 préjudiciable 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 suivante ont été réalisé.

3.3 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 figure 3.3 est represente la proportion d'individu non viable par iteration, assimulable à l'inverse de la robustesse. On observe ainsi qu'au cours des iterations, la population se stabilise est devient plus robuste au mutation.

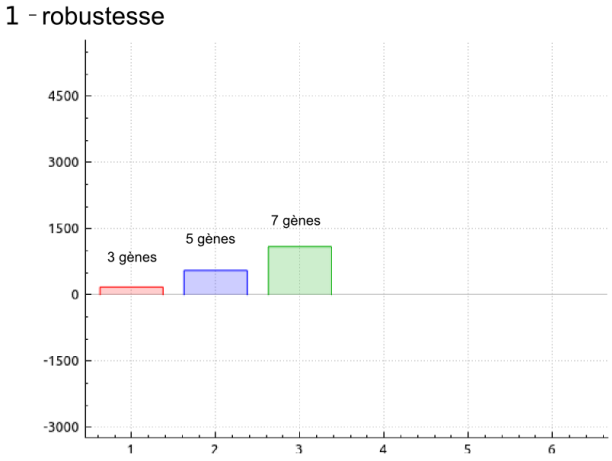
FIGURE 6 – Evolution de la robustesse au cours des iterations



3.4 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 même paramètres que précédemment. Ont été ensuite mesuré la robustesse pour chaques populations respective comme l'illustre la figure 3.4

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 experience invite à pensé à un compromis entre la complexité d'un réseau génétique et sa capacité d'adaptation. Un réseau trop complexe présenterait des difficulté à s'adapter.

3.5 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 évalué, la robustesse a été mesuré.
On observe sur la 3.5 que la population de 300 individus atteinds plus rapidement la stabilité. En revanche, la robustesse diffère de très peu entre les deux populations.

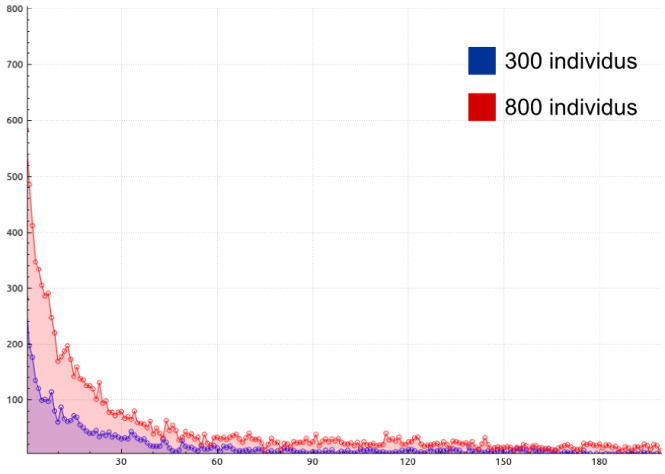
3.6 Autre experience non décrite par manque de temps

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

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

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

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



4 Conclusion