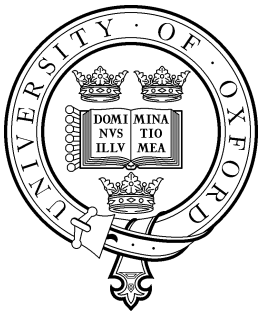
Programming Research Group

GREENSIM: A GENETIC REGULATORY NETWORK SIMULATOR

Christopher Fogelberg and Vasile Palade[1]

PRG-RR-08-07



Oxford University Computing Laboratory
Wolfson Building, Parks Road, Oxford OX1 3QD

[1]Email: christopher.fogelberg@comlab.ox.ac.uk

Abstract

Inference of complete genetic regulatory networks is a central problem in modern bioinformatics. However, because good biological data is still relatively rare, it is hard to evaluate new machine learning techniques for network inference. In this report we describe `GreenSim`, a modular, customisable and extensible genetic regulatory network simulator. It accurately models motifs, non-linear regulatory functions and can generate networks ranging in size from $N = 10^0$ to $N = 10^4$ genes. Code is available online and from the authors.

1 Introduction

Accurate inference of an organism's complete *genetic regulatory network* (GRN) is a central problem in bioinformatics. Knowledge of the GRN is necessary to understand an organism's ontogeny, its phylogenetic relationships and to accurately predict its response to external stimuli and drugs. Only when such relationships are understood can research be conducted quickly and efficiently.

Despite the development and increasing use of high-throughput biotechnology, well understood and validated biological data sets are rare and exist for only a few species. This means that realistically simulating biological data is essential to fully evaluate new machine learning and bioinformatic techniques.

In this paper, we describe `GreenSim`, the genetic regulatory network simulator. In the rest of this section (subsection 1.1) we summarise the features of GRN that must be simulated. Section 2 describes `GeneSim`, and section 3 describes Kyoda's simulator. Section 4 outlines the characteristics and modular structure of `GreenSim`, its usage, and analyses some networks generated by it. Section 5 concludes and outlines future work.

1.1 Characteristics of Genetic Regulatory Networks

The causal relationships between the genes in an organism make up its genetic regulatory network. This network is a directed graph, and an edge from the gene i to the gene j and annotated with a function means that the expression level of i regulates (contributes to, causes) the expression level of j.

Because of their evolutionary origins[16], GRN are characterised by certain common features across cell type and species. This section summarises the most important of these features, and a good GRN simulator should implicitly or explicitly create them.

At the structural level it is important to understand the different distributions across the in-degree and out-degree of genes (variables, nodes) in GRN. Research[2] suggests that GRN are not random directed graphs. Instead, the probability distribution for the out-degree appears to follow a power law[8] with $2 < \lambda < 3$. The distribution for the in-degree appears to be an exponential distribution, and μ is approximately 3 for prokaryotes and in the range 4–8 for eukaryotes[4; 11]. These distributions help create a modular structure in the network. There is no characteristic module size.

GRN are also characterised by the overabundance of *motifs*. A motif is a small subgraph which is much more common in GRN than it would be in a random graph. Motifs

include *auto-regulatory*, expanding *cascade*, *convergence* and triangular *feed-forward* motifs. In a feed-forward motif the gene i regulates j and k, and j regulates k as well. Motifs are detailed in [6]; it is important that they are appropriately present in any simulated GRN.

Two key aspects of the regulatory relationship between any pair of connected genes must be simulated. The first is that genetic regulatory functions are often complex, non-linear relationships whose functional form varies significantly from case to case. The gene i may up-regulate (down-regulate) the gene j very strongly (very weakly) and it may regulated it non-linearly. The nature of these relationships is described in more detail in [3; 5; 17].

Regulatory functions can also be conditional on the current phenotypic context[13]. Furthermore, the actual biological system is stochastic and discrete; however, a discrete simulation is not computationally tractable. Nor is a discrete model. This means that stochastic *biological noise*[12] will exist in any continuous representation of a GRN.

External phenotypic contexts excluded and assuming a weak central dogma, note that non-genetic regulatory influences can be ignored in GRN inference. This is because, under the weak central dogma, the GRN is assumed to be informationally complete. Non-genetic regulatory factors just manifest themselves as (more complex) genetic regulatory relationships.

2 GeneSim

`GeneSim` was developed at Duke University as part of a project which tried to understand songbird singing behaviour[7; 14; 15; 18; 19]. Used in most of the GRN research there, [18] and [19] describe `GeneSim` in detail.

It works as follows. A network of small size ($N \leq 100$) is generated and connected in a uniformly random way. Connections can also be manually specified. Time is measured in discrete intervals and every δ time steps the expression level is recorded as a sample. This interval represents the (hidden) changes in expression level that occur in between successive samples in a time series microarray data set.

$$y_{t+1} = y'_t + A \times (y'_t - b') + \epsilon' \tag{1}$$

Equation 1 defines the vector equation which calculates the expression level of all genes (capped at a maximum of 100 and a minimum of 0) at time $t+1$, given the expression level of all genes at time t. Column vectors are denoted with prime ('), and transposition is implicit. ϵ' represents the biological noise and is sampled from a vector Gaussian with $\mu = 0$ and σ_b as defined by the user. A represents the relationship between $y_{t+1}(k)$ and y_t. It defines a linear difference equation for each gene. b' represents the constitutive level of gene expression.

The constitutive level of gene expression can be explained as follows. Imagine that each gene has some "normal" level of expression. This is its constitutive level. If i up-regulates j and i is present at more than its constitutive level then it will cause j to be more strongly expressed. On the other hand, if i is expressed less strongly than usual

then the absence of i will cause a lower-than-usual expression of j. It is claimed by Yu et al. that this model leads to more biologically plausible time series.

Note two key aspects of `GeneSim`. Firstly, non-linear gene interactions are not modeled. These are essential for biological realism. Secondly, the network edge matching algorithm does not create motifs or biological distributions over k_{in} and k_{out}.

`GeneSim` also does not consider the phenotypically conditional nature of gene regulation. However this not a serious problem, for two reasons. Biological data sets typically come from just one phenotypic situation, and inferring a single network given samples generated from potentially contradictory regulatory networks is a different and much more difficult research question. In summary, `GeneSim` is a useful simulator, however it does not model several important features of biological GRN.

3 Kyoda's Simulator

The simulator in Kyoda et al. [10] uses ODEs to model the regulatory relationships. Each gene was randomly matched to between 1 and k other genes. These genes and random rate constants defined its regulatory ODE. The auto-regulatory motif was not allowed. Networks of up to $N = 100$ with $k = 2$ and $N = 20$ with $k = 8$ were simulated.

4 GreenSim

`GreenSim` is the GRN simulator we have developed in MatLab. Inspired by `GeneSim`, it aims to realistically simulate any kind and size of GRN. This section describes its functional characteristics (4.1), its modular structure (4.2), how to use it (4.3) and analyses some `GreenSim`-generated networks (4.4).

4.1 Characteristics

`GreenSim` can simulate GRN of size $N = [10^0, 10^4]$. It generates realistic non-linear regulatory functions, and ensures that the degree distributions are accurate. These distributions also lead to the emergence of motifs in the simulated networks. Section 5 discusses extensions that would allow it to model multiple phenotypic contexts.

4.2 Structure

`GreenSim` is designed in a modular fashion, and each of the modules can be replaced without affecting the others. For example, the edge matching module was recently replaced, and no other portions of the code needed to be modified. The algorithm is structured as follows.

First, *half edges* (in-edges and out-edges for each gene) are generated according to the appropriate distributions. Next, the half edges are matched by randomly permuting the genes and then matching the out-edges with the nearest in-edges. This reflects the biological reality (genes are frequently physically located near the genes they regulate)

and is also necessary to create realistic *technical noise*[9]. Technical noise is described shortly.

Next, the linear and non-linear regulatory functions are generated. Each gene has a linear regulatory function as described in equation 1. Call the change in the expression level of the gene k according to its linear regulatory function l_k and its expression level after the previous time step n_k.

A 2nd order non-linear regulatory function for each pair of genes which regulate k is defined as shown in equation 2. NL_k is a random matrix which defines a weight for the non-linear contribution of each pair of genes i, j $(i < j)$ which regulate k. The bracketed term in the equation, $\big((y_t - b)' \times (y_t - b)\big)$, takes the product of the expression level of each pair of genes. This product and NL_k are multiplied together member-wise and the members of the resulting matrix are summed to calculate the total non-linear regulatory contribution. The updated expression level for k is $n_k + nl_k + l_k$.

$$nl_k = \sum \big(NL_k \times (y_t - b)' \times (y_t - b)\big) \tag{2}$$

As in `GeneSim`, expression levels are updated at discrete intervals and can be sampled at intervals of size δ. Such a sequence of *samples* is called a *set of samples*. Samples can also be corrupted by parameterised levels of technical noise[9]. Technical noise refers to uncertain, missing or inaccurately measured gene expression levels in the set of samples. This occurs because of limitations in the biotechnology used to gather samples. Three kinds of technical noise are modeled.

Spot noise is the low, independent probability that any particular gene expression value in a set of samples will be covered. This can be caused in biological datasets by a faulty microarray or by poor handling of the microarray after the experiment has concluded.

Span noise is the low, independent probability that some contiguous subset of the genes in a sample is covered. Because of the edge matching algorithm such genes are likely to be directly involved in each others regulation. The size of the span is normally distributed according to user-specified parameters.

Value noise is added to the biological noise and further corrupts the gene expression levels. It represents the ambiguity that can be caused by using mRNA concentrations and fluorescence levels as a proxy for the level of gene activity[1].

4.3 Using GreenSim

In this subsection we describe how to use `GreenSim` to simulate GRN and generate samples from them.

Readers are encouraged to refer to the MatLab source files, comments and included help documentation for further information.

4.3.1 Generating a Network Structure

To generate a network structure, users call the `genNetwork(n, mu, maxReg, noise)` function. This function returns a network structure, and the parameters are as follows:

n is the number of genes

μ is the location parameter for the exponential distribution over the gene in-degree.

maxReg is the maximum regulatory influence that any one gene or pair of genes can have on another. A and NL_k are in the range $(0, 1)$ and are scaled by this parameter.

noise is the standard deviation of the biological noise in the network, σ_b.

In most cases, the value of the `noise` parameter in network generation should be no more than $0.5 \times$ `maxReg`. If this is not the case then the influence of regulatory genes may be obscured and the result will be a random walk.

4.3.2 Generating A Set of Samples

A set of samples can generated by calling `genSampleSet(net, inity, num, delta)`. This function returns a matrix of gene expression levels uncorrupted by noise. The (i, j)'th entry in the returned matrix is the expression level of the i'th gene in the j'th sample. The function's parameters are:

net is a network structure, as returned by `genNetwork`

inity is a column vector specifying the initial expression level of each gene.

num is the number of samples (columns) to be generated.

δ is the number of hidden gene expression updates between each sample.

In real biological situations, gene expression levels may change between samples. As in `GeneSim`, this is represented in `GreenSim` by a δ parameter and the gene expression values are updated δ times according to the regulatory functions and biological noise between each sample.

For example, if `num` $= 3$ and $\delta = 3$ then the gene expression values would be updated 9 times according to their regulatory functions, and the third, sixth and ninth vectors would make up a set of samples.

Note that the relative magnitude of `maxReg` and the δ used when generating samples will affect the difficulty of any network inference. We suggest that $\delta \leq \frac{1}{\texttt{maxReg}}$, else expression levels may change too much and in too complex a manner between samples.

A set of samples can be corrupted by the `corruptSample` function. This function takes a set of samples as its first parameter and then five other parameters for the technical noise. Covered gene expression values are denoted with -1.

Finally, a set of samples can be transformed using the `logSample` function. The log (base 2) of each gene expression value is calculated (adjusted for raw values less than $\epsilon = 0.000001$) and covered values in the transformed set of samples are denoted with 999.

4.3.3 Calculating Statistical Network Properties

The `genNetworkStats(net)` function calculates statistical metrics for a network. This function takes a network structure as a parameter and returns a statistics structure. The statistics it calculates are analysed in subsection 4.4.

4.3.4 Saving and Loading GreenSim Data

A network can be saved or loaded using the `writeNet` and `readNet` functions. Similarly, a sample can be saved or loaded using the `writeSample` and `readSample` functions and a network statistics structure can be saved or loaded using the `writeStats` and `readStats` functions.

For samples and statistics structures the `read...` functions take a source filename as a parameter. To save a sample or statistics structure the `save...` functions take a variable of the appropriate type as their first parameter and a target filename as the second parameter.

`writeNet` and `readNet` have the same parameters and parameter ordering as the functions for samples and statistics structures. However, networks are serialised to more than one file. This means that only a root filename needs to be specified in the function calls, and the functions will append the appropriate extensions to save or load the network.

To obtain an edge matrix which can be converted into a `dot` file for `GraphViz` the `writeEdgeMatrix` function can be used.

4.4 Examples and Analysis

This section describes some networks generated using `GreenSim`, ranging in size from 3 to 10^4 genes. Subsubsection 4.4.1 introduces the analysis by providing visualisations of networks' edge matrices, for networks ranging in size from 3 to 10^3 genes. Subsubsection 4.4.2 provides more detailed statistical analysis, for networks ranging in size from 3 to 10^4 genes.

4.4.1 Visual Analysis

This subsubsection contains figures and commentary depicting the edge matrices of networks ranging in size from 3 to 1000 genes. Each edge and each pair of edges to a gene defines a regulatory function which relates the expression level of the parent(s) at time t to the expression level of the child at time $t + 1$. The child's expression level can be calculated in full by using all parents and all pairs of parents, as described in subsection 4.2.

The network shown in figure 1 illustrates `GreenSim`'s ability to generate very small networks, and also shows how the combination of the different distributions over the in and out-edges can lead to auto-regulation and feed-forward motifs.

Figure 2 illustrates the automatic generation of modules, cascade and convergence motifs by the edge matching algorithm, even in small `GreenSim`-generated networks.

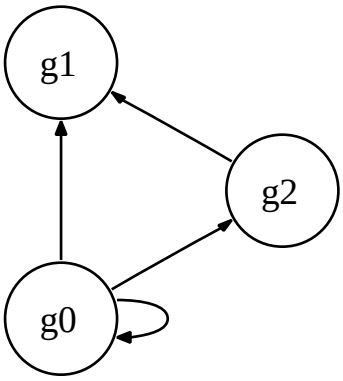


Figure 1: A `GreenSim` network with 3 genes.

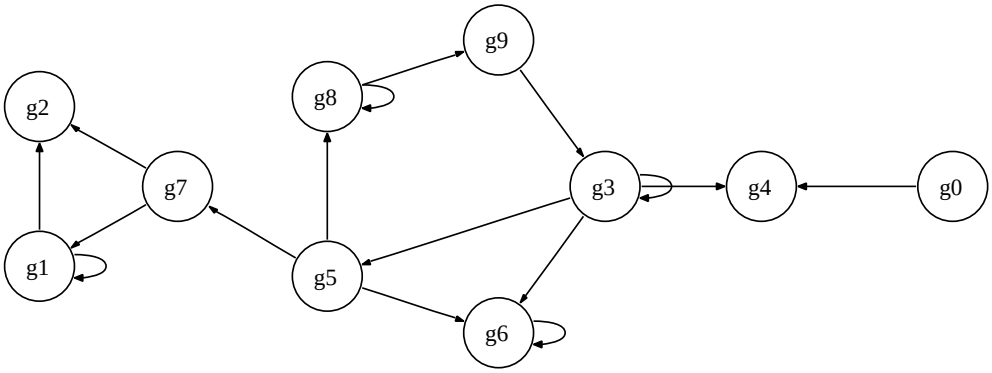


Figure 2: A `GreenSim` network with 10 genes.

Figure 3 shows the expected growth in module size as the network grows (a consequence of the power law distribution over out-edges). Cascade and convergent motifs also occur more often in larger networks.

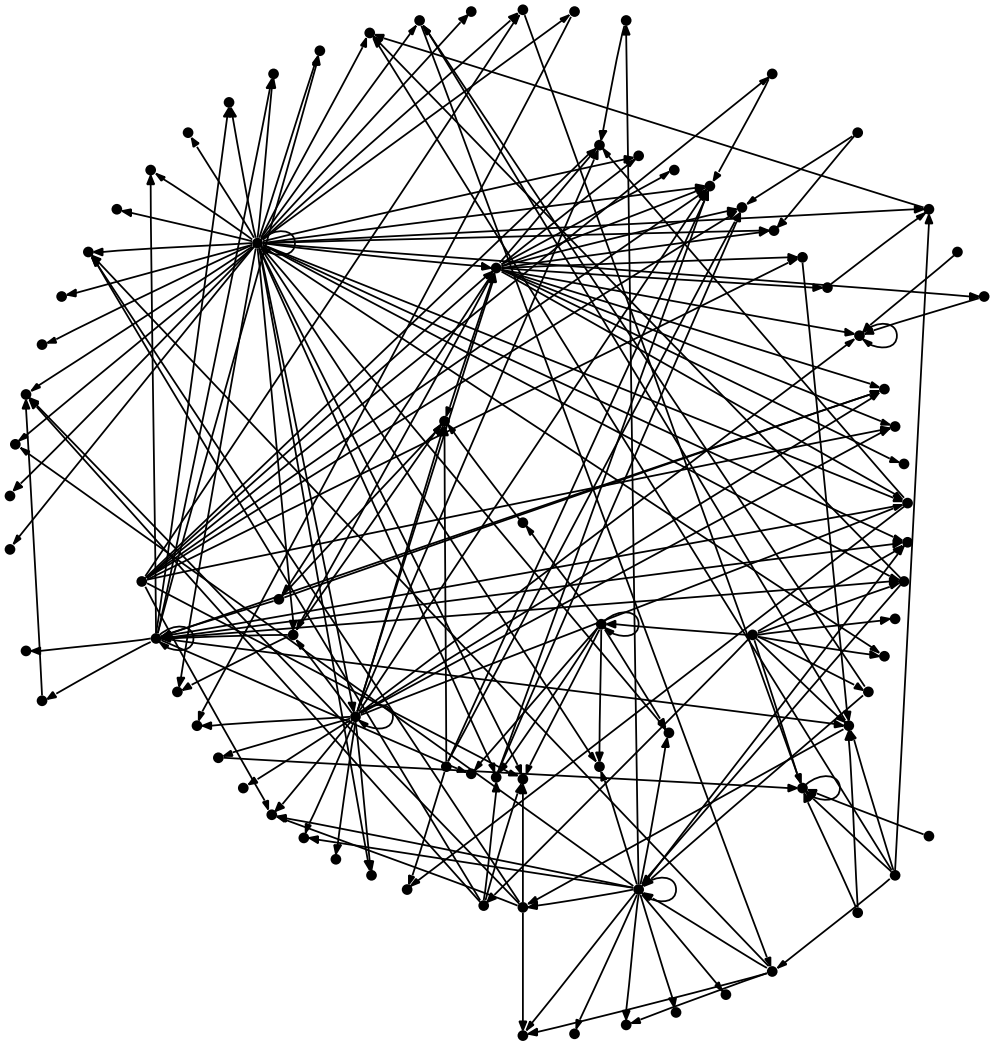


Figure 3: A `GreenSim` network with 100 genes.

Figure 4 is very difficult to interpret at the printed resolution. However, it is clear that maximum module size continues to grow while smaller modules and motifs also continue to exist.

Observation of these graphs shows that `GreenSim` networks possess many of the characteristics of real biological networks, including a messy modular structure, and the autoregulatory, feed-forward, cascade and convergence motifs.

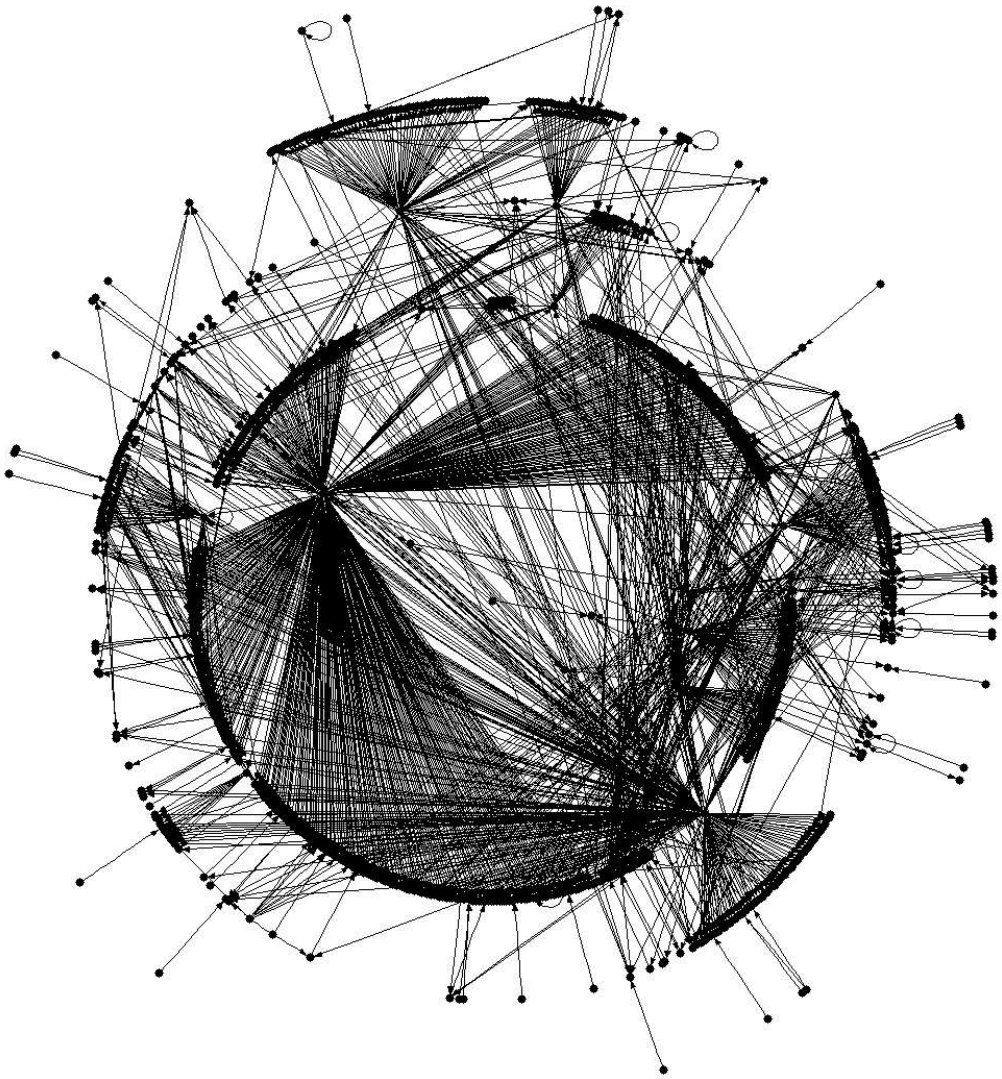


Figure 4: A `GreenSim` network with 1000 genes.

4.4.2 Statistical Analysis

This subsubsection presents and analyses some basic statistical properties of networks generated using `GreenSim`. Statistics for several different network sizes are shown in table 1 and more detailed discussion follows.

Table 1: This table contains the averages of 10 networks for each n. Statistics are displayed to 4 significant figures.

Statistic / Num. of genes	$n = 3$	$n = 10$	$n = 10^2$	$n = 10^3$	$n = 10^4$
Time (s)	0.1073	0.1404	0.2268	5.057	725.5
Total Edges	4.3	22	344.5	3518	35190
$max(k_{in})$	2.1	4.4	15.2	21.8	27.3
$max(k_{out})$	2.6	8.6	72.9	791.1	7968
Auto-regulatory motif	1.5	3.2	8.8	35.8	188.2
Feed-Forward Motif	0.3	13.1	723.4	4569	45150

The networks used to calculate the statistics displayed in table 1 were generated on an Asus M6A laptop (Pentium 4 1.8 GHz, 1 GB RAM, Win XP Pro). Observation of the results shows that the time it takes to generate a network is non-linear. Some of this is due to virtual memory paging issues. The remaining non-linearity appears to be created when the half edges are being matched.

For $n \geq 100$ we see that the number of edges per gene stabilises at ≈ 3.5. Given that the networks were generated with exponential in-degree $\mu = 3$ this is unsurprising. Furthermore in the full vectors of in and out-edges for each network we have observed that k_{out} and k_{in} are distributed correctly.

Analysis of the motifs is more interesting. It is clear that the incidence rate of the feed-forward motif is much larger in the mid-size networks. When $n = 3$ there is an average of 0.1 feed-forward motifs per gene and there are approximately 4.5 such motifs per gene when $n \geq 1000$. In contrast there are more than 7 feed-forward motifs per gene when $n = 100$. We suspect that this is a side-effect of the edge-matching algorithm but do not think it is a problem.

The change in the incidence rate of the auto-regulatory motif for very large networks is potentially more problematic. The source of this trend is difficult to determine. Given the network size, the correlation between the total number of edges and the incidence rate is almost 0. However, confusingly and independently of the number of genes, the correlation between the total number of edges and the auto-regulatory incidence rate is 0.792. There is also some correlation between network size and the incidence rate (0.747). The correlations of $max(k_{in})$, the network size and other combinations are all less than the correlation of the total number of edges and the incidence rate, independent of the network size, but greater than 0.6.

In short, naive correlations are uninformative and it is not clear what is driving this trend. However, it may be due to the distributions over k_{in} and k_{out} and the edge

matching algorithm. Because the range of k_{out} grows much more quickly than the range of k_{in}, and because the edge matching algorithm matches genes in descending order of the number of out-edges, if a network has "hub" genes which connect to the majority of other genes in the network then these genes may absorb all or most of the in-edges of a large proportion of most of the genes. This means that most genes cannot auto-regulate.

5 Discussion and Future Work

`GreenSim` is importantly different from previous simulators such as `GeneSim`. In particular, `GreenSim`:

- Explicitly and correctly models the in and out degree distributions.
- Implicitly models the other important features of a GRN, through its explicit modeling of the in and out degree distributions.
- Uses non-linear regulatory functions.
- Accurately models the various kinds of noise which exist in biological data sets.
- Is modular, easily customised and highly parameterised.
- Generates GRN ranging in size from 10^0 to 10^4 genes.

The visual analysis, statistical analysis and realistic regulatory functions suggests that `GreenSim` can be used to simulate and sample from realistic GRN of widely varying size.

One aspect of `GreenSim` that needs further investigation and which may need improvement is the edge matching algorithm. This is because it appears to under-represent the auto-regulatory motif in very large and very small GRN. If the biological literature confirms that this is inaccurate then a new matching algorithm would need to be developed.

Like `GeneSim`, the regulatory functions in `GreenSim` are only representative of a single phenotypic context. Although this is not a problem, for the reasons discussed in section 2, `GreenSim` could be extended to simulate the phenotypically context-dependent logical discontinuities in the regulatory functions that can be seen in biological organisms[13] as well.

For example, rather than generating just one set of functions (A and $\{NL_k\}$), `GreenSim` could be modified so that it generates m sets of functions. The set of functions (A^i and NL_k^i) used to generate each sample could be changed with some probability or according to environment-state variables. If a different edge matrix was necessary then the genes could be re-permuted and re-matched for each member of this set of regulatory functions.

Bibliography

[1] Yoganand Balagurunathan, Naisyin Wang, Edward R. Dougherty, Danh Nguyen, Michael L. Bittner, Jeffrey Trent, and Raymond Carroll. Noise factor analysis for cDNA microarrays. *Journal of Biomedical Optics*, 9(4):663–678, July/August 2004.

[2] Albert-Laszlo Barabasi and Zoltan N. Oltvai. Network biology: Understanding the cell's functional organisation. *Nature Review Genetics*, 5(2):101–113, February 2004. doi: 10.1038/nrg1272. URL `http://www.nature.com/nrg/journal/v5/n2/abs/nrg1272.html`.

[3] Hidde de Jong. Modeling and simulation of genetic regulatory systems: A literature review. *Journal of Computational Biology*, 9(1):67–103, 2002.

[4] Patrik D'haeseleer, Shoudan Liang, and Roland Somogyi. Genetic network inference: From co-expression clustering to reverse engineering. *Bioinformatics*, 18(8):707–726, 2000. URL `citeseer.ist.psu.edu/article/dhaeseleer00genetic.html`.

[5] Michael E. Driscoll and Timothy S. Gardner. Identification and control of gene networks in living organisms via supervised and unsupervised learning. *Journal of Process Control*, 16(3):303–311, March 2006. doi: 10.1016/j.jprocont.2005.06.010. URL `http://dx.doi.org/10.1016/j.jprocont.2005.06.010`.

[6] Christopher Fogelberg and Vasile Palade. Machine learning and genetic regulatory networks: A review and a roadmap. Technical Report CS-RR-08-04, Computing Laboratory, Oxford University, Wolfson Building, Parks Road, Oxford, OX1-3QD, April 2008.

[7] E. D. Jarvis, V. A. Smith, K. Wada, M. V. Rivas, M. McElroy, T. V. Smulders, P. Carninci, Y. Hayashizaki, F. Dietrich, X. Wu, P McConnell, J Yu, P. P. Wang, A. J. Hartemink, and S. Lin. A framework for integrating the songbird brain. *Journal of Computational Physiology A*, 188:961–80, December 2002.

[8] Stuart A. Kauffman. Antichaos and adaptation. *Scientific American*, 265(2):78–84, August 1991. ISSN 0036-8733.

[9] Lev Klebanov and Andrei Yakovlev. How high is the level of technical noise in microarray data? *Biology Direct*, 2:9+, April 2007. ISSN 1745-6150. doi: 10.1186/1745-6150-2-9. URL `http://dx.doi.org/10.1186/1745-6150-2-9`.

[10] Koji M. Kyoda, Mineo Morohashi, Shuichi Onami, and Hiroaki Kitano. A gene network inference method from continuous-value gene expression data of wild-type and mutants. *Genome Informatics*, 11:196–204, 2000.

[11] Harri Lähdesmäki, Ilya Shmulevich, and Olli Yli-Harja. On learning gene regulatory networks under the Boolean network model. *Machine Learning*, 52(1–2):147–167, 2003. ISSN 0885-6125.

[12] Matti Nykter, Tommi Aho, Miika Ahdesmäki, Pekka Ruusuvuori, Antti Lehmussola, and Olli Yli-Harja. Simulation of microarray data with realistic characteristics. *Bioinformatics*, 7:349, July 2006.

[13] E. Segal, Nir Friedman, N. Kaminski, A. Regev, and D. Koller. From signatures to models: Understanding cancer using microarrays. *Nature Genetics*, 37:S38–S45, June 2005. By invitation.

[14] V. A. Smith, E. D. Jarvis, and A. J. Hartemink. Evaluating functional network inference using simulations of complex biological systems. *Bioinformatics*, 18:S216–S224, 2002.

[15] V. Anne Smith, Erich D. Jarvis, and Alexander J. Hartemink. Influence of network topology and data collection on network inference. In *Pacific Symposium on Biocomputing*, pages 164–175, 2003.

[16] Kim Sterelny and Paul E. Griffiths. *Sex and Death : An Introduction to Philosophy of Biology (Science and Its Conceptual Foundations series)*. University Of Chicago Press, June 1999. ISBN 0226773043.

[17] Jiri Vohradskỳ. Neural network model of gene expression. *FASEB Journal*, 15: 846–854, 2001.

[18] J. Yu, V. A. Smith, P. P. Wang, A. J. Hartemink, , and E. D. Jarvis. Using Bayesian network inference algorithms to recover molecular genetic regulatory networks. In *International Conference on Systems Biology (ICSB02)*, December 2002.

[19] Jing Yu, V. Anne Smith, Paul P. Wang, Alexander J. Hartemink, and Erich D. Jarvis. Advances to Bayesian network inference for generating causal networks from observational biological data. *Bioinformatics*, 20(18):3594–3603, 2004.