Generating and Simulating Large Gene Regulatory Networks and Dynamical Analysis Applications

Sandy Kim

Advisor: Eric Deeds

Abstract

Critical temporal and spatial patterns of gene expression, vital to every organism are governed by a collection of genes and their interactions. Known as a gene regulatory network, they are often large and complex throughout nature, making it of great interest to reconstruct them. The global structures of these networks are often modeled using graphs with nodes and edges with simple representations of interactions, using a logic-based model. However, such models lack dynamic information. Dynamic models of specific Gene Regulatory Networks (GRNs) are based on differential equations, which consider dynamics. However, such dynamic models are much smaller than the GRNs found in most organisms. As such, little is currently known about the dynamic properties of GRNs on the scale of the whole-genome.

Here we generate Random GRNs (RGRNs) by randomly sampling both the interactions between genes in the network as well as the logic governing each regulatory interaction to capture large-scale network intricacies efficiently. We have demonstrated that current hardware can simulate the long time-scale behavior of networks of similar size and complexity to those that govern expression dynamics within the human genome. Also, we have shown applications of the developed framework in the analysis of dynamical behavior in a variety of generated large-scale RGRNs.

1 Introduction

A genome can encode hundreds to thousands of genes, each of which produces gene products that underlie the mechanisms of many cellular processes. Such cellular processes include cell differentiation, the cell cycle, and signal transduction. These gene products are vital and essential for nearly every process of life, such as development and responding to environmental cues. The temporal and spatial patterns at which these gene products are produced are governed by a collection of genes and their interactions, both with each other, and other substances, known as a gene regulatory network. Gene regulatory networks are often large and complex throughout nature, making it of great interest to reconstruct such networks to further the understanding of the intricacies of such complex biological mechanisms to shed light on higher-order structures and behaviors of a given organism [1].

There are many different computational methods that have been developed to model gene regulatory networks. One popular method of modeling is a discrete and logic-based model, introduced by Kauffman as Boolean networks [2]. These models are known for their simplicity in capturing the qualitative nature of gene regulatory networks. In this model, a gene can have two levels, active, represented by 1, or inactive, represented by 0, where these states are determined by all other genes in the network, effectively capturing the system's global state. Although Boolean networks have shown to provide insights regarding the existence and nature of the steady-states and the robustness of networks, it fundamentally does not model the dynamics of transcription factors due to its lack of detail [3].

However, it is important to note that most measurements taken from biological experiments are real-valued, not discrete. Thus, on the other end of the spectrum of network modeling lies a method that is continuous. Ordinary differential equations describe the instantaneous rate of change over time [4]. These models provide detailed information on the intricacies of the dynamics but require accurate data on the parameters, therefore it is only applicable to a few, specific systems [3]. But it is important to note that due to its ability to capture the quantitative nature of gene regulatory networks, these models hold significant predictive power [1] [5]. Yet, with greater detail comes a significant caveat: as the network size grows larger, these numerical computations become largely expensive. Due to this, most studies are done on smaller gene regulatory networks or a local area of a larger network. Therefore, there is a scarcity of models of large-scale networks that capture the essential biological details globally.

In this work, we generate and simulate large-scale gene regulatory networks, by using ordinary differential equations governed by Boolean logic and randomly sampling interactions between genes in order to model the complexity of networks on a global scale while maintaining computational efficiency and perform dynamical analysis on the generated networks. Findings from this study can be applied to shed light on the behavior of large gene regulatory networks.

2 Methodology

2.1 Modeling

To mathematically model random gene regulatory networks, we used a digraph-based representation, probability theory, Boolean logic, and the Hill equation to create systems of ordinary differential equations.
First, we defined two fundamental equations as the following:

$$\text{"AND"} : F(A \text{ and } B) = F(A) \wedge F(B) = F(A) \cdot F(B) \tag{1}$$

$$\text{"NOT"} : \text{not } F(A) = \overline{F(A)} = 1 - F(A) \tag{2}$$

And from De Morgan's Law and equations (1) and (2):

$$\begin{aligned}\text{"OR"} : F(A \text{ or } B) &= F(A) \vee F(B) = \overline{\overline{F(A)} \wedge \overline{F(B)}} \\ &= 1 - (1 - F(A)) \cdot (1 - F(B))\end{aligned} \tag{3}$$

Protein abundance for gene i is described by the ordinary differential equation:

$$\frac{d[\mathsf{TF}_i]}{dt} = k_{max} \cdot f_i(t) + k_{bas} - k_{deg} \cdot [\mathsf{TF}_i] \tag{4}$$

where $[\mathsf{TF}_i]$ is the concentration of transcription factor i and parameters k represent the different rates of the system. $f_i(t)$ is the promoter activity for each gene i and is calculated using Boolean logic functions and the promoter binding probabilities $P(j)$.
We used the standard Hill equation for activation and repression to define the probabilities of a transcription factor binding and activating or repressing a given gene as:

$$P_{act}(j) = \frac{[\mathsf{TF}_j]^n}{k_{prod}^n + [\mathsf{TF}_j]^n}; \quad P_{rep}(j) = \frac{k_{prod}^n}{k_{prod}^n + [\mathsf{TF}_j]^n} \tag{5}$$

I focused on a case where the Hill coefficient $n = 2$, which indicates at most, two transcription factors will bind to a given promoter region.
$f_i(t)$ is formulated by plugging $P(j)$ into combinations of (1) and (3).

2.2 Algorithm and Implementation

Let N be the number of total genes in the network.
We first generated the graph as a randomized $N \times N$ adjacency matrix to represent the network graph, where 0 represents no interaction, -1 represents repression, and 1 represents activation. In addition, I generated N initial conditions represented as an N-vector.
We reduced the interactions by omitting all 0s from the calculations, as they have no effect on the system, and generated random mixed Boolean expression trees, represented as Reverse Polish Notation (RPN) expressions.
We then evaluated the RPN expressions using a stack data structure.
The algorithm was implemented in Python 3. All ODEs were solved using the odeint function in the SciPy library, which implements standard numerical integration techniques.

2.3 Measuring Runtime and Scalability

We generated and simulated RGRNs ranging from 20 to 3,000 genes for three days of simulation time, timed each simulation from the execution to the termination of the code, and graphed the run time against the number of genes on a scatter plot. We used polynomial regression to find the best fit curve to give an indication of time complexity. Networks had a 10% edge density. Edge density is defined as $\frac{\text{number of edges that exist in a generated network}}{\text{all possible edges}}$.
We furthered the analysis of scalability by changing the network graph's edge density from 5% to 20% in increments of 5% and measuring the computational time. We keep the edge density at a maximum of 20%, as this range remains physiologically relevant in most organisms.

2.4 Stability Analysis

Let x^* be a fixed point and let $\eta(n) = x(t) - x^*$ be some small perturbation away from x^*. We can derive an approximate differential equation for η:

$$\dot{\eta} \approx \eta f'(x^*)$$

This linearization about x^* shows that the perturbation $\eta(t)$ grow exponentially if $f'(x^*) > 0$ and decays if $f'(x^*)$ [6]. We call a fixed point stable if $\eta(t)$ grows exponentially, and unstable if it decays exponentially.

2.4.1 Perturbation

We perform stability analysis on the generated networks by first, letting the system run to a steady-state, a condition in which the trajectories are unvarying or periodic. Then, we perturb the trajectories by a small magnitude, δx_0. Each of the perturbations is random, sampled from a N-dimensional hypersphere with a radius of 1.
We then graph the average distance between the perturbed trajectories and the steady-state trajectories over time. There are two ways to determine stability:

(1) Look at the graph for stability and if it follows exponential growth, the system is unstable, if it follows exponential decay, the system is stable, or if it follows a periodic pattern, the system is on a limit cycle attractor.
(2) Find the average largest Lyapunov exponent using the equation:

$$\lambda = \frac{1}{\tau} ln(\frac{||\delta x_\tau||}{||\delta x_0||})$$

where we integrate both trajectories in parallel for time τ, measure the distance between trajectories δx_τ, and rescale the separation between trajectories back to δx_0 [7]. If $\lambda > 0$, then the system is unstable. If $\lambda < 0$, then the system is stable. If $\lambda = 0$, we can make no conclusion.

2.4.2 Graphical Analysis

For systems of more than two genes, we reduce the dimensionality of genes to two using principal component analysis (PCA). Since PCA is a linear transformation, we can say the mapping of the original space to the principal component space is bilipschitz and therefore, topologically homeomorphic.
Given this, we analyze the stability from taking the principal components (PC) and creating a (PC1, PC2) plane, where we can visualize the solutions to the ordinary differential equations as trajectories in the phase space.

2.4.3 Time Delay Analysis

For systems that suggested chaotic behavior, we applied a Takens' embedding on the principal components.
Takens's theorem is a delay embedding theorem that states the following: A series of past values of a single scalar measurement, y, from a dynamical system can be used to form a vector that defines a point in a new space. Specifically, one constructs m-dimensional reconstruction space vectors $\vec{R}(t)$ from m time-delayed samples of the measurements $y(t)$, such that $\vec{R}(t) = [y(t), y(t-\tau), y(t-2\tau), \ldots, y(t-(m-1)\tau)]$. This reconstruction is guaranteed to be topologically identical to the full dynamics [8].
There is no way to know the optimal time delay τ and optimal embedding dimension m, a priori. One way to choose the best τ is to embed the data across many τ, calculate the normalized mutual information between the original and embedded data, and choose τ that minimizes normalized mutual information. One way to choose m is to embed the data across many m, calculate the between the embedded data of n dimensions and $n+1$ dimensions, and choose the $m = n$ that minimizes the number of false nearest neighbors between the two [9]. We followed these practices in estimating embedding parameters.

3 Results

3.1 Complexity

We show in Figure 1, each run time for generating large random networks from 0 to 3000 genes with an edge density of 10%.
Through polynomial regression, we were able to fit a curve:

$$f(x) = 2.024 * 10^{-4} x^2 - 1.138 * 10^{-2} x - 2.261$$

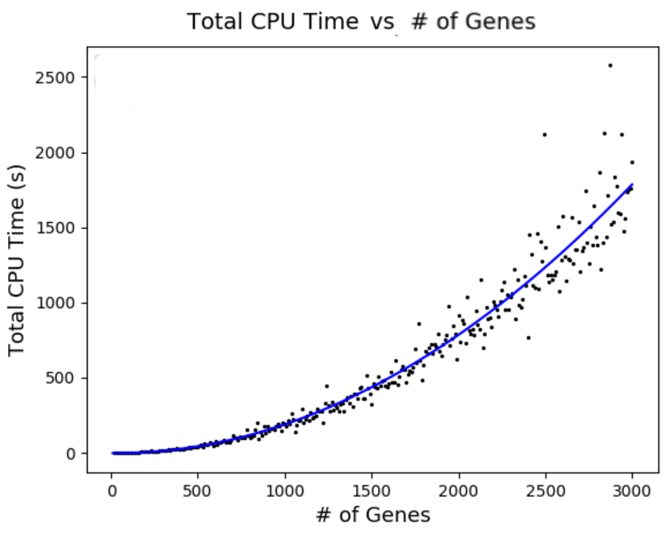

Figure 1: Computational run time against the network size. The black dots represent simulations and the blue line is a fitted polynomial curve.

Given the scale of the leading coefficient and the negative following terms, we are able to show that our method has low quadratic complexity with the number of genes. However, we do recognize as the network size grows larger, the variability of run time increases.

3.2 Scalability

We repeated our complexity analysis on edge densities of 5%, 15%, and 20% as shown in Figure 2, and found that our method scaled well, and as expected with increasing and decreasing edge densities.

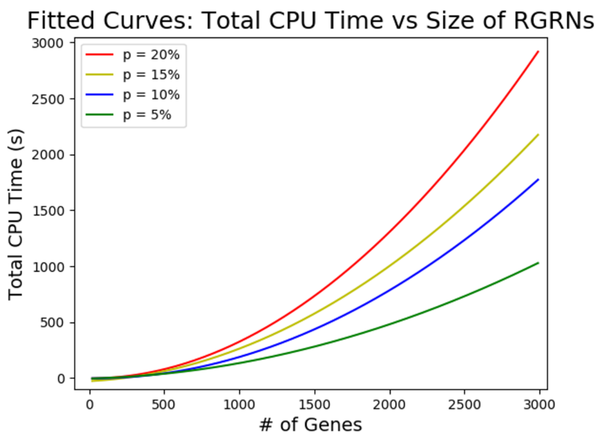

Figure 2: Computational run time against the network size for different edge densities. Each line represents a fitted polynomial curve. Green is for networks with edge density of 5%, blue for edge density of 10%, yellow for edge density of 15%, and red for edge density of 20%.

At 20% edge density, we found that it takes approximately 3000 seconds of run time to generate a network of 3000 genes.

3.3 Application of Dynamical Behavior Analysis

3.3.1 Example 1

As an example, we construct a randomly generated 100-gene regulatory network, shown in Figure 3.

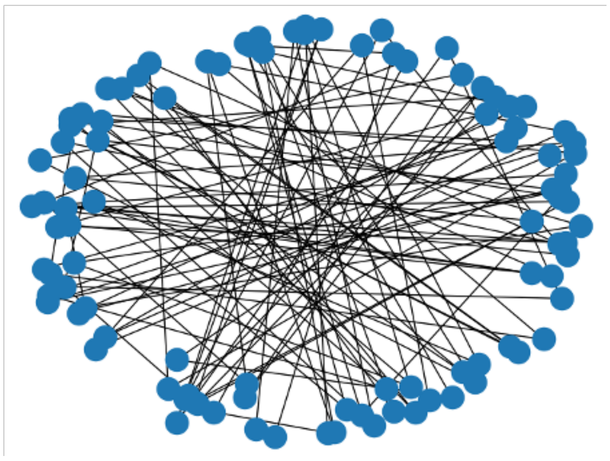

Figure 3: Schematic of the randomly generated 100-gene regulatory network where the nodes represent the genes, and the edges represent interactions. .

We take this network and implement it into our mathematical model, and computationally solve given randomly-generated initial conditions and graph the time-series solution over 8760 hours in simulation time, shown in Figure 4.

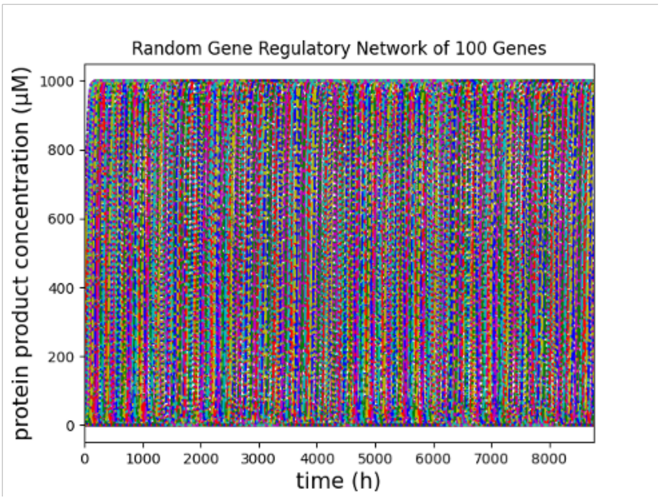

Figure 4: Time series solution graph against 8760 hours for the randomly generated 100-gene regulatory network

Given the scale of the network, we can see it is nearly impossible to differentiate each gene's protein product concentration.
We can take the system at steady-state and add random perturbations and measure the distance between the perturbed system and the steady-state over time.

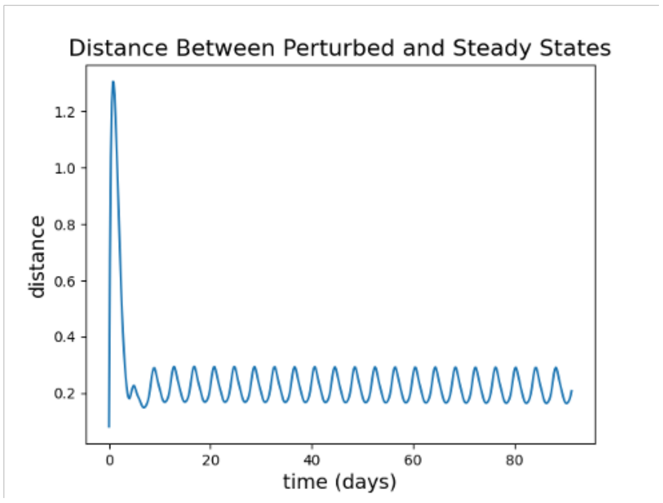

Figure 5: Distance between perturbed and steady-state against 90 days.

From Figure 5, we find that the distance follows a periodic behavior, so we can infer that the solutions follow a limit cycle attractor.
After applying PCA to the 100-dimensional series of ordinary differential equations, we can reduce the 100 genes to 2 genes (principal components) that are representative of the original network structure, and graph the 2 principal components against each other to obtain the solution curves in a phase space.

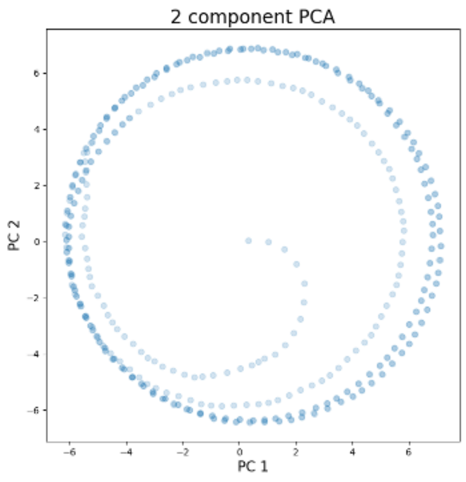

Figure 6: Trajectories of randomly generated 100-gene regulatory network in phase space visualized by graphing principal component 2 against principal component 1.

We can see from Figure 6, the trajectories will eventually converge to a limit cycle attractor.

3.3.2 Example 2

As another example, we construct a randomly-generated 200-gene regulatory repressilator [10] network, shown in Figure 7.

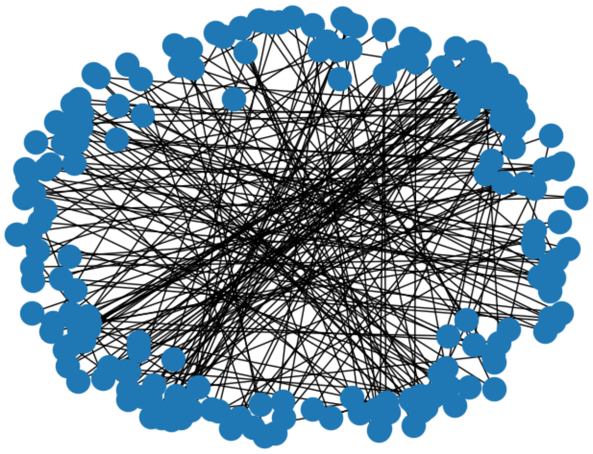

Figure 7: Schematic of the 200-gene regulatory repressilator network where the nodes represent the genes, and the edges represent interactions.

The resulting time-series solution over 87600 hours in simulation time is shown in Figure 8.

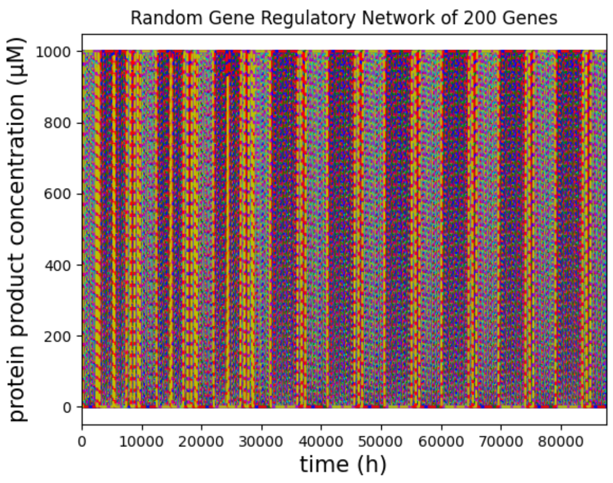

Figure 8: Time series solution graph against 87600 hours for the generated 200-gene regulatory repressilator network.

Again, given the large size of the network, an individual gene's protein product concentration is indistinguishable by eye. So, we perturb the system at steady-state and measure the distance between the perturbed and steady-state over time.

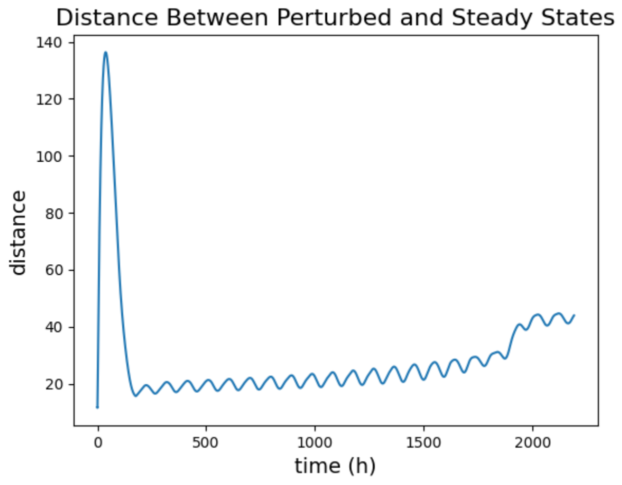

Figure 9: Distance between perturbed and steady-state against 2150 hours.

We can see near the end of Figure 9, the distance begins to increase, suggesting unstable or chaotic behavior.
Given that chaotic behaviors can only be exhibited in series of ordinary differential equations of 3-dimensions are higher, we apply PCA to the 200-dimensional series of ordinary differential equations and reduce it down to 3 principal components.

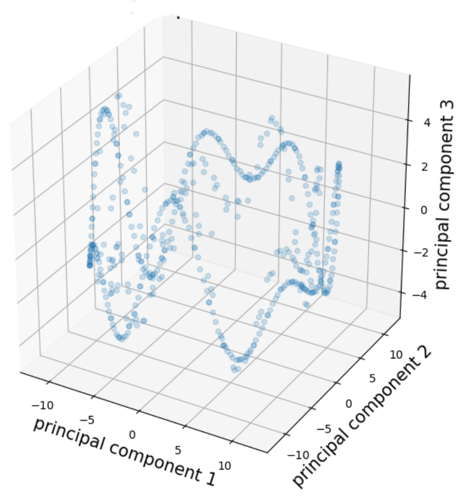

Figure 10: Trajectories of generated 200-gene regulatory repressilator network in phase space visualized by graphing principal components 1-3 against each other.

We can see from Figure 10, the trajectories are periodic along the first and second principal component, but aperiodic on the third. The apparent randomness in the third principal component leads us to explore chaos further by applying a Takens' embedding on it.

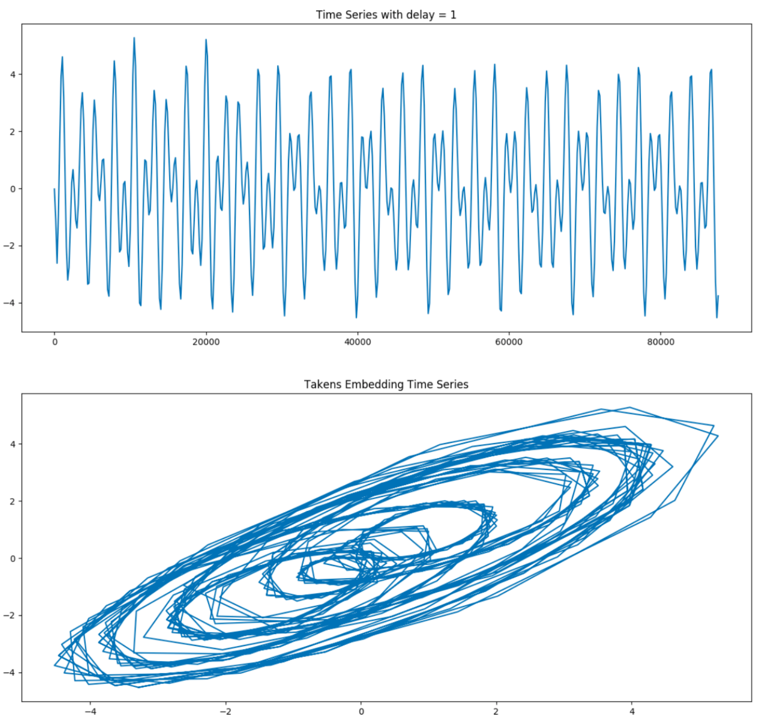

Figure 11: Takens' embedding of the principal component 3, where $\tau = 1$ and $m = 2$

We can see from the time series in Figure 11 that the oscillations are bounded, but aperiodic, and the projection onto the new space exhibits irregularity on the attractor.

4 Discussion

From our results, we can see that our method of generating and simulating large random gene regulatory networks can capture sufficient biological data on a genome-wide scale while being computationally feasible. We have reinforced this feasibility by demonstrating it is achievable from an edge density range that is physiologically relevant to most organisms.

In addition, with our simulation output formatted as a time series of protein product concentration for each gene, our generated networks allow for accessible applications of dynamical behavior analysis. When generating large gene regulatory networks, we have seen the time series solutions yielded from solving the series of ordinary differential equations for long-term behavior is hard to distinguish from one gene to another, undermining the ability to analyze dynamical behavior from that alone. By applying stability analysis by the Lyapunov exponent, we were able to simply perturb the system at its steady-state to get the system-wide behavior. Also, by leveraging the fact that Principal Component Analysis, a linear technique, we were able to simply visualize the trajectories the system takes to bolster suggested behaviors.

While there may be concerns due to the abstractness of this study's inherent mathematical nature, we hope that the ability to analyze such large systems will allow us to suggest phenomena for real gene regulatory networks in organisms in the future. For instance, in Example

1, given its stability towards a limit cycle attractor, we can say that organisms with similar 100-gene regulatory networks have a network that is robust in the face of external attacks. In Example 2, the possible presence of chaos in the 200-gene repressilator system can indicate that there exists a high dependency on the amount of protein product concentrated initially produced in much larger scale systems. This is suggestive that any deregulation of the system will impact the system heavily, causing it to behave dysfunctionally.

Although we have demonstrated a method that can analyze large-scale gene regulatory networks, there are areas in which that have yet to be created and refined. There is a lot of work that remains in analyzing chaotic behaviors. Although using Takens' embedding suggests the existence of chaotic behavior, it is far from rigorous proof. In future studies, there is an interest to characterize networks with specific behaviors (e.g. stable towards a fixed point, stable towards an attractor, unstable, chaotic). We hope with this, we can leverage machine learning to create a predictive model from generated networks and their behaviors in which we can apply to real biological data to infer what behavior a given organism's gene regulatory network will likely have.

Acknowledgements

This work was partially done under the UCLA Quantitative and Computational Biosciences Bruin in Genomics Summer Research Program and was partially funded by the National Institute of Mental Health R25: Undergraduate Research Experience in Neuropsychiatric Genomics.

References

[1] Frank Emmert-Streib, Matthias Dehmer, and Benjamin Haibe-Kains. Gene regulatory networks and their applications: understanding biological and medical problems in terms of networks. *Frontiers in cell and developmental biology*, 2:38–38, Aug 2014. 25364745[pmid].

[2] Stuart Kauffman, Carsten Peterson, Björn Samuelsson, and Carl Troein. Random boolean network models and the yeast transcriptional network. *Proceedings of the National Academy of Sciences*, 100(25):14796–14799, 2003.

[3] Guy Karlebach and Ron Shamir. Modelling and analysis of gene regulatory networks. *Nature Reviews Molecular Cell Biology*, 9(10):770–780, Oct 2008.

[4] Edda Klipp, Ralf Herwig, Axel Kowald, Christoph Wierling, and Hans Lehrach. *Systems Biology in Practice: Concepts, Implementation and Application*. Wiley-Blackwell, 1 edition, May 2005.

[5] T. Chen, H. L. He, and G. M. Church. Modeling gene expression with differential equations. *Pac Symp Biocomput*, pages 29–40, 1999.

[6] Steven H. Strogatz. *Nonlinear Dynamics and Chaos: With Applications to Physics, Biology, Chemistry and Engineering*. Westview Press, 2000.

[7] Iaroslav Ispolatov, Vaibhav Madhok, Sebastian Allende, and Michael Doebeli. Chaos in high-dimensional dissipative dynamical systems. *Scientific Reports*, 5(1):12506, Jul 2015.

[8] Stane Kodba, Matjaž Perc, and Marko Marhl. Detecting chaos from a time series. *European Journal of Physics*, 26(1):205–215, 2004.

[9] Elizabeth Bradley and Holger Kantz. Nonlinear time-series analysis revisited. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 25(9):097610, 2015.

[10] Michael B. Elowitz and Stanislas Leibler. A synthetic oscillatory network of transcriptional regulators. *Nature*, 403(6767):335–338, 2000.