Modification and Sensitivity of Equilibria in

Partially Characterized Genetic Networks

by Arvind Thiagarajan

S.B. Electrical Engineering and Computer Science MIT, 2013

S.B Biological Engineering MIT, 2013

Submitted to the
Department of Electrical Engineering and Computer Science
in Partial Fulfillment of the Requirements for the Degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

Massachusetts Institute of Technology

June 2018

Author: _____
Department of Electrical Engineering and Computer Science
May 25, 2018

Certified by: _____
Ron Weiss, Director of MIT Synthetic Biology Center, Thesis Supervisor
May 25, 2018

Accepted by: _____
Katrina LaCurts, Chair, Master of Engineering Thesis Committee

Modification and Sensitivity of Equilibria in Partially Characterized Genetic Networks
by Arvind Thiagarajan
Submitted to the Department of Electrical Engineering and Computer Science

May 25, 2018
In Partial Fulfillment of the Requirements for the Degree of Master of Engineering in Electrical Engineering and Computer Science

ABSTRACT
Engineering intracellular biological control systems, whether to perform artificial functions or to modify existing behavior, requires accurate characterization of genetic networks. Slight errors in characterization lead to significant changes in predicted behavior, and consequently to design errors that are difficult to debug. Metabolic engineering provides an empirical solution to this problem by varying many system parameters simultaneously to optimize the steady state level of a particular chemical species. Such approaches, though effective, require significant customization for the systems being studied. Here I propose and investigate, *in silico* via generation and analysis of random genetic networks, a more general approach in which a small number of modifications, selected empirically, are made to genetic networks. We found that feedback loop based modifications were most effective, and that responses to modifications depended significantly on network structure. Our work serves to advance a potential technique for efficient empirical design and modification of genetic networks.

Acknowledgements

I would like to thank Professor Ron Weiss for supervising my thesis project, and for introducing me to synthetic biology nearly a decade ago. Although I have worked on a number of topics since then, systems and synthetic biology continue to capture my interest and I appreciated the opportunity to work within the field again. For this reason, I would also like to thank Professors Timothy K. Lu of MIT and Michael B. Elowitz of the California Institute of Technology for facilitating my earlier experiences in synthetic and systems biology and helping me build intuition for the field. I am grateful to my friends and peers, who indulged my excitement about my work and provided feedback when my thoughts were stuck in loops. Finally, I appreciate the support my family has given me throughout this process.

Contents

List of Figures

1 Introduction

1.1 Motivation

Systems Biology is a burgeoning new field, encompassing and impinging on many aspects of other biological sciences. The subject deals with the quantitative analysis of systems architectures in biological networks that lead to emergent phenomena. Such analysis draws heavily on electrical engineering, physics, and control theory, and consequently appeals to scientists from a variety of fields. Systems Biology is most interesting, however, because biological systems are complex: a qualitative description of parts does not lead to a unique qualitative description of the whole, and consequently quantitative data must be used to determine the regimes between which a system has qualitatively different behavior.

Synthetic Biology is likewise a relatively new field, empowered by the same analytic tools as is systems biology. While systems biology focuses on the analysis of complex systems found in natural biological contexts, synthetic biology deals with the design, analysis, and implementation of novel genetic circuits and protein interaction networks within experimental organisms. Despite their differing subjects of inquiry, systems and synthetic biology both deal with biological logic and control. Indeed, many of the structures and motifs used widely today in synthetic biology were first discovered by systems biologists in the process of investigating natural systems. Just as insight in other engineering disciplines has often come from natural processes, so too does synthetic biology draw heavily from existing biological systems. With synthetic biology, however, we are afforded a novel opportunity, one which does not tend to present itself

outside of medicine and biology. In particular, we find ourselves in a position to augment existing systems, often with the very techniques we derived from studying them. Many of these systems perform complex regulatory functions which could be utilized for our gain and which would be difficult to recreate synthetically. However, these systems cannot usually be repackaged directly for our use – they require slight modifications to their functioning. Creating synthetic systems poses a clear challenge, but modifying natural systems is often much more difficult. As a result of this difficulty, synthetic biologists tend to avoid natural systems, preferring to engineer control networks that are as disjoint from endogenous networks as possible. Consequently, most work in synthetic biology is aimed at creating entirely novel systems on a smaller scale, as opposed to augmenting or building novel subsystems for large existing systems. Even in such cases, however, ensuring that the synthetic addition does not interact with endogenous networks is difficult, and thus such projects would also benefit from improvements in our ability to modify the behavior of endogenous networks precisely.

There are two difficulties associated with a rational design approach to modifying existing systems. The first is that endogenous genetic networks tend to be large, involving tens if not hundreds of genes, and can be relatively dense in regulatory connections between these genes. As such, characterizing the constituent parts of these networks can be time consuming. The second, and more crippling, difficulty is that small differences in the parameters characterizing these constituent parts can, when coupled with each other, lead to rather drastic differences in chemical behavior. One consequence of this is that errors in experimental measurements of these parameters, while individually minor,

together can lead to significant errors in our predictive model, and consequently to significant errors in our designs.

Metabolic engineering deals with the problem of modifying endogenous networks by eschewing simple designs and considering global modifications to the system under consideration [1]. This approach has proven successful in the past, but is still susceptible to a number of difficulties. First, just as with rational design, metabolic engineering requires full knowledge of the regulatory graph corresponding to the network of interest. Second, even after the entire network is known, metabolic engineering requires up-front augmentation of the system to allow for simultaneous over and under-expression of many chemical species. This is then guided using noisy measurements of the system parameters, mathematical determinations of the best perturbations to apply, and significant empirical adjustment of these analytic solutions [2]. Altogether, then, metabolic engineering is an intensive process, and one not ideally suited to the kind of experimentation required for early stage research. To be more concrete, metabolic engineering would not be a viable strategy in the face of uncertainty regarding choice of network or objective to optimize.

Despite its drawbacks, we believe that the approach utilized in metabolic engineering might contain useful intuition for more investigative modification of genetic networks. In particular, the reliance on empiricism to determine which modifications are most effective can be employed as easily with small, local modifications to the network as it can to large, global modifications. Any individual parameter might be changed relatively easily, and experiments in which different parameters are varied can be conducted in parallel. As such, if local modifications can be effective for most networks,

then identifying good local modifications will not be particularly intensive or time consuming. The question that we wish to investigate, then, is whether such modifications can be effective for most networks.

1.2 Background

Biological systems can be quite complex, but at the most basic level they are composed of three different reaction types: binding reactions, production reactions, and degradation reactions. In binding reactions, a number of molecules are joined to form a molecular complex which then participates in other reactions as a distinct entity. In production reactions, an existing molecular species mediates the generation of new molecules, potentially of another species. Finally, in degradation reactions, a species is gradually destroyed. In each reaction, the rate at which the reaction proceeds is linearly proportional to the concentrations of each of the constituent reactants, with identical reactant species treated as distinct reactants and with the constant of proportionality determined by the energetics of the reaction.

Production reactions are usually driven by genetic promoters, i.e. sequences of DNA preceding genes that control the rate at which those genes are transcribed to form mRNA and, eventually, proteins. Promoters often possess binding sites for other protein molecules, known as transcription factors. Bound promoters typically have significantly different expression rates than do unbound promoters, though both the qualitative and quantitative effect of this binding on the production rate vary from promoter to promoter. As a result, a promoter and the corresponding transcription factor can be used as a switch of sorts, allowing the output of one subsystem to serve as the input to another. This is the fundamental premise of intracellular biological control.

Using this basic analytic framework, many design principles can and have been ported into synthetic biology from more conventional engineering disciplines. Feed-forward loops have been used to prime engineered systems alternately to both transient and persistent changes in input signal [3]. Bi-stable toggle switch designs based on small-scale feedback loops allow transient signals to be transformed into enduring changes in downstream target concentrations [4]. Networks designed to possess no steady states have been employed to produce oscillating concentrations of various species [5]. These are just a few of the ways in which synthetic biology is able to produce novel control systems within cells. It is worthwhile to note that these techniques require a good deal of analysis and tuning in order to produce the desired results in any given system. Nonetheless, work in synthetic biology has shown the potential of employing older engineering methodologies in a biological context.

For the purposes of our investigation, however, we will not be making any assumptions about the presence of particular regulatory subgraphs, or motifs, in endogenous networks. These networks have evolved over time and involve many genes, and as such may employ entirely different design principles than those with which we engineer smaller artificial networks. The only assumption we will make, then, is that the networks we are considering have at least one steady state.

2 Setup

2.1 Problem Statement

Intracellular dynamics often serve to consolidate a number of factors to control the level or rate of production of a key compound. For instance, most metabolic regulation serves, in one way or another, to control the steady state level of adenosine triphosphate. This molecule, composed of a nucleotide base, a sugar, and three phosphate groups, serves as a carrier of energy within the cell. It imparts energy to various intracellular reactions by relinquishing one of its phosphate groups, in the process releasing energy which is consumed by the recipient reactions. This dephosphorylation converts adenosine triphosphate (ATP) to adenosine diphosphate (ADP).

While the combined supply of ATP and ADP available within the cell is relatively invariant to any number of intracellular signals, the ratio of concentrations between these two species is subject to dynamic control, as this ratio determines the amount of energy available to be consumed at any moment by the cell's various reactions. The conversion of ADP to ATP is mediated by various enzymes post-translationally. While this process, being central to metabolism, is quite well studied, many similar processes of comparable complexity are not. In such processes, the level of a key compound, much like ATP, is regulated post-translationally, and the reactions involved are much harder to uncover than are transcription reactions and promoter binding events.

In such a scenario, one or several products of a large genetic network are involved in a series of reactions that serve to regulate the concentration of this key compound. Because these reactions are difficult to elucidate precisely, it is hard to predict exactly

which components of the genetic network directly affect the regulation of the key compound, much less how they affect this regulation. Furthermore, the whole genetic network may itself be partially uncharacterized, i.e. only a subset of the network graph and parameters may be known. Under such constraints, one cannot model the control system fully and so cannot predict which modifications to the system will lead to desired changes in the steady state concentration of the key compound.

However, one can make small alterations to the genetic network and observe the effect that these alterations have on the level of the key compound. These small alterations must of course be restricted to the known subset of the genetic network. Whether such an empirical search is, on average and in the worst case, relatively certain to succeed, highly likely to fail, or subject to great variability - this is the question we seek to answer.

We seek to investigate this question *in silico*, by generating many possible genetic networks. For each such network, given uncertainty about downstream post-translational reactions, we can at best assess the impact our modifications have on arbitrary product of the genetic network. Put another way, if introducing a small modification to the production of an arbitrarily selected gene in the network affects the production of most other genes in the network, then we can be relatively sure that the empirical search we have described would be effective in practice, even when much of the genetic network itself is uncharacterized. Thus, we must measure, over many different genetic networks, the likelihood that a modification in the production of gene A affects the steady state level of gene B, for randomly selected but distinct genes A and B. We will describe our methodology in subsequent sections of this document.

2.2 Implementation Details

All subsequently described algorithms and procedures were implemented in Python 2.7. Heavy use was made of the numpy and scipy modules. Portions of the codebase were later rewritten in C++, with the pybind module used to communicate between code written in each language. The introduction of C++ was done in the hopes of improving performance, but these hopes were ultimately unfulfilled, as improvements in runtime were marginal and were offset by losses due to interfacing. The same was largely true with regard to attempts to parallelize analysis over the nodes in a given graph. In contrast, parallelization over the sample space of graphs was both straightforward and effective.

3 Representing and Analyzing Genetic Networks

3.1 Full Deterministic Model

A genetic network, as mentioned earlier, is composed of a set of genes and set of units, called operons, responsible for producing the proteins encoded by these genes. For the purposes of our exposition, we will consider a gene to be equivalent to the protein it encodes. Each operon includes a promoter and a gene, such that the promoter controls the expression of the gene. A promoter may bind to a number of the proteins in the network. The probabilities with which each binding partner of a given promoter will bind are not necessarily independent, i.e. the binding of one protein to a promoter may make it easier or harder for a different protein to bind to that promoter. As we will discuss later in this section, these binding probabilities are also affected by the concentrations of the proteins involved. Finally, the rate at which a protein is expressed by a given operon is determined by the set of proteins bound to the associated promoter.

Each genetic network can thus be represented by a "directed" hypergraph. In this hypergraph, a node corresponds to a gene in the network. An edge in a "directed" hyper graph consists of two sets of nodes, a source set and a sink set. In this hypergraph, an edge encodes an operon - the source set is the set of proteins that may bind to the promoter, while the sink set contains only the gene produced by the operon. Each edge is thus further characterized by a transfer function from the concentrations of the proteins in the source set to the production rate of the sole gene in the sink set.

It would be computationally infeasible to sample from the space of such hypergraphs. We will briefly elaborate on the size of this space. Specifying an edge from

a hypergraph like the one described requires selecting any subset of the nodes to be the source set and selecting a single node from the graph to be the sink node. This can be done in $n2^n$ ways, where n is the number of vertices in they hypergraph. Thus, a graph with n vertices and k edges is selected by choosing k edges from the $n2^n$ possible edges. Correcting for the indistinguishability of nodes, which leads to overcounting of the size of this space by a factor of $n!$, the size of this space scales as

$$\frac{1}{n!}\binom{n2^n}{k} \in O\left(\frac{2^{nk}}{n^{n-k}}\right)$$

As can be seen, this scaling is quite drastic in both hyperparameters, and especially in k. This size would only be exacerbated by the additional specification of each edge's transfer functions. As such, for the purposes of our investigation, we will simplify our search space greatly.

3.2 Simplified Model

We will consider only genetic networks in which each operon has exactly one protein that may bind to the promoter. This reduces our hypergraphs to simple directed graphs. In such a graph, the production of a single protein can still be controlled by the levels of multiple proteins in the network, i.e. there may still be multiple edges incident on a single node. The main distinction between this kind of network and a general network is that each of these control units operate independently, and the production from each is additive. We will now describe how the operons themselves will behave under our model, and thus how they may be characterized. Our descriptions will further limit the flexibility of our model, and while these restrictions certainly reduce the applicability of

our results, we contend that they are necessary to have any confidence in our sampling over the space of graphs.

3.3 Characterizing Operons

3.3.1 Dissociation Constants

As we have mentioned before, operons work to produce the associated protein at different rates depending on which proteins are bound to the promoter. In our simplified model, only one protein may be bound to the promoter. For the remainder of this discussion, we will denote the protein that binds to the promoter as protein A. The propensity with which a single copy of A binds can be understood using statistical mechanics. The relative probability of a finding a system in a particular configuration is given by

$$e^{-\frac{E}{k_B T}}$$

where E is the energy of the configuration, k_B the Boltzmann constant, and T the temperature. This value is known as the Boltzmann factor for this configuration. We can assume that A is comparatively immobile when bound to the promoter and thus the probability that A is bound is given by this expression, and not by a sum over multiple configurations, with E being the relative energy of binding between the promoter and A. That said, while this energy can be computed using molecular models, the normalization constant cannot be computed without integrating over all configurations of the system, and thus in practice the probability of A binding to the promoter cannot be computed. It must instead be measured experimentally.

The related quantity that is most often measured, usually using a technique called Surface Plasmon Resonance [6], is the dissociation constant. This constant is the ratio of

the rate at which A will unbind when bound to the promoter to the rate (per unit concentration) at which A will bind, i.e.

$$K_D = \frac{r_{unbinding}}{r_{binding}}$$

Since the overall binding rate must equal the unbinding rate at equilibrium, we have that

$$p_{unbound} * r_{binding}[A] = p_{bound} * r_{unbinding}$$

which in turn implies that

$$[A]^{-1}K_D = \frac{1 - p_{bound}}{p_{bound}} = \frac{1 - C^{-1}e^{-\frac{E}{k_BT}}}{C^{-1}e^{-\frac{E}{k_BT}}} = Ce^{\frac{E}{k_BT}} - 1 = De^{\frac{E}{k_BT}}$$

where C is the normalization constant, D the sum of Boltzmann factors for all nonbound configurations, and E is, again, the binding energy of the promoter and A. Thus, while we might not be able to calculate K_D from first principles, we can determine how changes in E will affect K_D. Since we are better able to predict E as a function of the promoter's DNA sequence, this suggests an experimental analogue to varying K_Ds in our sampled genetic network.

3.3.2. Expression Rates

As mentioned earlier, genes are expressed by operons at different rates depending on whether the operon's promoter is bound. In fact, there are at least two processes that are relevant for the actual production of a protein - the transcription of the gene to an mRNA, and the translation of the mRNA to a protein. Each transcription of a given gene takes a fixed amount of time, as does translation of the mRNA, and thus the true dynamics of the system must account for the time delay caused by transcription and translation. Indeed,

the difference between bound and unbound promoter states arises from differences in the promoter's ability to recruit enzymes that transcribe DNA. It is really the probability of recruitment that changes when the promoter is bound, and not the rate of transcription once it has begun. However, because we are only concerned about the equilibria of the systems we generate, we can treat all of these dynamics as continuous. This means that we can combine transcription and translation into a single production process that happens at a certain rate. Thus, for the purposes of our analyses, we need simply specify two rates: the rate at which the operon produces protein when its promoter is bound, and the corresponding rate when the promoter is unbound. In general, we shall refer to operons with higher rates when bound as activating units, and in the same vein we shall refer to operons with lower rates when bound as repressing units.

3.4 Network Dynamics

3.4.1 Combined Action of Edges

Let the node in a given network be numbered 1 through n and the edges 1 through k. Furthermore, for each edge e, let e_i and e_f be the starting and ending vertices of e. Furthermore, let r_e and s_e be the rates of expression for e when the promoter is bound and unbound, respectively. Finally, let D_e be the dissociation constant for e. From our earlier derivation, we have the following:

$$[e_i]^{-1} D_e = \frac{1 - p_{bound}}{p_{bound}} \implies p_{bound} = \frac{[e_i]}{D_e + [e_i]}$$

and consequently

$$p_{unbound} = 1 - p_{bound} = \frac{D_e}{D_e + [e_i]}$$

Thus, the effective rate of expression of node m by the operons of the network will be

$$\sum_{e|e_f=m} \frac{[e_i]r_e + D_e s_e}{D_e + [e_i]}$$

where the summand is merely a rearranged form of

$$p_{bound} * r_e + p_{unbound} * s_e$$

3.4.2 Time and Concentration Scales

It is worth taking a moment to consider the units for the various quantities we've been discussing before we proceed. There are only two dimensional quantities involved in the dynamics we have written down: chemical concentration and time. The per-edge expression rates (r_e and s_e) have units of concentration per time, while the per-edge dissociation constants (D_e) have units of concentration. Thus, as currently written, the net rate of expression for each node will have two arbitrary scale factors. We could fix one D_e and one r_e in each graph to be 1s and eliminate this redundancy. However, we will instead address the concentration scale redundancy later, via our sampling mechanism, and we will address the time scale redundancy via the introduction of degradation rates.

3.4.3 Degradation Rates

As presently described, our genetic networks only have production terms. New copies of proteins will get created over time, and the concentrations of all nodes will continually climb. Under such dynamics, no equilibria may exist. Cells prevent runaway growth by employing degradation mechanisms, by which specific proteins are created to bind and decompose other proteins. Under such mechanisms, the rate at which a given species of

protein is eliminated is determined by the probability that such a protein encounters a degrading protein.

Thus this degradation rate is proportional to the concentration of the protein being degraded. Furthermore, the constant of proportionality will be entirely determined by the concentration of the degrading protein involved. While some of these degrading proteins bind and degrade specific classes of substrates, most are generic and will target any protein. As such, in most cases, the constant of proportionality for degradation rates will be invariant across the identity of the protein being degraded. For this reason, we have chosen to fix all protein degradation rates to have identical constants of proportionality. Furthermore, since these constants have units of inverse time, we will fix them all to be 1, thus eliminating the time scale redundancy.

Accordingly, the final formula for the net rate of expression of node m is

$$\frac{d[m]}{dt} = \left(\sum_{e|e_f=m} \frac{[e_i]r_e + D_e s_e}{D_e + [e_i]} \right) - [m]$$

3.5 Finding Network Steady States

Now that we have established the dynamical equation for a genetic network, we are interested in finding the equilibria implied by these dynamics. We implemented two algorithms for doing this. The first relied on simulating the dynamics directly and allowing them to converge to equilibria, while the second relied on using a multidimensional Newton's Method to solve for the concentrations at which all net expression rates are zero. While we describe both methods below, we would like to note

up front that the root finding method was almost 5 times more performant than the direct simulation method.

3.5.1 Direct Simulation via Numerical Integration

The net expression rate given above can be computed simultaneously for each protein m. This was done by organizing all graph parameters in a three dimensional tensor $(e_i \times e_f \times (D_e, r_e, s_e))$ and encoding the dynamical equation above as a tensor computation. The resulting vector gives the time derivative of all protein concentrations at once. We integrated this derivative vector, recomputing it after every step, until the norm of the derivative vector fell below an arbitrary threshold of 10^{-6}. We used variable time steps for this integration, in a manner similar to many adaptive optimization methods used today. In particular, we maintained a running weighted sum of derivative norms that we used as the inverse of the time step. This sum S was initialized to 0 and updated according to the following rule:

$$S_{i+1} = \left| \frac{dv}{dt} \right| + cS_i$$

where v denotes the vector of protein concentrations. Ultimately, the value of $c = 0.9$ was chosen empirically to minimize time to convergence.

3.5.2 Solving Root Equations

The key component of this method involves recognizing that we are not interested in the dynamics of the system, but rather only in determining values of v at which the derivative vector is zero. Denote by $G(v)$ the expression for the derivative vector, i.e. $G(v) = \frac{dv}{dt}$.

We are thus looking to solve the equation $G(v) = 0$. To do this, we chose to analytically derive the Jacobian of $G(v)$:

$$J_{mj} = \frac{dG_m}{dv_j} = \frac{d}{d[j]}\left(\frac{d[m]}{dt}\right) = \left(\sum_{e|e_i=j,e_f=m} \frac{D_e(r_e - s_e)}{(D_e + [j])^2}\right) - \delta_{mj}$$

At each step, we approximated $G(v)$ linearly to solve for its root, and then used this approximate solution as the next iterate. This gave the following update rule:

$$v_{i+1} = v_i - J^{-1}G(v_i)$$

where the matrix-vector product with the inverse matrix was actually computed indirectly using an LU decomposition of J. This procedure was run until the norm of $G(v)$ fell below the same arbitrary threshold (10^{-6}) used in the previous method. As mentioned above, this Newton's Method strategy proved five times faster than actually simulating the system dynamics.

3.6 Strategies for Assessing Stability

An equilibrium for a genetic network is only worth analyzing if it is stable, as these are the only systems we will actually observe experimentally. As such, we needed to provide a way to assess whether an equilibrium was stable. A system is stable at a certain point if and only if small perturbations to the state of the system decay over time to 0, irrespective of the direction in which these perturbations are made. An equivalent way of stating this condition is that the repeated application of the linearized system, at the equilibrium point, should always give derivatives at least partially antiparallel to the introduced perturbation, i.e. that $pJp < 0$ at the equilibrium for all perturbations p. This statement will be true if and only if all eigenvalues of J have negative real parts.

3.6.1 Computing the Spectrum of the Jacobian

The most straightforward way to check whether all eigenvalues of J have negative real parts is to compute all the eigenvalues of J directly and check their real parts. There are a number of ways to compute the spectrum numerically, and all rely at core on the power method. Under this method, repeated application of the matrix J to a randomly selected vector (with intermediate normalizations to prevent numerical overflow) will converge to the eigenvector with eigenvalue of largest magnitude. Inductively, assume that you have identified the eigenvectors corresponding to the eigenvalues with the k largest magnitudes. Then to identify the next largest eigenvalue, simply repeat this procedure, beginning with a random vector orthogonal to the set of k eigenvectors you have so far. In this way, you will eventually have enumerated all eigenvectors and their corresponding eigenvalues.

3.6.2 Bounding the Spectrum of the Jacobian

The key insight associated with this method is that we don't need all of the eigenvalues - just the greatest and the least eigenvalues. The method begins, much like the previous one, with the power method. It is used to isolate the eigenvector with eigenvalue of largest magnitude. We then check to ensure that this eigenvalue has a negative real part by computing $\lambda_{max} = pJp$ for normalized eigenvector p. Assuming it is, let $J' = J - \lambda_{max} I$. Note that adding a multiple of the identity to a matrix simply shifts the original eigenvalues by that prefactor.

Now, since $\lambda_{max} < 0$ was the eigenvalue of J with greatest magnitude, it follows that every other eigenvalue of J is greater than λ_{max}, as any eigenvalue less than λ_{max}

would be negative and thus have magnitude greater than λ_{max}. As a result, since eigenvalue λ in J corresponds to eigenvalue $\lambda - \lambda_{max}$ in J', it follows that all eigenvalues of J' must be positive, and consequently that the eigenvalue of J' with greatest magnitude will correspond to the greatest eigenvalue of J. Thus, applying the power method to J' and adding λ_{max} to the eigenvalue obtained from it should give the greatest eigenvalue of J. If this eigenvalue has negative real part, then all the eigenvalues of J must have negative real parts.

3.6.3 Theoretical and Empirical Comparisons of Strategies

The number of iterations required for the power method to converge scales exponentially with the ratio between the eigenvalue of second greatest magnitude and the eigenvalue of greatest magnitude. In other words, when the eigenvalue of greatest magnitude is much larger than the next eigenvalue in this sequence, the power method converges in exponentially fewer iterations. Assuming, for simplicity of calculation alone, a geometric distribution of eigenvalues of J for a randomly sampled genetic network, it follows that roughly the same number of iterations of the power method will be required for each eigenvalue uncovered.

However, the same cannot be said for the computation of the eigenvalues of J'. To see this, consider a network with n nodes, with spectrum $\{-c^i,\ 0 \leq i < n,\ 0 < c < 1\}$. The number of iterations of the power method expended per eigenvalue of J will be $\log_c \epsilon$ for some small but positive ϵ, and thus the total number of matrix multiplications and normalizations will be $n \log_c \epsilon$.

Now consider computing the spectrum bounds for the same network. The initial computation of λ_{max} will require the same $\log_c \epsilon$ iterations of the power method.

However, the spectrum of J' is now $\{1 - c^i,\ 0 \leq i < n,\ 0 < c < 1\}$, and the ratio between the two eigenvalues with greatest magnitude is

$$\frac{1 - c^{n-2}}{1 - c^{n-1}} \approx \frac{1}{1 + c} \approx 1 - c$$

As such, the second application of the power method will take $\log_{1-c} \epsilon$, which scales quite poorly for small c.

In practice, the extra factor of n in the runtime for the full spectrum computation appears to overwhelm this ratio effect, and we observe that the spectrum bounding method runs approximately three times faster than the full spectrum computation method.

4 Sampling Genetic Networks

Even with the restricted space of genetic networks we are considering, there are an infinite number of such networks and no obvious way to select representative graphs uniformly from this space. Nonetheless, we designed a scheme by which to sample graphs from this space in a systematic way. At a high level, for each sample, we specify the number of nodes in our graph, select which edges to include, and finally specify the parameters for these edges.

4.1 Variable Hyperparameters

While there are many random graph models that capture various desirable properties, it is difficult for us to know what invariants a biological network must respect, if any. However, because of the localized manner in which mutations arise in evolutionary processes, we assume that any such invariant properties in our graph sampling must also be local.

To give a concrete example of such a property, we choose not to fix the total number of edges in our graph, but we are comfortable concentrating the degree of any given node around a particular value. Considerations like these lead, conveniently, to natural and independent sampling of per-edge properties in our graph, as we will describe later. For the moment, we will instead discuss the various local invariants that we specify as hyperparameters of our sampling model.

4.1.1 Minimum In-Degree of Nodes

In the model that we have described, the concentration of a node is only directed affected by the operons for which it is a product, i.e. the edges for which it is the sink node. Consequently, any node without an incident edge would never be produced and thus would not contribute at all to the dynamics of the network. It would instead serve as a redundant way of expressing the same network without the noninteracting node.

To avoid this redundancy, we decided to require that each node have at least one incident edge. For full generality, we also allowed for the specification of the minimum number of incident edges per node as a hyperparameter of the model, but we ultimately did not experiment with varying this hyperparameter.

4.1.2 Maximum In-Degree of Nodes

In a similar vein, we wanted to be able to actively sample from sparse graphs and dense graphs and compare results between the two sets. To allow for this and similar experiments to be conducted, we introduced the maximum number of incident edges per node as a hyperparameter that the user could specify.

4.1.3 Dissociation Constant Distribution

Following the paradigm of local sampling per edge, we needed to select a way by which the user could specify the distribution from which per-edge dissociation constants should be drawn. Under a physically motivated model for the inter-edge variance in dissociation constants, each promoter was originally identical. Subsequent mutations in promoter sequence and protein sequences lead to variations in binding energy and specificity per edge. As per this model, binding energies would themselves follow a Gaussian

distribution, the result of applying the central limit theorem to independent mutations in a given promoter sequence. Exponentiating this random variable leads to a log-normal distribution. In the limit of long time-scale evolution, the variance in these binding energies gets large and the log-normal distribution tends towards a very broad uniform distribution. Thus, inspired by this model and the simplicity of specification, we decided to introduce a hyperparameter in our model, specifying the maximum value of a per-edge dissociation constant, and to sample all dissociation constants from the uniform distribution between zero and this maximum value.

4.1.4 Expression Rate Distribution

We described earlier how the expression rates in our models are a combination of probabilities of polymerase recruitment, transcription times, and translation times. Since the latter two are roughly constant for a fixed length of protein, this combined parameter should follow a log-normal distribution for the probability of recruitment, and a somewhat tightly peaked Gaussian distributions for the length of the protein. Consequently, the combined parameter follows an approximately log-normal distribution. Under the same reasoning we employed for dissociation constants, we choose to specify this as a bounded uniform distribution. We introduced a hyperparameter to our model to specify the maximum possible expression rate. Each expression rate is then sampled from the uniform distribution between zero and this maximum value.

We note that as specified, the expression rates for both bound and unbound promoters are sampled from the same distribution. While this is in fact the default in our model, we understand that in most cases this is not empirically representative, and so we correct for this assumption with another hyperparameter, as specified in Section 4.1.5.

4.1.5 Ratio between Leak Expression and Full Expression

As we mentioned in Section 3.3.2, activating units have higher expression rates when bound than when unbound, while repressing units have higher expression rates when unbound than when bound. Since all expression rates in our model are initially sampled from the same uniform distribution, we decided to enforce the ordering of expression rates per operon by scaling the appropriate expression rate for each operon down by a constant factor. That factor, i.e. the ratio between the average leak expression and the average full expression of an operon, is specified in our model as a hyperparameter.

While this does not guarantee the appropriate ordering of expression rates in each operon, it does greatly increase the probability that the ordering is correct, i.e. a number sampled from the uniform distribution between 0 and 1 is very unlikely to be larger than a number sampled from the uniform distribution between 0 and 100.

4.1.6 Ratio of Activating and Repressing Units

The hyperparameter described in Section 4.1.5 allows for relative scaling of expression rates, but it also presupposes a labeling over edges, wherein each edge is designated either an activator or a repressor. By default, these labels are assigned at random, with both labels being equally likely. However, it is quite possible that networks with different ratios of activating and repressing operons have more or less modifiable equilibria. For this reason, we introduced the ratio of probabilities for assigning each label as a model hyperparameter.

4.2 Sampling Algorithm

For a fixed number of nodes n, we sampled the edges of the graph by iterating through the nodes. For each node, we would then select the incident edges to include. We wished to select these edges by independently including or discarding each possible incident edge to the node with probability 0.5. However, because of our hyperparameters constraining the minimum and maximum number of incident edges per node, we couldn't sample the edges for inclusion independently in this way. Instead, we decided to first sample the number of edges to be selected incident to the node. To respect the restricted distribution over this number of edges, we computed the relative probabilities as

$$\binom{n-1}{k}$$

for potential number of incident edges k. The rationale for this value is that there are $n-1$ possible incident edges to the node (we disallow edges from a node to itself) and k edges to select.

Normalizing this distribution for k between the minimum and maximum number of incident edges, we obtain a discrete probability distribution for k and draw a sample from it. For this value of k, we then select k incident edges uniformly at random. We do this iteratively, by selecting an incident edge at random and including it with probability $\frac{k'}{n'}$, where k' (initially k) is the number of remaining incident edges that need to be included and n' (initially $n-1$) is the number of incident edges that have yet to be included or discarded.

When an edge is selected, the dissociation constant and two expression rates are drawn from their respective uniform distributions. The edge is then labeled either an activator or repressor, and the scaling factor is applied to the appropriate expression rate.

This entire process is repeated for all selected incident edges, and then for all nodes in the graph. The result is stored in a three dimensional tensor for subsequent computations.

4.3 Optional Master Switch Node

4.3.1 Motivation

The sampling model we have described selects graphs in which each node will have, across all sampled graphs, the same distribution of properties. While this symmetry is unassuming in some ways, it also often unrealistic. In many networks, there is at least one protein that serves as a master regulator or switch. This protein both controls and is controlled by most, if not all, of the other proteins in the network. In such networks, it is the level of the master regulator that we would ultimately like to modify, and thus it is worth introducing a slightly modified algorithm to sample from the space of such networks.

4.3.2 Modified Sampling Algorithm

The modified sampling algorithm first designates the master switch node and includes all $2(n-1)$ edges (incident and exiting) between the master switch node and all other nodes. The two edges between the master switch node and any given other node in the graph are required to have identical labels, and the label for each other node is selected randomly, with probability 0.5, to be either activator or repressor. The subgraph induced by all non-master-switch nodes is then sampled according to the original model on $n-1$ nodes, with the maximum and minimum in-degree per node each reduced by one to account for the edges to and from the master switch node. The dissociation constants and expression rates are all selected as in the original model.

5 Network Modifications

This section will describe the three categories of network modifications we examined in the course of this work. For each category, we provide the intuition underlying its use, the mathematical formalism required to include the modification in the dynamical equations for a network, and finally a theoretical basis for determining the impact of the modification on a fully characterized network.

5.1 Constant Expression

5.1.1 Motivation

This type of modification involves adding an operon that expresses a single protein in the network at a fixed rate. It requires a constitutive promoter, i.e. one that doesn't bind any transcription factor but nonetheless recruits polymerases at a non-negligible rate. In some sense, increasing the level of a protein in a genetic network is the simplest modification one could make. In most cases, the effect this has on the protein being directly modified will be straightforward: the concentration of the protein will go up by the expected amount. In some cases, a strong feedback loop will cause the protein level to increase drastically, or even fall as a threshold is passed and a negative feedback loop activated. For small increases in expression, however, such drastic changes will be unlikely to occur. The more interesting category of change is, unsurprisingly, the effect this modification has on the concentrations of other proteins in the network. For that reason, this type of modification constitutes the easiest way to measure how the concentrations of different proteins in the network depend on each other. This modification is characterized

by the choice of protein being overexpressed and the rate at which the added operon is overexpressing the protein.

5.1.2 Formalism

Let x be the vector of concentrations of proteins in the genetic network, labeled in some canonical order. Using earlier notation and definitions, we have that

$$\frac{dx}{dt} = G(x)$$

in the absence of network modifications. Now, let r be the vector of overexpression rates introduced as part of this modification. While the experiments we conducted only involved r having a single non-zero component, specifying it as a full vector allows for generalization and also simplifies our notation. In this specification, r_i is the rate at which our added operons are expressing protein i in the network. With this modification, the new dynamics are simply

$$\frac{dx}{dt} = G(x) + r$$

5.1.3 Calculation of Theoretical Impact

Let y be the solution for our dynamics, i.e. $G(y) + r = 0$. If we vary r, we expect y to change as well, and we would like to understand this dependency. We do this by taking a total derivative of this equation with respect to r:

$$\frac{d}{dr}\left(G(y) + r\right) = J(y)\frac{dy}{dr} + I = 0$$

where J, as before, is the Jacobian of G with respect to x, evaluated at y. Note that $\frac{dy}{dr}$ is itself a matrix, as both y and r are vectors, i.e. the derivative with respect to r is a gradient operator, applied element-wise to its operand. Simplifying gives us that

$$\frac{dy}{dr} = -\left(J(y)\right)^{-1}$$

Using this with a fully characterized network, we can always compute the directional derivative due to a particular change in r by right multiplying the matrix $\frac{dy}{dr}$ with the change in r. Furthermore, this equation also provides a numerical algorithm to achieve a desired change in y. We need merely approximate the change in y linearly as

$$\Delta y = \left(\frac{dy}{dr}\right)\Delta r$$

and solve for the Δr that will give the desired Δy. By taking measured steps in the direction of Δr and iterating, we will converge to the new value of r necessary to achieve a desired value of y.

5.2 Positive Self-Regulating Feedback Loop

5.2.1 Motivation

This category of network modification involves adding an operon that acts as a loop edge, i.e. an operon that controls and is controlled by the same protein. We fix these operons to be activators with unbound expression rates of essentially zero. Furthermore, we fix their dissociation constants and bound expression rates to be very large, such that the net expression rate from the operon is given by

$$\frac{[A]}{K_D + [A]} R_{operon} \approx \frac{[A]}{K_D} R_{operon} \approx R_A[A]$$

i.e. the expression rate of the operon is directly proportional to the concentration of protein being controlled.

This modification is very similar to the previous one in that it serves primarily to change the concentration of a single protein in the network, with secondary charges permeating through the network from there. The main difference is that the feedback allows for small changes in this R to lead to much bigger changes in equilibria of the system. Intuitively, where the previous modification led to linear changes in $[A]$, this modification will lead to exponential changes in $[A]$.

5.2.2 Formalism

Much like with the last modification, we will express our formalism more generally, to represent the simultaneous addition of multiple operons with feedback. Using the same notation as before and letting R be the constants of proportionality as expressed in Section 5.2.1, we have that the dynamics of the modified system are given by

$$\frac{dx}{dt} = G(x) + R \odot x$$

where $R \odot x$ denotes the element-wise product between these two vectors.

5.2.3 Calculation of Theoretical Impact

Again, using the same notation as in Section 5.1.3, we are interested in computing $\frac{dy}{dR}$ and we will do this by taking a full derivative of the equilibrium condition equation with respect to R. This is shown below:

$$G(y) + R \odot y = 0$$

$$\frac{d}{dR}\left(G(y) + R \odot y\right) = J\frac{dy}{dR} + Q = 0$$

where Q is a square matrix that satisfies

$$Q_{ij} = \delta_{ij} y_i + R_i \left(\frac{dy}{dR}\right)_{ij}$$

Letting S_R denote the diagonal matrix that satisfies $\left(S_R\right)_{ii} = R_i$ and S_y denote the diagonal matrix that satisfies $\left(S_y\right)_{ii} = y_i$, we can simplify our result as follows:

$$J\frac{dy}{dR} + Q = \left(J + S_R\right)\frac{dy}{dR} + S_y = 0$$

which can be solved for $\frac{dy}{dR}$, giving

$$\frac{dy}{dR} = -\left(J(y) + S_R(R)\right)^{-1} S_y(y)$$

While this computation requires slightly more time than the equivalent computation for the first network modification that we described, the overall time complexity is the same as before. Using this, we can again solve for $\frac{dy}{dR}$ and also determine the change in R required to produce a desired change in y.

5.3 Modifying Dissociation Constants

5.3.1 Motivation

The first two categories of modification that we described, while technically alterations to the network, were really just ways to persistently perturb a single node in the network.

This modification focuses instead on perturbing the edges to the network. We had a choice, when considering this perturbation, of modifying either the dissociation constants or the expression rates of a selected operon. We decided to modify only the dissociation constants because this modification is easier to do experimentally than a modification of the expression rates. This is because measurements of the expression rates require flooding the system with or draining the system of transcription factor (to force the promoter to be consistently bound or unbound respectively) and then measuring the expression rate, a process which is both slower and less amenable to high throughput methods than an SPR experiment to measure dissociation constants of modified promoters.

5.3.2 Formalism

Unlike with the previous two categories of modification, the form of the system's dynamic equations isn't changed. This is because no new edges or nodes are being added, and instead old edges are simply being modified. However, while the form of the dynamics may not change, the way the dynamics depend on the modifications is actually more involved than with the previous two categories of modification. In particular, the Jacobian of the main graph, which remained unchanged during the previous modifications, is now a function of the variable being modified.

To be concrete, let D be the matrix denoting all dissociation constants for the system, such that D_{ij} is the dissociation constant for the operon in which protein j binds to the promoter to control the expression of protein i. An infinite element of D corresponds to a nonexistent edge in the network. Furthermore, let F be the matrix denoting changes in expression upon promoter binding, such that F_{ij} is the increase in

expression rate due to protein j binding to the promoter in the operon that controls the expression of protein i. With this notation, we have that the Jacobian of the system dynamics is given by

$$J_{ij} = \frac{D_{ij}F_{ij}}{(D_{ij} + [j])^2} - \delta_{ij}$$

5.3.3 Calculation of Theoretical Impact

Since the dynamics themselves haven't changed in form, our equilibrium conditions have reverted to their original form, i.e. $G(y) = 0$. Taking a total derivative of this equation with respect to D gives

$$\frac{\partial G}{\partial y}\frac{dy}{dD} + \frac{\partial G}{\partial D} = 0$$

where the first term is a tensor contraction and the second term is a three dimensional tensor. We are again looking to solve for $\frac{dy}{dD}$, and while higher-order tensor contractions are more time consuming to invert than matrix multiplication, they can be done in a similar way. We will not be able to express $\frac{dy}{dD}$ in closed form as a result, but we do note that $\frac{\partial G}{\partial y} = J$ and $\left(\frac{\partial G}{\partial D}\right)_{ikj} = -\frac{\delta_{ik}[j]F_{ij}}{(D_{ij} + [j])^2}$ are easily computable.

6 Metrics

As we experiment with applying different network modifications and varying hyperparameters, both in network sampling and in the modifications themselves, we need good ways to examine and understand the implications of our experiments. In particular, we want to understand two things. First, if our sampling is sufficient, we would like to measure in some way the effectiveness of our various modifications. Secondly, we wish to know whether there are characteristics of different nodes that make them better or worse as candidates at which to target modifications. The two metrics we propose, sensitivity and relative sensitivity, address the first and second questions respectively. These metrics must be customized slightly based on the type of modification we are considering. In this section, we describe both metrics and also describe how they are used to organize our data.

6.1 Sensitivity

A modification is effective if it allows one to elicit a small change in the concentration of some target concentration via a comparatively small change in some hyperparameter of the modification. This definition is motivated by the power of the modification, for if a slight change in the modification yields a much larger change in the equilibrium concentrations, then the modification is very effective at controlling the system.

We would like our definition of sensitivity to capture this notion of efficacy. One choice for this definition that would capture the correct intuition at a local level would be

$$\frac{dy_i}{df(\theta)}$$

where y, as in Section 5, is the vector of equilibrium concentrations for the statement, θ is some hyperparameter of the modification, and f is some function of θ. In particular, f should be chosen such that $df = \frac{df}{d\theta} d\theta$ reflects the experimental effort required to change θ by $d\theta$. The choice of f thus depends on the category of modification under consideration, and we will justify the choices we made in Sections 6.1.1 - 6.1.3.

While this definition is certainly acceptable, it is less than optimal. The locality of the derivative is a disadvantage in the context we are considering. Network modifications will typically not be arbitrarily small, nor do we expect the response of the system to scale linearly and continuously with the magnitude of the modification in general. For instance, a modification that initially has little to no effect on the system but, after a certain threshold, induces a large change in y is still considered potent. This example would suggest that we should simply use the maximum achievable change in y as our sensitivity metric, but while such a choice would represent the efficacy of discontinuous transfer functions appropriately, it would fail to capture the power of modifications that do affect y continuously. For this reason, we decide to use the following metric as our measure of sensitivity:

$$S = \max_{\Delta\theta} \frac{\Delta y_i}{\Delta f(\theta)}$$

Computing this metric involves scanning over different values of θ. Since

$$\frac{dy_i}{df(\theta)} = \lim_{\Delta\theta \to 0} \frac{\Delta y_i}{\Delta f(\theta)}$$

it follows that $S \geq \frac{dy_i}{df(\theta)}$. Thus, in cases where this derivative is relatively constant as a function of θ, i.e. in loosely regulated systems, $S \approx \frac{dy_i}{df(\theta)}$, and in cases where y only changes after θ (starting at θ_0) crosses a threshold (θ_T), i.e. in tightly regulated systems, $S \approx \frac{y_i(\theta_T) - y_i(\theta_0)}{f(\theta_T) - f(\theta_0)}$.

6.1.1 Constant Expression

An element from this category of modification is characterized by two attributes: the protein being expressed, and the rate at which it is expressed. Varying this expression rate can be done in two ways - by changing the DNA sequence responsible for recruiting polymerases and initiating transcription, or by changing the number of copies of the operon included as part of the modification. While changing the recruiting sequence might be more effective for achieving a particular rate of expression, it is far less automatable and requires some analysis of the sequence. Changing copy number, on the other hand, can be done relatively quickly, and by picking the initial single operon to have a suitably small expression rate, the copy number can be used to scan over possible expression rates with as fine resolution as may be desired. Thus, for the purposes of our work, we choose θ to be the expression rate r_j for whichever protein j is being expressed, and we choose $f(\theta) = \theta$.

6.1.2 Positive Self-Regulating Feedback Loop

Although the action of operons from this category of modification is dissimilar to the action of operons from the previous category, the practical implementation is quite similar. In particular, an element from this category of modification is also characterized by two attributes: the protein being expressed, and the constant of proportionality between the concentration of the protein and the rate at which it is expressed. We described earlier how a linear relation between concentration and expression rate is achieved by having a very high dissociation constant (i.e. a very weakly binding promoter) and a comparatively high expression rate. Modifying either of these two attributes directly is difficult. Any change to the promoter will either be insignificant or will disrupt the linear relation between concentration and expression rate. On the other hand, the expression rate is already large, and so making it larger by increasing the probability of polymerase recruitment is impractical. Thus, for this category, even more so than for the last, adjusting behavior using copy number is essential. For this reason, we choose θ to be the constant of proportionality R_j for whichever protein j is being expressed, and we choose $f(\theta) = \theta$.

6.1.3 Dissociation Constants

Changing dissociation constants of existing operons in the network involves altering promoter sequences in an effort to modify binding energies. It is difficult to predict, *a priori*, what binding energies are achievable, in part because it is difficult to specify precisely how much we are willing to change the promoter sequence. Nonetheless, when altering the dissociation constant of the operon controlled by protein j and controlling

protein i, $\theta = D_{ij}$. In order to determine the best choice of function $f(\theta)$, let us consider a simple model of how modifications to promoter sequence will affect the dissociation constant. Assume that, all else being equal, each base alteration changes the binding energy of the promoter and protein j by a fixed amount ϵ in expectation. From statistical mechanics, we know that the dissociation constant will change by a multiplicative factor of $e^{\beta\epsilon}$ for temperature-dependent constant β. Thus, when k bases are altered, the dissociation constant is scaled by a factor of $e^{\beta k\epsilon}$, i.e. $D'_{ij} = D_{ij}e^{\beta k\epsilon}$ or, alternately,

$$\ln D'_{ij} = \ln D_{ij} + \beta k\epsilon \implies k \propto \Delta\left(\ln D_{ij}\right)$$

Since k is the parameter an experimentalist would have the most control over, it follows that $f(\theta) \propto \ln\theta = \ln D_{ij}$ would be an appropriate choice for this category of modification.

6.2 Relative Sensitivity

In the event that all sensitivities are large, independent of the many hyperparameters we use to sample our networks and of the specific node or edge being modified, we can rest easy knowing that localized empirical modifications are almost certain to be effective. However, if this is not the case, i.e. if sensitivity depends in some way on these various hyperparameters, we would like to have a way to measure how effective a particular modification is on average, relative to the other modifications that could be made on the same network.

In particular, if a particular modification exhibits great sensitivity in its effect on y_i for a particular protein i, this may be attributable to the general effectiveness of the modification, but it might also be due to a high level of sensitivity of y_i to all

modifications. There is no well-defined way to compare our sensitivity metric across different categories of modification, but within a category, the values of the metric should be directly comparable. As such, we can obtain a measure of the portion of sensitivity due to the modification itself by dividing a sensitivity measure for y_i by the average sensitivity measure for y_i across all modifications of that type. We call this measure the relative sensitivity or adjusted sensitivity.

6.3. Distributions and Scatterplots

As we have mentioned, we wish to understand both how effective a given type of modification may be generally, and also how various hyperparameters associated with node and edge properties affect the likelihood of efficacy. These two goals require different methods of aggregating our sensitivity and relative sensitivity data. In order to describe these methods, we must first slightly clarify our terminology. When measuring the sensitivity of y_i with respect to parameter θ_k, indexed in some canonical way, we denote the measure with S_{ik}.

To collect data, we would, for a fixed category of modification and for each sampled network, randomly sample possible i and possible k, subject to the constraint that θ_k does not directly affect y_i. Thus, θ_k cannot be r_i, R_i or D_{ij} for any j. Having sampled these pairs, we would compute the sensitivity and relative sensitivity for each. We then plotted a probability distribution of the sensitivity over all collected samples. This frequency distribution indicates how likely that category of modification is to be effective, i.e. the more tightly concentrated that distribution is, the less effective the modification will be on average.

We also generated scatterplots wherein potentially relevant node and edge hyperparameters were used as x-coordinates and the relative sensitivities were used as y-coordinates. This allowed us to determine more easily which hyperparameters actually affected the sensitivity of a particular modification. With these two aggregation methods, we were able to make sense of the data we generated, and we will discuss some of these results in the next section.

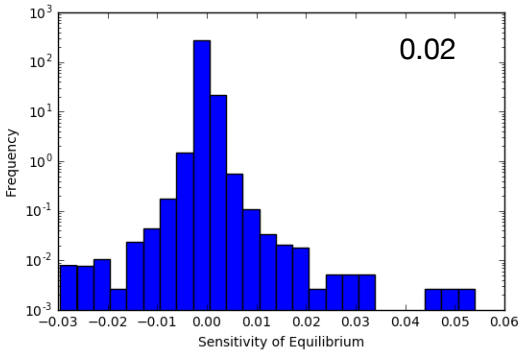
7 Results and Discussion

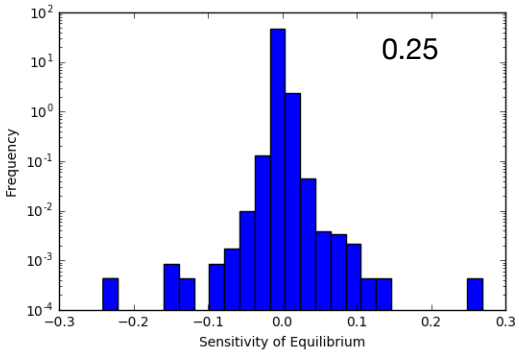
In this section, we'll present some of the data generated during our investigations. We'll explore how our two metrics were affected by model hyperparameters constraining operon parameters, different network modifications, and model hyperparameters constraining graph structure. In all of the the experiments we ran, we sampled networks with 20 regular nodes, with the master switch networks having a single additional master switch node. The choice of 20 as the size of our vertex set was somewhat arbitrary - we needed a number large enough that an experimentalist might plausibly wish to avoid from full characterization, but not so large that it would limit our own computational sampling.

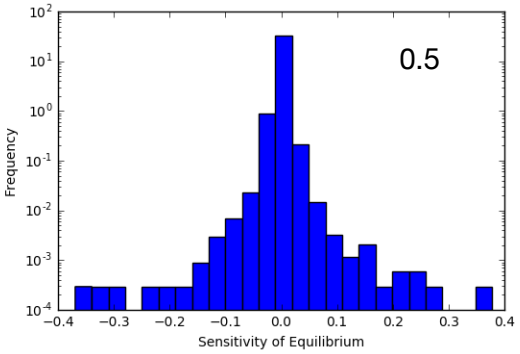
7.1 Network Modifications

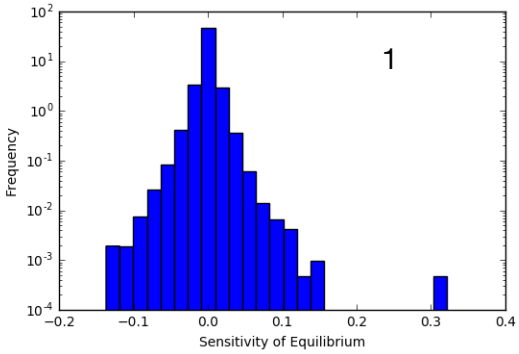
7.1.1 Constant Expression

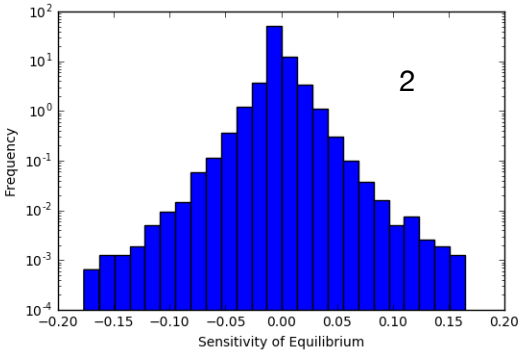
As mentioned earlier, the constant expression network modification is the most fine-grained of the three categories we are experimenting with, and consequently we decided to use it first to investigate how varying the hyperparameters for operon characteristics affects our sensitivity measure. To do this, we fixed the maximum in-degree of network nodes to be 10 and sampled over different ratios between the dissociation constant scale and the expression rate scale. We first did this for networks in which the leak expression rates were not scaled down and each operon was equally likely to be an activator or a repressor. The various sensitivity distributions produced are shown below in Figure 7-1.

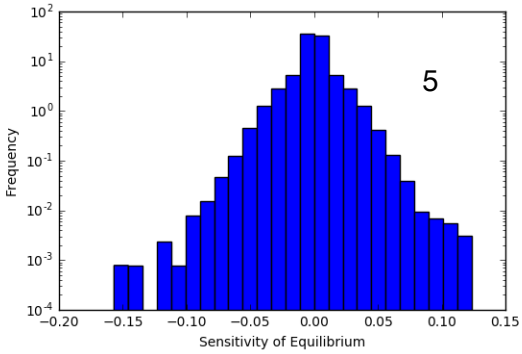

0.02
Frequency
Sensitivity of Equilibrium

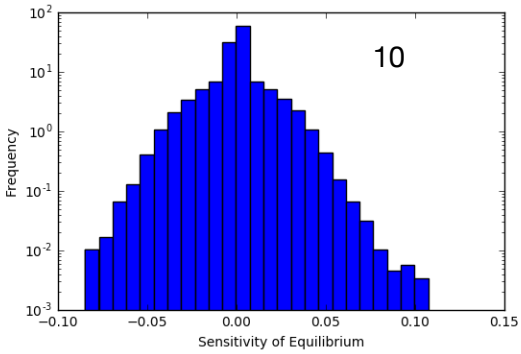

0.25
Frequency
Sensitivity of Equilibrium

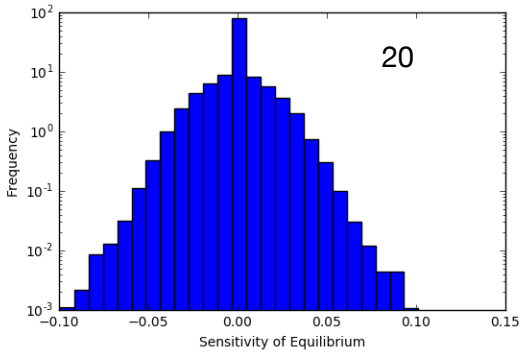

0.5
Frequency
Sensitivity of Equilibrium


1
Frequency
Sensitivity of Equilibrium


2
Frequency
Sensitivity of Equilibrium


5
Frequency
Sensitivity of Equilibrium


10
Frequency
Sensitivity of Equilibrium


20
Frequency
Sensitivity of Equilibrium

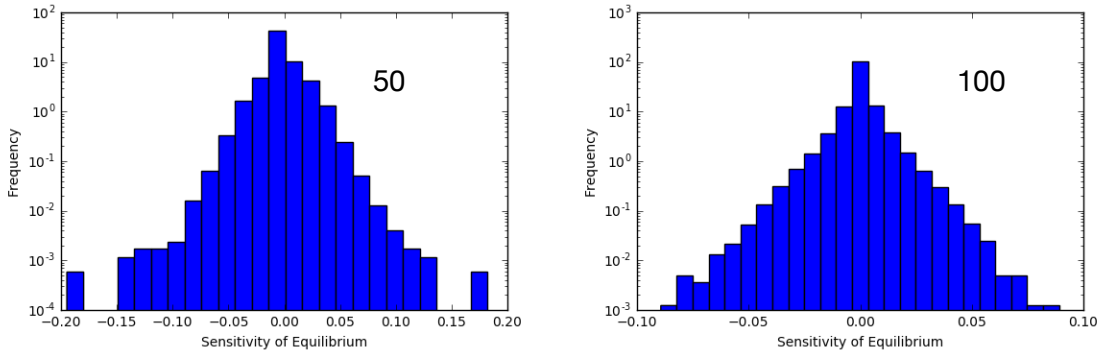


Figure 7-1. Sensitivity of Fully Random Networks under Constant Expression Modification. All sampled networks had 20 nodes, maximum in-degree of 10, equal probabilities assigned to activator and repressor operons, and leak expression rates unscaled with respect to full expression rates. Each distribution is labeled with the ratio of dissociation constant scale to expression rate scale used to generate the component data.

It is worth noting that in all distribution graphs shown here, the frequencies are plotted on a log scale. The decay away from a sensitivity of 0 in either direction is thus much steeper than a Gaussian decay. In the distributions from Figure 7-1, the shapes of the curves look similar across different values of the dissociation constant-expression rate scale ratio, but the width of the distribution varies with this parameter. It appears to be widest between the values of 0.25 and 2, and this behavior is reproduced when scaling of leak expression is included. In Figures 7-2, 7-3, and 7-4 below, we show the corresponding distributions with leak expression scaled down by a factor of 100, while also varying the relative probabilities of seeing activating and repressing operons.

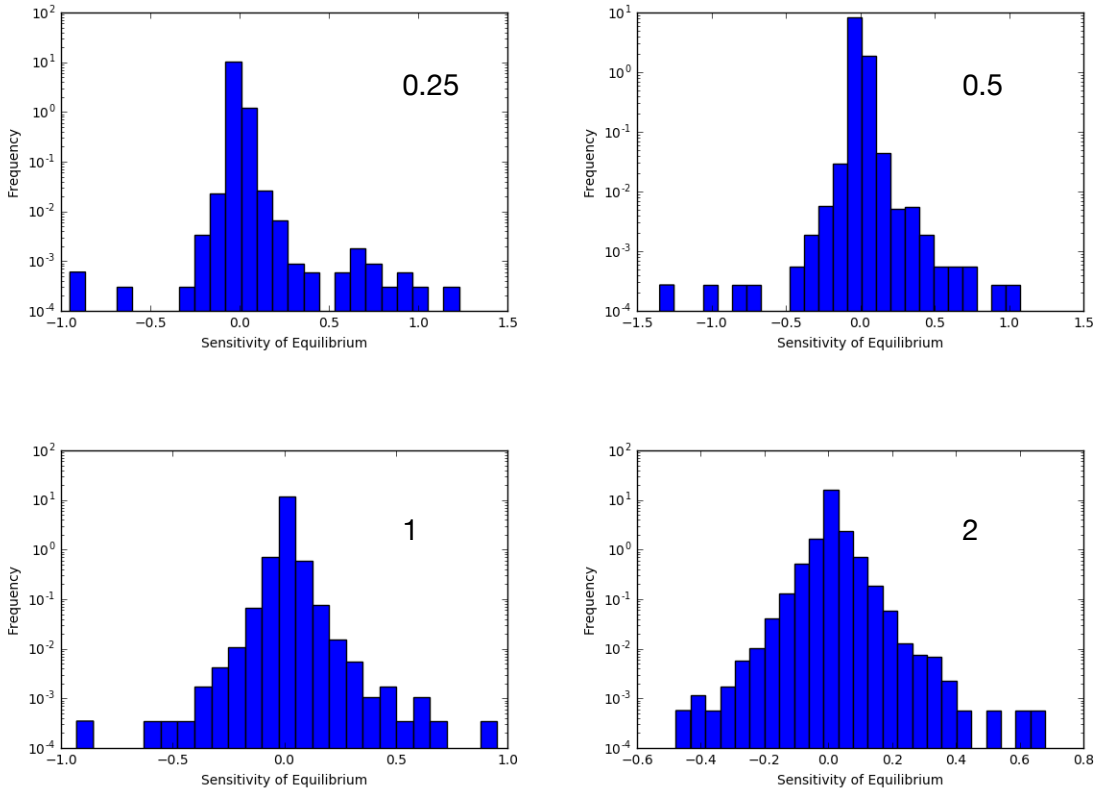


Figure 7-2. Sensitivity of Fully Random Networks under Constant Expression Modification. All sampled networks had 20 nodes, maximum in-degree of 10, twice as much probability assigned to repressor as to activator operons, and leak expression rates scaled down by a factor of 100 with respect to full expression rates. Each distribution is labeled with the ratio of dissociation constant scale to expression rate scale used to generate the component data.

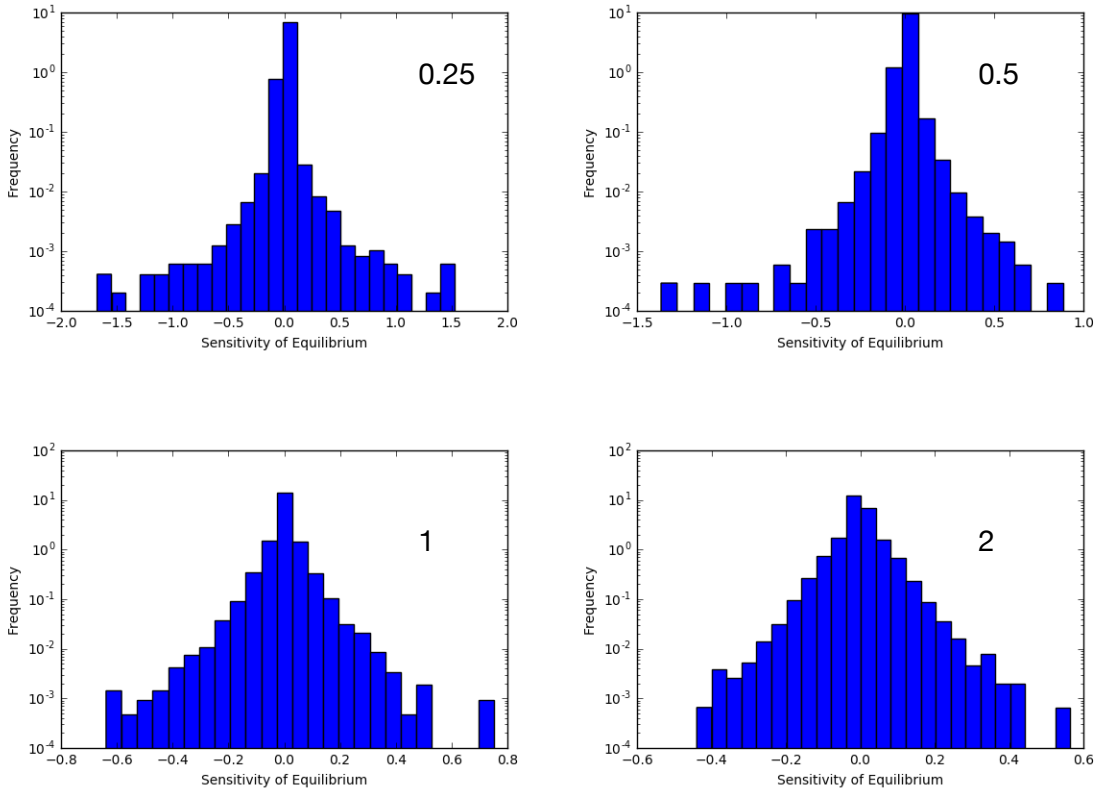


Figure 7-3. Sensitivity of Fully Random Networks under Constant Expression Modification. All sampled networks had 20 nodes, maximum in-degree of 10, equal probabilities assigned to activator and repressor operons, and leak expression rates scaled down by a factor of 100 with respect to full expression rates. Each distribution is labeled with the ratio of dissociation constant scale to expression rate scale used to generate the component data.

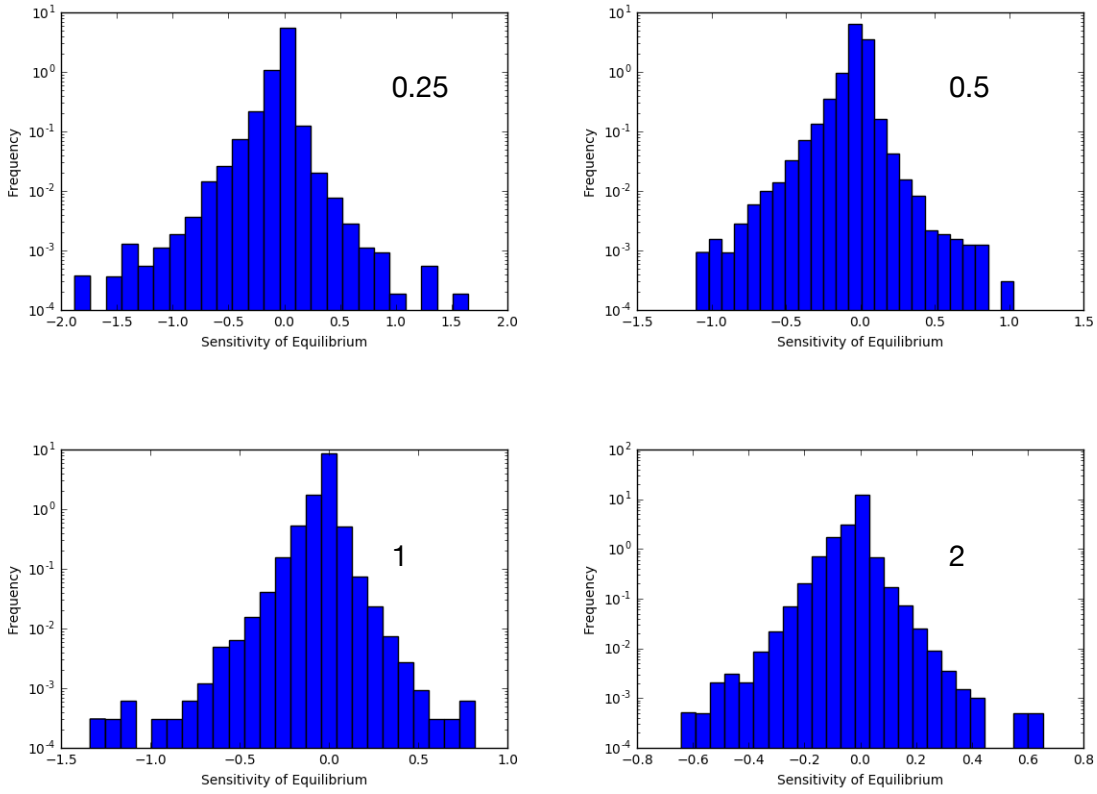


Figure 7-4. Sensitivity of Fully Random Networks under Constant Expression Modification. All sampled networks had 20 nodes, maximum in-degree of 10, ten times as much probability assigned to activator as to repressor operons, and leak expression rates scaled down by a factor of 100 with respect to full expression rates. Each distribution is labeled with the ratio of dissociation constant scale to expression rate scale used to generate the component data.

As is readily apparent, scaling down the leak expression leads to an approximately 10-fold increase in sensitivity on average, but does not seem to interact with the dependence on dissociation constant-expression rate scale ratio. Furthermore, increasing the probability of activator operons seems to only slightly increase the probability of observing intermediate values of the sensitivity. The preferred scale ratio of approximately 1 for operon parameters can be explained to some degree. The dissociation constant in some sense sets the threshold between "on" and "off" expression for a given operon. Thus, if expression rates are too far below this threshold, nothing will be produced, whereas if expression rates are too far above this threshold, production will always be saturated. Furthermore, the drastic effect of scaling down leak expression is very predictable, as without it, leak expression and full expression will be equal in expectation, and thus the operons themselves will often have little to no effect on the dynamics.

For the remainder of our presentation, we will only show data for which dissociation constants and expression rates were sampled at the same scale. Shown below in Figure 7-5 is the data under this condition collected for constant expression modifications. Also shown are the relations between two graph structure hyperparameters and the log relative sensitivity. These two parameters are the in-degree and out-degree of the node of the network to which the modification is applied. There are two main conclusions to be drawn from this data. First is that both parameters are not remotely sufficient for predicting relative sensitivity. They are accompanied by a great deal of noise and a relatively small level of correlation. The uncertainty in the relative sensitivity increases as in-degree is reduced, and is maximized when the out-degree is about half the

size of the vertex set, i.e. 10. The median sensitivity seems to be relatively unaffected by the in-degree of the node, and positively correlated with the out-degree of the node.

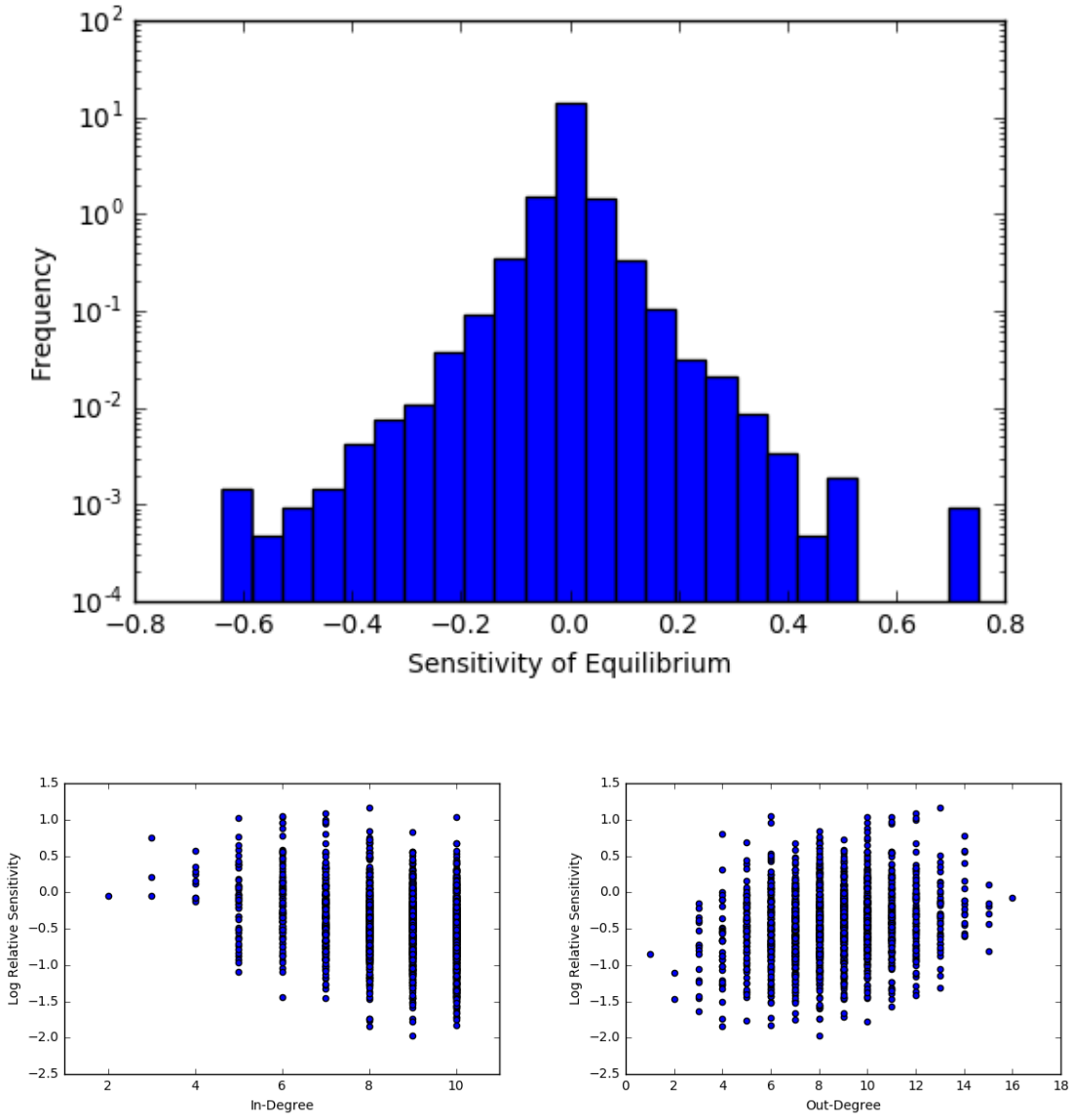


Figure 7-5. Benchmark Sensitivity Measures of Fully Random Networks under Constant Expression Modification. All sampled networks had 20 nodes, maximum in-degree of 10, dissociation constants and expression rates drawn uniformly on identical scales, equal probabilities assigned to activator and repressor operons, and leak expression rates scaled down by a factor of 100.

7.1.2 Positive Self-Regulating Feedback Loop

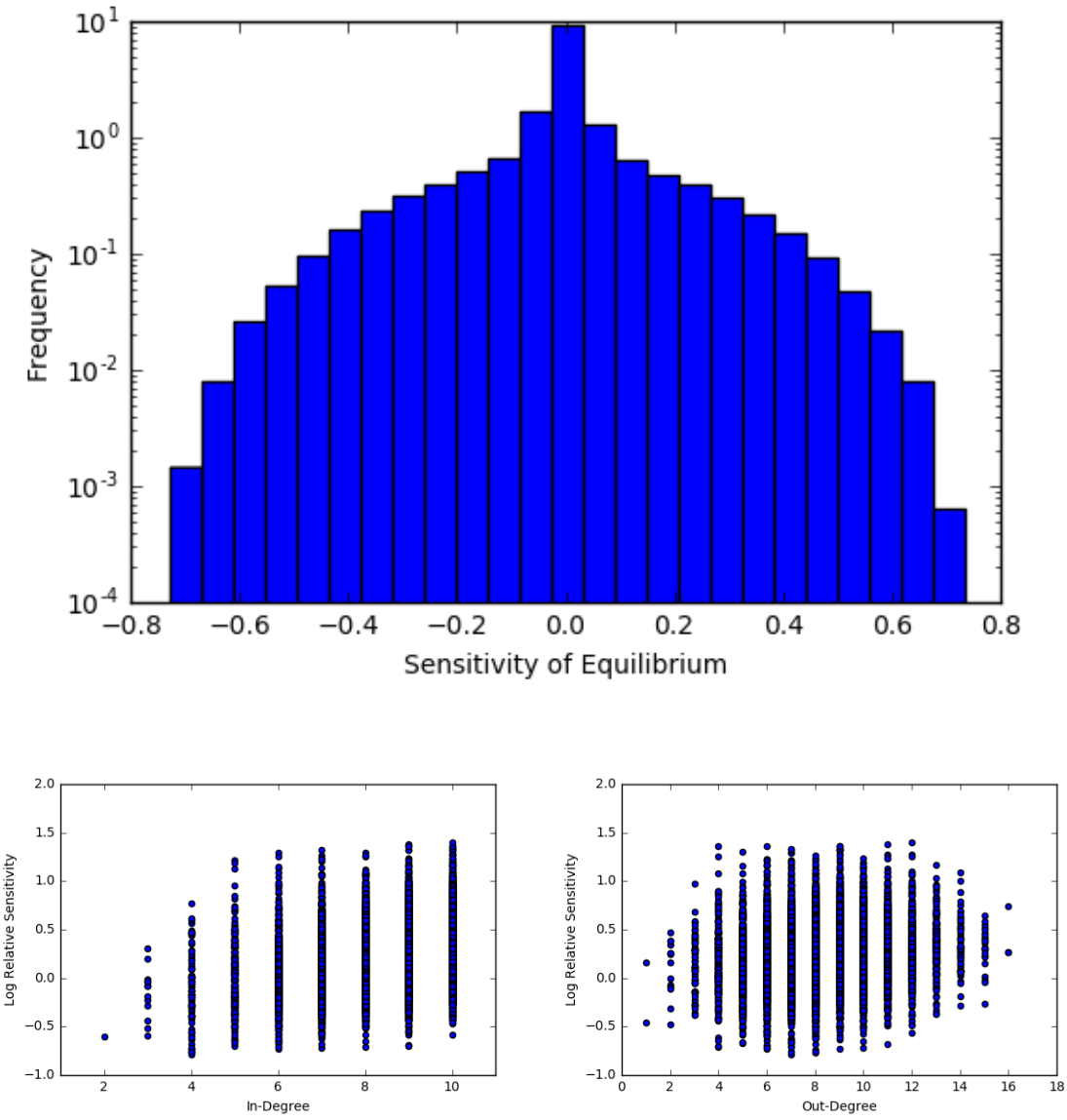


Figure 7-6. Benchmark Sensitivity Measures of Fully Random Networks under Positive Feedback Loop Modification. All sampled networks had 20 nodes, maximum in-degree of 10, dissociation constants and expression rates drawn uniformly on identical scales, equal probabilities assigned to activator and repressor operons, and leak expression rates scaled down by a factor of 100.

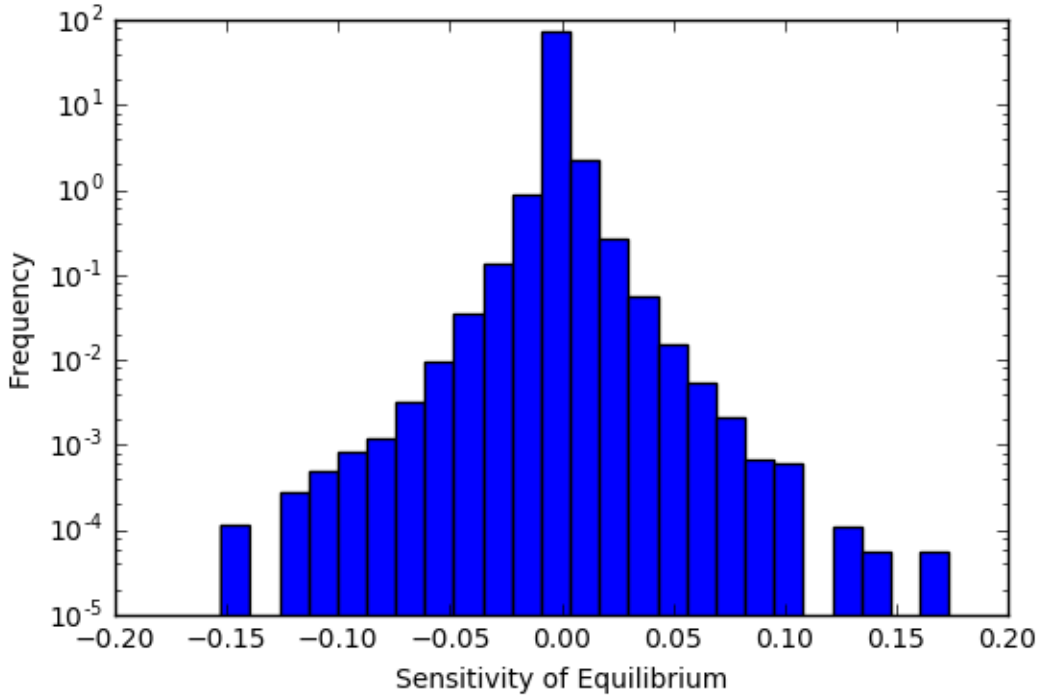
Shown above in Figure 7-6 is the performance of the positive feedback loop network modification. This category of modification was expected to be more performant, if only because it functions as an exponential version of the constant expression category. That said, it is worth noting that the improved performance does not manifest as increases in the max sensitivity observed, but rather as significant increases in the probabilities of observing the larger sensitivity values that were already being observed. Indeed, the distribution of sensitivities under this category of modification much more closely resembles a normal distribution (quadratic in log-scale), as opposed to the very tightly concentrated distributions observed under the constant expression modification.

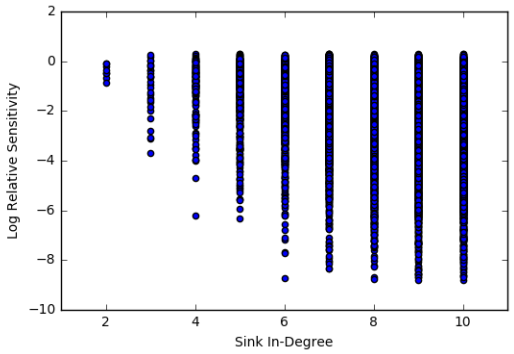
The relation between relative sensitivity and node out-degree for this modification is roughly the same as the corresponding relation for the constant expression modification, but the same cannot be said for the relation between relative sensitivity and node in-degree. Indeed, there is a slight positive correlation between node in-degree and the median value of sensitivity observed. As before, however, any correlation that we observe is greatly outweighed by the noise in the measurement. Nonetheless, it is worth noting that there may be a fundamental difference between how feedback and constant expression affect specific nodes as a function of the node's controllers.

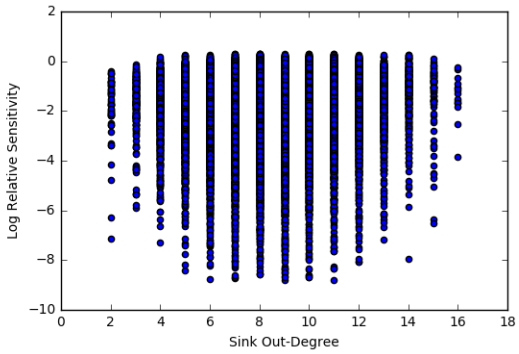
7.1.3 Dissociation Constants

Shown below in Figure 7-7 is the performance of the dissociation constant network modification. This category of modification cannot be compared as easily to the other two, as the sensitivity is being taken with respect to the effective number of basepairs altered, and not the change in dissociation constant directly. As such, the maximum sensitivity observed should not be contrasted with the corresponding value for the other

two distributions. That said, the shape of the distribution can be compared to the shapes observed under the other two modifications. The shape seen here much more closely resembles the fast decaying shape observed under the constant expression category of modification than it does the Gaussian (quadratic) shape observed under the positive feedback loop category of modification. As such, the data would suggest that these modifications are less effective than the positive feedback loop modifications.







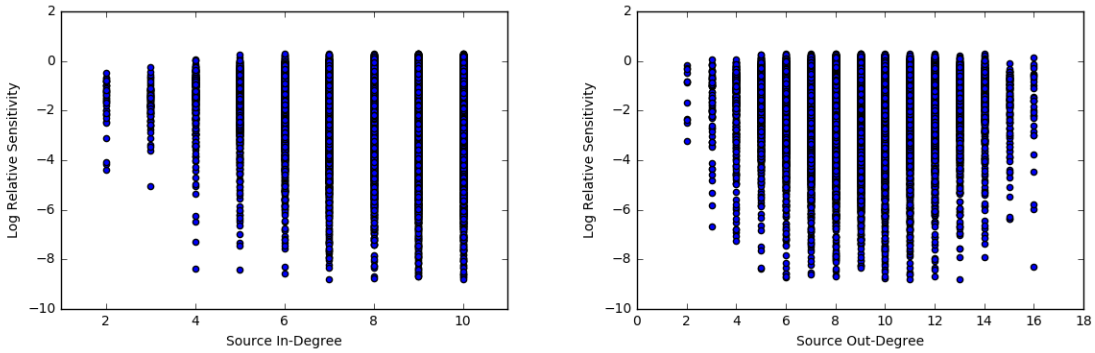


Figure 7-7. Benchmark Sensitivity Measures of Fully Random Networks under Dissociation Constant Modification. All sampled networks had 20 nodes, maximum in-degree of 10, dissociation constants and expression rates drawn uniformly on identical scales, equal probabilities assigned to activator and repressor operons, and leak expression rates scaled down by a factor of 100.

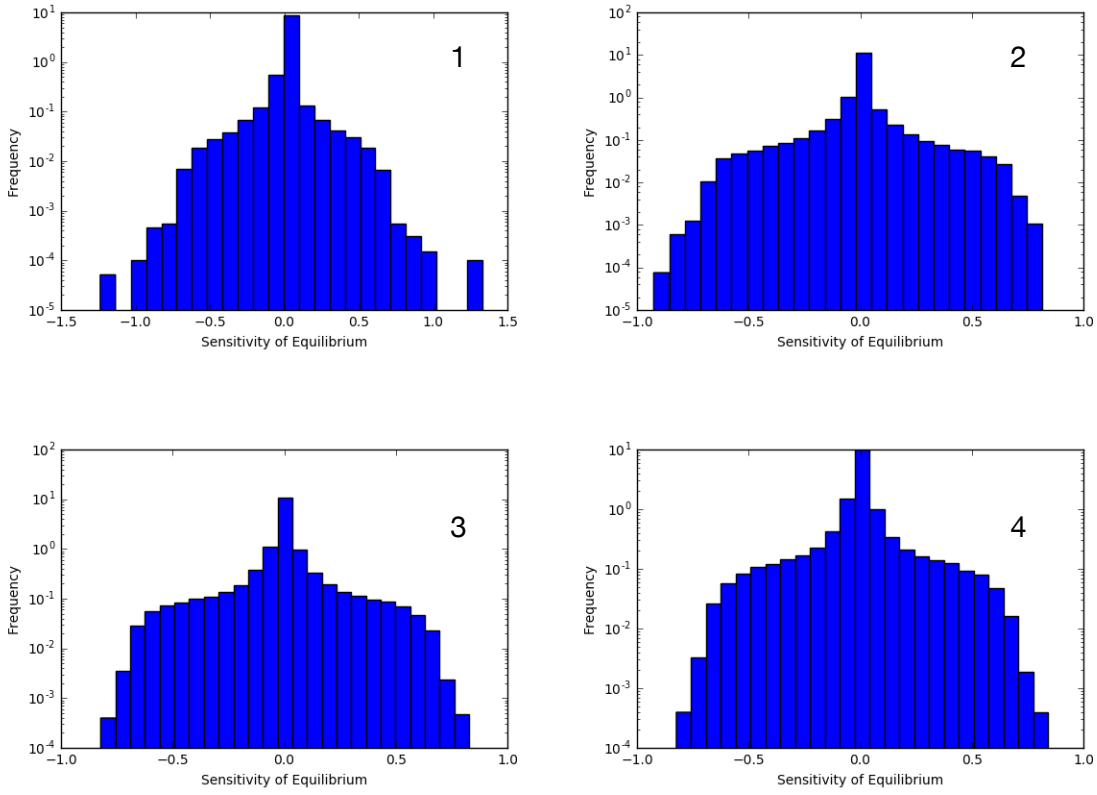
The justification for this, at a coarse level, is simply that the number of possible edges for which the dissociation constant could be modified scales with the square of the number of vertices, and thus if the number of impactful edges scales linearly with the number of impactful nodes, there is a fair chance that even highly effective modifications get pushed to the tail of the distribution by the increased chance of sampling ineffective modifications. It is also worth noting that the relative sensitivity measures in this figure are normalized with respect to the maximum sensitivity across the targeted node, as opposed to the average sensitivity used earlier. This was done to counteract the effect of ineffective edges, which will be much more prevalent in this scenario than ineffective nodes were under the previous two categories of modification. In these graphs, then, data concentrated at a higher value of log relative sensitivity suggests a distribution under which data with high relative sensitivity is more likely. Thus, for both sink and source

nodes of the edge being modified, a lower in-degree appears to be correlated with higher median relative sensitivity. The relation of relative sensitivity to sink and source out-degree, however, remains the same as was observed under both previous modifications.

7.2 Varying Graph Connectivity

In this section, we will explore the effects of varying the maximum in-degree of nodes in a network for both fully random networks and networks with a master switch. In both cases, the number of non-master-switch nodes will be 20. Both experiments were done using positive feedback loop modifications, simply because those were the most potent examples we had observed thus far, and so, we reasoned, were the most likely to show significant change as other hyperparameters of the model were perturbed.

7.2.1 Fully Random Networks



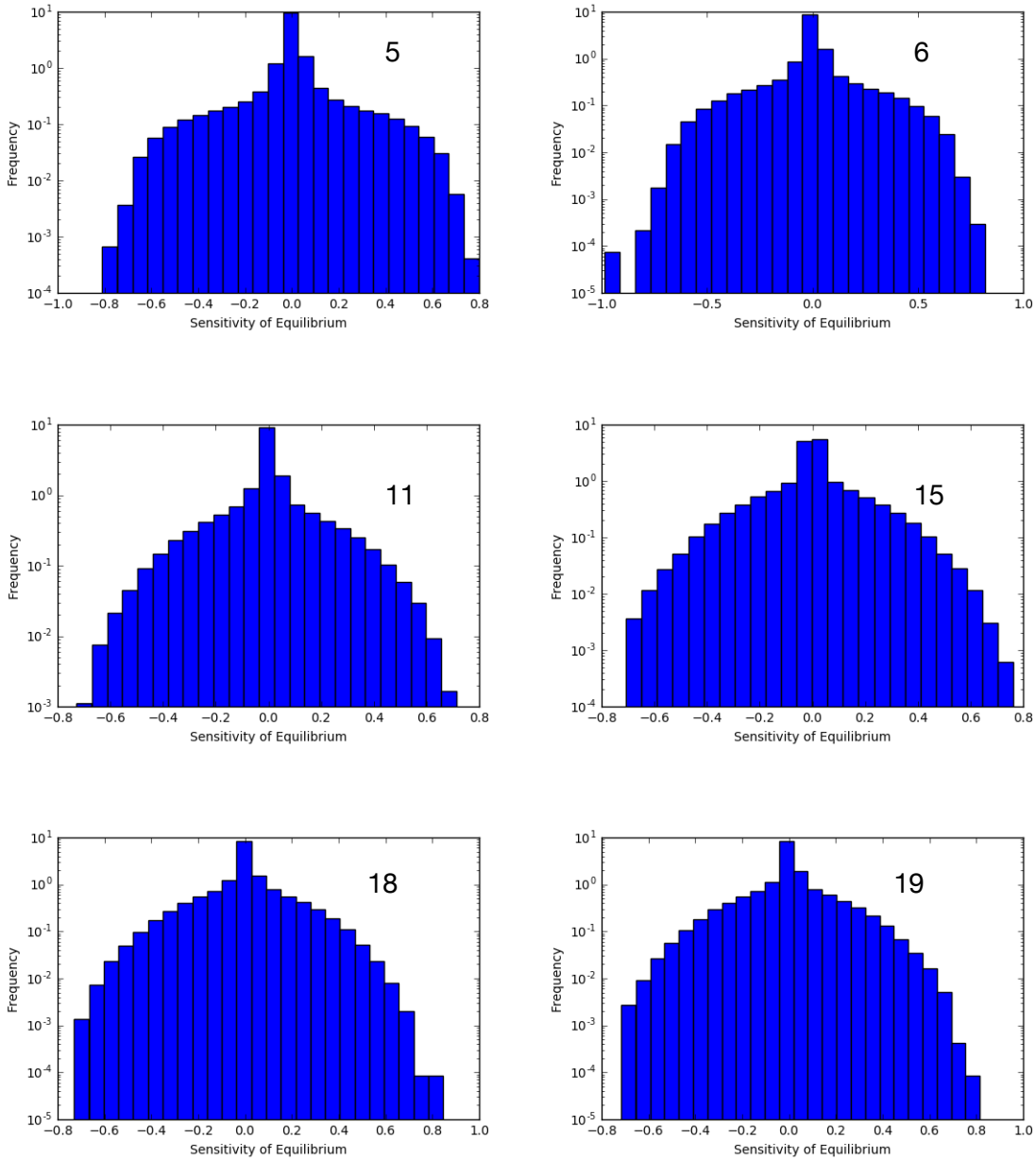
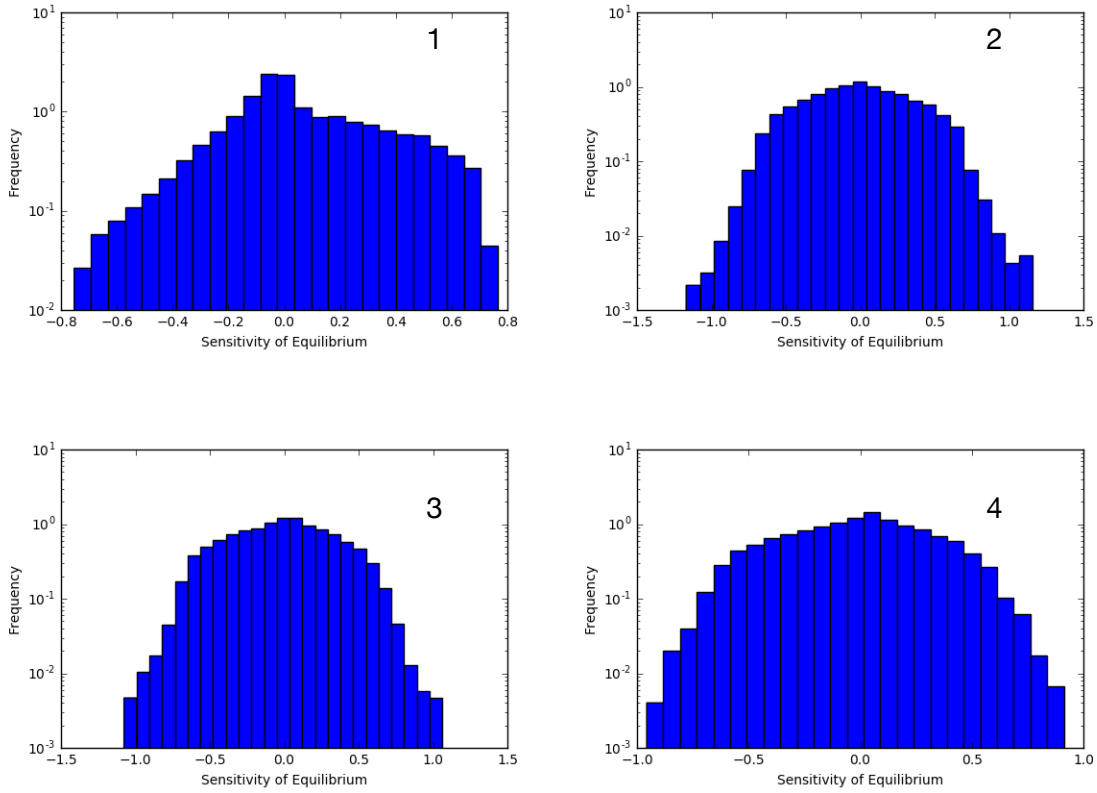


Figure 7-8. Sensitivity of Fully Random Networks under Positive Feedback Loop Modification. All sampled networks had 20 nodes, dissociation constants and expression rates drawn uniformly on identical scales, equal probabilities assigned to activator and repressor operons, and leak expression rates scaled down by a factor of 100. Each distribution is labeled with the maximum in-degree of nodes used to generate the component data.

As shown above in Figure 7-8, the coverage of the distribution of sensitivities decreases as the maximum in-degree of the network's nodes is slowly increased. That said, it is not a continuous change, i.e. the larger sensitivities observed when the maximum in-degree is one or two do not appear to fit the shape of the normal distribution we had observed earlier, and that shape is recovered only after the maximum in-degree has been raised enough to cause those higher sensitivity values to be lost. We also note that there continuous to be a large, discontinuous peak tightly concentrated at a sensitivity of zero.

7.2.2 Networks with a Master Switch Node



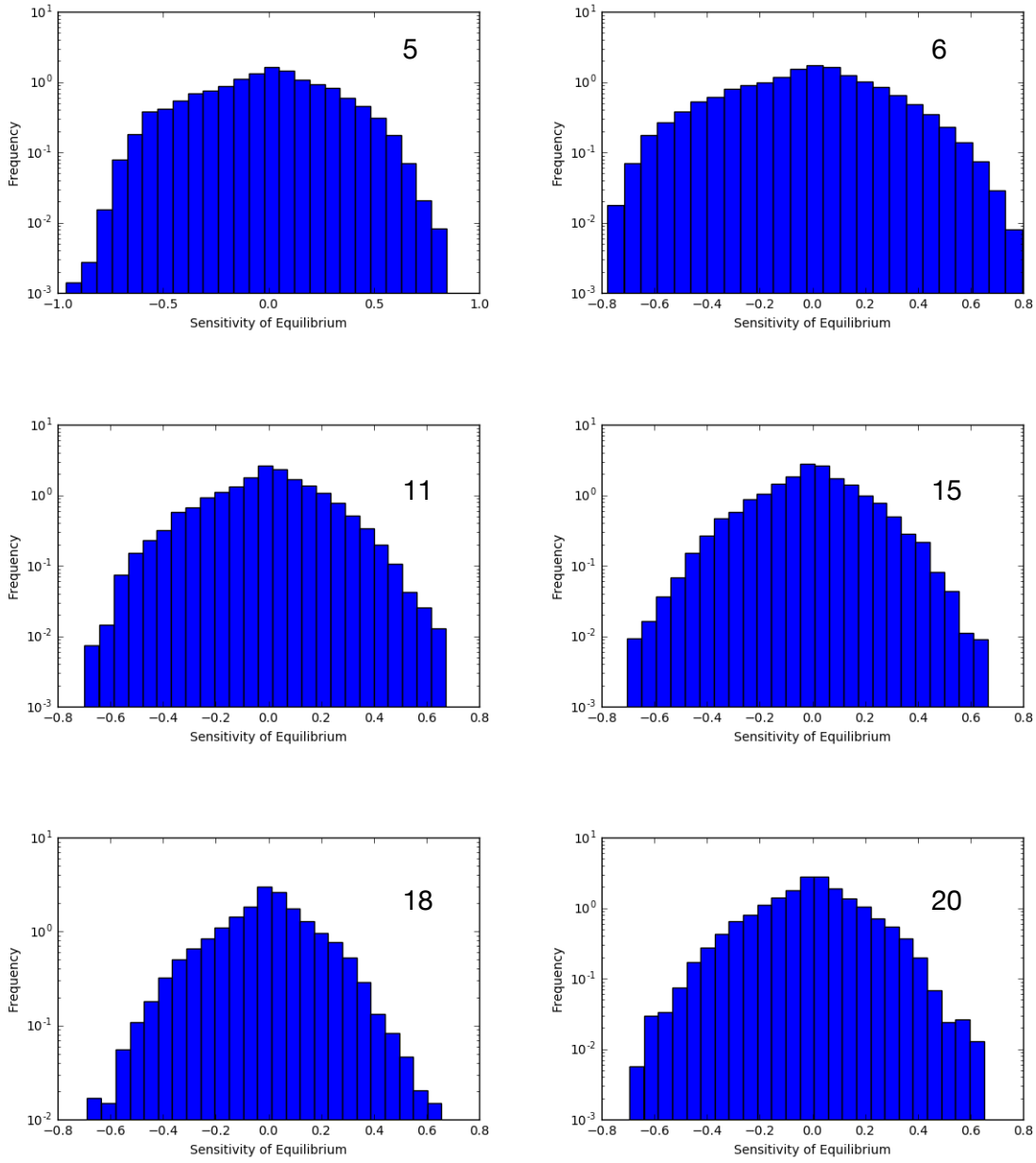


Figure 7-9. Sensitivity of Master Switch Nodes under Positive Feedback Loop Modification. All sampled networks had 20 non-master-switch nodes, dissociation constants and expression rates drawn uniformly on identical scales, equal probabilities assigned to activator and repressor operons, and leak expression rates scaled down by a factor of 100. Each distribution is labeled with the maximum in-degree of nodes (not master switch) used to generate the component data.

The results of varying maximum in-degree of nodes in a master switch network are shown above in Figure 7-9. The behavior we observe is very similar, though more pronounced, than the behavior we observed with fully random networks. The width of the distribution decays more slowly as maximum in-degree increases, and the distribution achieves the shape of a normal distribution even while at full width. Furthermore, the concentrated central peak we observed under the fully random network experiments is dispersed here. It is worth noting that the sensitivities here are only measured for the master switch node, i.e. we are unconcerned with how effectively the concentration of any other node can be changed. All in all, master switch nodes perform differently enough that they may well be worth investigating in greater detail in the future.

8 Extensions and Future Work

8.1 Inferring Graph Structure Empirically

One of the reasons for which full characterization of a network is prohibitive is simply that not all nodes and connections in a genetic network are equally useful for predicting dynamics and equilibria. This problem would be greatly mitigated if, e.g., one could expand and traverse portions of a network selectively, only unrolling them if there was evidence that those portions would prove informative and useful. Although we have not yet shown that the sensitivity of a node provides additional information about the impact of the node's neighbors on the dynamics of the system, this would be a reasonable conclusion to draw given the way we defined sensitivity. Additional work to clarify this assumption would perhaps be worth doing, for with such confirmation, networks could be partially characterized in such a way as to efficiently extract information only about parts of the graph truly necessary to predict and modify network dynamics.

To be more concrete, one could imagine running experiments to determine a node's neighbors in a genetic network based on a probabilistic decision that takes into account the sensitivity of that node. Finding the right stochastic process by which to make such a decision will require forming a better understanding of how the average efficacy of a modification involving a specified node is dependent on the efficacies of modifications involving neighboring nodes.

8.2. Training Predictive Models

In the course of our investigation, we found that inferences could not readily be drawn about the efficacy of a specified modification based on individual properties of the node or edge being directly modified. The reason for this was that the distributions we were looking at were simply too complicated. There are two possibilities associated with this complexity. One is that there is no computable function of a node's properties that will allow one to predict sensitivity ahead of time, and the other is that there is such a function but that it is much more complicated than any of the degree computations we attempted. The best way to determine which possibility is correct is to attempt to find such a function.

Deep learning might well prove useful on a problem like this. In the problem we have described, there are a number of attributes of information available to us, either about the node or edge being modified or about neighbors in the network. While running our experiment *in silico*, we have access to all the information we could wish for about our network, but the experimentalist making use of this empirical technique would have a much more limited set of information available. However, no information is truly unavailable to the experimentalist; rather, some information is already available, while other information would simply be costly to obtain. One way to train a neural net model on data that is nominally available but should not be used unless absolutely necessary is to add noise to pieces of information, such that noise is added to features at levels commensurate to the costliness of the feature. Such a model would then only use costly information when absolutely necessary. In this way, we could train a model to predict, in an experimentally feasible way, the likelihood that a specific modification will be effective. Poor model performance would indicate that no predictive function exists.

9 Conclusion

In conclusion, we tested the viability of a local variant of metabolic engineering by sampling random genetic networks and testing the predicted efficacy of our strategy on these sampled networks. We generated data under a variety of different conditions, specified via hyperparameters that allowed us to vary the structure of our networks and the distributions from which operon parameters were sampled. We established that, without additional information about the network being examined or the protein being targeted for modification, only one category of modification (involving the addition of positive feedback) showed promise as a vehicle for empirically changing the equilibria of a partially characterized network.

We were able to draw limited conclusions about the effect of local graph connectivity on the value of targeting a particular protein in a network. We were also able to relate modification efficacy to the relative magnitudes of promoter dissociation constants and operon expression rates, as well as to the local connectivity patterns within the network. We also established some theoretical results about the optimal modification hyperparameters required to achieved a desired equilibrium for a system.

Finally, we presented strategies for using empirical sensitivity measurements to more efficiently characterize unknown networks and to determine modification efficacy with as little biological experimentation as possible. These strategies may prove useful in extending this work in the future and in turning this strategy into a practical experimental tool.

10 References

[1] Yang, Y.T., Bennet, G. N., San, K.Y. Genetic and Metabolic Engineering. Electronic Journal of Biotechnology Volume 1, Number 3 (1998).

[2] Rocha, I., Maia, P., Evangelista, P., et al. OptFlux: an open-source software platform for in silico metabolic engineering. BMC Sys Biol. 45(4) (2010).

[3] Entus, R., Aufderheide, B., & Sauro, H.M. Design and implementation of three incoherent feed-forward motif based biological concentration sensors. Syst Synth Biol 1: 119–128 (2007).

[4] Gardner T.S., Cantor C.R. & Collins J.J. Construction of a genetic toggle switch in Escherichia coli. Nature 403: 339-342 (2000).

[5] Elowitz, M.B. & Leibler, S. A synthetic oscillatory network of transcriptional regulators. Nature 403: 335-8 (2000).

[6] Rich, R.L. & Myszka, D.G. Higher-throughput, label-free, real-time molecular interaction analysis. Analytical Biochemistry. **361** (1): 1–6 (2007).

[7] Levine, J.H., Lin, Y., & Elowitz, M.B. Functional Roles of Pulsing in Genetic Circuits. Science 342: 1193 (2013)

[8] Alon, U. An Introduction to Systems Biology. Design Principles of Biological Circuits. Chapmann & Hall/CRC. Boca Raton, FL. 2007.