

Constructing Mathematical Models of Gene
Regulatory Networks for the Yeast Cell Cycle and
Other Periodic Processes

by

Anastasia Deckard

Graduate Program in Computational Biology and Bioinformatics
Duke University

Date: _____

Approved:

John Harer, Supervisor

Steven Haase

Sayan Mukherjee

David Schaeffer

Dissertation submitted in partial fulfillment of the requirements for the degree of
Doctor of Philosophy in the Graduate Program in Computational Biology and
Bioinformatics
in the Graduate School of Duke University
2014

ABSTRACT

Constructing Mathematical Models of Gene Regulatory
Networks for the Yeast Cell Cycle and Other Periodic
Processes

by

Anastasia Deckard

Graduate Program in Computational Biology and Bioinformatics
Duke University

Date: _____

Approved:

John Harer, Supervisor

Steven Haase

Sayan Mukherjee

David Schaeffer

An abstract of a dissertation submitted in partial fulfillment of the requirements for
the degree of Doctor of Philosophy in the Graduate Program in Computational
Biology and Bioinformatics
in the Graduate School of Duke University
2014

Copyright © 2014 by Anastasia Deckard
All rights reserved except the rights granted by the
Creative Commons Attribution-Noncommercial Licence

Abstract

We work on constructing mathematical models of gene regulatory networks for periodic processes, such as the cell cycle in budding yeast, using biological data sets and applying or developing analysis methods in the areas of mathematics, statistics, and computer science. We identify genes with periodic expression and then the interactions between periodic genes, which defines the structure of the network. This network is then translated into a mathematical model, using Ordinary Differential Equations (ODEs), to describe these entities and their interactions. The models currently describe gene regulatory interactions, but we are expanding to capture other events, such as phosphorylation and ubiquitination. To model the behavior, we must then find appropriate parameters for the mathematical model that allow its dynamics to approximate the biological data.

This pipeline for model construction is not focused on a specific algorithm or data set for each step, but instead on leveraging several sources of data and analysis from several algorithms. For example, we are incorporating data from multiple time series experiments, genome-wide binding experiments, computationally predicted binding, and regulation inference to identify potential regulatory interactions.

These approaches are designed to be applicable to various periodic processes in different species. While we have worked most extensively on models for the cell cycle in *Saccharomyces cerevisiae*, we have also begun working with data sets for the metabolic cycle in *S. cerevisiae*, and the circadian rhythm in *Mus musculus*.

To my family: Betty (Mom), Morgan, Gary, Tom.

Contents

Abstract	iv
List of Tables	ix
List of Figures	x
Acknowledgements	xiii
1 Motivation	1
2 Background and Significance	3
2.1 Building Models	3
2.1.1 Finding Nodes	4
2.1.2 Finding Edges	5
2.1.3 Mathematical Models	8
2.1.4 Global Network Model	9
2.2 Infrastructure and Methodology to Support Model Building	10
2.2.1 Testing	10
2.2.2 Data Management & Integration	10
2.2.3 Iterative and Flexible Analysis	11
3 Mathematical Models	13
3.1 Gene Regulation	14
3.2 Modeling regulation: Hill Functions	15
3.3 Post-Translational Regulation	17

3.4	Proxies for complexes and similar proteins	18
3.5	Representing Models for Computation	21
3.6	Conclusions	24
3.7	Future Work	24
4	Finding Nodes	25
4.1	Methods for Detecting Periodicity	26
4.2	Synthetic Data	28
4.2.1	Distinguishing Periodic Shapes	29
4.2.2	Sampling Density vs. Experiment Length	33
4.3	Biological Data	35
4.3.1	Preference in shapes	37
4.3.2	Differences in p-values/scores between algorithms	37
4.3.3	Finding Dominant Periods	40
4.3.4	Constructing lists of periodic genes	43
4.4	Conclusions	44
4.5	Future Work	46
5	Finding Edges	47
5.1	Using biological data to identify edges	47
5.2	Computationally-predicted binding	49
5.3	Methods for inferring edges	52
5.4	Developing synthetic networks for testing	53
5.4.1	Generating networks with optimization	54
5.4.2	A library of synthetic periodic networks	58
5.5	Local Edge Machine (LEM)	60
5.6	Conclusions	62

5.7	Future Work	64
6	Finding Network Parameters	66
6.1	Finding Parameters for a Network: Evolutionary Algorithms	66
6.1.1	Create Initial Population	67
6.1.2	Simulate Population	68
6.1.3	Assign Fitness	68
6.1.4	Selection	70
6.1.5	Reproduction & Mutation	71
6.1.6	Testing	72
6.1.7	Optimization with unknowns	73
6.2	Using biological data to inform parameters	73
6.3	Enforcing topology in the face of changeable parameters	75
6.4	Models of the Yeast Cell Cycle	76
6.5	Mapping the search space	78
6.6	Conclusions	80
6.7	Future Work	80
7	Data Management & Integration	83
7.1	Gene Information	83
7.2	Mapping gene identifiers	84
7.3	Results on integrating chIP-chip data	88
7.4	Conclusions	90
7.5	Future Work	90
	Bibliography	92
	Biography	100

List of Tables

3.4.1 mRNA proxies for complexes/groups	20
4.2.1 Functions for synthetic signals for testing periodicity detection	31
4.3.1 Numbers of genes selected as periodic for cutoffs	39
4.3.2 Time ranges covered by the yeast data sets	41
6.1.1 Parameter limits used in modeling ODEs	68
6.2.1 Different scales in gene expression time series data sets	74
6.2.2 Different scales in times series data sets normalized with different packages	74
7.3.1 Integrated ChIP-chip data for yeast	89

List of Figures

2.2.1 Iterative lifecycle for building models	12
3.1.1 Overview of transcriptional gene regulation	15
3.2.1 Hill functions and their behavior	17
3.3.1 Modeling Ubiquitination	19
3.3.2 Modeling Phosphorylation	20
3.5.1 Regulation equations for a network topology	22
3.5.2 Binary operator tree to represent regulation logics	23
4.2.1 Synthetic signals for testing periodicity detection	30
4.2.2 Performance on separating periodic from non-periodic signals	32
4.2.3 Biases for signal shape in periodicity detection algorithms	34
4.2.4 Detecting periodic signals for different numbers of cycles	35
4.3.1 Top ranked signals from each periodicity detection algorithm	38
4.3.2 Detecting dominant periods in yeast cell cycle data sets	42
4.3.3 Detecting dominant periods in mouse circadian rhythm data sets	43
4.3.4 Periodic genes in the Yeast Cell Cycle	44
4.3.5 Periodic genes in the Mammal Circadian Rhythm	45
5.1.1 Networks built from biological evidence of regulation	50
5.2.1 Predicted promoter occupancy across time	52
5.4.1 Synthetic 10 node gene regulatory networks for testing	59
5.4.2 Synthetic 20 node gene regulatory networks for testing	60

5.5.1 LEM evaluates a combination of regulators	61
5.5.2 Performance of LEM on synthetic models	63
6.1.1 Overview of Evolutionary Algorithm	67
6.4.1 Optimizing parameters for simplified models	79
6.5.1 Exploring the search space of simplified Yeast Cell Cycle models	81
7.2.1 Systematic or standard names also as alias names	86
7.2.2 Merging or deletion of genes	87
7.2.3 Alias maps to multiple Genes	88

Symbols

K	1/2 maximal activation/repression for Hill function.
n	Hill coefficient in Hill function.
\vee , OR	An OR logic gate.
\wedge , AND	An AND logic gate.

Abbreviations

APC	Anaphase Promoting Complex.
CDK	Cyclin Dependent Kinase.
CLOCCS	Characterizing Loss Of Cell-Cycle Synchrony.
EA	Evolutionary Algorithm.
GRN	Gene Regulatory Network.
ODE	Ordinary Differential Equation.
ORF	Open Reading Frame.
PSSM	Position-Specific Scoring Matrices.
SGD	Saccharomyces Genome Database.
TF	Transcription Factor.
LEM	Local Edge Machine.
DL	de Lichtenberg.
LS	Lomb-Scargle.
JTK	JTK_CYCLE.
PH	Persistent Homology.
SW	SW1PerS.

Acknowledgements

I would like to thank my advisor, John Harer, for listening patiently to my ideas, kindly answering my questions, and always being excited about whatever project we were working on. Your enthusiasm and encouragement made projects fun, and your ability to keep me (mostly) on task made progress possible.

I would like to thank my co-advisor, Steven Haase, for being glad when I asked a question, encouraging us to say what we mean, and appreciating the database.

I would like to thank my committee members Sayan Mukherjee and David Schaffer. Thank you for asking insightful questions that improved my work and taking the time to answer my questions.

I would like to thank the members of the Harer Group: Jose Perea, Kevin McGoff, Xin Guo, Colbert Sesanker. We have had a lot of interesting technical discussions and some fun non-technical discussions.

I would like to thank the members of the Haase Lab: Chun-Yi Cho, Ariana Eily, Laura Simmons Kovacs, Michael Mayhew, and Crystal Baker, and especially Sara Bristow, Christina Kelliher, and Adam Leman. You have gracefully handled huge numbers of biology questions, requests for data, and even more requests to review giant lists of data and graphics.

I would also like to say how much I have enjoyed our collaboration between the computation/math people of the Harer group and the biology people of the Haase lab.

I would like to thank members of my family, whose support, encouragement, and general love of science made this possible. Mom, thank you for being supportive and interested in my work and teaching me that difficult tasks can be the most rewarding. Morgan, thank you for being there to talk and for the home-baked food. Tom, thank you for cheering me on and making sure I have waterproof shoes. Mom and Tom, thank you for proofreading this dissertation. Gary, thank you for being there and for reminding me that there is much more to life than my computer. My enjoyment of life in general is greatly improved by sharing it with you.

1

Motivation

The cell cycle is a periodic process in which a cell replicates its parts and divides into two cells, and this process is then repeated in each of these cells, and so on. The proper regulation of the cell cycle is critical to growth, development, and reproduction. In multicellular organisms, problems in the regulation of the cell cycle can lead to disease.

Given the size and complexity of the regulatory network for the cell cycle, modeling is becoming critically important in representing and studying this system. The overarching goal of this project is to construct quantitative models of the cell cycle that can be manipulated to study this process and its components. These models would be used to inform the design of future experiments; the results of these experiments can then be used to improve the model.

Additionally, these models can be used to test assumptions about proposed networks. It is commonly believed that the activity of cyclins and cyclin-dependent kinases (CDKs) form the central oscillator in the cell cycle. More recently, it has been proposed that a network of transcription factors form an autonomous oscillator that controls the transcriptional program and is coupled to, but not controlled by, the

cyclin/CDK oscillator (Simmons Kovacs et al., 2012). Where the predictions from a quantitative model of a proposed network compares well to experimental data, it indicates that the model adequately captures the mechanism of interest.

Constructing a model of a gene regulatory network requires finding genes (the nodes) and then the interactions between the genes (the edges), which defines the structure (topology) of the network. We then translate this topology into a mathematical model, using Ordinary Differential Equations (ODEs), to describe these entities and their interactions. To model the behavior, we must then find appropriate parameters for the mathematical model of this network.

To construct models of the cell cycle, we propose an approach that uses large-scale data integration and a pipeline of algorithms. These will be used to identify the important components to be used in models, and how they interact. However, these will not be used to construct large-scale models, which are often not informative at the lower levels of detail we are interested in. Instead, they will be used to construct smaller-scale models that capture the important actors and actions within the networks. Additionally, many data sources will be incorporated to increase our confidence in the model and to reveal underlying mechanisms.

While we work currently with data from yeast for constructing models of the cell cycle, these approaches are designed to be applicable to other models that exhibit periodic processes.

Background and Significance

Historically, building models of biological processes has used a reductionist approach (You, 2004). These bottom-up models are often smaller and simpler, but are well curated and understood mechanistically. More recently, systems biology emphasizes the use of data that are becoming available from large-scale, high-throughput methods in a more top-down approach. However, the challenge of building informative models from these large amounts of data grows rapidly. Additionally, the complexity and size of the underlying system means that our intuition fails when trying to construct this type of model.

2.1 Building Models

The process of building mathematical models of gene regulatory networks for periodic processes can be separated (to some extent) into steps: find the nodes, find the edges, apply a mathematical framework, and then use these to construct a network model. The first step is to identify genes with periodic expression (nodes) and then the interactions between the genes (edges), which defines the structure of the network. This network is then translated into a mathematical model to describe these entities

and their interactions. To model the global behavior, the appropriate parameters must be found for the mathematical model that allow its dynamics to approximate the biological data.

2.1.1 Finding Nodes

To build a network, it is necessary to identify the entities that must be included in the network. Nodes are the actors in a network; these usually correspond to mRNA or protein levels. As we are interested in building gene regulatory networks of periodic processes, we are looking for genes that are being periodically expressed. There exist many methods for detecting periodicity in data, often originating in fields outside of biology, and many have been applied to identifying periodicity in biological data. We can divide these methods into broad classes based on how (or whether) they use a reference signal: sinusoidal curves as a reference signal, user-defined reference patterns, and methods that do not use a reference pattern.

In this first group are approaches that measure the strength of the periodicity and determine the period length by comparing the signal to sinusoidal curves with different periods. These include methods that transform a signal in the time domain into the frequency domain, as with Fourier transforms, while others fit sinusoidal curves directly to the target signal. COSOPT (Straume, 2004) compares a signal to cosine curves with different phases and periods to measure their correspondence and then uses empirical resampling to compute significance. In De Lichtenberg et al. (2005), a Fourier method and a measure of amplitude (as an indicator of the strength of regulation) are used to score signals. Permutations are then used to compute significance. Lomb-Scargle (Lomb, 1976; Scargle, 1982) uses a method based on Fourier methods, but can accommodate unevenly sampled and missing data and return significance scores.

Another set of algorithms compare the signal to reference curves that are specified

by the user (which may be sinusoidal curves). The method of Luan and Li (2004) generates a single spline function to capture the pattern of known periodic genes, and then uses this shape-invariant model to identify other periodic genes. JTK CYCLE (Hughes et al., 2010) computes how the samples are increasing or decreasing relative to one another, for both the reference curve and signal. It then measures the statistical significance of the correlation between these patterns for the reference curve and the signal.

Some methods do not use a reference pattern, but instead attempt to discover patterns that exist in the data. Address Reduction (Ahnert et al., 2006) measures the algorithmic compressibility of the signal; a signal that is more compressible indicates there is a pattern that might be of biological interest. An application of Persistent Homology by Cohen-Steiner et al. (2010) determines pairs of minima and maxima of a time series. If there is only one minimum and maximum pair, it is considered to be a perfect oscillation; additional oscillations in the time series will create more minimum-maximum pairs, indicating a less perfect curve.

A recently developed algorithm, SW1PerS, was designed to overcome some of the limitations in the the above methods (Perea and Harer, 2013). SW1PerS transforms the signal into a high-dimensional set of points and interprets periodicity of the original signal by measuring the circularity of the point cloud. We have been testing and refining this method using synthetic data. We have also used this approach on several biological data sets to find genes that are periodically expressed (Perea et al., 2014).

2.1.2 Finding Edges

After identifying the nodes in the network, our next task is to find the interactions between the nodes. These interactions are called the edges in a network. The goal of network inference (a.k.a. reverse engineering of networks or network reconstruction)

is to use experimental data to find edges.

To infer a GRN to model network dynamics, it is necessary to run time-series experiments to collect large quantities of data; unfortunately, it is difficult to specify the amount of data needed beforehand, and both the number and spacing of time points can affect the performance of the inference (Hecker et al., 2009). Additionally, the amount of data needed increases as the system becomes more complex, the data becomes more dynamic, the network increases in size, or the desired resolution increases. As the number of nodes in a network increases, the number of possible topologies and their parameters increase exponentially (van Riel, 2006). Another issue is the quality of the data (Van Someren et al., 2002). Data quality is decreased by noise from measurements, low concentrations, lack of cell synchrony, or variations in individuals/tissues. Furthermore, the information contained in the data may be much less than the quantity of data. Additionally, a single experiment will only show the behavior of the system under one specific condition. While progress has been made on constructing gene regulatory networks from experimental data (Gardner and Faith, 2005; Hecker et al., 2009; Van Someren et al., 2002), this is still considered an important and challenging area.

Network inference methods use time series data from experiments (e.g. microarrays for several time points in wild-type, knockout/knockdown/overexpression, or perturbation experiments) and/or steady-state data from multiple perturbation experiments. For using only steady-state data, it is possible to infer that nodes interact, but not necessarily the direction of the interaction because there is no timing information, which gives undirected edges. This is the case for mutual information or correlation approaches. For time-series data, there is an inherent direction to actions; this gene turns on and then this gene responds, which indicates the direction of the edge.

Many algorithms have been developed to infer gene regulation, and cover a wide

range of methodologies from different disciplines. ARACNE uses an information theory measurement, called mutual information (MI), of the dependence between genes (Margolin et al., 2006), and TimeDelay-ARACNE is a modification that uses time series data to indicate the direction of the interactions (Zoppoli et al., 2010). In a similar vein, CLR (Context Likelihood of Relatedness) also uses MI (Faith et al., 2007), and tlCLR (time-lagged CLR) uses timing information to indicate the direction of the edges. Banjo uses Dynamic Bayesian Networks to model the system and a greedy search with multiple restarts to the most probable network to explain the data (Yu et al., 2004). Inferelator models the network as a system of linear ODEs and uses Least Angle Regression (LARS) to find directed interactions between genes, while enforcing sparseness in network connectivity (Bonneau et al., 2006). A more recent version includes tlCLR for model selection, followed by the Inferelator for model refinement and parameter discovery (Greenfield et al., 2013). TIGRESS is another method that uses Least Angle Regression (LARS) for feature selection, run many times on randomly perturbed data as a stability selection step (Haury et al., 2012). GENIE3 is a tree-based ensemble method; while it only uses steady-state measurements, the authors state that it is able to predict directed edges in some cases by comparing the symmetry of predictions between nodes (Huynh-Thu et al., 2010).

A new method is being developed, called Local Edge Machine (LEM) by Kevin McGoff, Xin Guo, John Harer, and me, at Duke University. This method infers combinatorial logic for the regulators of each gene from time series data. It uses Hill functions to model the interactions, gradient descent to find the best fits to the data, and a Bayesian model to compute the probability of a combinatorial logic given the data. We have tested this method on synthetic data and run it on biological data.

2.1.3 Mathematical Models

Models of gene regulation can be constructed with varying levels of abstraction. As genes do not directly interact, it is necessary to implicitly or explicitly model factors such as: synthesis and degradation of proteins and mRNA, binding of regulatory elements, different interactions between multiple regulatory elements, activation and inhibition, post-translation modifications, interactions between proteins and/or metabolites, etc. Models can be described as physical or influential (Gardner and Faith, 2005). A physical strategy attempts to identify the actual physical interactions between DNA and other entities; therefore, it must explicitly model non-gene entities and their interactions. An influence strategy focuses on finding relationships between changing levels of gene products to infer their interactions (even though they don't physically interact); therefore it implicitly models the effects of other entities without actually describing their interactions.

There are many different representations that can be used to model a gene regulatory network. Networks can be modeled using correlation networks, Boolean networks, Bayesian networks, linear and nonlinear ordinary differential equations (ODEs), partial differential equations, and stochastic equations (De Jong, 2002; Hecker et al., 2009). This research focuses on nonlinear ODE models because they can model oscillations (Hecker et al., 2009) and have been widely used in modeling dynamical systems in general, and more specifically for modeling regulatory networks (De Jong, 2002).

Even within the modeling framework of nonlinear ODEs, many different systems have been used to describe gene regulatory networks and other biological interactions that affect them. From experimental evidence, the transcriptional response of a gene to regulation is nonlinear and has a sigmoidal shape; an extreme version of this response is known as an ultra-sensitive response. In more mechanistic models, where

genes and proteins and gene-protein and protein-protein binding are modeled, ODEs with mass action kinetics may be used. These models can display ultra-sensitive responses through the interactions of multiple nodes (Buchler and Louis, 2008) and can have oscillations (Francois and Hakim, 2004). S-systems are another mathematical system that captures combinatorial activation and repression of multiple regulators with nonlinear ODEs (Savageau and Voit, 1987). Hill functions, one for activation and one for repression, are sigmoidal functions that model cooperativity (Hill, 1913). These functions are used in many models of gene regulation.

2.1.4 Global Network Model

While nonlinear differential equations are well suited to modeling needs, this increases the complexity of the models. Most systems of nonlinear ODEs cannot be solved analytically, so they must be solved numerically, which can require considerable computations for large systems. A much larger sample size is needed to be able to pinpoint nonlinear interactions and there are usually more parameters to fit, which increases the size of the search space that must be considered (Hecker et al., 2009).

To address these issues, we use several strategies. The first strategy is to use fast simulation of the mathematical models. The second strategy is using global heuristic optimization techniques that are designed to deal with finding optimal solutions in large and difficult search spaces. The third strategy is using data to inform the selection of smaller subnetworks of interest. Additionally, the solutions found may not be unique; they could represent one of many possible sets of topologies and parameters.

For example, in Chen et al. (2004), a cell-cycle model is constructed by identifying the molecular steps involved, defining the types of rate law to use, and then either using experimentally derived rate constants where available or determining those computationally. This model qualitatively agrees with the physiological behavior

of 120 mutant strains, but shows inconsistencies between model and data for 11 mutants.

2.2 Infrastructure and Methodology to Support Model Building

2.2.1 Testing

The lack of “gold-standard” data sets for testing these methods is another major challenge. Without comprehensive tests, it is very difficult to establish the accuracy of inferring networks, and even more difficult to compare accuracy claims between different methods (Gardner and Faith, 2005). This is also true for methods that are designed to detect periodically transcribed genes or regulatory interactions between genes.

For regulatory networks, the yearly Dialogue for Reverse Engineering Assessments and Methods (DREAM) challenge includes a synthetic (in silico) challenge. Now in its eighth year, each in silico challenge is different. Networks have ranged in size from ten to a hundred genes, include time series or steady state data, can be models from different organisms, and in the DREAM2 in silico challenge, included an oscillating network.

To address this issue in part, we create synthetic data sets to test our methods, and other available methods. The synthetic data sets are designed to capture the characteristics of real data, and all parts of the data were engineered with known parameters. We can therefore run these data sets through different methods and compare the output to the known parameters.

2.2.2 Data Management & Integration

With the large volumes of data generated in this field, it becomes necessary to implement strategies for managing data. Another concern is making sure that all data used is properly versioned and sourced; data in this field, even from one experiment,

are not static. Microarray experiments can be normalized with different algorithms, or groups of microarray data can be normalized with different experiments. Annotations can change: putative genes can be merged or deleted, new genes can be added, and names can change. We use data generated by different groups, at different times, from different technologies. These must be integrated as seamlessly as possible to ensure the most complete and accurate view of the underlying data.

2.2.3 Iterative and Flexible Analysis

Building a model is not one pass through a pipeline of analysis that leads to one final result. The outputs of model building may serve several possible roles: a hypothesis that can lead to new experiments, a group of possible solutions, or even an intermediary result that can suggest new types of information that might be needed to improve the model. We therefore view the construction of models as an iterative lifecycle, which relies on the close interactions of math and biology (Figure 2.2.1).

As the system is iterative, it must be flexible enough to include new data and generate new analyses. There are often many questions that data can answer, so the ability to explore several potential solutions is needed. There are often many answers to these questions that are “just as good”, and even more answers if one is willing to relax the criteria to reflect uncertainty about initial assumptions.

Additionally, the system needs to be flexible to include different types of organisms and different biological processes. While we currently focus on building models of the yeast cell cycle, these methods are all general enough that they can be applied to building models of other types of periodic processes. For example, there are time series data from the mammalian circadian cycle (Hughes et al., 2009), plant root clock (Moreno-Risueno et al., 2010), and the yeast metabolic cycle (Tu et al., 2005).

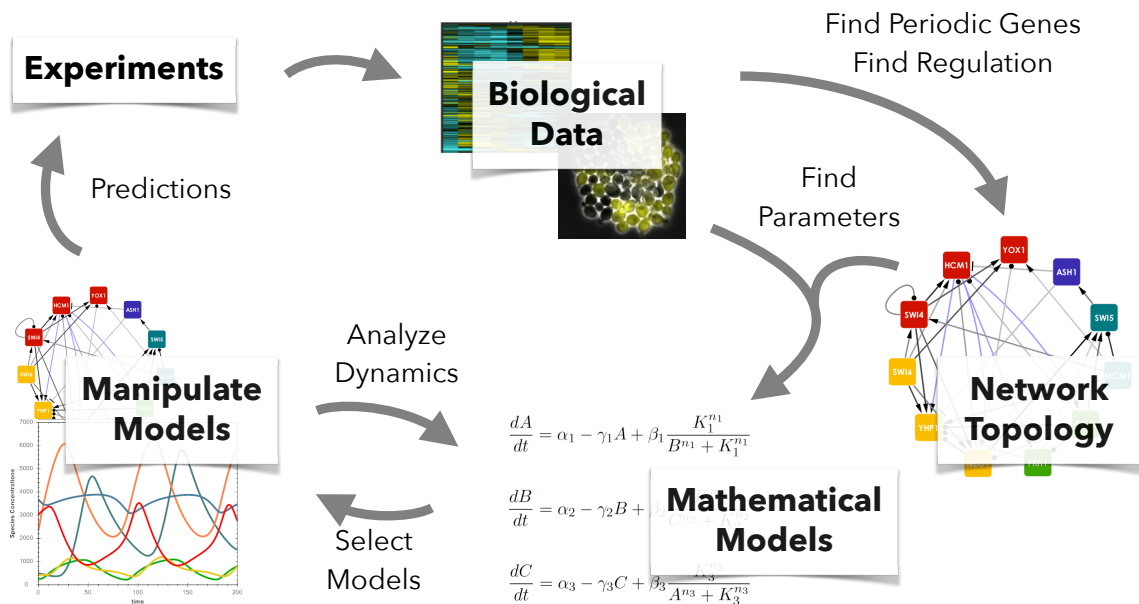


FIGURE 2.2.1: Building mathematical models of biological networks, shown as an iterative process. Steps include gathering data, finding the nodes and edges of a network, building a mathematical model from these, selecting and analyzing the dynamics of the model, and finally using the model to make predictions or inform what additional data is needed. Experiments are then run and new data are used to improve the model on the next iteration.

Mathematical Models

Mathematical modeling is more than developing equations to describe a system; it is the collecting of actors and behaviors of a system, cataloguing assumptions about what is important to model and parts that can be simplified, and building an understanding of what purpose the model will fulfill. So while we might not start building a model until information has been gathered, we first need to evaluate what can be gained from a model and the general form we desire in a model. Only then can we determine what information is needed to begin building a model. That is why we describe the mathematical models first, even though we do not generally start building them until we have collected data (Chapter 4 Finding Nodes and Chapter 5 Finding Edges).

We wish to build mathematical models that can both emit dynamics like the oscillating patterns of gene expression and capture the underlying structure of the gene regulatory network. A model that is capable of prediction is the ultimate goal; a change in the structure of the model would lead to a change in the model's function, thus predicting the result of the same change made in the biological network. However, even short of large-scale models for predicting system-wide changes, many

of the smaller steps in building models can be highly informative. Even errors can be informative; they can indicate when an assumption about the biological system might be wrong or reveal a discord with the math that was chosen to represent it. They can also help to show which parts of the system might be important, parts might need an expanded or deeper exploration, or parts that could be simplified. The most illuminating part of model building is that it shows where more information is needed, and sometimes even suggests what kind of information should be sought.

3.1 Gene Regulation

Models of gene regulation cover varying levels of abstraction. As genes do not directly interact, it is necessary to implicitly or explicitly model factors such as: synthesis and degradation of proteins and mRNA, binding of regulatory elements, activation and inhibition, post-transcriptional modifications, post-translational modifications, interactions between proteins and/or metabolites, et cetera. At the more mechanistic, actual physical interactions between DNA and DNA-binding proteins, protein-protein interactions can be explicitly modeled; therefore, such a model must explicitly account for non-gene entities and their interactions. A more abstract modeling approach might only attempt to capture relationships between changing levels of various gene's products (even though they don't physically interact) by implicitly modeling the effects of other entities without actually describing their interactions.

Within the modeling framework of nonlinear ODEs, many different systems have been used to describe gene regulatory networks and other biological interactions that affect them. From experimental evidence, the transcriptional response of a gene to regulation is nonlinear and has a sigmoidal shape. Hill functions, one for activation and one for repression, are sigmoidal functions that model cooperativity (Hill, 1913) and are very popular for modeling transcriptional regulation.

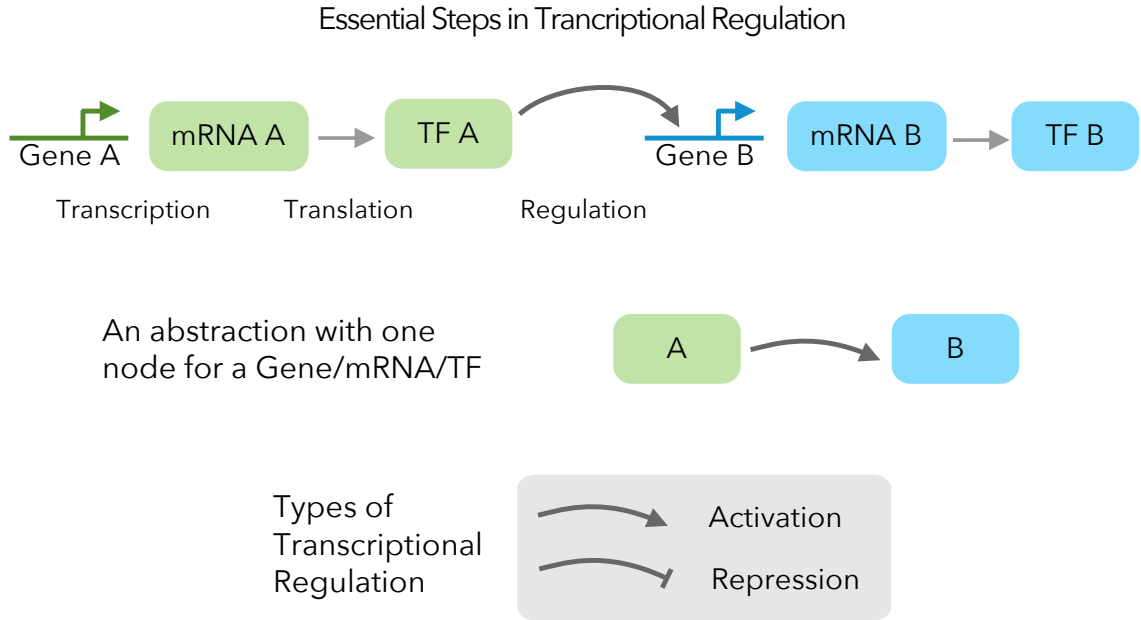


FIGURE 3.1.1: Overview of transcriptional gene regulation. Transcriptional regulation includes the steps of a gene being activated, transcription creating mRNA, translation creating proteins, and those proteins then regulating a downstream gene. A more abstract model simplifies the system by grouping gene, mRNA, and protein into one node, where the intermediate processes are implicitly modeled. For transcription, regulation can be either activation or repression.

3.2 Modeling regulation: Hill Functions

To model the behaviors of genes and their interactions, we use systems of Ordinary Differential Equations (ODEs). Regulatory effects of the interactions have been modeled using Hill functions (De Jong, 2002; Alon, 2007; Polynikis et al., 2009).

The ordinary differential equation for a gene X_i under regulation is written in the form:

$$\frac{dX_i}{dt} = \alpha_i - \gamma_i X_i + f_i(X_1, X_2, \dots, X_m) \quad (3.2.1)$$

where X_i is the gene being regulated, α_i is the basal transcription rate constant, γ_i is the degradation rate constant, and f_i is the input function for one or more

transcription factors X_1, X_2, \dots, X_m that act on gene X_i . Note that a transcription factor (TF) can act either as an activator or a repressor.

Where multiple TFs act on the gene, the input function models their combined effect on the regulation of the gene. These can be written as combinations of AND and OR logic gates. For example, an input function of two activators joined by OR, with an AND gate for the repressor is written:

$$f_1(X_1, X_2, X_3) = (a_{1,1}(X_1) \vee a_{1,2}(X_2)) \wedge r_{1,3}(X_3) \quad (3.2.2)$$

We generally assume that repression always overrides other combinations, and use an AND gate to represent this behavior. There are several ways to interpret these gates mathematically. Here we choose to use multiplication for an AND gate and addition for an OR gate.

The regulatory effect of a transcription factor will be modeled using Hill functions, which have a sigmoidal shape (Hill, 1913). Regulation can be either activation, which increases transcription of the target gene, or repression, which decreases transcription of the target gene. The Hill functions, either activation a or repression r , for a gene X_i being regulated by transcription factor X_j are:

$$a_i(X_j) = \beta_{i,j} \frac{X_j^{n_{i,j}}}{X_j^{n_{i,j}} + K_{i,j}^{n_{i,j}}} \quad r_i(X_j) = \beta_{i,j} \frac{K_{i,j}^{n_{i,j}}}{X_j^{n_{i,j}} + K_{i,j}^{n_{i,j}}} \quad (3.2.3)$$

where K is the 1/2 maximal activation/repression coefficient, β is the maximal transcription rate, and n is the the Hill coefficient. X_j is the amount of the transcription factor. Each Hill function can define different parameters, so the are indexed by (i, j) for the gene X_i being regulated by transcription factor X_j . These Hill functions give sigmoidal response curves (Figure 3.2.1).

The Behavior of Hill Functions

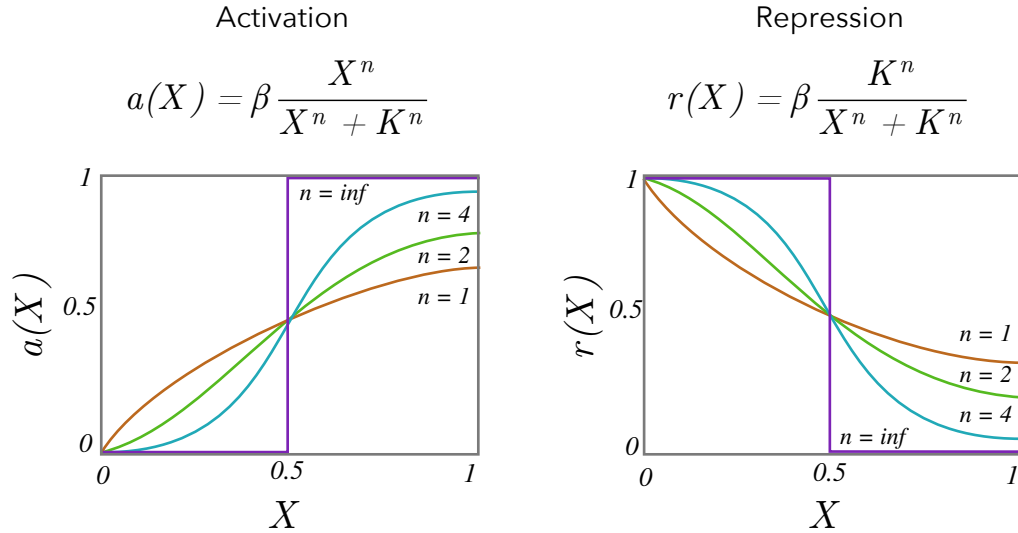


FIGURE 3.2.1: Hill functions for activation and repression. The behavior of the Hill functions are plotted for an increasing amount of the regulator X .

3.3 Post-Translational Regulation

In some cases, it is useful to expand the model and include post-transcriptional or post-translational modifications. Modifications to the mRNA, after transcription but before translation, are called post-transcriptional. Modifications occurring to the proteins, are called post-translational. To model post-transcriptional modifications, the amount of the protein must be modeled separately, so our combined node will be broken into two nodes, one for mRNA and one for Protein.

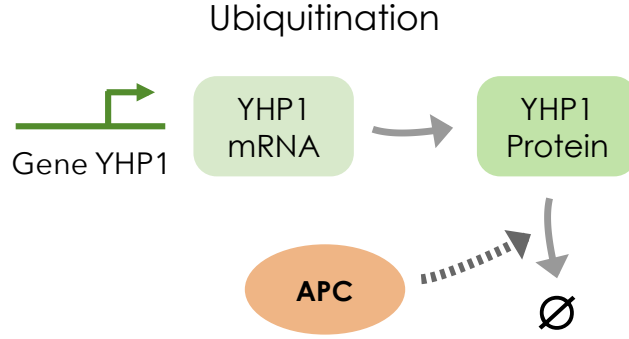
Ubiquitination is the process in which a post-translational regulatory protein called a ubiquitin ligase attaches a ubiquitin to a protein, which marks the protein for degradation by the proteasome. Mathematically this is represented by increasing the rate of degradation of a protein/TF as a function of increasing amounts of the ubiquitin ligase, as the degradation rate is no longer constant. In the cell cycle, an important ubiquitin ligase is the Anaphase Promoting Complex (APC), which begins the transition from anaphase to metaphase. For example, the protein Yhp1

is ubiquitinated by the APC, which increases its rate of degradation (Ostapenko and Solomon, 2011). A diagram of this process and possible equations that can be used to model this process are shown in Figure 3.3.1. Nrm1, Clb2, and the components of the APC are also ubiquitinated by the APC (Ostapenko and Solomon, 2011; Prinz et al., 1998).

Phosphorylation/Dephosphorylation is the process of adding/removing a phosphate group from a protein, which modifies its activity. Kinases phosphorylate proteins and phosphatases dephosphorylate proteins. For a transcription factor, phosphorylation/dephosphorylation modifies its ability to regulate its downstream targets either by making the TF more or less active or by dictating that it should be moved in or out of the nucleus. Multiple phosphorylation sites often exist on these proteins, and multiple phosphorylations/dephosphorylations can occur leading to different levels of activity. As a simplification, we assume states are combinations of low, high, on, and off. For example, the SFF complex is phosphorylated by Clb2, which moves SFF from a state of low activity to high activity (Pic-Taylor et al., 2004; Reynolds et al., 2003). A diagram of this process and possible equations that can be used to model this process are shown in Figure 3.3.2. The SBF and MBF complexes start in a low activity state, but can be changed to a high or off activity state by phosphorylation. The Ace2 and Swi5 proteins start in an off state and are switched to an on state by phosphorylation. We also assumed that the reverse of the modification happens automatically, but at a much slower rate.

3.4 Proxies for complexes and similar proteins

To simplify a model, we amalgamate nodes that have similar or cooperative behaviors. We then use a “proxy” node for representing the behavior of interest. Two situations for which this simplification can be made are protein complexes and genes with similar expression dynamics (Table 3.4.1). A *complex* is group of interact-

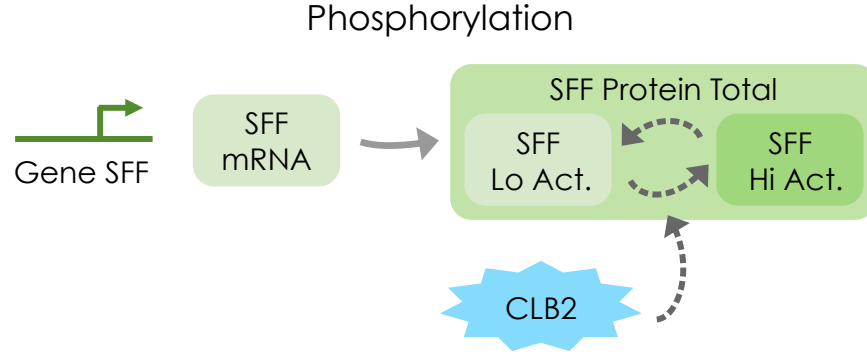


$$\begin{aligned} \dot{YHP1}_{rna} &= a_1 h(SBF_{lo}) + a_2 h(SBF_{hi}) \\ &\quad - \beta_1 YHP1_{rna} + \gamma \\ \dot{YHP1}_{prot} &= k_t YHP1_{rna} - \beta_2 YHP1_{prot} \\ &\quad - k_1(APC)(YHP1_{prot}) \end{aligned}$$

FIGURE 3.3.1: The protein Yhp1 is ubiquitinated by APC, which increases its rate of degradation (moved to state \emptyset , which means removed from system). The equations show a simplified ODE model of this process.

ing proteins that perform a function. Some components may always be available, but other components that are required for the complex to function might only be available at certain times. We then select this limiting component as the proxy for the whole complex. A *group* represents a set of nodes that have similar dynamics and are believed to activate the same targets. One set of dynamics is selected as representative of the group.

These simplifications are accurate as long as the underlying behavior is maintained. For example, in a condition where the dynamics of one member of a group is changed relative to the others, then that group would need to be split and represented by two or more nodes.



$$\begin{aligned}
 \dot{SFF}_{\text{rna}} &= a_1 h(\text{SBF}_{\text{lo}}) + a_2 h(\text{SBF}_{\text{hi}}) + \dots \\
 &\quad - \beta_1 SFF_{\text{rna}} + \gamma \\
 \dot{SFF}_{\text{lo}} &= k_t SFF_{\text{rna}} - \beta_2 SFF_{\text{lo}} \\
 &\quad - k_{1f} \cdot \text{CLB2}_{\text{prot}} \cdot SFF_{\text{lo}} + k_{1r} SFF_{\text{hi}} \\
 \dot{SFF}_{\text{hi}} &= -\beta_3 SFF_{\text{hi}} \\
 &\quad + k_{1f} \cdot \text{CLB2}_{\text{prot}} \cdot SFF_{\text{lo}} - k_{1r} SFF_{\text{hi}}
 \end{aligned}$$

FIGURE 3.3.2: The SFF complex is phosphorylated by the Clb2 which changes SFF from a state of low activity (lo act.) to high activity (hi act.). The equations show a simplified ODE model of this process.

Table 3.4.1: To simplify the model, several proteins are represented by one protein. A protein complex can be represented by protein that is believed to limit the behavior of the whole complex. A group of proteins that have similar dynamics and downstream targets can be represented by one protein.

Name	Proxy	Members	Type
SBF	SWI4	SWI4 SWI6 STB1	complex
MBF	MBP1	MBP1 SWI6 STB1	complex
APC	CDC20	CDC20	complex
SFF	NDD1	NDD1 FKH1 FKH2 MCM1	complex
PLMs	HCM1	HCM1 PLM2 TOS4	group
CLN1 2	CLN2	CLN1 CLN2	group
ACE2 SWI5	ACE2	ACE2 SWI5	group

3.5 Representing Models for Computation

One issue typically encountered in working with models is how to read and write, store, and share information between multiple computer programs. Modeling requires import and export with programs for visually editing the model, editing parameters and equations, visualization, simulation, and optimization. Additionally, we often require a format, at least for the topology, that can be easily read and modified by humans with different modeling needs, such as biologists, mathematicians, and programmers. Generally, no one format meets all our needs for modeling tasks, so we often use a mixture of formats as needed. For models with parameters found by optimization, for example, we have written a module to save networks in the Antimony language (Smith et al., 2009), which is a human-readable text format. This format can then be converted to SBML (Hucka et al., 2003), which is supported by a wide range of analysis, simulation, and visualization packages. We have also written converters to save network models with parameters to python classes that can be simulated with our custom python simulation and plotting tools.

One challenge we considered was how to represent the topology of a network in a way that could be easily read and edited by people (biologists, mathematicians, and programmers), that could be read and written by software, and that could flexibly represent the topology without defining mathematical functions or parameters. A text format was created to represent “regulation equations” along with a parser in C++ to read the format. An example network topology and corresponding regulation equations are shown in Figure 3.5.1.

In the regulation equations, the target gene is on the left-hand side, and all regulators and how they interact are on the right-hand side. The input function for two or more transcription factors are modeled with logic functions using AND and OR gates, which act on the activator/repressor functions. Additionally, we can

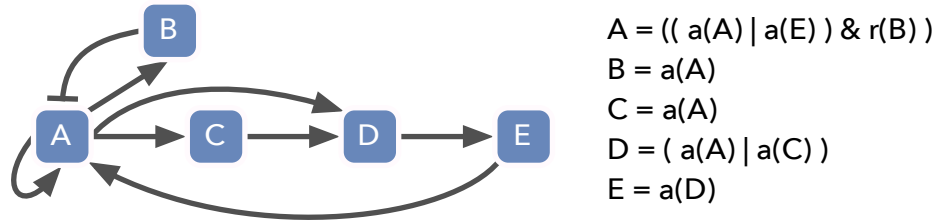


FIGURE 3.5.1: An example of a network topology (left) and the regulation equations (right) used to represent it.

also represent missing information about regulators by replacing them with a “u” for unknown function, and the logic gates by replacing them by a “?” symbol. Each logic gate and its inputs must be surrounded by parentheses so that the order can be interpreted unambiguously. For example, an input function of two activators joined by OR, which are overridden by a repressor would be:

$$W = ((a(X) | a(Y)) \& r(Z))$$

As we allow only two inputs per logic gate, the input function (right-hand side) can be modeled as a binary tree made of nested logics. To translate this nested mathematical structure into a data structure, we view it as an operator tree. Its structure can then be defined as a grammar, shown in Extended Backus-Naur Form (EBNF):

```

<expression> ::= <node>
<node> ::= <regulation> | <operation>
<operation> ::= '(' <node> <operator> <node> ')'
<operator> ::= '&' | '|'
<regulation> ::= 'a(' <factor> ')' | 'r(' <factor> ')'
<factor> ::= <letter>(<letter> | <digit>)*

```

We can now parse the mathematical representation into a tree, shown in figure 3.5.2. This is how the program stores the logic and regulation of the input function. For each transcription factor, all constants needed for the Hill function are stored and

the leaf regulation nodes in the tree point to this information in memory. When we need to evaluate the mathematical expression during simulation, it calls an evaluate function on the tree that will recursively evaluate each subtree. This system is very flexible in that it can store any type of logic function, but it is also good for performance because it does not require any evaluation or searching during simulation time.

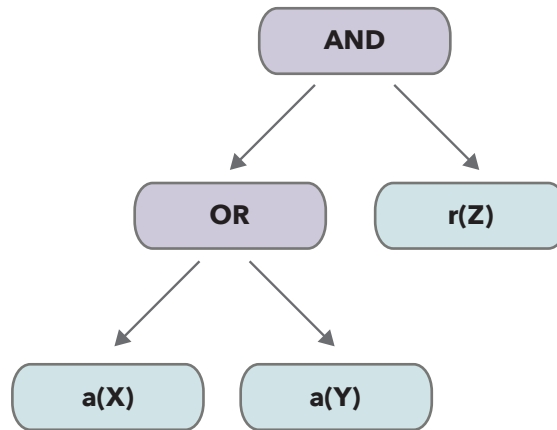


FIGURE 3.5.2: A binary operator tree to represent the regulation logic: $W = ((a(X) | a(Y)) \& r(Z))$

To simulate these models, we developed custom simulation software that handles parsing simplified representations of gene regulatory networks, creating data structures that represent the mathematical models, simulating the models, and tracking their output. Simulations are computationally expensive; when it is necessary to run large numbers of simulations (for example, when finding parameters) it is critical that they be executed as rapidly as possible. Simulations of the system of ODEs are performed by CVODE (Cohen and Hindmarsh, 1996), which is part of the SUNDIALS package. CVODE is fast and offers methods for stiff systems. For example, we simulated 1000 identical oscillating networks with 7 nodes and 15 edges; in Matlab this took 502-631 seconds, while our C++ program with CVODE took 4-5 seconds.

3.6 Conclusions

We have examined many types of mathematical models (for example, boolean networks, bayesian networks, and rule-based networks) and determined that systems of ODEs best match the information we have available and our goals in model building. Within the realm of ODEs, we have examined several formalisms, such as S-systems, linear ODEs, and gene-protein interaction with mass action kinetics, and determined that Hill functions will best meet our modeling needs. We have used these set up several models with parameters that are capable of producing oscillatory dynamics that do not damp. Additionally, we have been able to adjust topologies and parameters to emit a variety of dynamics.

3.7 Future Work

While we have begun to model post-translational modifications, such as ubiquitination and phosphorylation, we believe that these models will need more development and testing. These mechanism are especially important in being able to build models for cell cycle data sets; several of the data sets we study are for mutant conditions that alter proteins responsible for phosphorylation and ubiquitination.

Finding Nodes

The first step in building a GRN model is to identify genes to include. Many methods for detecting periodicity or patterns in data exist; often coming from other areas of science to be applied to biological data. While these methods may perform well for their original area of intended use, it is often unclear how well they will perform on a given biological data set, or on different types of biological data sets. This is a very important factor, given that we are interested in periodic signals that come from different processes in different organisms and were collected in different experimental settings. These signals can therefore vary in their shape as defined by the underlying process, systematic imperfections due to the system or method used to collect data, and the density of samples that were collected. For example, in the time series microarray experiments that we study, there are two to three cycles collected and the sampling densities of different experiments range from 6 to 24 samples per cycle. To recover the fullest sets of periodically expressed genes from different data sets, we would like to quantify the performance of periodicity detection algorithms across a full range of variations present in biological data sets.

4.1 Methods for Detecting Periodicity

After examining many algorithms for periodicity detection, we selected de Lichtenberg (DL) (De Lichtenberg et al., 2005), Lomb-Scargle (LS) (Glynn et al., 2006), JTK_CYCLE (JTK) (Hughes et al., 2010), and Persistent Homology (PH) (Cohen-Steiner et al., 2010). We also consider a new method, SW1PerS, developed by Jose Perea and John Harer at Duke University (Perea and Harer, 2013) and we have been working together on applying it to biological data. These methods each employ different measures of periodicity. A comparative study of the de Lichtenberg, Lomb-Scargle, Persistent Homology, JTK_CYCLE and de Lichtenberg methods was carried out in Deckard et al. (2013). Some, but not all, of the information presented in this chapter overlaps with information in Deckard et al. (2013) and Perea et al. (2014); refer to these papers for additional details on the periodicity detection algorithms and results.

The method described by de Lichtenberg (DL) is a Fourier style method that measures significance using permutations (De Lichtenberg et al., 2005). The method measures both periodicity and amplitude to determine whether a signal is periodic. A background distribution for periodicity for each signal is generated by creating a set of random signals by permuting the samples of that signal. The significance is the proportion of permuted signals with Fourier score at least as large as the original signal's observed Fourier score. A background distribution for the amplitude is generated by creating new profiles from all the profiles by selecting a value from a randomly selected gene profile at each time point. The significance of the regulation (amplitude) is measured as the proportion of permuted profiles with standard deviation at least as large as the observed signal's standard deviation. The implementation in R from Orlando et al. (2008) was used.

The Lomb-Scargle (LS) method was developed for detecting periodicity in astro-

physical data and was designed to be robust to missing time points (Lomb, 1976; Scargle, 1982). It is similar to Fourier style methods; a set of sinusoidal signals that cover a range of periods are compared to the time series to measure the correspondence. The significance of each of these is calculated, and the period of the most significant fit is returned. We use an implementation in R from Glynn et al. (2006) that uses the Lomb-Scargle normalized periodogram as defined in Press and Rybicki (1989).

JTK_CYCLE (JTK) was developed in the biological field to examine circadian rhythms and is a statistical method (Hughes et al., 2010). A set of reference signals, which can be a cosine curve or signal specified by the user, is generated to cover a range of periods and phase shifts. Signals are examined to determine how their points are decreasing or increasing in relation to one another. This is achieved by first converting the values in the signal to a rank, and then a pair-wise comparison of values to determine if each value is greater than or less than each other value. The increasing/decreasing pattern of the time series is then compared to the increasing/decreasing pattern for each period and phase shift of the reference signal to determine the statistical significance of the correlation. It uses the Jonckheere-Terpstra test and Kendall's tau. The implementation in R was provided by the author of the paper.

Persistent Homology (PH) is a method from computational topology (Cohen-Steiner et al., 2010). PH normalizes the data and then creates subtle pairings of minima and maxima of a time series using different window sizes to cover a range of periods of interest. A measure is obtained by summing the difference (persistence) between the maximum and the minimum of each pair. If there is only one minimum and maximum pair, the signal is considered to be a perfect oscillation. Additional oscillations in the time series will create more minimum-maximum pairs, which will increase the score, indicating a less perfect profile. The method is insensitive both to

amplitudes and sinusoidal shape. The last author, Yuriy Mileyko, of Cohen-Steiner et al. (2010) provided an implementation of the algorithm written in C++.

SW1PerS (SW) also comes from the field of computational topology (Perea and Harer, 2013). SW directly measures the periodicity of a signal by quantifying how the signal repeats. It uses a window of size n , and the values within the window define a single point in n -dimensional space. The window slides along the signal, tracing a trajectory through the n -dimensional space. If the signal is periodic with window size n , then as the sliding window encounters the second cycle, the trajectory will come back to its starting point, creating a circular shape. This circular shape is an n -dimensional point cloud, which can be then be normalized and de-noised. To measure the degree of circularity and the size of the hole in the middle, and therefore the periodicity, the 1-Persistent Homology algorithm is used. This method makes no assumptions about the shape of the repeating pattern.

4.2 Synthetic Data

The synthetic data sets were designed to mimic features in biological data sets. The synthetic data are generated with known parameters so that the algorithms' results can be qualitatively measured and compared. Three factors were varied in the synthetic data set: the shape of the signal, the noise applied to the signal, and the sampling density of the signal. All other parameters (except phase shift), were held constant. This synthetic data set contains 1,000 signals for all combinations for Gaussian noise $SD = \{0, 25, 50\}$, # samples = $\{50, 25, 17\}$, and twelve signals (ten periodic and two non-periodic).

The set was created first by selecting a fixed period and amplitude, but the phase shifts were allowed to vary from 0 to the length of the period, selected from a uniform distribution. The amplitude was 100 (peak-to-trough height), period length was 100 and the signals covered 200 units of time, so each signal contained two cycles. The

shapes of periodic signals included were cosine, two cosine signals with different amplitudes, cosine that is peaked, two peaked signals with different amplitudes, cosine with an exponential trend, cosine with a linear trend, cosine that is damped, sawtooth, square waves, and a contracting cosine signal. Non-periodic shapes were flat and linear (Figure 4.2.1, Table 4.2.1). One thousand signals were generated for each signal shape, with no noise (SD=0), and then Gaussian noise was added to these sets with standard deviations of 25% (SD=25) and 50% (SD=50) of the peak-to-trough height. This set of curves was then used to create several datasets with different, evenly-spaced sampling densities: 50 samples, 25 samples, and 17 samples across two periods.

4.2.1 Distinguishing Periodic Shapes

The first analysis measures how well an algorithm can distinguish a periodic signal from a non-periodic signal under increasing levels of noise and decreasing sampling density for each type of periodic shape. The algorithms JTK, LS, PH, which can examine a range of periods, searched for periods of length 40 to 160. The implementations of DL and SW currently only search for single periods, so they were set to search for signals with a period of 100. To measure how well an algorithm can separate two types of signals, we use the Receiver Operating Characteristic (ROC). Given a ranked list of scores and score cutoffs; we can count the number of signals above the cutoff that are periodic (True Positive) or non-periodic (False Positive). Receiver operating characteristic (ROC) curves measure this tradeoff between the True Positive Rate and the False Positive Rate as the score cutoff is varied. The area under curve (AUC) of the ROC curve is a summary of this measure, where an AUC of one indicates all periodic cases were separated from non-periodic cases for some p-value or score (Figure 4.2.2). As expected, performance degrades as the noise increases and the sampling density decreases. However, this does not happen at an

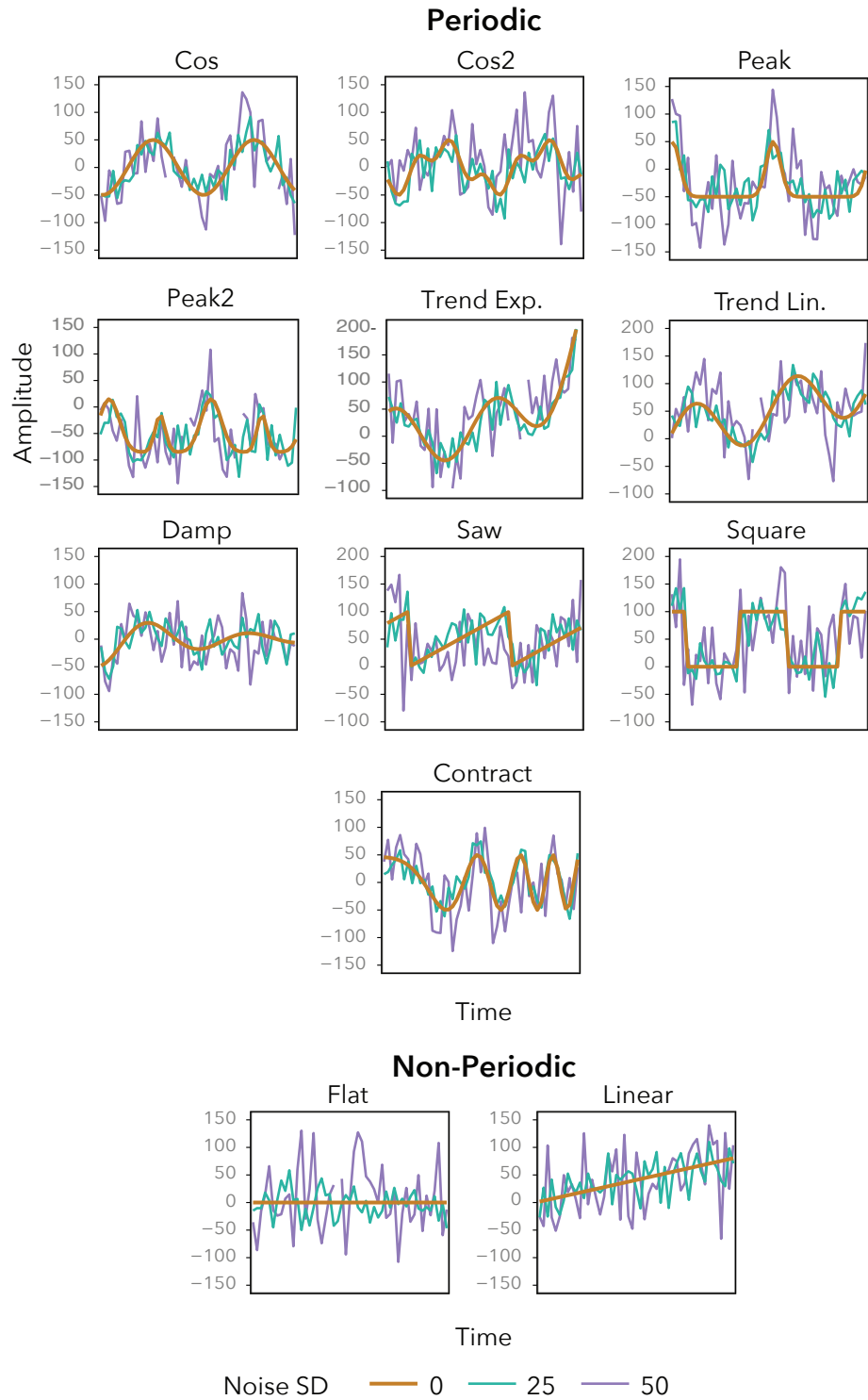


FIGURE 4.2.1: Synthetic signals for testing periodicity detection. There are ten periodic signals and two non periodic signals. Each signal has three levels of noise, first no noise ($SD=0$), and then Gaussian noise with $SD=25, 50$.

Table 4.2.1: Functions for synthetic signals for testing periodicity detection: cosine (cos), two cosine signals with different amplitudes (cos2), cosine that is peaked (peak), two peaked signals with different amplitudes (peak2), cosine with an exponential trend (trend exp), cosine with a linear trend (trend lin), cosine that is damped (damp), sawtooth (saw), square waves (square), and a contracting cosine signal (cont).

Shape	Function f for signal
cos	$f = \text{amp} * \cos(2*\pi/\text{per} * t - \text{pshift}*(2*\pi/\text{per}))$
cos 2	per2 = per * 0.3333 amp2 = amp * 0.50 pshift2 = (pshift + (per2 * 0.25)) % per $f = \text{amp} * \cos(2*\pi/\text{per} * (t - \text{pshift}))$ $+ \text{amp2} * \cos(2*\pi/\text{per2} * (t - \text{pshift2}))$ The combined signal height is then adjusted to the original amplitude
peak	peak = 20 $f = \text{amp} * (-1 + 2 * \text{fabs}(\cos(\pi/\text{per} * t - \text{pshift}*(\pi/\text{per}))))**\text{peak}$
peak2	peak1 = 10, peak2 = 40 amp2 = amp * 0.70 pshift2 = (pshift1 + (per * 0.5)) % per f = amp1 * (-1 + 2 * abs(cos(pi/per * (t - pshift1))))**peak1 + amp2 * (-1 + 2 * abs(cos(pi/per * (t - pshift2))))**peak2
trend exp	trende = 0.027 $f = \text{amp} * \cos(2*\pi/\text{per} * (t - \text{pshift}))$ $+ \exp(\text{trende} * t)$
trend lin	trendl = 0.5 $f = \text{amp} * \cos(2*\pi/\text{per} * t - \text{pshift}*(2*\pi/\text{per}))$ $+ (\text{trendl} * t)$
damp	damp = 0.01 $f = \text{amp} * \cos(2*\pi/\text{per} * t - \text{pshift}*(2*\pi/\text{per})) * \exp(-\text{damp}*t)$
saw	$f = 2 * \text{amp} * (((t - \text{pshift}) \% \text{per}) / (\text{per}-1))$
square	$f = 2 * \text{amp} * \text{round}(((t - \text{pshift}) \% \text{per}) / (\text{per}-1))$
contract	$f = \text{amp} * \text{math.cos}(2*\pi/\text{per} * (t ** 2 / \text{per} - \text{pshift}))$
flat	0
linear	(slope * t)

even rate for all shapes. The ability to detect contracting signals degrades the fastest for all algorithms except PH. All the algorithms have difficulty separating periodic damped shapes from non-periodic shapes. The algorithms performance on cosine and square signals degrades the least quickly; at almost every noise level and sampling rate these have the highest AUC scores. SW performs on average as well as JTK and LS for most noise and sampling density combinations. In the most challenging case, with the highest noise and lowest sampling density, SW outperforms LS and JTK.

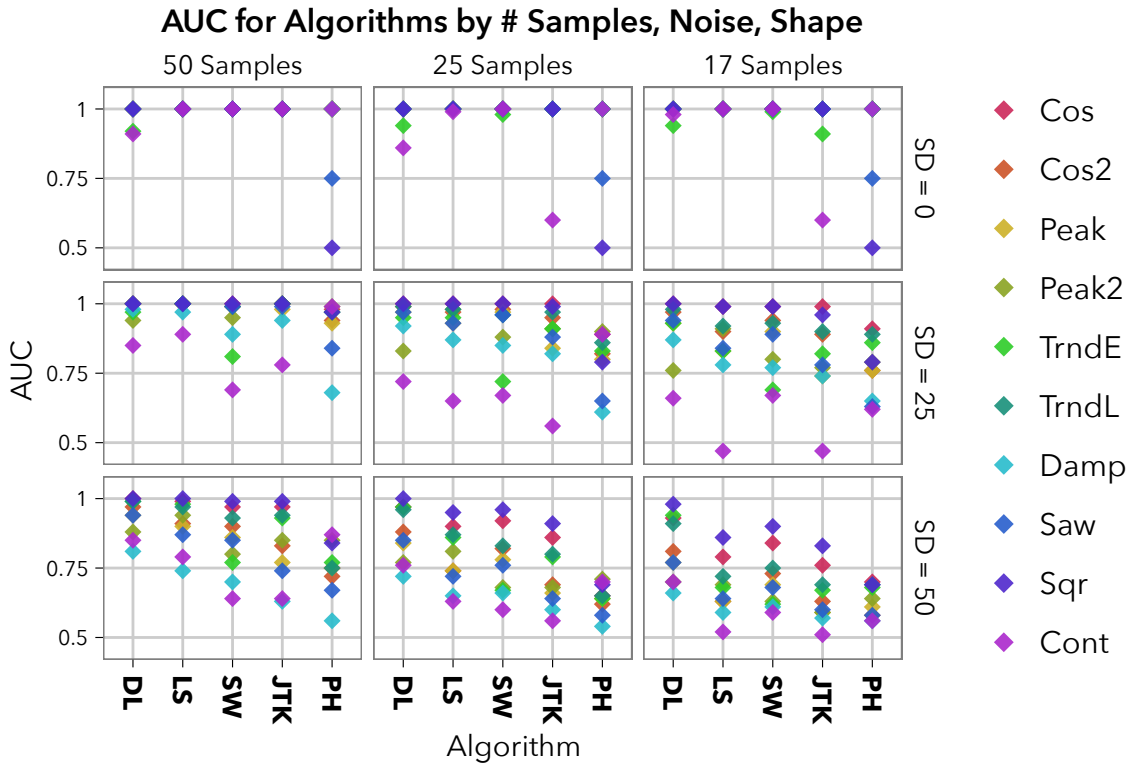


FIGURE 4.2.2: Algorithm performance on separating period from non-periodic signals. These plots show the degradation in performance for classifying periodic signals, from non-periodic signals (linear and flat) under increasing Gaussian noise with standard deviation = $\{0, 25, 50\}$ and decreasing number of samples = $\{50, 25, 17\}$. One thousand signals were created for each noise level, number of samples, and shape. The $-\ln$ (p-value or score) was used. (Perea et al., 2014)

In our second analysis, we examined whether the algorithms have biases for or against the different signal shapes. The differences in how each algorithm defines

periodicity can be seen by plotting the distribution of scores for each shape (Figure 4.2.3). Bias is indicated when the distribution of the scores for shapes are separated from one another. For JTK and LS, we can see a strong bias for cosine signals, which receive the best scores. DL groups most of the signals together, except double peak and contracting signals that receive worse scores and the trended signals receive a wide range of scores, from best to worst. Both JTK and LS exhibit a strong bias for cosine shapes. JTK's next preference is for peaked shapes. After cosine, LS prefers square and cosine double signals. For SW, there is a mixture of cosine, cosine 2, cosine damped, and square signals near the top of the rankings. These are followed closely by peaked and sawtooth signals.

4.2.2 Sampling Density vs. Experiment Length

One question that arises in designing experiments to capture periodic behavior is whether it is better to densely sample fewer periods or to sparsely sample more periods. To test how the number of cycles alters algorithm performance, we constructed one data set with period = 100 and peak to trough amplitude = 100. We applied Gaussian noise with SD = {0, 25, 100}. We then took a fixed number of samples covering either 1, 2, or 3 cycles. Due to restrictions with PH and JTK with running on one or more periods, we had to take additional samples to make it 1.08 periods (with 56 and 28 samples instead of 50 and 25 samples, respectively). We made data sets for 50 samples and for 25 samples. We ran these data sets through the algorithms searching for periods 96 to 108. We then generated ROC plots and computed the AUC. We plotted the AUC to show how well each algorithm performs at separating periodic from nonperiodic signals. When viewing the results, we can see that performance on damped signals degrades greatly as more periods are included. This is expected; the more cycles we include, the more the signal become damped and is obscured by noise. The performance of JTK and LS on trended signals appears

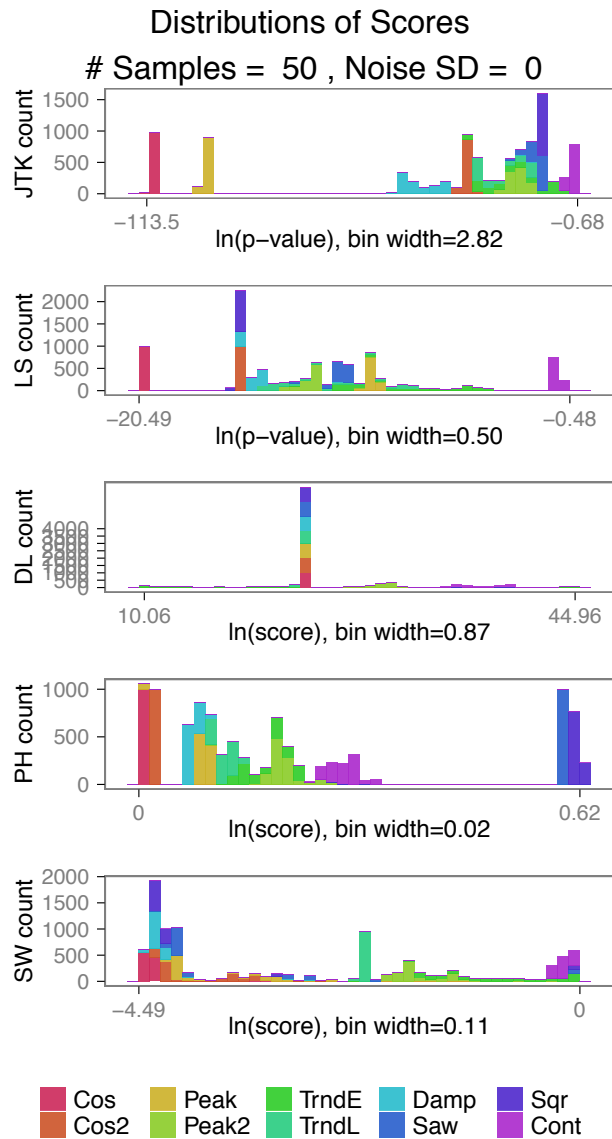


FIGURE 4.2.3: Biases for curve shapes for each algorithm (rows). Distributions of scores are by shape with no noise (Gaussian noise $SD=0$). The x-axis shows the log of the scores, ranging from the lowest (best score) to the highest (worst score) returned by the algorithm. The y-axis shows the number of signals receiving the score. (Perea et al., 2014)

to decrease going from 2 to 3 periods, while their performance on peaked increased slightly on most cases.

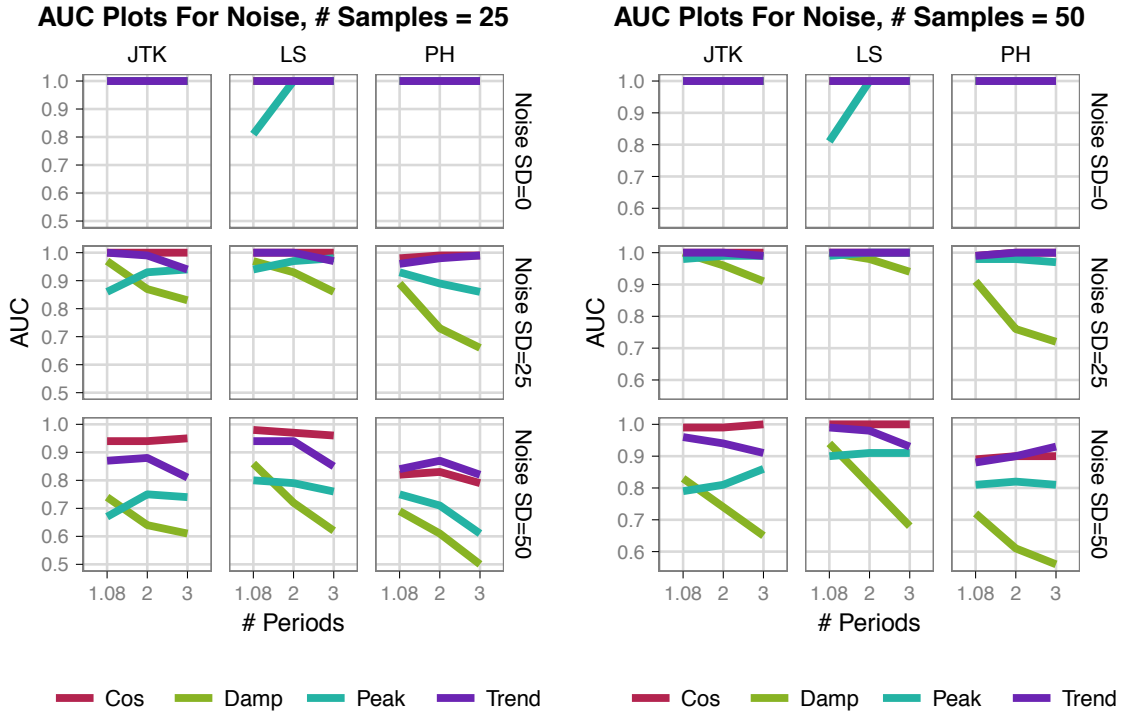


FIGURE 4.2.4: Performance on detecting periodic signals for different numbers of cycles. The AUC was computed for the algorithm’s results on 1, 2, or 3 periods with a given number of samples. Due to restrictions with PH and JTK with running on one or more periods, we had to take additional samples to make it 1.08 periods (with 56 and 28 samples instead of 50 and 25 samples).

4.3 Biological Data

To see how the algorithms performed on biological data, we ran the algorithms on three different data sets. The microarray experiments discussed here were designed to measure periodic gene expression in different processes and organisms, which might include different signal shapes.

The yeast cell cycle wild-type data (WT) from Orlando et al. (2008) shows pe-

riodic gene expression during the cell cycle in budding yeast, *S. cerevisiae*. A population of wild-type cells were synchronized and samples were taken at 16 minute intervals. Only the first replicate was used. The period for the cell cycle in this experiment is estimated to be approximately 95 minutes, and the data sets cover a recovery period and roughly two cell cycles. This data set contains 15 samples, but only the last 13 are used to omit a stress response. The number of samples per cycle is: $13/2 = 6.5$ samples.

The yeast metabolic data from Tu et al. (2005) is from *S. cerevisiae* that were grown to a high density, briefly starved and then given low concentrations of glucose. Samples were taken approximately every 23-25 minutes (sampling was not even at all time points). As some of the algorithms cannot handle uneven sample intervals, we evened the sample times in the data by changing the unevenly sampled times to every 24 minutes. The yeast metabolic cycle is estimated to be 300 minutes, and this data set covers approximately three cycles and has 36 samples. The number of samples per cycle is: $36/3 = 12$ samples.

The mammal circadian rhythm data from Hughes et al. (2009) is from wild-type mice (*M. musculus*) that were synchronized by entraining them to an environment with 12 hours of light and 12 hours of dark for one week. They were then placed in total darkness. Samples were taken from the liver of several mice every hour. The period of the circadian rhythm is 24 hours, and this data set covers two circadian cycles and has 48 samples. The number of samples per cycle is: $48/2 = 24$ samples.

The yeast cell-cycle data set is less densely sampled than the other data sets, and appears less noisy. The yeast metabolic cycle and mammal circadian data appear to be noisier than the yeast data sets, and are more densely sampled.

We have used SW, LS, JTK, PH, and DL to evaluate our data sets. LS, JTK, and PH were run on the same range of periods for each data set. As DL and SW only support searching for one period, we ran them on the estimated period.

4.3.1 *Preference in shapes*

In Figure 4.3.1, the top scoring gene expression profiles from each algorithm on each data set are shown. All of the probes were rank ordered on p-value (LS, JTK) or score (DL, PH, SW). Several probes received the same top score: cell cycle PH: 253, metabolic cycle DL: 26, circadian DL: 52. In these cases we sorted by probe id and selected the top ten. Only profiles with annotated symbols were included.

We used these results to illustrate what preferences the algorithms display for the shapes of the signals present in each of the datasets. Shapes very similar to cosine curves appear in the top scores from LS; and these shapes have similar peak heights. We can see curves that look equally periodic, although less similar to cosine shapes, in the other algorithms' top scores. The DL algorithm prefers higher amplitude profiles, which works well for the yeast cell-cycle data: two of the top five scores shown are cell-cycle genes (NRM1 and CLB1). In the circadian data set, however, DL appears to give high rankings to very noisy signals (Egr1). There are also many profiles that are more peaked in the yeast metabolic data, and both JTK and DL give these peaks good scores. PH detects extremely peaked profiles, which works well for the yeast metabolic data, but returns results that do not appear to be periodic for the mammal circadian (Adam1b). These appear to be singular peaks which might be more associated with noise than with periodic expression. However, in the yeast metabolic data, many of the top results show very steep peaks (AAH1); these appear to be truly periodic profiles as they have the expected three peaks and are regularly spaced. SW gives top rankings to a variety of shapes; some are more peaked (Coq10b in circadian) and some are flatter at the top (HHT1 in metabolic).

4.3.2 *Differences in p-values/scores between algorithms*

It would be useful to be able to compare the results from several algorithms. However, selecting sets of periodic genes by p-value or score poses an interesting problem. We

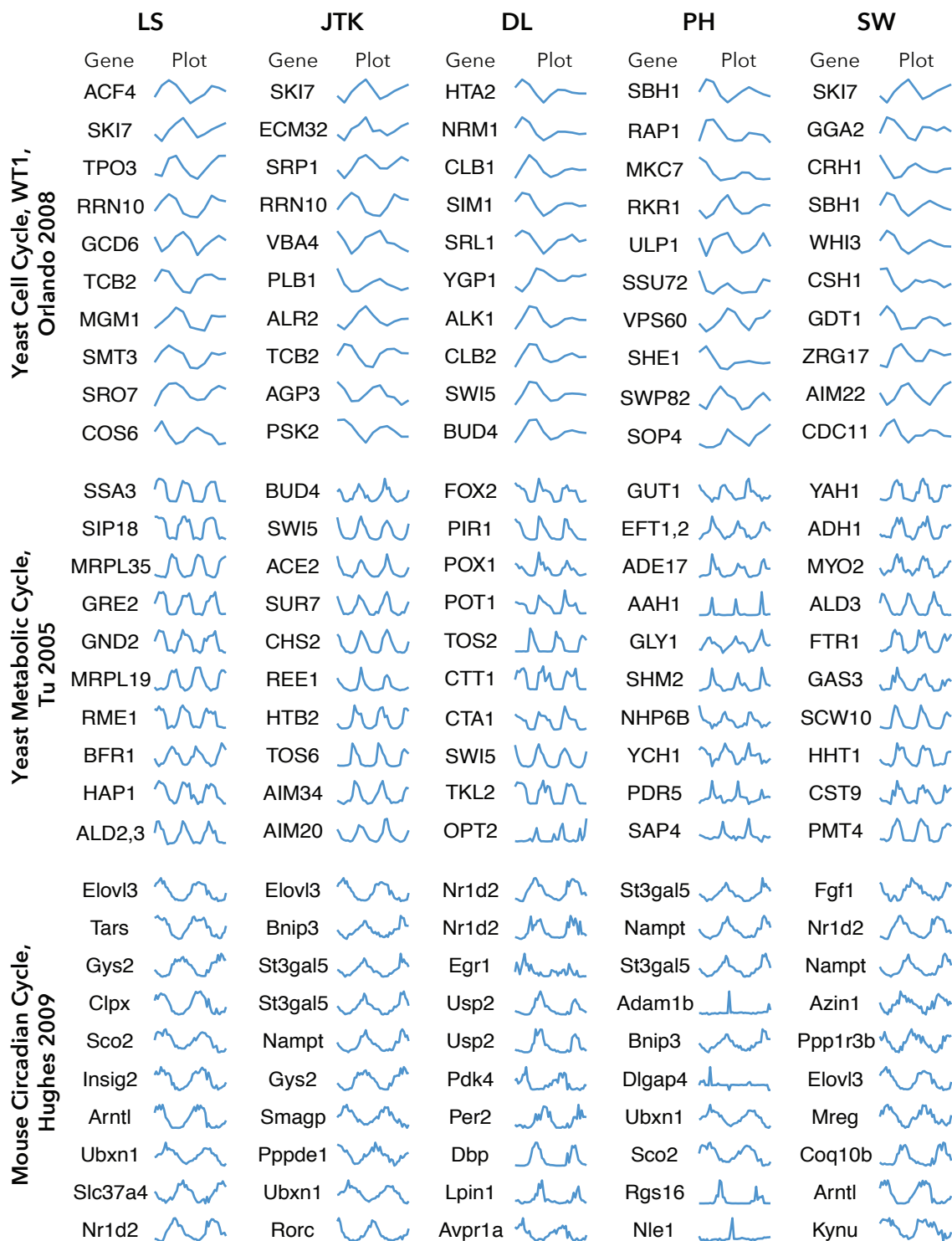


FIGURE 4.3.1: Top ranked signals from each periodicity detection algorithm. The plots are normalized from the minimum to the maximum of the signal. Probes that did not have a symbol name were excluded.

Table 4.3.1: Numbers of genes selected as periodic for cutoffs on p-values and q-values. These are computed for each data set for algorithms that return significance measures (LS and JTK).

Data Set	Alg	p<0.1	p<0.05	p<0.01	q<0.1	q<0.05	q<0.01
Cell Cycle	LS	111	4	0	0	0	0
	JTK	1334	800	208	0	0	0
Met. Cycle	LS	4303	3744	2660	3550	2918	1819
	JTK	5043	4626	3828	4626	4237	3396
Circadian	LS	10331	6570	3849	3624	2845	1577
	JTK	13597	9991	6204	6830	5607	3935

do not have a way of directly comparing scores, as returned by PH, DL, and SW, to p-values, as returned from JTK and LS. For the algorithms with p-values, selecting one significance level can return gene lists with very different numbers of genes. For example, in the yeast cell-cycle data, there are 800 probes with $p < 0.05$ from JTK, but only 4 genes from LS. We do not therefore have a method for selecting a cutoff for each method that is based on some common measure of periodicity or significance.

For the algorithms that return significance measures, JTK and LS, we counted the number of genes at several cutoffs for p-values and q-values (Table 4.3.1). The q-values were calculated using Benjamini-Hochberg FDR (Benjamini and Hochberg, 1995). For the yeast cell cycle, but on an older set of data, 800 probes were identified using a Fourier algorithm with correlation to measure periodicity (Spellman et al., 1998). On the same cell cycle data set we used here, 1,271 genes were identified as having periodic transcription using DL (Orlando et al., 2008). For the yeast metabolic cycle, 3,552 genes were called periodic using LS to determine the period and autocorrelation to measure the periodicity ($p < 0.05$) (Tu et al., 2005). For the mouse liver circadian data, 3667 periodic genes were identified by the intersection of the COSOPT and Fisher’s G approaches ($q < 0.05$) (Hughes et al., 2009); In another study on the same data set, 5425 were identified as periodic using JTK_CYCLE (BH $Q < 0.05$) (Hughes et al., 2010).

Another issue is how to interpret and compare the p-values that are returned by these methods. LS and JTK return p-values, and these p-values change if the sampling density changes. Therefore the periodicity of two identical profiles, one with twice as many time points as the other, are not directly comparable using the final p-values from these algorithms. Additionally, The work of Kallio et al. (2011) and Futschik and Herzel (2008) suggest that the significance of the results is generally overestimated, which can be caused by the null model having more randomness than exists in the data. This causes the significance values to be excessively optimistic and overestimates the number of periodic genes.

4.3.3 Finding Dominant Periods

In many cases, the exact period length in a set of data is unknown or there could be multiple harmonics in the dataset. In these cases, it is desirable to be able to determine the dominant period or periods that exist in a dataset. To do this, we can run the dataset through algorithms, such as LS or JTK, that can search for a wide range of periods within the dataset. These algorithms compute the p-values for every period examined for each signal. We can then look at the number of signals that have p-values below a given cutoff for each period length. These can be plotted as a distribution, and the peaks indicate where there are dominant periods within the data sets.

The yeast cell cycle data sets wt and clb1-6 from Orlando et al. (2008) and cdc28 from Simmons Kovacs et al. (2012), were trimmed to 13 time points to omit the recovery period that contains a stress response. The start times, end times, sampling interval, and total length of time covered by the data are shown in Table 4.3.2. The length of time covered by these datasets are different; for the wt and clb1-6, the time covered is 192 minutes, while for the cdc28 data, the length of time is 240 minutes. We ran these data sets with JTK and LS to compute p-values for

Table 4.3.2: The start and the end time of the replicate 1 and 2 (r1, r2) for wild-type (wt), clb1-6, and cdc28 yeast cell cycle data sets. The total time covered by the data sets.

Data Set	Time Start	Time End	Interval	Time Total
wt r1	62	254	16	192
wt r2	70	262	16	192
clb1-6 r1	70	262	16	192
clb1-6 r2	62	254	16	192
cdc28 r1	50	290	20	240
cdc28 r2	50	290	20	240

period lengths of 50 to 200. This corresponds to searching for approximately 4 cycles to 1 cycle within the datasets: $192/50 = 3.8$ cycles to $192/200 = 0.96$ cycles for wt1 and clb 1-6; and $240/50 = 4.8$ cycles to $240/200 = 1.2$ cycles. We then plotted the period versus the number of probes with a p-value below a given cutoff (Figure 4.3.2). There is a peak at 100 for wt1 and wt2 that indicates a potential dominant period. The lengths of the cycles for wt1 and wt2 were previously estimated in Orlando et al. (2008) using CLOCCS Orlando et al. (2007). The estimate for wt1 was 77.1 minutes for mother cells and 118.5 minutes for daughter cells (77.1 plus daughter specific phase of 41.4); if we use the average as a rough approximation of how the mother and daughter cycle lengths might combine, it yields 97.8 minutes. The estimate for wt2 was 85.0 minutes for mother cells and 120.1 minutes for daughter cells (85.0 plus daughter specific phase of 35.1); averaging these yields 102.55 minutes. These CLOCCS estimates of approximately 100 minutes for both wt replicates are very close to the period of 100 estimated from the p-value distributions presented here.

The mouse liver circadian rhythm data was run through JTK and LS searching for periods of length 4 to 48 hours. When plotted as number of probes with a p-value below the cutoff versus the period, we can see the dominant peak at 24 hours, as expected (Figure 4.3.3). There are also smaller peaks at 12 and 8 hours .

Number of Probes for p-value Cutoffs by Period

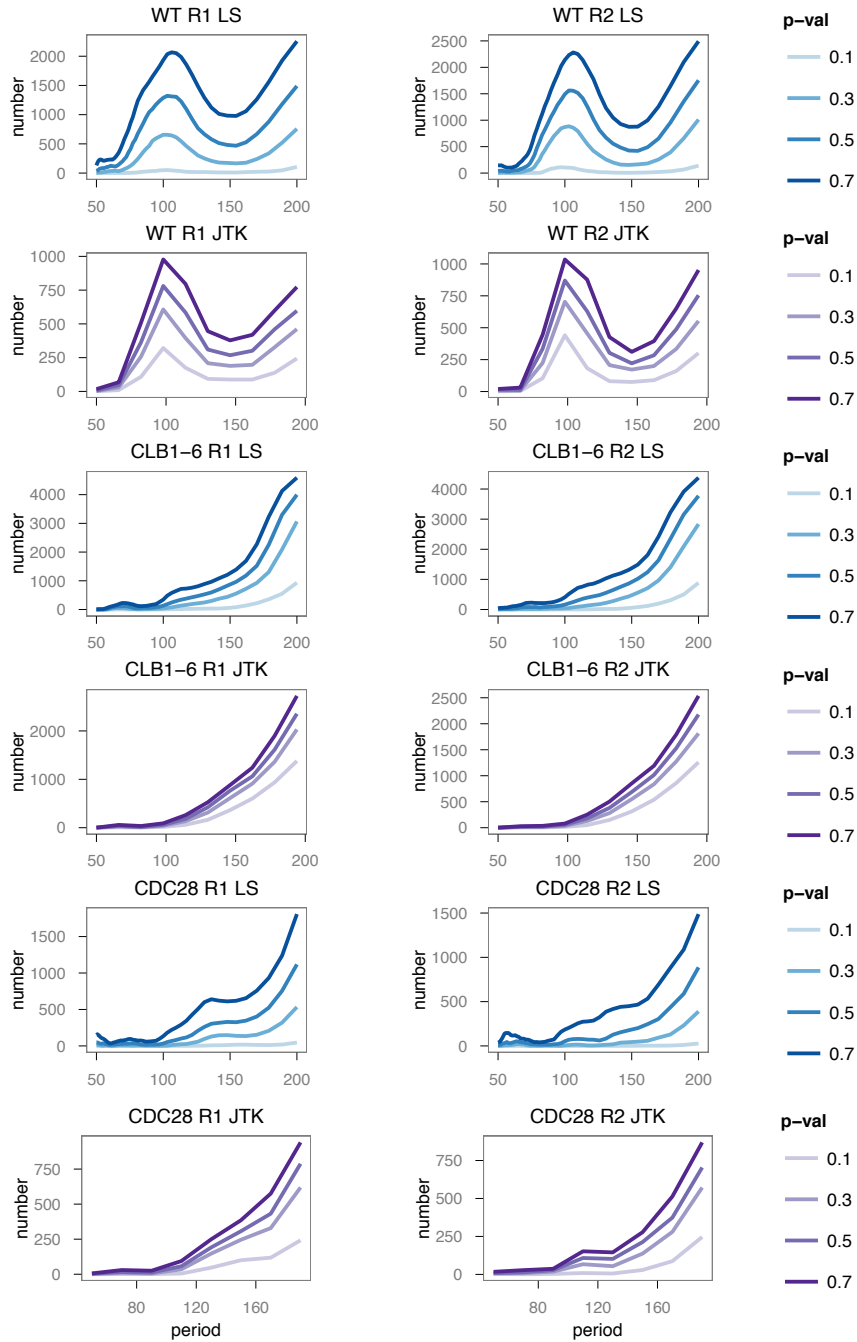


FIGURE 4.3.2: To detect dominant periods within a data set, we plot the number of probes that have p-values less than given cutoffs. Shown are replicate 1 and 2 (r1, r2) for wild-type (wt), clb1-6, and cdc28. Plots are from the p-values from LS and JTK.

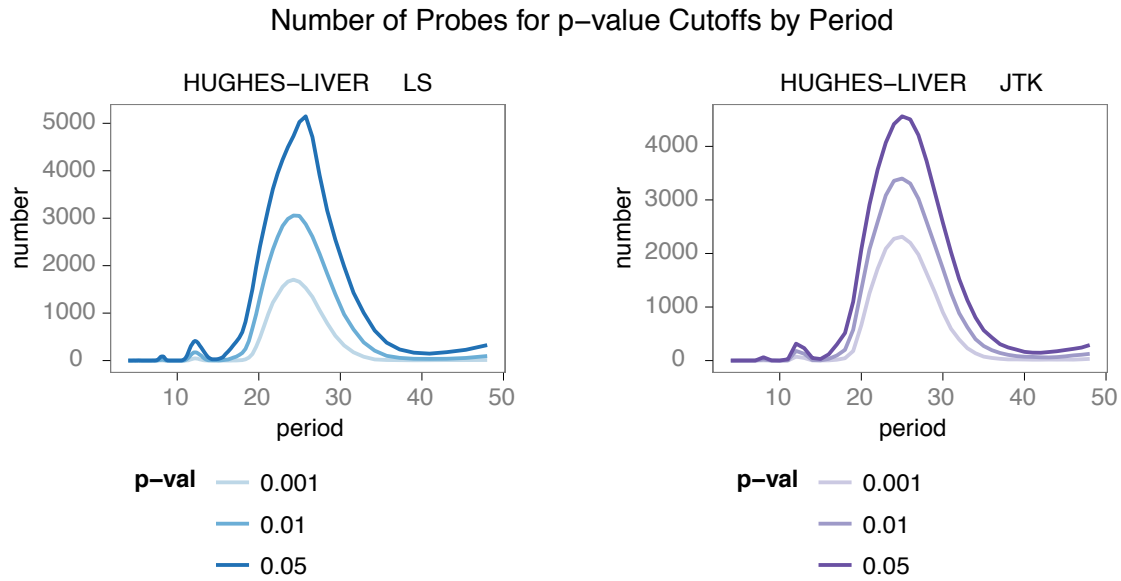
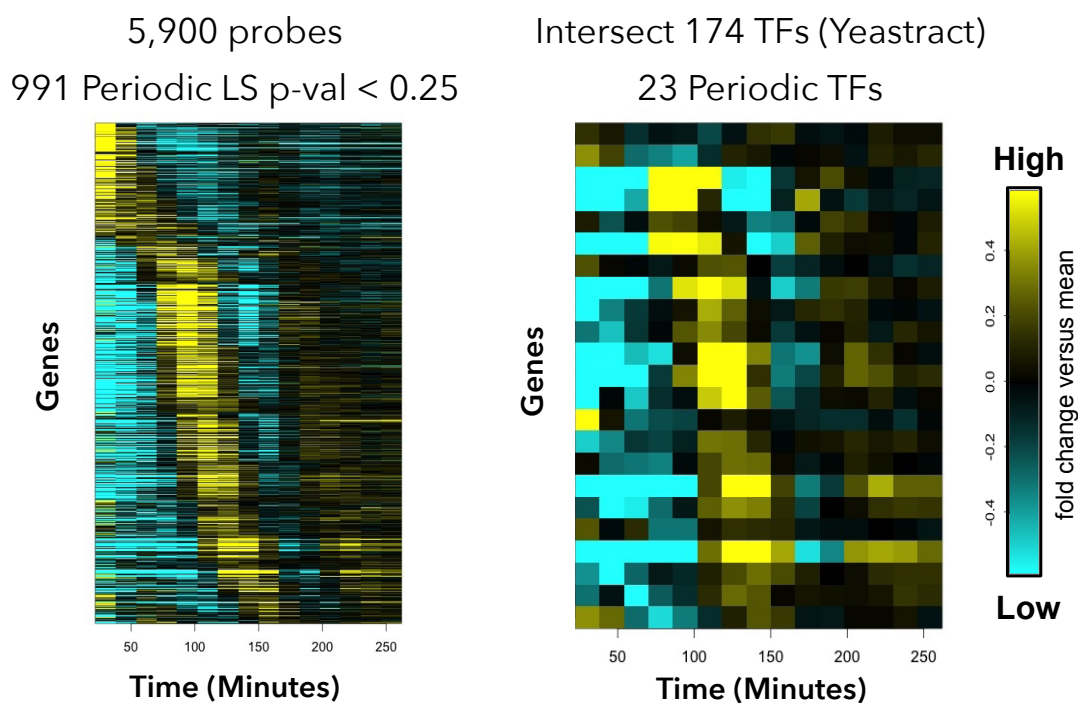


FIGURE 4.3.3: To detect dominant periods within a data set, we plot the number of probes that have p-values less than given cutoffs. Shown are replicate 1 and 2 (r1, r2) for wild-type (wt), *clb1-6*, and *cdc28*. Plots are from the p-values from LS and JTK.

4.3.4 Constructing lists of periodic genes

To construct lists of periodic genes to use as nodes, we take all genes above certain cutoffs from the algorithms. As we are mainly interested in transcription factors, we then take the intersection of the periodic genes lists and lists of known transcription factors in the system. For budding yeast, we use the list of transcription factors from Yeastract (Teixeira et al., 2006) (Figure 4.3.4). For the mouse, we use the list of transcription factors from Ravasi et al. (2010), with the transcription factor *Cry2* added (Figure 4.3.5).

It is important to note that there is no “one list” of periodic genes that are generated from this analysis. Each gene list is selected for a need, and a different list can be selected to fulfill a need for a different type of analysis. For example, we may wish to relax the confidence requirements to make a bigger list or select only results from certain algorithms to use the shape bias of the algorithm. Additionally, we



Heatmaps by Christina Kelliher, Bio Dept., Duke University

FIGURE 4.3.4: Heatmaps of the gene expression for periodic genes from the yeast cell cycle from selecting a cutoff (left) and taking the intersection of known transcription factors (right). The rows are genes, and the columns are times points. Heatmaps were generated by Christina Kelliher, Biology Department, Duke University.

may relax the restriction of only using TFs; we might want “terminal nodes” in the network that do not regulate other genes (we call these “lightbulbs” of the system). For building more mechanistic models, it might be desirable to include genes that are not TFs, but are involved in post-translational modifications, such as kinases.

4.4 Conclusions

Our findings suggest that curve shape has the largest impact on the scoring of biological signals by these periodicity detection algorithms, especially under conditions of higher noise or lower sampling rate. Algorithms such as LS, DL, and JTK rely on comparing data to reference curves (LS and DL assume a sinusoidal curve, JTK can

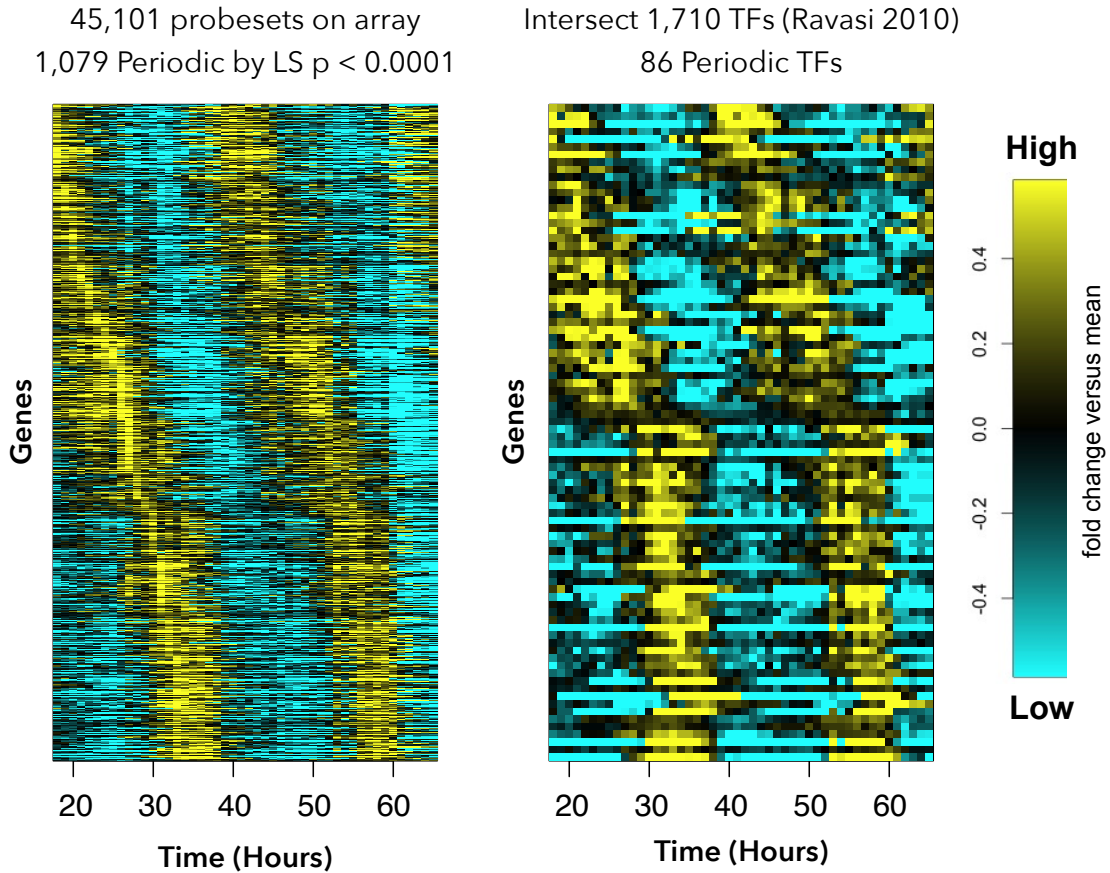


FIGURE 4.3.5: Heatmaps of the gene expression for periodic genes from the mammalian circadian rhythm from selecting a cutoff (left) and taking the intersection of known transcription factors (right). The rows are genes, and the columns are time points. Heatmaps were generated by Christina Kelliher, Biology Department, Duke University.

use a user-specified curve); therefore, they will perform most accurately when the data matches the reference curves. In an ideal situation with no noise and high resolution, all the algorithms perform well at distinguishing periodic from non-periodic signals. As noise increases or sampling rate decreases, the ability of these algorithms to classify non-standard curve shapes degrades much more rapidly than for true cosine shapes.

The algorithms also show differences in the shapes of signals that they rank most

highly. JTK and LS show a strong bias for cosine signals, DL ranks most shapes at an intermediate level, and SW1PerS gives a mixture of cosine, cosine 2, cosine damped, and square signals top rankings. Given their different underlying ideas of periodicity, these algorithms could be used together to recover a more comprehensive set of periodic signals within a data set. This is especially useful when the signal attributes are not known beforehand.

Mostly importantly for our process of building network models, we have created lists of nodes (genes) that can be used in the next steps of the process.

4.5 Future Work

For future work, it would be of great interest to collect other algorithms, especially newer algorithms that show any performance improvement over existing algorithms. We have mainly focused on cosine-based shapes, but biological signals can contain non-cosine based curves and those that reflect responses to complex interactions. The type and amount of noise and also sampling densities should also be expanded to encompass what is seen in data collected from other technologies. For example, the microarray data we have shown here is collected for 2 or 3 cycles and has less than 50 samples. We would like to also explore using different noise models, as these could vary by organism, system, and technologies used. It would also be of interest to study these methods on sampling rates that reflect the longer and higher resolution time series that are seen in single-cell time lapse microscopy.

Finding Edges

There are many obstacles to discovering edges in a gene regulatory network. We aim to overcome these obstacles by using multiple data sets, where available: replicates, experiments from different conditions, from different labs, and using different technologies. This will facilitate working with larger networks, overcoming variability in noise and quality, and also finding behaviors under different conditions.

5.1 Using biological data to identify edges

The first type of biological evidence we have been incorporating to identify edges is genome-wide binding analysis. This method takes a set of known transcription factors and identifies where they bind in the genome. In Simon et al. (2001), the genome-wide targets of nine known cell cycle transcription factors (Mbp1, Swi4, Swi6, Mcm1, Fkh1, Fkh2, Ndd1, Swi5, and Ace2) were identified using ChIP-chip. In Lee et al. (2002), 141 transcription factors, including twelve that were specifically identified as cell cycle related, had their binding sites mapped using ChIP-chip. In Harbison et al. (2004), 203 transcription factors, which were considered to be a nearly complete set of all TFs in the yeast genome, mapped to their binding locations using

ChIP-chip. In Workman et al. (2006), 30 transcription factors (including seven cell cycle genes: Mcm1, Swi4, Swi6, Fkh2, Ndd1, Ace2, Swi5), were mapped to their binding locations for two conditions: control (grown on YPD) and stress (exposed to MMS) using ChIP-chip. Following the method in Simmons Kovacs et al. (2012), these p-values were combined using Fisher's method.

So that we could use data from these ChIP-chip experiments, we needed to map the results to a common set of gene names and combine the p-values. These experiments were loaded into the database we developed and mapped to the standard set of budding yeast genes downloaded from SGD (Cherry et al., 2011). In the database we now have 1,917,021 p-values for 1,207,994 edges, which are used to calculate combined p-values and return filtered results. This covers 197 TFs and 6,382 target genes. While we have been incorporating ChIP-chip data to reveal potential edges in the network, this indicates only binding, and not direct regulation. Additionally, this binding information is from a limited set of conditions and includes many false positives and false negatives.

Additional information that is manually curated is also stored in a database we have been developing. Sara Bristow, Adam Leman, Christina Kelliher, and I have manually collected, verified, sourced, and recorded additional data on potential edges for the network. This includes data from experimental studies, other databases, and literature. We collected data on transcription factors not included in the ChIP-chip studies, and also on the function of the regulation: activation, repression, both, unknown. One source of information is Yeastract, as it provides a huge resource on edges in budding yeast and the experiments used to determine edges (Teixeira et al., 2006). Additionally, data on proteins that are not transcription factors, such as cyclins (Clbs and Clns) and the APC, and their affects on downstream targets (increased activity, reduced activity, degradation, localization) are collected. Other information can also be collected, such as the time when the genes start to turn on

(as measured by 1/2 their left edge in a given cycle), which indicates the timing of their activation, or when the genes are at their maximal expression, which indicates when they might be most likely to be regulating their downstream targets.

We can then query this database to generate networks based on evidence; not whole functional models, but lists of edges given user-defined restrictions on sets of nodes, edge types, and confidence criteria. These lists of data can then be used in Cytoscape (Shannon et al., 2003) to automate drawing network diagrams (Figure 5.1.1). In this example, we have:

- selected a list of periodically transcribed genes from our node-finding steps,
- taken only those that are transcription factors or cyclins,
- taken the nodes for which we have edge evidence,
- selected a cutoff for ChIP-chip p-values and included manually curated edges,
- included information on the edge mechanism (transcriptional, post-translational, etc.) and regulation function,
- included the timing of expression based on when the expression is 1/2 its peak (1/2 the left edge).

This allows us to compile “evidence-based” networks that we can use for further network construction. Additionally, these lists of edge information can be used as priors for inferring edges.

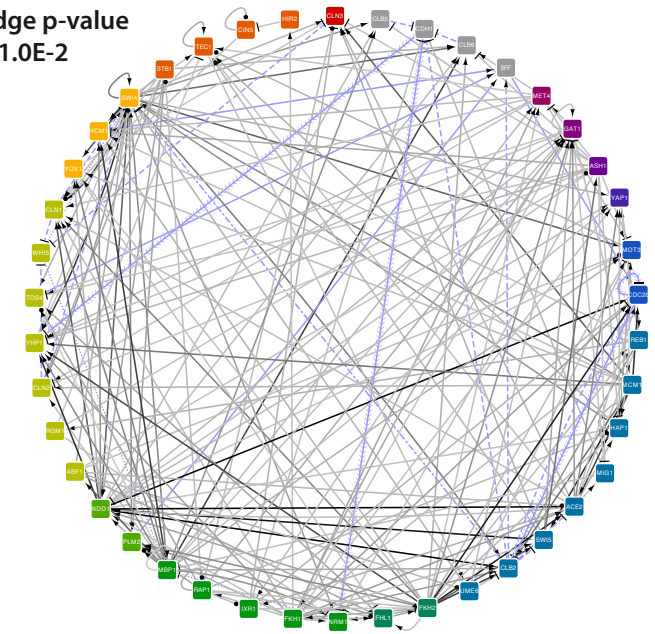
5.2 Computationally-predicted binding

Previous studies in yeast (Badis et al., 2008; Macisaac et al., 2006; Zhu et al., 2009) have identified the DNA-binding specificities for transcription factors using data from protein binding microarrays, ChIP-chip, and other types of experiments. The

- + Periodically Expressed
- + Transcriptions Factors or Cyclins
- + Confidence (Line Color)
 - Literature
 - Combined CHIP-chip p-val
 - Higher Confidence
 - Lower Confidence
- + Mechanism (Line Type)
 - Phosphorylation
 - Physical
 - Physical/Phosphor
 - Transcriptional
 - Ubiquitination
- + Regulation (Line End)
 - Activator
 - Repressor
 - Unknown
- + Expression Timing (Node Color)

G1	S	G2/M		

A) Edge p-value < 1.0E-2



B) Edge p-value < 1.0E-6

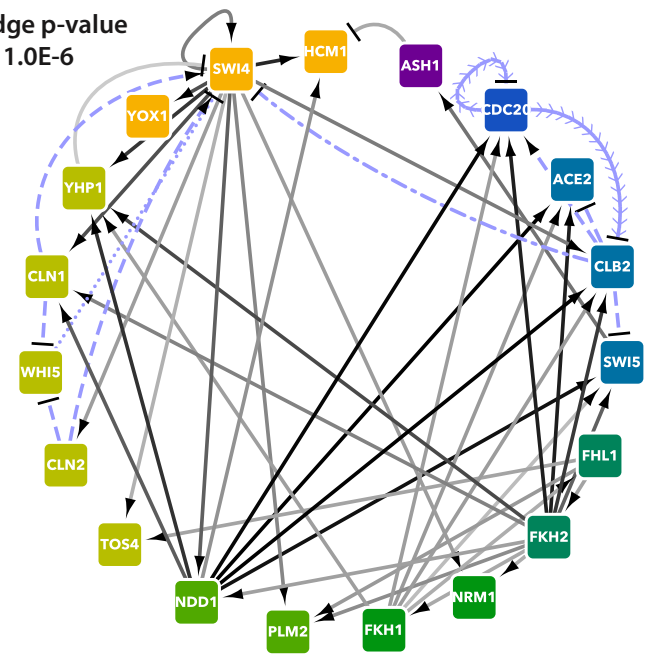


FIGURE 5.1.1: Networks built from biological evidence of regulation. These networks are visualized with cytoscape from data in the database.

binding sequence specificities between a TF and a DNA sequence can be expressed as position-specific scoring matrices (PSSMs). A manually curated set of PSSMs from the studies above is available from JASPAR (Portales-Casamar et al., 2010), which includes Ace2, Fhl1, Fkh1, Fkh2, Hcm1, Mbp1, Mcm1, Swi4, Swi5, Yhp1, and Yox1. These can then be used to predict the binding at other sequences, such as promoters, across the genome. Once again this indicates only binding, and not direct functional regulation.

The COMPETE software package by Wasson and Hartemink (2009) models the binding of nucleosomes, transcription factors, and other DNA binding factors to the genome. It uses binding specificities from PSSMs and concentrations of the factors as input to the model. Binding of all the components are modeled simultaneously, which allows for competition between the factors. The output is a probabilistic description of occupancy for each binding factor at each position in the DNA sequence.

To model a changing system, the cell cycle was divided into five time segments. In each segment, we selected transcription factors that were becoming active, as indicated by their increasing levels in the wild-type microarray data. The PSSMs from Macisaac et al. (2006) were used, as this set had more cell cycle genes of interest. We then selected three TFs per segment, and set their concentrations ($=0.01$) to turn them “on”, with all other transcription factors being left “off”. COMPETE was run on each time segment, to compute the occupancy of the transcription factors and nucleosomes in the promoter regions of 76 periodic transcription factors. The section of the genome covering the gene, 5000 bp upstream, and 5000 bp downstream of the gene were evaluated by COMPETE. We examined the occupancies for 500 bp upstream of the start site, which is assumed to include the promoter region (Figure 5.2.1).

For the promoter region of HCM1 in the first time segment, COMPETE predicts that Mbp1, Swi6, and Swi4 are binding and that a nucleosome has been ejected.

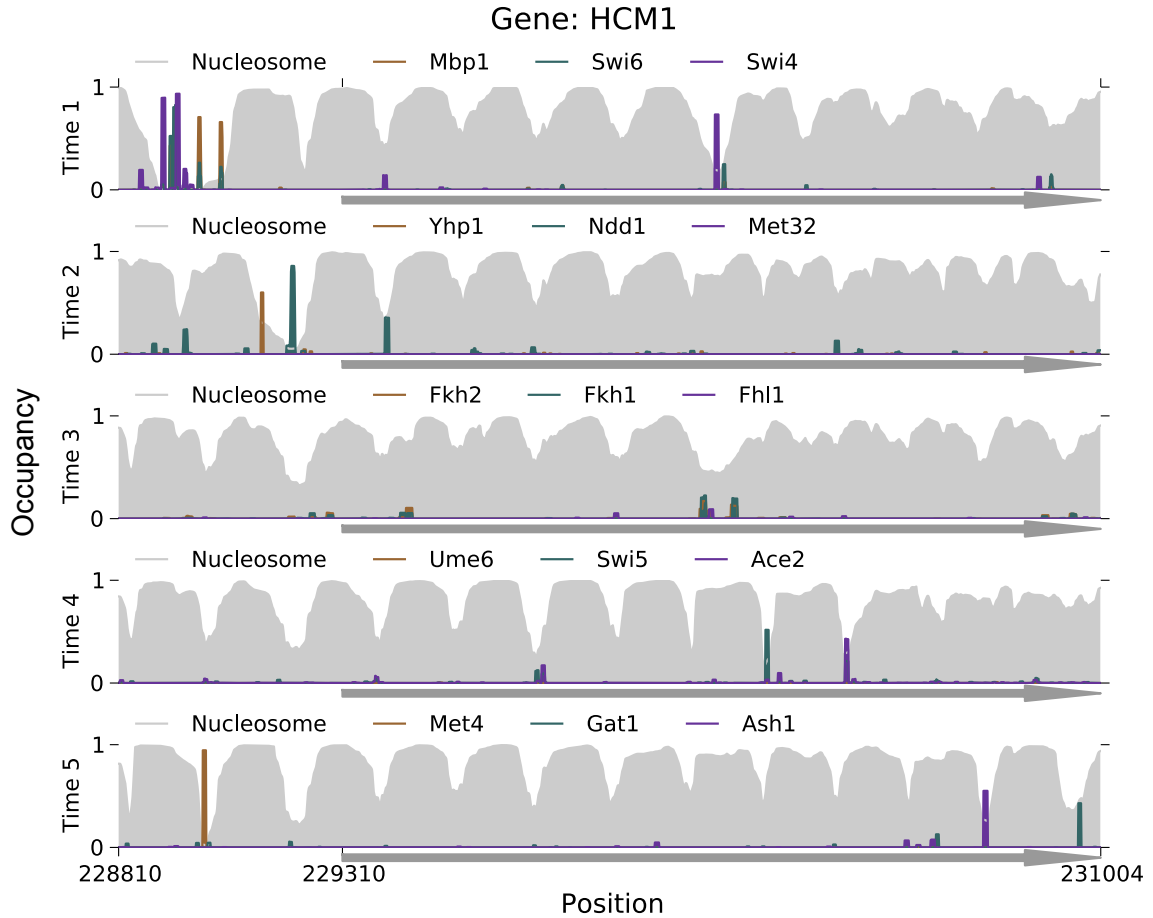


FIGURE 5.2.1: Predicted promoter occupancy by COMPETE for the gene HCM1. Each row is a time segment. The location and direction of the gene are shown by the arrow on the x-axis.

This matches predictions that HCM1 is a target of SBF, a complex that includes Swi4 and Swi6, and MBF, a complex that includes Mbp1 and Swi6.

5.3 Methods for inferring edges

Evidence of binding is not direct evidence of regulation; therefore, we are also looking into methods of determining regulatory relationships, i.e. when a change in one gene's expression level is linked to a change in another gene's expression level. This could find cases that support existing binding evidence, indicate that previ-

ously undiscovered relationships may exist, or find that binding does not indicate regulation.

Many algorithms have been developed to infer gene regulation, and cover a wide range of methodologies from different disciplines. ARACNE uses an information theory measurement (Margolin et al., 2006), and TimeDelay-ARACNE is a modification that uses time series data to indicate the direction of the interactions (Zoppoli et al., 2010). CLR (Context Likelihood of Relatedness) also uses an information theory measurement (Faith et al., 2007), and tlCLR (time-lagged CLR) uses timing information to indicate the direction of the edges. Banjo uses Dynamic Bayesian Networks to model the system and searches for the most probable network to explain the data (Yu et al., 2004). Inferelator models the network as a system of linear ODEs and uses Least Angle Regression (LARS) to find directed interactions between genes, while enforcing sparseness in network connectivity (Bonneau et al., 2006). A more recent version includes tlCLR for model selection, followed by the Inferelator for model refinement and parameter discovery (Greenfield et al., 2013).

A new method is being developed, called Local Edge Machine (LEM) by Kevin McGoff, Xin Guo, John Harer, and me. One of the most critical issues we encountered is how test the performance of these methods. We therefore needed to create synthetic networks with periodic dynamics that would serve as a ground truth.

5.4 Developing synthetic networks for testing

When testing a method, it is important to have test cases where all the information of the system is known beforehand so it can be compared to the results that are returned by the algorithms being tested. For gene regulatory networks, there are several sources of data or network generators. These include the DREAM challenges, which include challenges that includes network inference. While the second challenge for network inference contained periodic networks, and the fourth challenge had an

example network that was periodic, these networks do not wholly meet our needs for testing inference on periodic data.

There are several features found in periodic networks that we would like to exist in our periodic synthetic networks. The dynamics of the network should contain a variety of signals shapes, not just pure cosine curves. For example, we would like to see peaks, asymmetric on increase and decrease, flat sections in the peak or trough, and combinations of multiple signals caused by influence from multiple regulations. There should be variations in the amplitudes within the expression levels for network, and also a variety of networks that cover different ranges of amplitudes. The complexity of the networks should also vary on number of nodes and the number of regulators. Some networks should also include lightbulbs; nodes that are controlled by the network, but do not regulate any other nodes in the network (they have inputs but no outputs).

5.4.1 Generating networks with optimization

To build these networks, we wanted to specify topologies without specifying the dynamics that they would produce. Using a curve fitting algorithm would therefore not work; we needed a flexible way to find parameters that would allow for periodic network dynamics. To accomplish this, we modified our Evolutionary Algorithm (See section 6.1). An Evolutionary algorithm is an optimization technique that works on populations of candidate solutions with parameters that vary from one another. Each network in the population is simulated, and the extent to which its dynamics match the desired outcome is scored by an objective function. The networks with the worst scores are removed from the population, and the networks with the best scores then reproduce, but with mutated parameters, to create the next generation of networks. This is repeated until a score threshold is passed, or the algorithm is halted after a given number of generations. For generating networks for testing, we perform

multi-objective optimization, which computes the fitness of the network based on several measurements: periodicity, amplitudes of the signals, K parameter of the Hill functions, and β (also called the V_{max}) parameter of the Hill functions.

To take a given topology and find a set of parameters that generate periodic dynamics, we changed the objective function of the evolutionary algorithm. Instead of fitting the dynamics to a set of curves, we modified the objective function to measure the periodicity of the gene's expression level at a given period length. The measure of periodicity we used was the Lomb-Scargle periodogram (Lomb, 1976; Scargle, 1982), as defined by Press and Rybicki (1989):

$$P_g(\omega) = \frac{1}{2\hat{\sigma}^2} \left[\frac{\left[\sum_{i=1}^N (g(t_i) - \bar{g}) \cos(\omega(t_i - \tau)) \right]^2}{\sum_{i=1}^N \cos^2(\omega(t_i - \tau))} + \frac{\left[\sum_{i=1}^N (g(t_i) - \bar{g}) \sin(\omega(t_i - \tau)) \right]^2}{\sum_{i=1}^N \sin^2(\omega(t_i - \tau))} \right] \quad (5.4.1)$$

where $\omega = (1 / \text{the period length})$ is the frequency used in computing the periodogram $P_g(\omega)$; g is the time series for a gene; $g(t_i)$ is the expression level at time t_i ; and \bar{g} is the average expression level for the gene. The τ and $\hat{\sigma}^2$ are computed as in Press and Rybicki (1989). Note that we only compute the periodogram for the frequency of interest; the larger the value, the more periodic the signal at that frequency. The periodicity error for gene g is computed as

$$err_{per}(g) = 1/P_g(\omega) \quad (5.4.2)$$

$$err_{per} = \frac{1}{G} \sum_{g=1}^G err_{per}(g) \quad (5.4.3)$$

The err_{per} is the sum of the periodicity errors for each gene g , divided by the total number of genes G . The optimization works to maximize the periodicity score for each node's dynamics in the network, which generates a network that has periodic dynamics for all of its nodes.

We also optimize for the amplitudes of the dynamics (times series max - time series min) to be within a certain range. The amplitude error for gene g is computed as

$$amp = max(g) - min(g)$$

$$err_{amp}(g) = \begin{cases} |amp - amp_{max}| & amp > amp_{max} \\ |amp - amp_{min}| & amp < amp_{min} \end{cases} \quad (5.4.4)$$

$$err_{amp} = \frac{1}{G} \sum_{g=1}^G err_{amp}(g)$$

where amp_{max} is the desired maximum signal height and amp_{min} is the desired minimum signal height. The err_{amp} is the sum of the errors for each gene g , divided by the total number of genes G .

To ensure an edge actually exists, the K parameter in the hill function must be within the range of its regulator; otherwise the gene will not respond to input from its regulator. Therefore, we also punish errors for being out of this range. For each regulator r of each gene g , the error is computed as

$$K_{min} = min(r) + buffer$$

$$K_{max} = max(r) - buffer$$

$$err_K(g, r) = \begin{cases} |K - K_{max}| & K > K_{max} \\ |K - K_{min}| & K < K_{min} \end{cases} \quad (5.4.5)$$

$$err_K(g) = \sum_{r=1}^R err_K(g, r)$$

$$err_K = \frac{1}{G} \sum_{g=1}^G err_K(g)$$

where the r is the time series of the regulator and the $buffer$ is a small amount that pads the min and the max. The $err_K(g, r)$ is the error for gene g and regulator r .

The $err_K(g)$ is the sum of the errors for each gene g , divided by the total number of genes G .

While there are bounds set on each type of parameter in the network, the combination of multiple Hill functions leads to addition and multiplication of individual β values. The system of having a β for each Hill function is very flexible for computation and defining networks, but their combinations can be quite large relative to the desired maximal transcription rate. For example, if we move the β outside of the hill functions and look at how these define the total maximal transcription rate for a combination of regulators:

$$(h_a(X) + h_a(Y)) * h_r(Z) \quad (5.4.6)$$

$$(\beta_1 h_a(X) + \beta_2 h_a(Y)) * \beta_3 h_r(Z) \quad (5.4.7)$$

$$(\beta_1 + \beta_2) * \beta_3 \quad (5.4.8)$$

it becomes obvious that the total transcription rate can be much higher than its components. We therefore apply a desired minimum and maximum to the total maximum transcription rate for all the regulators, and optimize this. The total V_{max} or β error for gene g is computed as

$$vmax = eval(\beta)$$

$$err_{vmax}(g) = \begin{cases} |vmax - vmax_{max}| & vmax > vmax_{max} \\ |vmax - vmax_{min}| & vmax < vmax_{min} \end{cases} \quad (5.4.9)$$

$$err_{vmax} = \frac{1}{G} \sum_{g=1}^G err_{vmax}(g)$$

The $err_{vmax}(g)$ is the sum of the errors for each gene g , divided by the total number of genes G .

Each of the types of the errors is weighted and then summed to produce the total

error for the network.

$$\begin{aligned}
 w_{per} &= 10,000 \\
 w_{amp} &= 10 \\
 w_K &= 100 \\
 w_{vmax} &= 100 \\
 err_{net} &= w_{per}err_{per} + w_{amp}err_{amp} + w_Kerr_K + w_{vmax}err_{vmax}
 \end{aligned}
 \tag{5.4.10}$$

5.4.2 *A library of synthetic periodic networks*

After developing methods to generate synthetic networks that are periodic, we set out to develop a library of networks that could be used for testing inference methods on periodic data. We were able to generate networks that met our specifications. They were periodic with a specified period and had desired amplitudes that varied within the specified ranges. We were also able to generate networks with different amplitude regimes, such as all amplitudes within ranges of 10 to 100, or 100 to 10,000. The dynamics contained a variety of signal shapes, such as peaks, asymmetric on increase and decrease, flat sections in the peak or trough, and combinations of multiple signals caused by influence from multiple regulations. The complexity of the networks could be varied by size, from 3 to 20 nodes (our largest network attempted). The complexity of the networks could also be varied by the overall connectivity of the network and the number of inputs to regulatory functions. Networks have also been constructed with and without lightbulbs (nodes with inputs from the network, but no outputs back into the network). For each topology, we have run the EA several times, which produces different sets of network parameters. The amplitude or Vmax restrictions are also varied. This creates sets of parameters and their resulting dynamics for each topology.

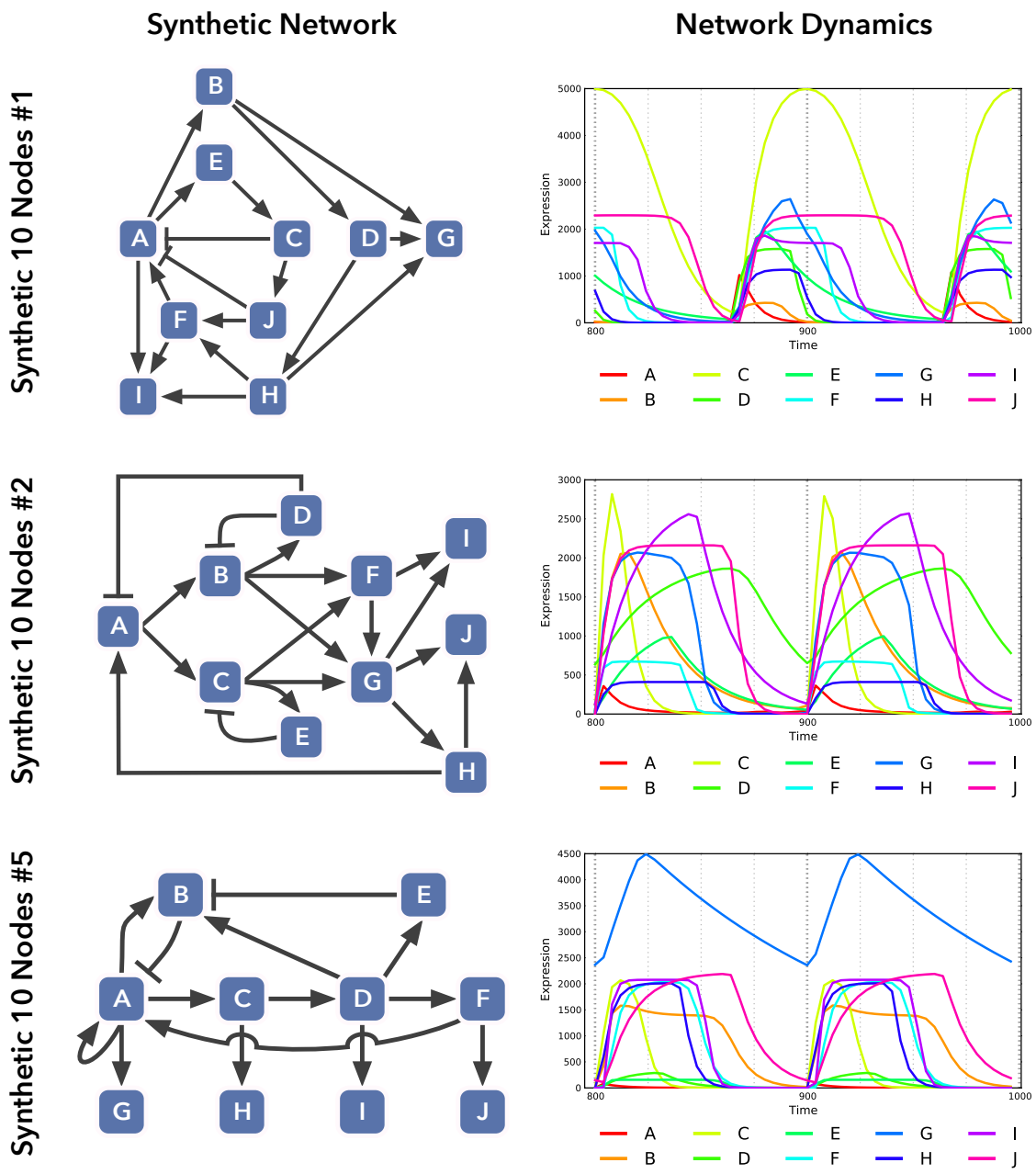


FIGURE 5.4.1: Synthetic gene regulatory networks for testing. These networks are built for testing algorithms on inferring edges and parameters for time-series data.

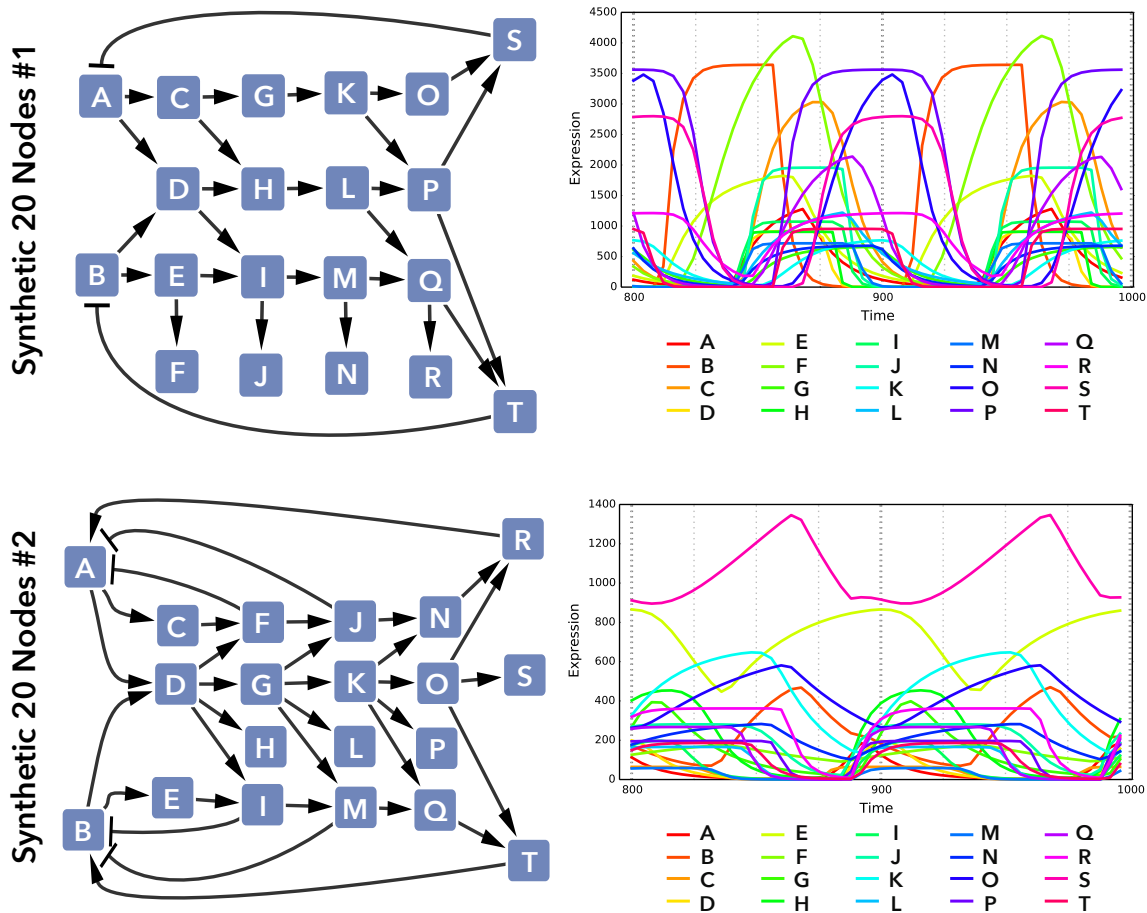


FIGURE 5.4.2: Synthetic gene regulatory networks for testing. These networks are built for testing algorithms on inferring edges and parameters for time-series data.

5.5 Local Edge Machine (LEM)

A new method for discovering regulation is being developed, called Local Edge Machine (LEM) by Kevin McGoff, Xin Guo, John Harer, and me. Some, but not all, of the information presented in this section overlaps with information in McGoff et al. (2014); please refer to this paper for additional information on the method and results. LEM looks for functional regulation in the network; where a change in one gene’s expression level predicts the change in another gene’s expression level. It uses Hill functions to model the interactions, gradient descent to find the best fits to

the data, and a Bayesian model to compute the probability of a combinatorial logic given the data. It can additionally incorporate prior information about activation or repression for transcription factors.

Local Edge Machine (LEM) predicts regulators of each gene in a network. For each gene, it enumerates all potential combinations of regulators. For each combination of regulators, it evaluates how well it can match the real data of the gene's expression (Figure 5.5.1).

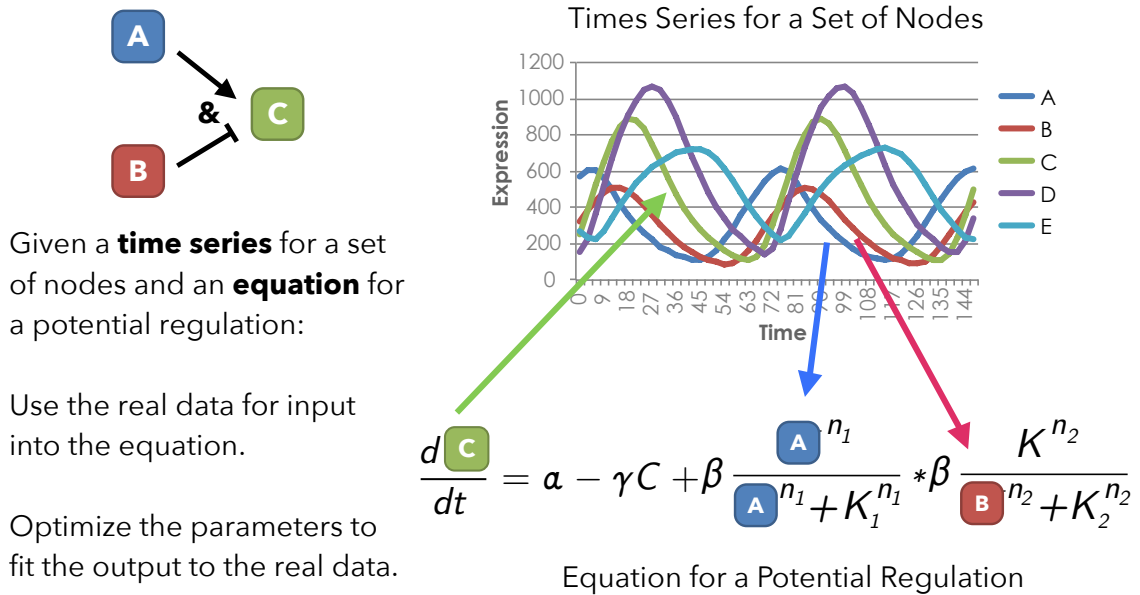


FIGURE 5.5.1: LEM uses the data for the regulators to simulate the regulatory function, and optimizes the parameters to find the best fit .

The regulatory function uses Hill functions, as described previously:

$$a_i(X_j) = \beta_{i,j} \frac{X_j^{n_{i,j}}}{X_j^{n_{i,j}} + K_{i,j}^{n_{i,j}}} \quad r_i(X_j) = \beta_{i,j} \frac{K_{i,j}^{n_{i,j}}}{X_j^{n_{i,j}} + K_{i,j}^{n_{i,j}}} \quad (5.5.1)$$

We assume that a repressor always overrides activators, and is therefore included with an AND gate (multiplication). For example, an input function f of two activators joined by OR, with an AND gate that overrides the activation for the tran-

scription factors X_1, X_2, X_3 :

$$f_1(X_1, X_2, X_3) = (a_{1,1}(X_1) + a_{1,2}(X_2)) * r_{1,3}(X_3) \quad (5.5.2)$$

Self-regulation is also allowed. The ODE for a gene under regulation includes this regulation function, the basal transcription rate constant α , and the degradation rate constant γ

$$\frac{dX_i}{dt} = \alpha_i - \gamma_i X_i + f_i(X_1, X_2, \dots, X_m) \quad (5.5.3)$$

For the regulatory function, the values from the real time series data are used for the transcription factor. Gradient descent with multiple initial starts is used to optimize the ODE parameters. The objective function used in the optimization is how well the solution from the ODE matches the real time series data from the gene. Once the best fitting parameters are found for each regulatory function, a Bayesian model is used to compute the probability of a combinatorial logic given the data.

LEM has been run all against all the synthetic network topologies, for two sets of parameters per topology. We currently run LEM using 1 to 3 regulators, and results here are shown using only one regulator (Figure 5.5.2). The performance of LEM is compared to two other network inference algorithms called Inferelator (Greenfield et al., 2013) and TD-ARACNE (Zoppoli et al., 2010). The measure of performance is computed using a Receiver Operator Characteristic (ROC) plot, and is summarized with the Area Under Curve (AUC). On these synthetic networks, LEM shows the best performance with AUC scores of approximately 0.9 (1 is a perfect score).

5.6 Conclusions

We have mapped data from five chIP-chip experiments in budding yeast to a set of gene IDs from the SGD. We have been able to load 2,652,112 p-values into our

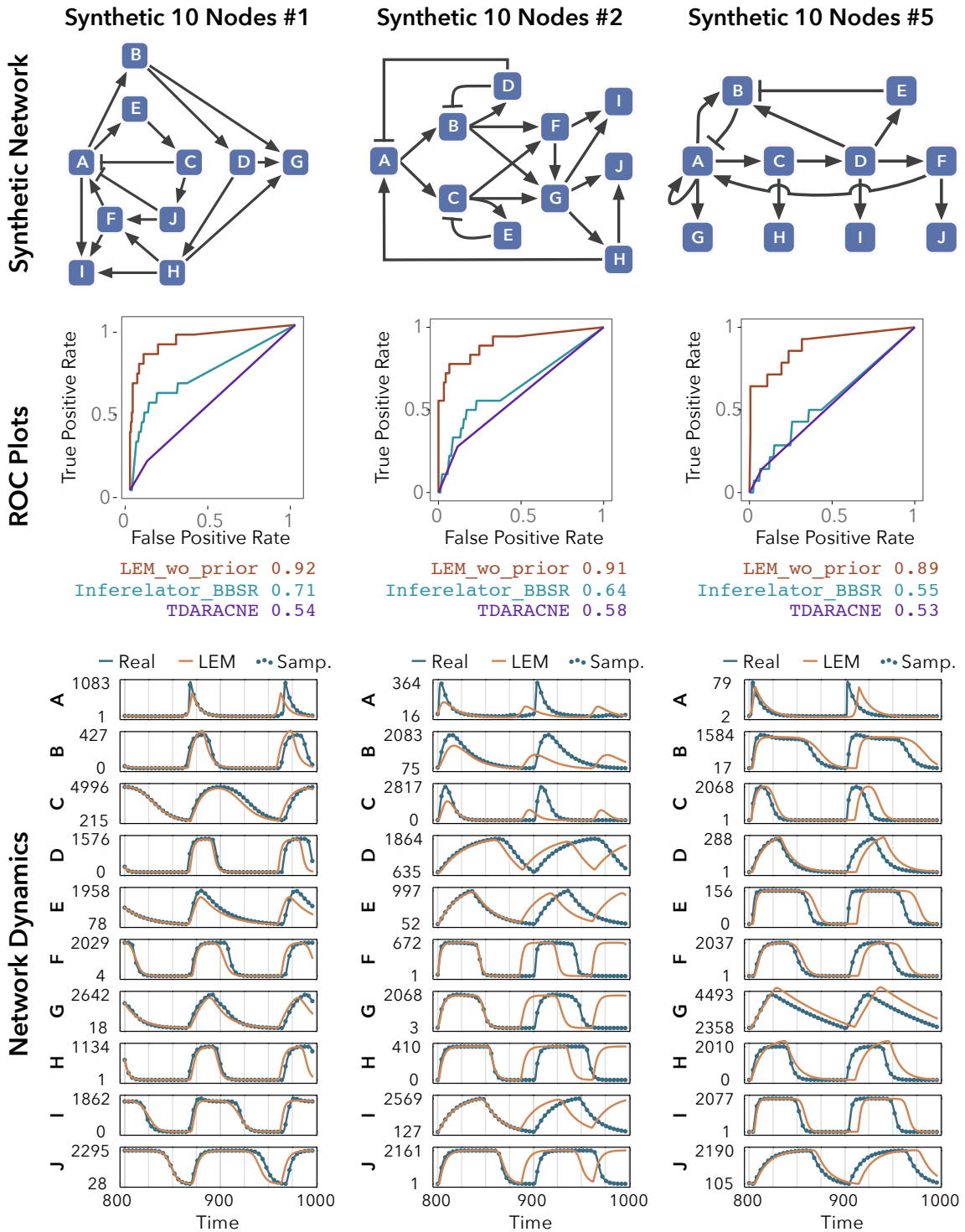


FIGURE 5.5.2: LEM results on synthetic models with 10 nodes, original network shown (top row). ROC plots and AUC scores are shown for LEM, Inferelator, and TD-Aracne (second row). The dynamics constructing using LEM’s top pick for each gene’s regulator are shown (bottom row).

database for 197 proteins and 6,382 genes. These give us confidence information on 1,206,994 potential bindings between protein and DNA that we can use to evaluate the possible existence of an edge. We have additionally begun storing manually curated information on potential edges. COMPETE has also been used to model the binding of multiple transcription factors present at different times.

To test inference methods, it was necessary to create synthetic networks models to generate test data. We have generated networks with 3, 5, 10, and 20 nodes, with varying levels of complexity in their regulatory structures (two topologies with 3 nodes, four topologies with 5 nodes, seven topologies with 10 nodes, and three topologies with 20 nodes). There are currently sixteen topologies, each with several sets of parameters that generate oscillatory dynamics.

These synthetic networks have been used in testing the inference algorithm LEM (Local Edge Machine). Initial results are promising, especially compared to other inference algorithms we have tested. Combining these types of edge information provides us with lists of potential edges that can be used in our next step, constructing networks and finding parameters for them.

5.7 Future Work

For circadian data, we have begun examining and collecting gene regulation data. The first data set we are using in the genome-wide binding analysis. For the mouse circadian rhythm, the Chip-seq data of Koike et al. (2012) used seven known circadian transcription factors: BMAL1, CLOCK, NPAS2, PER1, PER2, CRY1, and CRY2. Their study included measurements for six time points across one circadian cycle. As this data only indicates binding, we have also begun manually curating data from literature. The goal would be to build a compiled data set similar to what we have created for yeast, so that we can query, visualize, and incorporate this data in our network building process.

From testing networks of different sizes, we also know that the performance of the inference degrades as the size of the network increases; it would be of great interest to know if the density of the connectivity or the stability of the network also influences the performance of the inference. We would like to expand our set of synthetic networks to include larger networks and more variation in the connectivity of the networks; for example, networks that contain redundancy or are densely regulated. Of additional interest is the robustness of the networks to parameter or concentration changes; we are therefore very interested in examining the stability of our oscillatory networks and potentially adding the capability to select for more stable networks during optimization.

Finding Network Parameters

Once we have selected nodes and edges, we have a network topology. We have decided on a mathematical model for this topology, so now we want to find parameters that allow the network model to produce dynamics that match the biological data. To accomplish this, the parameters are optimized such that the model's dynamics will resemble the biological data.

6.1 Finding Parameters for a Network: Evolutionary Algorithms

Evolutionary algorithms (EAs) have been used previously to find parameters for ODE models of networks that exhibit nonlinear (Wahde and Hertz, 2000; Sakamoto and Iba, 2001; Deckard and Sauro, 2004) or oscillatory behavior (Paladugu et al., 2006; Francois and Hakim, 2004). EAs are a type of global heuristic optimization that does not require computing derivatives. Our evolutionary algorithm finds candidate solutions by evolving sets of parameters for a network topology. The initial population consists of N networks with parameters that are selected from a uniform distribution within set ranges. The initial conditions for the network are set from the biological data; the initial concentration of each gene is set to the expression

level in the data for the first time point. For each generation, each network in the population is simulated to generate dynamics. Each network's fitness is determined by comparing its dynamics to the biological data. The fittest networks then reproduce to create the next generation. After a specified fitness is reached or a specified number of generations pass, the algorithm stops and returns a network with the best fitting parameters that were found. We wrote our Evolutionary Algorithm in C++.

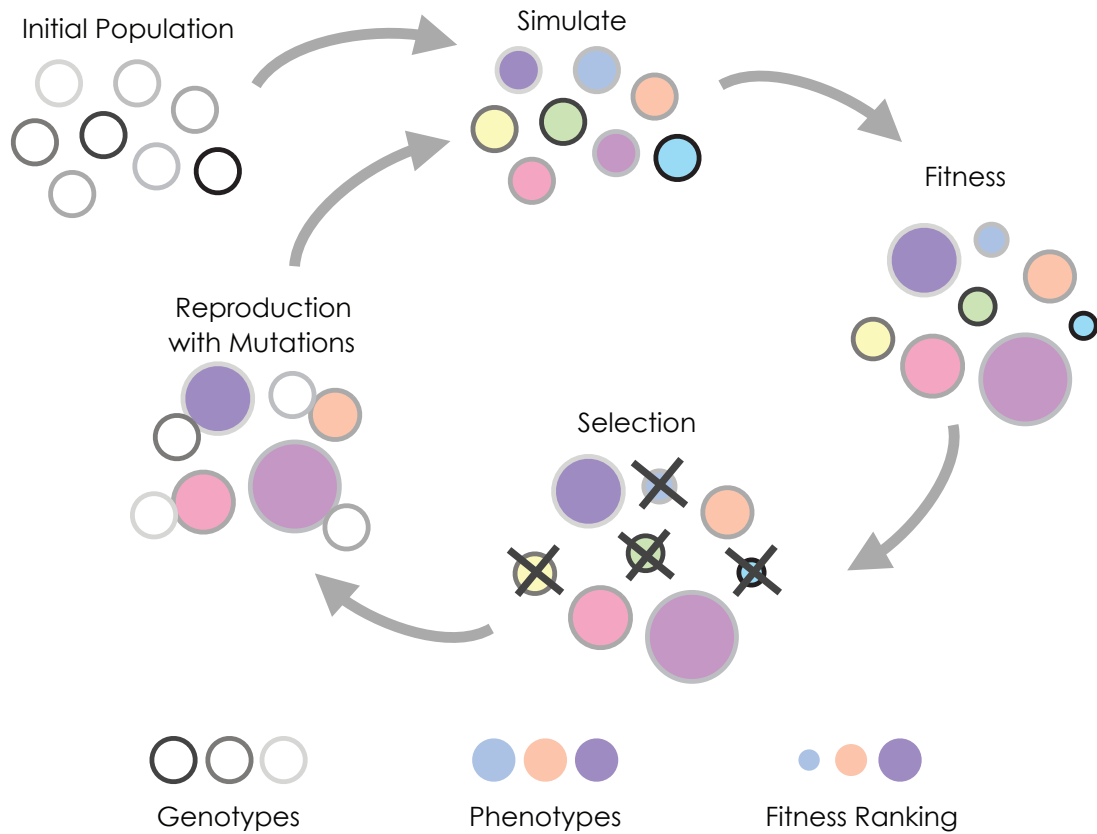


FIGURE 6.1.1: An overview of the steps in an Evolutionary Algorithm.

6.1.1 Create Initial Population

The initial population consists of a set number of networks with parameters that are selected from a distribution. The parameters are initialized within set ranges from a uniform distribution. Several different ranges were used. For models of the yeast

Table 6.1.1: To simplify the model, several proteins are represented by one protein. A protein complex can be represented by protein that is believed to limit the behavior of the whole complex. A group of proteins that have similar dynamics and downstream targets can be represented by one protein.

Parameter	Min	Max
α basal transcription rate constant	0.01	1
γ degradation rate constant	0.01	1
K activation/repression coefficient	min(regulator)	max(regulator)
β maximal transcription rate	10	1000
n Hill coefficient	1	8

cell cycle we selected parameter ranges based on their ability to emit signals with similar amplitudes and shapes to the data (Table 6.1.1). The initial conditions for the network are set from the data; the initial concentration of each gene is set to the expression level in the biological data for the first time point. If there is no data for a gene included in the topology, then we set the initial concentration to a value selected from a uniform distribution.

6.1.2 Simulate Population

The simulation runs for each pair of adjacent ordered time points $(start, end) = (t_1, t_2), (t_2, t_3), \dots, (t_{T-1}, t_T)$, and then stores the concentration of the gene and its first derivative for each end time point. As we are working with a periodic system, we must simulate the data for several periods to verify that the pattern repeats.

6.1.3 Assign Fitness

The fitness function measures the difference between the biological data and the dynamics of the model, which is called the error. Larger errors indicate worse fitness. We first calculate the error for each time point for each gene and combine them, and then we combine the scores from each gene to make the score for the network.

For genes $g = 1, \dots, G$ and a time points $t = 1, \dots, T$, the expression level of gene g at time t is denoted by $y_{data}(g, t)$ for the biological data and by $y_{sim}(g, t)$ for the

model simulation. For each gene g at each time point t , we calculate the error of the simulation data compared to the biological data. This measure consists of two parts: the absolute error between the values at each time point

$$err_{point}(g) = \frac{1}{T} \sum_{t=1}^T |y_{data}(g, t) - y_{sim}(g, t)| \quad (6.1.1)$$

and the absolute error between the estimated first derivatives at each interior time point

$$err_{slope}(g) = \frac{1}{T-2} \sum_{i=2}^{T-1} |[y_{data}(g, t+1) - y_{data}(g, t-1)] - [(y_{sim}(g, t+1) - y_{sim}(g, t-1))]| \quad (6.1.2)$$

These sums are normalized by the number of time points T . The errors for each gene g are then summed and normalized by the number of genes G , and are stored as two scores for the network.

$$err_{point} = \frac{1}{G} \sum_{g=1}^G err_{point}(g) \quad (6.1.3)$$

$$err_{slope} = \frac{1}{G} \sum_{g=1}^G err_{slope}(g) \quad (6.1.4)$$

The two error scores are then weighted and added together to make the final error score for the network

$$w_{point} = 1$$

$$w_{slope} = 1000 \quad (6.1.5)$$

$$err = w_{point}err_{point} + w_{slope}err_{slope}$$

The total error is used as the final measure of fitness for the network. The use of an error term for the slope is very important for measuring the accuracy of the period

for oscillations; it is possible to encounter dynamics of a different period that fits all the sampled points (an alias of the desired period length). In practice this has generally appeared as signals with much smaller periods (higher frequencies) that alias the period of interest.

6.1.4 Selection

Selection determines which networks survive to create offspring for the next generation. It is based on the fitness scores. Currently there are two types of selection that are implemented: truncation selection (aka elitism) and fitness proportionate selection. Truncation selection is deterministic while fitness proportionate selection is probabilistic (Michalewicz and Fogel, 2004, Section 7.4).

In both methods, we first sort the population from the best fitness (lowest error) to the worst fitness (highest error). For Elitism, we simply save a set number of the best individuals and delete the remainder. Fitness proportionate selection is slightly more involved. We ignore the actual fitness scores and instead use the network's position in the fitness ranking to determine its probability of surviving

$$p(pos) = \left(\frac{N - pos}{N} \right)^s \quad pos = 0, 1, \dots, N - 1 \quad (6.1.6)$$

Where p is the probability of survival given the network's position in the ranking of N networks in the population, to the power of the selection pressure s . The probability of survival for the network in position 0 (the first and best) network is 1 so it will always be selected for survival and reproduction.

We consider different types of selection and selection pressures to deal with the tradeoffs of exploration versus exploitation during optimization. It is generally believed that elitism will cause faster convergence, often at the expense of diversity in the population (Črepinšek et al., 2013). This is also affected by the percentage

of networks that survive, which is another parameter that is difficult to tune and requires adjustment for each problem.

We additionally apply a uniqueness criterion, which prevents the population from filling up with identical networks and causing the population to lose diversity and stagnate. The current uniqueness criterion is based on fitness; if the difference between two scores is less than a threshold, the networks are considered to be duplicates and the second one is removed. Note that this precludes any sort of neutral mutations, from accumulating in the population because the criterion acts upon the fitness score and is ignorant of any distinctions in the genotype (the model parameters and topology). To avoid this, it is also possible to base uniqueness on the genotype.

6.1.5 Reproduction & Mutation

If there are S surviving networks and a carrying capacity N for the population, then each surviving network makes $O = (K/S) - 1$ offspring. Programmatically, they each make O clones which are then mutated. The parent networks remain in the population to compete against their offspring. This guarantees that the best solutions are never lost. Note that there is no mechanism for differential reproduction (i.e. fitter networks have more offspring) on a per generation basis. However, as a fitter network may survive for several generations longer than a less fit network, so the fitter network will produce more offspring during its lifetime.

Mutation rates determine how much and how often the parameters are altered. Rather than establishing set mutation rates, we instead use a system where the mutation rates vary for each offspring. Each offspring has a parameter for the frequency and magnitude of its mutations. The magnitude provides an upper limit on a normal distribution for the actual size of the mutation. The offspring will have a continuum of few to many parameters being changed and a continuum of small to possibly large mutations. For example, one offspring will have a few large parameter changes while

another will have many small parameter changes. No assumptions are made about when exploration or exploitation would be best; instead the raw material for both is provided at all stages and selection can determine what is optimal. No crossover operators are used on the networks.

Since we adopt this strategy of mutations and we preserve the best solution at each generation, we can allow the maximum frequencies and magnitudes of mutations to be very high. For example, at the highest level of mutation, 90% of the parameters in the model can be changed by an order of magnitude. This allows for large jumps across the search space, which is not truly evolutionary; these individuals act more as distant relatives coming from another population and help to maintain diversity.

6.1.6 Testing

To test the program, we created synthetic network topologies. We then filled in parameters to create oscillations, simulated the network to get its dynamics, and then sampled this data at regular intervals across several periods. The topology and the target data were run through the program to see if it could find parameters to replicate the behavior of the original model. The program can find parameters that generate dynamics that match the period and, to a lesser extent, match the amplitude. The program also does well at matching the overall shapes of the curves. It should be noted that the parameters found can be very different from the original parameters, indicating that there may be several different sets of parameters that can approximate the data. This issue with identifiability cannot be directly resolved from the data alone.

As shown above, there are several different parts of the EA, each with different parameters that can be tuned to alter how the EA operates. Therefore the efficiency of an EA relies upon how these are all set; which is itself a difficult optimization problem (Michalewicz and Fogel, 2004, Section 10.6). The implementation we have

developed has performed well to date, and we consider our parameters to be acceptably tuned to the current problem. However, accept that there is “no free lunch” (Wolpert and Macready, 1997) and that our parameters and possibly parts of the EA must be adjusted as our network models change.

6.1.7 Optimization with unknowns

Another feature of our evolutionary algorithm that can be used during optimization is the addition of “anonymous nodes” in the network. In cases where an especially promising topology cannot be optimized, there may exist an obvious gap. This gap could be filled in by an anonymous node, which would be set to interact with the genes around the gap of interest. The parameters for this gene and its interactions would be optimized with the rest of the network. If this resulted in a particularly successful optimization, then this anonymous gene’s profile could be used to locate real genes with similar profiles. This actual gene and any interactions could then replace the anonymous gene in the network, and the network would then be re-optimized to determine whether the gap was correctly filled.

This can also be used to optimize features that do not have available data. For example, when we model phosphorylation, we do not have information on the quantities of proteins that are in a high or low state. We can model these two states as anonymous nodes, and use the sum of them as we know the total amount of protein.

6.2 Using biological data to inform parameters

The ODE models with Hill functions have parameters for the basal transcription rate constant, decay rate constant, maximal transcription rate constant, $1/2$ maximal activation/repression coefficient, and the hill coefficient. Any information about the values for these parameters, even if only to provide a range of values, would greatly restrict our possible search space and improve the biological accuracy of our models.

Table 6.2.1: Different scales in different times series gene expression data sets. The amplitude (amp) of a signal is measured as the maximum minus the minimum (peak-to-trough height). Shown as normalized and provided by authors.

Data Set	Min	Max	Min Amp	Max Amp	Mean Amp
Yeast Cell Cycle WT1	77.33	28570.64	7.28	19283.79	842.03
Yeast Cell Cycle WT2	76.72	28513.39	8.05	17340.34	811.42
Yeast Metabolic Cycle	0	156.04	0.004	92.40	5.21
Mammal Circadian	20.6	1218383	0.8	530872.7	3358.205

Table 6.2.2: Different scales for Yeast Cell Cycle WT1 normalized with different packages. The amplitude (amp) of a signal is measured as the maximum minus the minimum (peak-to-trough height).

Data Set	Min	Max	Min Amp	Max Amp	Mean Amp
RMA	4.00	14.84	0.05	8.61	1.30
GCRMA	1.78	15.79	0	12.97	1.97
MAS5	0.02	8017.00	0.44	5960.30	491.54
dChip	77.33	28712.94	13.18	26844.3	1558.14

One of the issues in determining parameters is determining units; in microarray data, the units are arbitrary. For different organisms, we might assume that the minimum/maximum amounts of mRNA might be different, which would need to be accounted for in parameter range selection. If we download different data sets, we will see different minimum and maximum expression levels, and different amplitudes (maximum - minimum for each signal). However, these have used different normalizations: dCHIP in Orlando et al. (2008), Genespring v7 in Tu et al. (2005), and GCRMA in Hughes et al. (2009) (Figure 6.2.1). RMA and GCRMA return the expression measures in log (base 2). Comparing the scale of different normalizations on the same data set indicates the scope of the variation (Figure 6.2.2). While different scaling is of biological concern, this information is not readily accessible; the larger concern is to be aware of the scale of the data and to rescale the data or scale the parameter ranges such that the parameters in a model can produce dynamics that match the scale of the data.

Several studies have addressed the kinetic parameters in gene regulatory networks. Wang et al. (2002) measured the decay of mRNA in *S. cerevisiae*. The strain had a temperature-sensitive mutation in RNA polymerase II, so that transcription could be inactivated. After inactivation, samples were taken at 0, 5, 10, 15, 20, 30, 40, 50, 60 minutes and mRNA levels were quantified using microarrays. The decay rate constant was found as the value that minimized a least squares fit to an exponential decay model. These were measured under a single condition, and if mRNA decay is regulated then these represent only one dynamic response in decay rates out of many possible responses.

6.3 Enforcing topology in the face of changeable parameters

Once we have selected a topology, the optimization should find parameters that reflect the effect of all the edges in the network of the global dynamics. One common problem we have encountered when optimizing networks is that the parameters can be adjusted to nullify an edge in the network. One case is when one (or more) maximal transcription rate β in the combination of Hill functions is set to zero, or is so much smaller than the other β , that it does not make a contribution to the regulation. For example, in the combinatorial regulation of X_1 with two activating Hill functions $a_{1,2}$ and $a_{1,3}$

$$f_1(X_2, X_3) = \beta_1 a_{1,2}(X_2) + \beta_2 a_{1,3}(X_3) \quad (6.3.1)$$

If the β_1 is close to zero, or $\beta_1 \ll \beta_2$, then the edge X_2 activates X_1 is not contributing to the dynamics of X_1 , and that edge does not truly exist. To overcome this, we select a range of parameters that reflects our willingness to believe that two edges can have different influence and still be considered relevant. We currently use one hundred; if an edge has one hundred times less effect on its target, we are willing

to consider that it might not be a relevant edge.

Another issue is the value of $1/2$ maximal activation/repression coefficient K . This defines the level of the regulator where the response in the regulated gene changes from a lower state to a higher state. For example, a Hill function $a_{1,2}$ representing the edge X_2 activates X_1

$$a_{1,2}(X_2) = \beta \frac{X_2^n}{X_2^n + K^n} \quad (6.3.2)$$

If $K \ll \min(X_2)$ then the activation is never turned on. If $K \gg \max(X_2)$ then the activation is never turned off. In either case, the edge X_2 activates X_1 is not contributing to the dynamics of X_1 , at least for the provided range of values of X_2 , and therefore that edge does not truly exist. However, this constraint holds only for this condition where the regulator has this minimum and maximum; in another condition the range of the regulator's values could change such that $a_{1,2}$ does respond to changes in X_2 , and the edge would then be functional.

6.4 Models of the Yeast Cell Cycle

The time series microarray data from the Yeast cell cycle data has noise and is very damped. While ODEs can be fit to dynamics that are damped and/or noisy, these measurements are not the behavior we wish to capture. What we really wish to model is repeating oscillations of gene expression that occur in a single cell as it moves through several cell cycles. Microarray data measures gene expression from a population of cells, so that each time point is an average of the gene expression in all the cells. These populations are synchronized such that they should all be in the same phase of the cell cycle at the beginning of the experiment, but this synchrony is not complete and the loss of synchrony increases over time, especially in budding yeast where asymmetrical divisions between mother and daughter leads to a longer time before the daughter begins its next cell cycle (Orlando et al., 2007).

An algorithm to measure this loss of synchrony, Characterizing Loss Of Cell-Cycle Synchrony (CLOCCS), accounts for such sources of asynchrony as: variations in phase for the starting population, variations in time to complete a cell cycle, a longer first G1 phase for the starting population to recover after being synchronized, and a longer time in G1 phase for the smaller daughter cells (Orlando et al., 2007, 2009; Mayhew et al., 2011). CLOCCS considers the distributions of phases in the population through time and models the population as a branching process, where each new generation of daughter cells is a branch. It then uses Bayesian inference and cell-cycle markers to determine the parameters for the sources of variation in cell cycle phase.

Given the estimates from CLOCCS characterizing the loss of synchrony, removing these effects from the original data would reveal the underlying gene expression dynamics in a single cell. A deconvolution algorithm was developed for this purpose, which returns gene expression profiles for one cell cycle of the mother (Guo et al., 2013). The time series it returns is no longer damped and it is also smoothed. As additional benefits, the deconvolved data is more densely sampled, with 100 samples for one cell cycle, as opposed to approximately 7 samples per cycle in the original microarray data.

We can then use the deconvolved data to make our desired data set: repeating oscillations of gene expression that occur in a single cell as it moves through several cell cycles. The mother's cell cycle, one cycle with 100 points, is down-sampled by ten and then repeated for five cycles. This data set then has 50 points for 5 cycles, for 10 points per cycle. This time series data set can now be used in optimizations to build models that can match the timing of the original cell cycle data, and will have continued, undamped oscillations.

One application of the EA optimization with deconvolved data is to test small, simplified network topologies of core components to see how well they match the

timing of the data and maintain oscillations. For example, we built small, 4-node networks with different edge structures (Figure 6.4.1). In the top three examples, the EA was able to find parameters for the model that output dynamics similar to the deconvolved data. The last network shows an example of a network that the EA was unable to find parameters that would allow a good fit for the dynamics.

The optimization ran 1,000 networks for 1,000 generations, selecting an elite number of 50 at the end of each generation. One million networks were evaluated per run, and one run requires 5-10 minutes on a laptop (Macbook pro, 2.9 GHz Intel Core i7, 8 GB of RAM).

6.5 Mapping the search space

As the optimization samples parameters in the search space for the network, it also computes fitness scores for each network. Using the parameters as coordinates in n -dimensional spaces, where n is the number of parameters, and fitness is an additional dimension, we can generate an n -dimensional fitness landscape. The n -dimensional form can reveal whether there are multiple peaks that indicate different sets of parameters exist that give similarly good answers. This shows issues with identifiability. These fitness landscapes can also show whether these peaks are steep or gradual. This is of interest in biological systems; wide areas in the fitness landscape where a system continues to operate indicate that the system is robust to changes in parameters. This robustness is considered a desirable and likely property of biological systems; a model that shows more robustness might be the better explanation of a biological system.

To visualize the search space, 2-D slices can be plotted with one parameter on each axis and color to represent the height of the fitness landscape. The best (lowest minimized) scores are the peaks in the fitness landscape. The areas of most interest are the areas around the peaks where the networks display functional oscillations. A

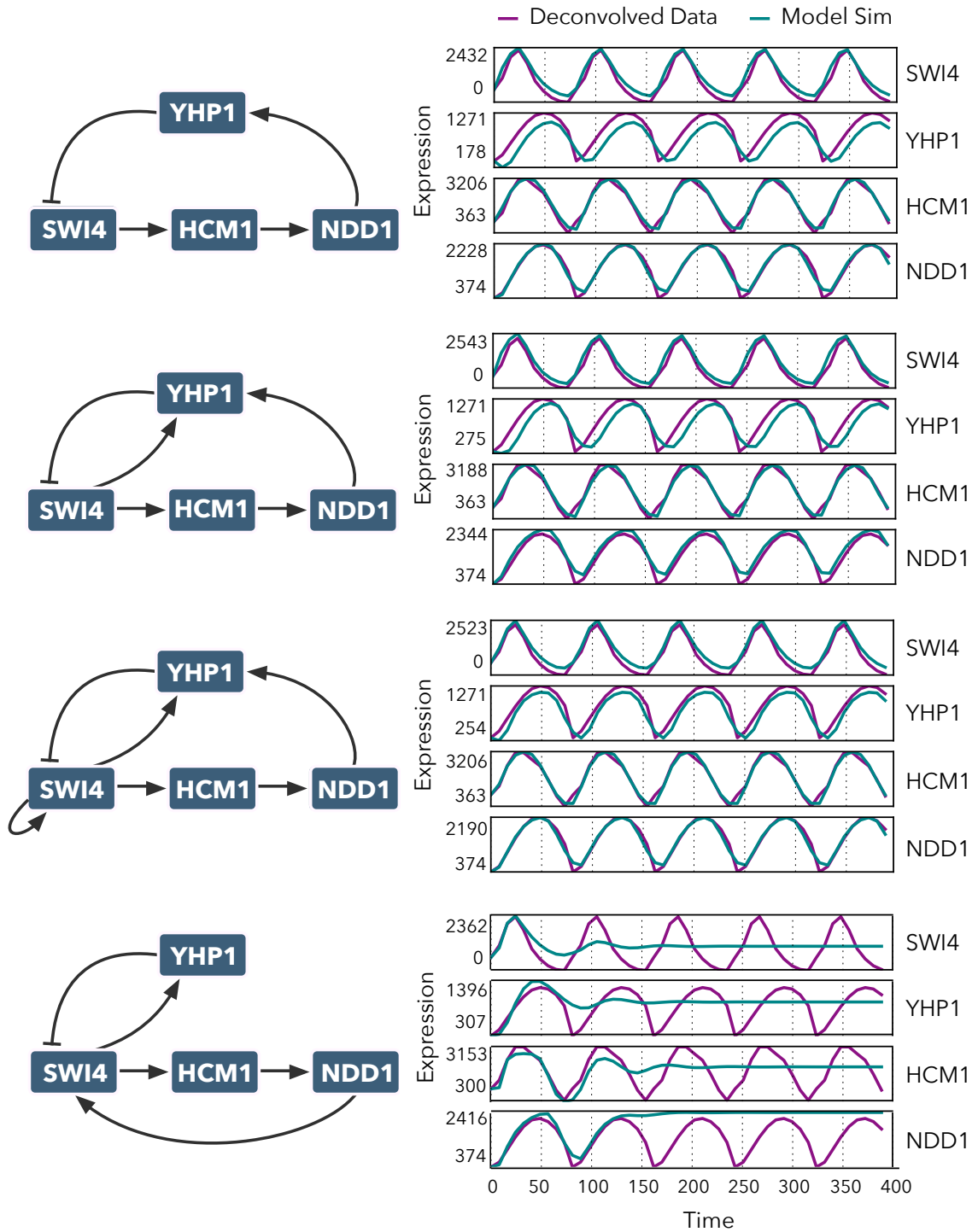


FIGURE 6.4.1: Optimizing parameters for simplified models. On the left are several network topologies for simplified components in the yeast cell cycle network. The plots on the right show the dynamics of the deconvolved data compared to the simulation of the model with optimized parameters.

cutoff score associated with functional oscillations is selected by examining networks from the EA that have worse scores but still appear to oscillate. These plots can be used to compare the fitness landscapes for different networks for the two parameters near the best solutions (Figure 6.5.1). As these are only projections, it indicates what the search space looks like, but we can not determine higher dimensional structure from the plots.

6.6 Conclusions

An Evolutionary Algorithm was developed to find parameters that would allow models to emit dynamics that fit data. This EA has shown acceptable speed and accuracy in finding network models that can match data. Its performance has scaled well on networks up to ten nodes and increasing complexity. Additionally, it has been possible to introduce or modify constraints on the parameters to find networks that meet our specifications.

We have run this algorithm on synthetic networks to test its ability to recover the original dynamics of the system. The algorithm has been able to reproduce the dynamics of the synthetic networks, but the parameters found can be different from the original parameters. It has also been used on simplified models of the cell cycle network to fit to deconvolved data for the cell cycle, and has returned parameters that can emit dynamics similar to the original data.

6.7 Future Work

We have currently optimized networks up to ten nodes, but would like to increase the size of the networks we optimize. Of additional interest is the robustness of the networks to parameter or concentration changes; we are therefore very interested in examining the stability of our oscillatory networks and potentially adding the capability to select for more stable networks during optimization.

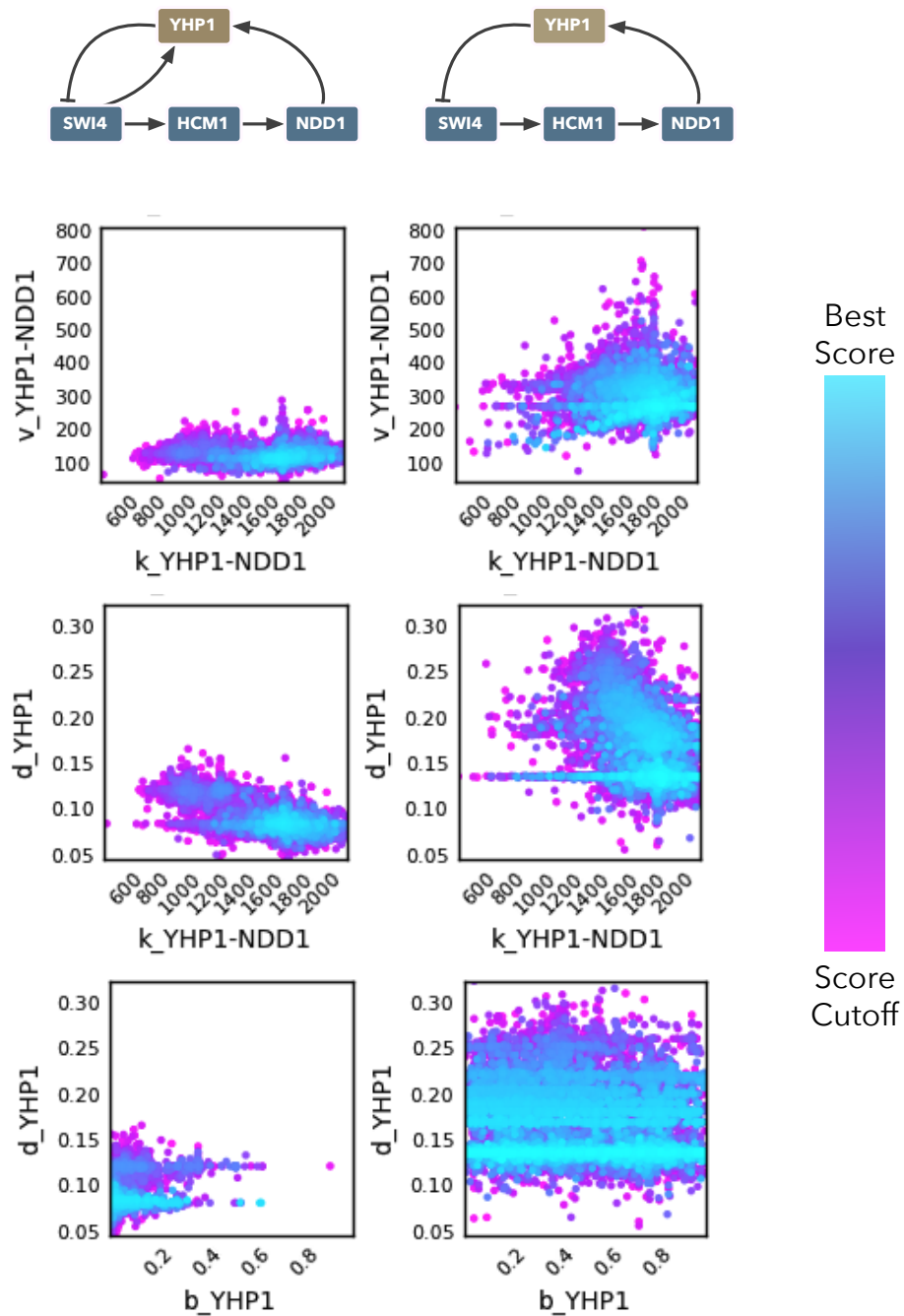


FIGURE 6.5.1: Exploring the search space of simplified Yeast Cell Cycle models. For two different simplified topologies of the yeast cell cycle, we compare the search space. These plots are two-dimensional slices of the fitness landscape described by two variables in the functions for the ODE of YHP1. The color indicates the fitness.

Models must be able to match data from many experimental manipulations. To perform this type of optimization, we create models to represent different experiments, for example by fixing topologies and/or fixing certain concentrations that match each of the experimental conditions. Each condition-specific model is then simulated with the shared parameters, and compared to the data set from that condition. The fitness then measures how well the parameters perform across all conditions, and the parameters will be optimized for these multiple experiments.

Data Management & Integration

To work efficiently and accurately with large amounts of data, it becomes necessary to manage and integrate the data. One of our initial goals was to take several data sets from *S. cerevisiae* and to integrate all the data sets so that they mapped to a set of genes. While this initially seemed straightforward, the data was collected from 2001 to 2013 by different labs and using different microarrays, and we encountered issues trying to map all the probe ids or gene symbols. Some issues were that data may be labeled using standard names instead of systematic names, ORFs previously considered to be genes can be deleted or merged, the common names for genes might be updated, and so on. To get the fullest and most accurate view of the data, it is critical to map data from different sources to the correct gene.

7.1 Gene Information

We use the set of active genes from the Saccharomyces Genome Database as a standard set to which all other data is mapped. A gene is a single entity that is identified by a single SGD ID and a single systematic name. It might also have a standard name and multiple alias names that it may have been referred to historically.

For example, the gene CLN3 has the information: SGD ID: S000000038, Systematic Name: YAL040C, Standard Name: CLN3, Aliases: DAF1, FUN10, WHI1.

As our standard set of genes, we downloaded data from SGD (Cherry et al., 2011) using their Yeastmine tool (Balakrishnan et al., 2012). This standard set allows us to map any existing data to a gene by looking up by systematic name, standard name, or alias names. Tracking the status of genes allows us to make updates to our mappings when new active genes are added, or active genes are changed to deleted or merged. The database currently contains 6,702 genes, which covers the entire set of *S. cerevisiae* genes that are mapped on the Affymetrix Yeast 2 Genome Array.

The set of genes on the Affymetrix Yeast 2 Genome Array does not only cover genes classified as “Verified ORFs” by the SGD; it also covers other classifications. We therefore pull all gene records in the classifications included by the array. There are 908 probe-gene mappings for ORFs that are dubious or uncharacterized. There are 168 probe-gene mappings for pseudogenes and transposable elements. We therefore pull all records for:

```
feature_type: ORF, transposable_element_gene, pseudogene.  
qualifier: Verified, Dubious, Uncharacterized
```

Integration and management is handled by a mySQL database. Scripts are used to import data and cleanly map the data to a standard set of genes. Other scripts are used to link together data, which can then be filtered and returned to the user. Additionally, all information added to the database has information on the source/references for the data including version and link information.

7.2 Mapping gene identifiers

Gene annotations are not static. This presents a challenge for integrating multiple data sets. The gene information from data sets, especially older data sets, might not have gene identifiers that directly align with gene identifiers maintained in current

databases. We have created set of rules to help ensure that automatic integration of data is performed as safely as possible. This also serves to highlight cases where data integration would be unsafe, and therefore potentially misleading. One critical piece of information we use is SGD's status information for genes, which marks gene records in SGD as being active, merged (with another gene record), or deleted.

Our first two cases that need to be considered when mapping occur when two gene records are both active (Figure 7.2.1). These highlight that it is important to use the most restrictive identification available before trying less restrictive identification information, e.g. first try SGD ID, then systematic name, then standard name, and only if all these have been exhausted, examine the alias names. It would be unsafe to perform a search for a given *id* using a condition like *if (id == sgd_id OR id == systematic_name OR id == alias_name)* and attempt to use this for automated mapping.

1. **Some genes names are listed as a systematic_name for one sgd_id and as an alias for a different sgd_id.** There are nine cases where this occurs, which might include symmetric cases. Therefore, a search on all the systematic_name/standard_name should be exhausted before attempting a search on the alias_names.
2. **Some genes names are listed as a standard_name for one sgd_id and as an alias for a different sgd_id.** There are 96 cases where this occurs, which might include symmetric cases. A search on all the systematic_name/standard_name should be exhausted before attempting a search on the alias_names.

Another set of rules deals with changes of status for gene records in the SGD (Figure 7.2.2). These could have occurred at some point after data was collected, so these

1.

sgd_id_1	systematic_name_1	standard_name_1	alias_names_1	sgd_id_2	systematic_name_2	standard_name_2	alias_names_2
S000028422	YLR154W-C	TAR1	YLR154W-A	S000028675	YLR154W-A	NULL	YLR154W-C
S000007623	YNL067W-A	NULL	YNL067W-B	S000028810	YNL067W-B	NULL	YNL067W-A
S000007252	YOL013W-B	NULL	YOL013W-A	S000028811	YOL013W-A	NULL	YOL013W-B

2.

sgd_id_1	systematic_name_1	standard_name_1	alias_names_1	sgd_id_2	systematic_name_2	standard_name_2	alias_names_2
S000000127	YBL031W	SHE1	NULL	S000000027	YAL029C	MYO4	FUN22 SHE1
S000000349	YBR145W	ADH5	NULL	S000002327	YDL168W	SFA1	ADH5
S000000271	YBR067C	TIP1	NULL	S000003113	YGL145W	TIP20	TIP1
S000000117	YBL021C	HAP3	NULL	S000004929	YMR312W	ELP6	TOT6 KTI4 HAP3
S000000009	YAL011W	SWC3	SWC1	S000005490	YOL130W	ALR1	SWC3
S000000071	YAR018C	KIN3	FUN52 NPK1	S000005759	YOR233W	KIN4	KIN31 KIN3

FIGURE 7.2.1: An example of mapping data to a gene where it is necessary to check the systematic name (1) or the standard name (2) separately and before comparing to the alias names.

need to be checked when integrating older data sets. On the other hand, these changes could occur after data has been integrated, and therefore an update to gene records might require remapping of the data to keep it as current as possible.

1. **sgd_id is active but did not exist in previous version.** The new SGD record should be added to the database.
2. **sgd_id merged with other sgd_id.** Older data sets will map to the merged gene record or might map to both the merged and active gene (where a single gene was previously believed to be two genes, and data was collected for both

2.

sgd_id	systematic_name	standard_name	alias_names	status	feature_type	qualifier
S000000518	YCL012W			Merged	ORF	
S000000520	YCL014W	BUD3	YCL012W	Active	ORF	Verified

3.

sgd_id	systematic_name	standard_name	alias_names	status	feature_type	qualifier
S000002137	YAL034C-B		YAL035C-A	Active	ORF	Dubious
S000002138	YAL042C-A		YAL043C-A	Active	ORF	Dubious
S000002139	YAL056C-A		YAL058C-A	Active	ORF	Dubious

4.

sgd_id	systematic_name	standard_name	alias_names	status	feature_type	qualifier
S000000512	YCL006C			Deleted	ORF	

5.

sgd_id	systematic_name	standard_name	alias_names	status	feature_type	qualifier
S000001920	YFR024C			Deleted	ORF	
S000002968	YFR024C-A	LSB3	YFR024C	Active	ORF	Verified

FIGURE 7.2.2: An example of mapping data to a gene where it is necessary to check the systematic name (1) or the standard name (2) separately and before comparing to the alias names.

locations). Newer records will map to the new gene record.

3. **Systematic names do not have an sgd_id, even though they can be be aliases of other genes.** For example, YAL035C-A, YAL043C-A, YAL058C-A all have records in the Simon et al. (2001) data set, but none of these have a corresponding SGD ID (even as being deleted or merged). However, these can exist as aliases of other genes. In this case, it would not be possible to search for the systematic_name and see it was merged. It would only be possible to search for the name in the alias.

4. **Systematic names have an `sgd_id` and are marked as deleted, but are not aliases of other genes.** In this case, the data is mapped to the deleted gene, and there is no way to map it to an active gene.
5. **Systematic names have an `sgd_id` and are marked as deleted, but are aliases of other genes.** These appear to behave as merged genes. For example, YFR024C.

However, even following the above rules, there are still cases when mapping an alias to a specific gene record can fail. The first of these is when one gene symbol (or systematic name) is listed as an alias for more than one gene record (Figure 7.2.3). These cannot be resolved automatically given the data available; The safest course is to reject the data until more information can be included or a human can review them.

Gene Records			
Alias	# Genes	Systematic Names	Symbols
CAL1	2	YBR023C YGL155W	CHS3 CDC43
CCT1	2	YDR212W YGR202C	TCP1 PCT1
COD1	2	YEL031W YPR105C	SPF1 COG4
CRT1	2	YJL210W YLR176C	PEX2 RFX1
CSL5	2	YKL080W YGR140W	VMA5 CBF2

FIGURE 7.2.3: Examples where a gene alias maps to one or more records.

7.3 Results on integrating chIP-chip data

We wanted to use the chIP-chip data sets for budding yeast to learn about potential regulatory relationships. Five data sets were collected. Simon et al. (2001) included the genome-wide targets of nine known cell-cycle transcription factors (Mbp1, Swi4, Swi6, Mcm1, Fkh1, Fkh2, Ndd1, Swi5, and Ace2). Lee et al. (2002) included 141

Table 7.3.1: The amount of ChIP-chip data that was successfully integrated for budding yeast.

p-values loaded	2,652,112
TF/gene pairs	1,206,994
TFs	197
Target Genes	6,382

transcription factors, including twelve that were specifically identified as cell cycle related. Harbison et al. (2004) included 203 transcription factors, which were considered to be a nearly complete set of all TFs in the yeast genome. We used only the YPD data set. The data set from Workman et al. (2006) contained 30 transcription factors (including seven cell cycle genes: Mcm1, Swi4, Swi6, Fkh2, Ndd1, Ace2, Swi5). We used the data sets for two conditions: control (grown on YPD) and stress (exposed to MMS).

We next needed to integrate five chIP-chip data sets for budding yeast, which we did by mapping them to a set of gene IDs. Our database contains 6,702 gene records, each with a SGD ID, downloaded data from SGD (Cherry et al., 2011) using their Yeastmine tool (Balakrishnan et al., 2012). These include only ORFs, transposable elements, and pseudogenes. The chIP-chip data sets have their entries marked with the systematic name or standard name (symbol). A set of scripts to automate integrating the data were developed based on the information in the previous section (Mapping the gene identifiers). These read each TF/gene pair from the chIP-chip data, and then attempts to map the TF and the gene to a gene record in the database. It then first attempts to match the systematic name to a systematic name in the database. If that fails, it then attempts to map the standard name to a standard name in the database. Using just these steps, we have been able to consolidate 2.6 million p-values for 1.2 million TF/gene pairs (Table 7.3.1).

However, not all the data from the chIP-chip data sets was included. There are gene IDs in the data sets that might not have a record in the database, as we do

not keep all the SGD records. Some of the systematic or standard names used the data sets have been deprecated and are now stored as an alias name. While we can map these gene IDs to alias names in the database, it is less straightforward. A gene alias could map to two or more gene records, which we did not have a way to resolve without manually intervening. Some genes have been merged with another active genes; in some cases there are records for the merged gene and active gene with the merged gene listed as an alias. In other cases, the merged name is just listed as an alias of the active gene. The entries in the data sets could exist for the merged gene, the active gene, or both.

7.4 Conclusions

We have encountered many issues in integrating gene data for budding yeast. Some issues occur because gene annotations are not static. Experiments have occurred at different times, using different technologies, and used different methods for mapping their data; the annotations attached to the data reflect these factors. The largest challenge was understanding how these records can change, and how to create simple rules for combining data that takes these into account. We have been able to take several sets of data and integrate them, at least partially. We have integrated the majority of data from five chIP-chip experiments against a collection of gene records in our database. These combined p-values for transcription factor binding and other types of edge data have also been combined our lists of periodic genes, which we have used in constructing our networks.

7.5 Future Work

To handle more diverse data sets, especially from different technologies, it would be necessary to expand the number of gene records that are stored in our database. Automating the ability to import genes by alias names and mapping merged genes

to active genes, and how to detect and report when multiple entries map to the same gene record, would also increase the amount of data we can include from data sets. Adding a user interface to the database and access to the import/export scripts would allow more people to have access to the data. It would also be beneficial to track gene information changes as they are made available, and to be able to notify us that changes have occurred.

Bibliography

Ahnert, S., Willbrand, K., Brown, F., and Fink, T. (2006), “Unbiased pattern detection in microarray data series,” *Bioinformatics*, 22, 1471–1476.

Alon, U. (2007), “An introduction to systems biology: design principles of biological circuits,” Chapman & Hall/CRC.

Badis, G., Chan, E. T., van Bakel, H., Pena-Castillo, L., Tillo, D., Tsui, K., Carlson, C. D., Gossett, A. J., Hasinoff, M. J., Warren, C. L., Gebbia, M., Talukder, S., Yang, A., Mnaimneh, S., Terterov, D., Coburn, D., Li Yeo, A., Yeo, Z. X., Clarke, N. D., Lieb, J. D., Ansari, A. Z., Nislow, C., and Hughes, T. R. (2008), “A Library of Yeast Transcription Factor Motifs Reveals a Widespread Function for Rsc3 in Targeting Nucleosome Exclusion at Promoters,” *Molecular Cell*, 32, 878–887.

Balakrishnan, R., Park, J., Karra, K., Hitz, B., Binkley, G., Hong, E., Sullivan, J., Micklem, G., and Cherry, J. (2012), “YeastMine—an integrated data warehouse for *Saccharomyces cerevisiae* data as a multipurpose tool-kit,” *Database: The Journal of Biological Databases and Curation*, 2012, bar062.

Benjamini, Y. and Hochberg, Y. (1995), “Controlling the false discovery rate: a practical and powerful approach to multiple testing,” *Journal of the Royal Statistical Society. Series B (Methodological)*, 57, 289–300.

Bonneau, R., Reiss, D. J., Shannon, P., Facciotti, M., Hood, L., Baliga, N. S., and Thorsson, V. (2006), “The Inferelator: an algorithm for learning parsimonious regulatory networks from systems-biology data sets de novo.” *Genome Biology*, 7, R36.

Buchler, N. E. and Louis, M. (2008), “Molecular Titration and Ultrasensitivity in Regulatory Networks,” *Journal of Molecular Biology*, 384, 1106–1119.

Chen, K., Calzone, L., Csikasz-Nagy, A., Cross, F., Novak, B., and Tyson, J. (2004), “Integrative analysis of cell cycle control in budding yeast,” . . . *Biology of the Cell*.

- Cherry, J. M., Hong, E. L., Amundsen, C., Balakrishnan, R., Binkley, G., Chan, E. T., Christie, K. R., Costanzo, M. C., Dwight, S. S., Engel, S. R., Fisk, D. G., Hirschman, J. E., Hitz, B. C., Karra, K., Krieger, C. J., Miyasato, S. R., Nash, R. S., Park, J., Skrzypek, M. S., Simison, M., Weng, S., and Wong, E. D. (2011), “Saccharomyces Genome Database: the genomics resource of budding yeast,” *Nucleic Acids Research*, 40, D700–D705.
- Cohen, S. and Hindmarsh, A. (1996), “CVODE, a stiff/nonstiff ODE solver in C,” *Computers in physics*, 10, 138–143.
- Cohen-Steiner, D., Edelsbrunner, H., Harer, J., and Mileyko, Y. (2010), “Lipschitz Functions Have Lp-Stable Persistence,” *Foundations of Computational Mathematics*, 10, 127–139.
- Črepinšek, M., Liu, S.-H., and Mernik, M. (2013), “Exploration and exploitation in evolutionary algorithms: A survey,” *ACM Computing Surveys (CSUR)*, 45, 35–33.
- De Jong, H. (2002), “Modeling and simulation of genetic regulatory systems: a literature review,” *Journal of Computational biology*, 9, 67–103.
- De Lichtenberg, U., Jensen, L., and Fausbøll, A. (2005), “Comparison of computational methods for the identification of cell cycle-regulated genes,” *Bioinformatics*, 21, 1164–1171.
- Deckard, A. and Sauro, H. M. (2004), “Preliminary Studies on the *In Silico* Evolution of Biochemical Networks,” *Chembiochem*, 5, 1423–1431.
- Deckard, A., Anafi, R. C., Hogenesch, J. B., Haase, S. B., and Harer, J. (2013), “Design and analysis of large-scale biological rhythm studies: a comparison of algorithms for detecting periodic signals in biological data,” *Bioinformatics*, 29, 3174–3180.
- Faith, J. J., Hayete, B., Thaden, J. T., Mogno, I., Wierzbowski, J., Cottarel, G., Kasif, S., Collins, J. J., and Gardner, T. S. (2007), “Large-Scale Mapping and Validation of Escherichia coli Transcriptional Regulation from a Compendium of Expression Profiles,” *PLoS Biol*, 5, e8.
- Francois, P. and Hakim, V. (2004), “Design of genetic networks with specified functions by evolution in silico,” *Proceedings of the National Academy of Sciences*.
- Futschik, M. E. and Herzel, H. (2008), “Are we overestimating the number of cell-cycling genes? The impact of background models on time-series analysis.” *Bioinformatics*, 24, 1063–1069.

Gardner, T. and Faith, J. (2005), “Reverse-engineering transcription control networks,” *Physics of Life Reviews*, 2, 65–88.

Glynn, E. F., Chen, J., and Mushegian, A. (2006), “Detecting periodic patterns in unevenly spaced gene expression time series using Lomb–Scargle periodograms,” *Bioinformatics*, 22, 310–316.

Greenfield, A., Hafemeister, C., and Bonneau, R. (2013), “Robust data-driven incorporation of prior knowledge into the inference of dynamic regulatory networks.” *Bioinformatics*, 29, 1060–1067.

Guo, X., Bernard, A., Orlando, D. A., Haase, S. B., and Hartemink, A. J. (2013), “Branching process deconvolution algorithm reveals a detailed cell-cycle transcription program.” *PNAS*, 110, E968–977.

Harbison, C. T., Gordon, D. B., Lee, T. I., Rinaldi, N. J., Macisaac, K. D., Danford, T. W., Hannett, N. M., Tagne, J.-B., Reynolds, D. B., Yoo, J., Jennings, E. G., Zeitlinger, J., Pokholok, D. K., Kellis, M., Rolfe, P. A., Takusagawa, K. T., Lander, E. S., Gifford, D. K., Fraenkel, E., and Young, R. A. (2004), “Transcriptional regulatory code of a eukaryotic genome,” *Nature*, 431, 99–104.

Haurly, A.-C., Mordelet, F., Vera-Licona, P., and Vert, J.-P. (2012), “TIGRESS: Trustful Inference of Gene REgulation using Stability Selection.” *BMC Systems Biology*, 6, 145.

Hecker, M., Lambeck, S., Toepfer, S., and Van Someren, E. (2009), “Gene regulatory network inference: Data integration in dynamic models—A review,” *Biosystems*, 96, 86–103.

Hill, A. V. (1913), “The Combinations of Haemoglobin with Oxygen and with Carbon Monoxide. I,” *Biochemical Journal*, 7, 471.

Hucka, M., Finney, A., Sauro, H. M., Bolouri, H., Doyle, J. C., Kitano, H., Arkin, A. P., Bornstein, B. J., Bray, D., Cornish-Bowden, A., Cuellar, A. A., Dronov, S., Gilles, E. D., Ginkel, M., Gor, V., Goryanin, I. I., Hedley, W. J., Hodgman, T. C., Hofmeyr, J.-H., Hunter, P. J., Juty, N. S., Kasberger, J. L., Kremling, A., Kummer, U., Le Novere, N., Loew, L. M., Lucio, D., Mendes, P., Minch, E., Mjolsness, E. D., Nakayama, Y., Nelson, M. R., Nielsen, P. F., Sakurada, T., Schaff, J. C., Shapiro, B. E., Shimizu, T. S., Spence, H. D., Stelling, J., Takahashi, K., Tomita, M., Wagner, J., Wang, J., and SBML Forum (2003), “The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models.” *Bioinformatics*, 19, 524–531.

- Hughes, M., Hogenesch, J. B., and Kornacker, K. (2010), “JTK_CYCLE: An Efficient Nonparametric Algorithm for Detecting Rhythmic Components in Genome-Scale Data Sets,” *Journal of Biological Rhythms*, 25, 372–380.
- Hughes, M. E., DiTacchio, L., Hayes, K. R., Vollmers, C., Pulivarthy, S., Baggs, J. E., Panda, S., and Hogenesch, J. B. (2009), “Harmonics of circadian gene transcription in mammals,” *PLoS genetics*, 5, e1000442.
- Huynh-Thu, V. A., Irrthum, A., Wehenkel, L., and Geurts, P. (2010), “Inferring regulatory networks from expression data using tree-based methods.” *PLoS ONE*, 5, e12776.
- Kallio, A., Vuokko, N., Ojala, M., Haiminen, N., and Mannila, H. (2011), “Randomization techniques for assessing the significance of gene periodicity results.” *BMC Bioinformatics*, 12, 330.
- Koike, N., Yoo, S.-H., Huang, H.-C., Kumar, V., Lee, C., Kim, T.-K., and Takahashi, J. S. (2012), “Transcriptional architecture and chromatin landscape of the core circadian clock in mammals.” *Science*, 338, 349–354.
- Lee, T. I., Rinaldi, N. J., Robert, F., Odom, D. T., Bar-Joseph, Z., Gerber, G. K., Hannett, N. M., Harbison, C. T., Thompson, C. M., Simon, I., Zeitlinger, J., Jennings, E. G., Murray, H. L., Gordon, D. B., Ren, B., Wyrick, J. J., Tagne, J.-B., Volkert, T. L., Fraenkel, E., Gifford, D. K., and Young, R. A. (2002), “Transcriptional Regulatory Networks in *Saccharomyces cerevisiae*,” *Science Signaling*, 298, 799.
- Lomb, N. (1976), “Least-squares frequency analysis of unequally spaced data,” *Astrophysics and Space Science*, 39, 447–462.
- Luan, Y. and Li, H. (2004), “Model-based methods for identifying periodically expressed genes based on time course microarray gene expression data.” *Bioinformatics*, 20, 332–339.
- Macisaac, K. D., Wang, T., Gordon, D. B., Gifford, D. K., Stormo, G. D., and Fraenkel, E. (2006), “An improved map of conserved regulatory sites for *Saccharomyces cerevisiae*,” *BMC Bioinformatics*, 7, 113.
- Margolin, A. A., Nemenman, I., Basso, K., Wiggins, C., Stolovitzky, G., Favera, R., and Califano, A. (2006), “ARACNE: An Algorithm for the Reconstruction of Gene Regulatory Networks in a Mammalian Cellular Context,” *BMC Bioinformatics*, 7, S7.

Mayhew, M. B., Robinson, J. W., Jung, B., Haase, S. B., and Hartemink, A. J. (2011), “A generalized model for multi-marker analysis of cell cycle progression in synchrony experiments.” *Bioinformatics*, 27, i295–303.

McGoff, K., Guo, X., Deckard, A., Kelliher, T., Leman, A., Haase, S. B., and Harer, J. (2014), “The Local Edge Machine: Inference of Dynamic Models of Gene Regulation,” in preparation.

Michalewicz, Z. and Fogel, D. B. (2004), *How to Solve It: Modern Heuristics*, Springer, New York.

Moreno-Risueno, M. A., Van Norman, J. M., Moreno, A., Zhang, J., Ahnert, S. E., and Benfey, P. N. (2010), “Oscillating gene expression determines competence for periodic Arabidopsis root branching.” *Science*, 329, 1306–1311.

Orlando, D., Lin, C., Bernard, A., Wang, J., Socolar, J., Iversen, E., Hartemink, A., and Haase, S. (2008), “Global control of cell-cycle transcription by coupled CDK and network oscillators,” *Nature*, 453, 944–947.

Orlando, D. A., Lin, C. Y., Bernard, A., Iversen, E. S., Hartemink, A. J., and Haase, S. B. (2007), “A probabilistic model for cell cycle distributions in synchrony experiments.” *Cell Cycle*, 6, 478–488.

Orlando, D. A., Iversen, E. S., Hartemink, A. J., and Haase, S. B. (2009), “A branching process model for flow cytometry and budding index measurements in cell synchrony experiments.” *The annals of applied statistics*, 3, 1521–1541.

Ostapenko, D. and Solomon, M. J. (2011), “Anaphase promoting complex-dependent degradation of transcriptional repressors Nrm1 and Yhp1 in *Saccharomyces cerevisiae*.” *Molecular Biology of the Cell*, 22, 2175–2184.

Paladugu, S., Chickarmane, V., Deckard, A., Frumkin, J., McCormack, and Sauro, H. (2006), “In silico evolution of functional modules in biochemical networks,” *IEE Proc. Syst. Biol*, 153.

Perea, J. and Harer, J. (2013), “Sliding Windows and Persistence: An Application of Topological Methods to Signal Analysis,” .

Perea, J., Deckard, A., Haase, S. B., and Harer, J. (2014), “SW1PerS: Sliding Windows and 1-Persistence Scoring; Discovering Periodicity in Gene Expression Time Series Data,” submitted to *Bioinformatics*.

- Pic-Taylor, A., Darieva, Z., Morgan, B. A., and Sharrocks, A. D. (2004), “Regulation of cell cycle-specific gene expression through cyclin-dependent kinase-mediated phosphorylation of the forkhead transcription factor Fkh2p.” *Molecular and cellular biology*, 24, 10036–10046.
- Polynikis, A., Hogan, S., and diBernardo, M. (2009), “Comparing different ODE modelling approaches for gene regulatory networks,” *Journal of Theoretical Biology*, 261, 511–530.
- Portales-Casamar, E., Thongjuea, S., Kwon, A., Arenillas, D., Zhao, X., Valen, E., Yusuf, D., Lenhard, B., Wasserman, W., and Sandelin, A. (2010), “JASPAR 2010: the greatly expanded open-access database of transcription factor binding profiles,” *Nucleic Acids Research*, 38, D105–D110.
- Press, W. H. and Rybicki, G. B. (1989), “Fast algorithm for spectral analysis of unevenly sampled data,” *Astrophysical Journal*, 338, 277–280.
- Prinz, S., Hwang, E. S., Visintin, R., and Amon, A. (1998), “The regulation of Cdc20 proteolysis reveals a role for APC components Cdc23 and Cdc27 during S phase and early mitosis.” *Current biology*, 8, 750–760.
- Ravasi, T., Suzuki, H., Cannistraci, C. V., Katayama, S., Bajic, V. B., Tan, K., Akalin, A., Schmeier, S., Kanamori-Katayama, M., Bertin, N., Carninci, P., Daub, C. O., Forrest, A. R. R., Gough, J., Grimmond, S., Han, J.-H., Hashimoto, T., Hide, W., Hofmann, O., Kamburov, A., Kaur, M., Kawaji, H., Kubosaki, A., Lassmann, T., van Nimwegen, E., MacPherson, C. R., Ogawa, C., Radovanovic, A., Schwartz, A., Teasdale, R. D., Tegnér, J., Lenhard, B., Teichmann, S. A., Arakawa, T., Ninomiya, N., Murakami, K., Tagami, M., Fukuda, S., Imamura, K., Kai, C., Ishihara, R., Kitazume, Y., Kawai, J., Hume, D. A., Ideker, T., and Hayashizaki, Y. (2010), “An Atlas of Combinatorial Transcriptional Regulation in Mouse and Man,” *Cell*, 140, 744–752.
- Reynolds, D., Shi, B. J., McLean, C., Katsis, F., Kemp, B., and Dalton, S. (2003), “Recruitment of Thr 319-phosphorylated Ndd1p to the FHA domain of Fkh2p requires Clb kinase activity: a mechanism for CLB cluster gene activation.” *Genes & Development*, 17, 1789–1802.
- Sakamoto, E. and Iba, H. (2001), “Inferring a system of differential equations for a gene regulatory network by using genetic programming,” *Proc. Congress on Evolutionary Computation*.
- Savageau, M. A. and Voit, E. O. (1987), “Recasting nonlinear differential equations as S-systems: a canonical nonlinear form,” *Mathematical biosciences*, 11, 546–551.

- Scargle, J. (1982), “Studies in astronomical time series analysis. II-Statistical aspects of spectral analysis of unevenly spaced data,” *Astrophysical Journal*, 263, 835–853.
- Shannon, P., Markiel, A., Ozier, O., Baliga, N. S., Wang, J. T., Ramage, D., Amin, N., Schwikowski, B., and Ideker, T. (2003), “Cytoscape: a software environment for integrated models of biomolecular interaction networks.” *Genome Research*, 13, 2498–2504.
- Simmons Kovacs, L., Mayhew, M., Orlando, D., Jin, Y., Li, Q., Huang, C., Reed, S., Mukherjee, S., and Haase, S. (2012), “Cyclin-Dependent Kinases Are Regulators and Effectors of Oscillations Driven by a Transcription Factor Network,” *Molecular Cell*.
- Simon, I., Barnett, J., Hannett, N., and Harbison, C. (2001), “Serial regulation of transcriptional regulators in the yeast cell cycle,” *Cell*.
- Smith, L. P., Bergmann, F. T., Chandran, D., and Sauro, H. M. (2009), “Antimony: a modular model definition language.” *Bioinformatics*, 25, 2452–2454.
- Spellman, P. T. P., Sherlock, G. G., Zhang, M. Q. M., Iyer, V. R. V., Anders, K. K., Eisen, M. B. M., Brown, P. O. P., Botstein, D. D., and Futcher, B. B. (1998), “Comprehensive identification of cell cycle-regulated genes of the yeast *Saccharomyces cerevisiae* by microarray hybridization.” *Molecular Biology of the Cell*, 9, 3273–3297.
- Straume, M. (2004), *Methods in Enzymology*, vol. 383 of *Methods in Enzymology*, Elsevier.
- Teixeira, M. C., Monteiro, P., Jain, P., Tenreiro, S., Fernandes, A. R., Mira, N. P., Alenquer, M., Freitas, A. T., Oliveira, A. L., and Sá-Correia, I. (2006), “The YEAS-TRACT database: a tool for the analysis of transcription regulatory associations in *Saccharomyces cerevisiae*.” *Nucleic Acids Research*, 34, D446–51.
- Tu, B., Kudlicki, A., Rowicka, M., and McKnight, S. (2005), “Logic of the yeast metabolic cycle: temporal compartmentalization of cellular processes,” *Science*, 310, 1152–1158.
- van Riel, N. (2006), “Dynamic modelling and analysis of biochemical networks: mechanism-based models and model-based experiments,” *Briefings in Bioinformatics*, 7, 364–374.

- Van Someren, E., Wessels, L., and Backer, E. (2002), “Genetic network modeling,” *Pharmacogenomics*, 3, 507–525.
- Wahde, M. and Hertz, J. (2000), “Coarse-grained reverse engineering of genetic regulatory networks,” *Biosystems*.
- Wang, Y., Liu, C., Storey, J., Tibshirani, R., Herschlag, D., and Brown, P. (2002), “Precision and functional specificity in mRNA decay,” *Proceedings of the National Academy of Sciences of the United States of America*, 99, 5860.
- Wasson, T. and Hartemink, A. J. (2009), “An ensemble model of competitive multi-factor binding of the genome,” *Genome Research*, 19, 2101–2112.
- Wolpert, D. H. and Macready, W. G. (1997), “No free lunch theorems for optimization,” *IEEE Transactions on Evolutionary Computation*, 1, 67–82.
- Workman, C. T., Mak, H. C., McCuine, S., Tagne, J.-B., Agarwal, M., Ozier, O., Begley, T. J., Samson, L. D., and Ideker, T. (2006), “A systems approach to mapping DNA damage response pathways.” *Science*, 312, 1054–1059.
- You, L. (2004), “Toward computational systems biology,” *Cell biochemistry and biophysics*, 40, 167–184.
- Yu, J., Smith, V., Wang, P., and Hartemink, A. (2004), “Advances to Bayesian network inference for generating causal networks from observational biological data,” *Bioinformatics*, 20.
- Zhu, C., Byers, K. J. R. P., McCord, R. P., Shi, Z., Berger, M. F., Newburger, D. E., Saulrieta, K., Smith, Z., Shah, M. V., Radhakrishnan, M., Philippakis, A. A., Hu, Y., De Masi, F., Pacek, M., Rolfs, A., Murthy, T., LaBaer, J., and Bulyk, M. L. (2009), “High-resolution DNA-binding specificity analysis of yeast transcription factors,” *Genome Research*, 19, 556–566.
- Zoppoli, P., Morganella, S., and Ceccarelli, M. (2010), “TimeDelay-ARACNE: Reverse engineering of gene networks from time-course data by an information theoretic approach,” *BMC Bioinformatics*, 11, 154.

Biography

Anastasia Deckard was born in Orange, California on April 13, 1979. She attended California State University, Fullerton, and received her Bachelor of Science in Computer Science, with a minor in Mathematics in May, 2008.

While at Duke, she has been funded by the Biochronicity: Time, Evolution, Networks, and Function (D12AP00025) grant from DARPA and Duke Center for Systems Biology (5P50GM081883-05) grant from NIH/NIGMS.

Publications:

Deckard A, Anafi RC, Hogenesch JB, Haase SB, and Harer J. Design and analysis of large-scale biological rhythm studies: a comparison of algorithms for detecting periodic signals in biological data. *Bioinformatics*. 29(24), 3174–3180.

Deckard A, Bergmann FT, Sauro HM. Enumeration and Online Library of Mass-Action Reaction Networks. arXiv:0901.3067v1 [q-bio.MN]. 2009 (preprint, but has been cited in other publications).

Deckard A, Bergmann FT, Sauro HM. Supporting the SBML layout extension. *Bioinformatics*. 2006, 22(23):2966-7.

Paladugu SR, Chickarmane V, Deckard A, Frumkin JP, McCormack M, Sauro HM. In silico evolution of functional modules in biochemical networks. *IEE Proceedings-Systems Biology*. 2006, 153(4):223-35.

Deckard A, Sauro HM. Preliminary studies on the in silico evolution of biochemical networks. *Chembiochem*. 2004, 5(10):1423-31.