Inesc-Id Technical Report

Analysis of the Dynamic Behavior of Gene Regulatory Networks

José Vellez Caldas Arlindo L. Oliveira Ana T. Freitas

July 2006

Abstract

Recent technological advancements have led to a dramatic increase in the quantity of available biological data. In this context, a general challenge nowadays is to transform the data into useful information. One of the most important problems is how to model and infer a gene regulatory network (GRN) from microarray expression data. A GRN consists of a set of mutually-influencing genes. Influence is exerted by several means, such as transcriptional regulation or post-translational modifications. There are several computational approaches for modeling and inferring GRNs. In this technical report we study the following topics: modeling the yeast's cell cycle GRN and analyzing its dynamics in both wild type and mutant strains, using Boolean networks; inferring parameters in artificial GRNs, from artificial expression data, using a particular piecewise linear equations model.

Keywords: Bioinformatics, computational biology, gene regulatory network, Boolean network, cell cycle, piecewise linear equation, machine learning, gradient descent.

Contents Index

Figure Index

Table Index

1. Introduction

A cell reacts to its environment by selectively producing and changing the functional state of several proteins. A protein's function is specified by a sequence of amino acids: it may act as an enzyme, it may transport certain metabolites in and out of the cell, and so on. Proteins are produced from messenger ribonucleic acid (mRNA) molecules. An mRNA molecule's basic building blocks are, essentially, the bases adenine (A), guanine (G), cytosine (C) and uracil (U). An mRNA molecule defines a unique code from which a protein can be derived. mRNA itself is produced using a part of the DNA molecule called a gene. A gene, since being part of the DNA molecule, shares the same kind of bases with the mRNA molecules, with the exception of uracil, which is substituted by thymine (T). The process of going from DNA to protein is summarized in the following sequential steps:

- transcription: an mRNA molecule is created from a gene; this molecule completely specifies the protein that should be produced later;
- translation: the mRNA molecule, which conveys information regarding the protein to be produced, is used to synthesize it; one mRNA molecule may be used in the production of several units of the same type of protein.

Figure 1.1 illustrates the process.

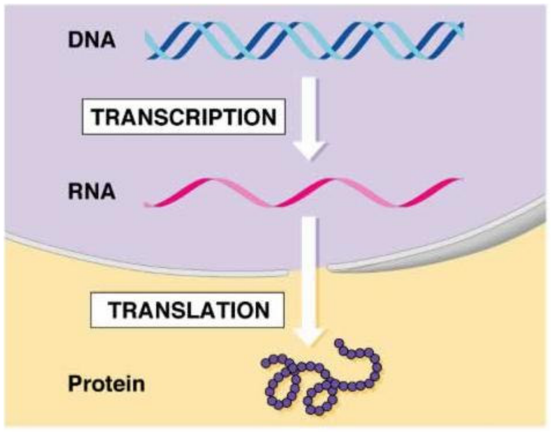

Fig. 1.1 – Transcription followed by translation.

In figure 1.1, the rose-colored area refers to the cell's nucleus (in eukaryotic (i.e. with cellular nucleus) organisms, the nucleus contains the DNA molecule). The mRNA molecule migrates to the cytoplasm, where it will be used by a ribosome to produce the specified protein (amino acids are required in this latter step).

There are several factors which affect the overall quantity of protein in an active form. Some of these are mRNA production and degradation rates, protein synthesis rate, protein degradation rate, or even post-translational modifications, such as phosphorylation, that alter a protein's activation state. The process of regulating mRNA production rate is called transcriptional regulation. This process is commonly studied in computational biology, so we describe it here in more detail. Each gene has a region, called the promoter, upstream of the region that defines the code for a protein. A transcription factor, which is itself a type of protein, can bind to a gene's promoter region. As a consequence of this binding, the mRNA production for the gene will increase or diminish. Some genes code for transcription factors, thereby regulating other genes' mRNA production rate. A system of genes that influence each other by means of this kind of regulation is called a transcriptional regulatory network. Understanding transcriptional or general GRNs is crucial for improving the diagnostic and treatment of diseases like cancer [1].

Currently, there is only partial information regarding which genes interact with each other and the nature of their interactions, given a metabolic context. Even in extensively studied biological processes, such as glycolysis, the underlying GRN hasn't been completely discovered [2]. Besides, GRNs are extensive and complex [3], making it difficult to intuitively understand their dynamics [4].

Recently, there has been a steep increase in the quantity of available biological data. This is being caused by improvements in already-existing experimental techniques, such as DNA sequencing, and also by the emergence of new technologies such as microarrays. A microarray is a simultaneous measurement of each gene's mRNA expression level in a neighborhood of cells. A sequence of microarray experiments, during a cellular process, originates a time series. There are several publicly-available sets of microarray experiments, some of them frequently used in bioinformatics [5]. Besides microarrays, there are other kinds of biological data, such as DNA sequence or protein expression measurements, that are useful in discovering genetic regulation processes. Given the enormous quantity of raw data available, there is the challenge of transforming it in useful biological information. Two of the most studied problems are how to model and infer GRNs, and how to compare protein or DNA sequences in order to discover whether they share a significant motif.

This technical report will focus on the problem of modeling and inferring GRNs. Frequent approaches to this problem rely, for example, on the use dynamic Bayesian networks [6-8] or systems of linear differential equations [9-10]. Although most of the analyzed approaches use, as input data, microarray expression levels, there are methods which combine microarrays with other kinds of data [11]. One of the key obstacles in evaluating a method's capacity to detect correct GRNs is the often incomplete knowledge about them. Frequently, a method detects many relations which are neither supported nor denied by biological literature, and that is a bane of precise algorithm evaluation. It is important to refer that most studied GRNs come from simpler organisms than the human, such as yeast (*saccharomyces cerevisiae*), a unicellular eukaryotic organism.

In this technical report, we study two approaches for modeling and inferring GRNs – boolean networks and piecewise linear equations. We intend to understand the potential and applicability of each formalism. Boolean networks are studied under the perspective of modeling, via a case study – the cell cycle. Piecewise linear equations are studied under both perspectives, modeling and inference, using several artificial networks; we study the dynamics that the equations allow for; we apply optimization algorithms to artificial GRNs, in order to infer parameters that best explain their respective time series. In the first part of this technical report, we introduce several useful concepts. In the second part, we analyze and extend the results on a Boolean model of the cell cycle [12]. In the third part, we describe a particular piecewise linear equations model; we describe an optimization algorithm; we discuss results for parameter inference in several artificial GRNs. In the last part, we conclude on the possibilities and limitations of each approach, and conjecture on possible future research directions.

2. Gene regulatory networks

In this chapter, we present several useful concepts for the comprehension of our work. We describe the cell cycle GRN and the concept of recurring motif in a GRN. We introduce three approaches to the problem of modeling and inferring GRNs: Boolean networks, linear differential equations (via a case study) and piecewise linear equations. We also describe a tool called Genetic Network Analyzer (GNA), which has the purpose of qualitatively simulating and analyzing a GRN.

2.1. The cell cycle

The cell cycle is an essential process to every living being. It consists of a sequence of events which allow for the division of a cell into two daughter cells, each having a copy of the mother cell's DNA molecule. The main events that comprise the cell cycle are:

- G1 (gap one) – The cell grows and prepares to replicate its DNA. At the end of this part of the cell cycle, the cell makes a commitment to divide itself;

- S – The cell replicates its DNA;

- G2 (gap two) – The cell prepares for mitosis (division into two twin cells). In this phase, cell growth is observed;

- M (mitosis) – The cell divides into two twin cells. This part of the cell cycle involves four phases – prophase, metaphase, anaphase and telophase.

The set of the first three phases is called interphase. Figure 2.1 portrays the cell cycle.

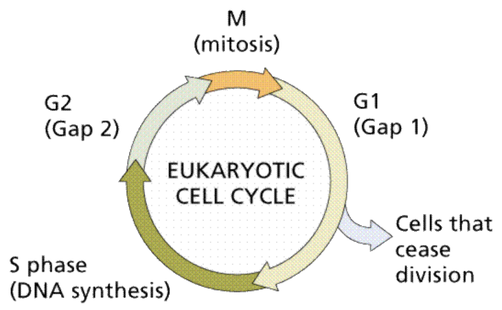

Fig. 2.1 – The sequence of processes in the cell cycle.

The progression from a phase to the next one is significantly controlled by two types of proteins – cyclins and cyclin-dependent kinases (CDK). The purpose of a kinase is to catalyze the phosphorylation (i.e. the supply of a phosphate group) of a target molecule. This transformation may change the target molecule's activation state. Thus, in the context of the cell cycle, selective phosphorylation is a way of regulating the passage from a phase to the next one. CDKs are cyclin-dependent, that is, they need to be activated by cyclins. A particular CDK-cyclin combination defines which molecules are targets. It is common for the same CDK to react with different cyclins in different cell cycle phases, thereby acting on different molecules. Each type of cyclin is characteristic of a particular phase of the cell cycle. Cell cycle progression is also monitored via a series of checkpoints that check whether, for example, the replicated DNA is damaged or not. In general, higher organisms have more complex cell cycles. In this technical report we study the yeast cell cycle, where there is only one CDK protein, named cdc28 (cell division cycle 28) and less types of cyclins than, for example, in the human being.

2.2. Recurring motifs in gene regulatory networks

R. Milo *et al* [13] have proposed a simple algorithm to identify recurring motifs in directed graphs. The graphs are representations of several networks – transcriptional GRNs, biological neural networks, food chains, electronic circuits and the internet. The algorithm searches for motifs of three or four nodes. Motifs with a low p-value are returned. The results show that each network has a set of significant motifs, and that these sets partially overlap (i.e. there are significant motifs that appear in more that one network). One of these motifs is the feed-forward loop. It appears in transcriptional GRNs and biological neural networks. Figure 2.2 illustrates the feed-forward loop.

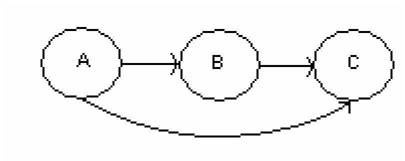

Fig. 2.2 – Feed-forward loop.

One of the possible roles of a feed-forward loop is to activate a gene only if the activation signal is persistent, and inactivate it more quickly in the absence of that signal. S. Mangan and U. Alon [14] make a theoretical analysis of the motif's function. We have created an artificial feed-forward loop, using a piecewise linear equation model, and inferred its parameters from artificial expression data, to validate that the parameter inference algorithm is capable of recovering parameters from motifs that are commonly present in transcriptional GRNs.

2.3. Modeling gene regulatory networks

2.3.1. Modeling using differential equations

We have analyzed an approach based on first degree differential equations [10]. The derivative for the expression of a gene *i* is given by:

$$X'_i(t) = G_i(t) - \lambda_i X_i(t) + \xi_i(t) \qquad (2.01)$$

where *$G_i(t)$* is the gene's transcription rate in instant *t*, λ_i is the gene's constant rate of degradation and *$\xi_i(t)$* is noise. The function *$G_i(t)$* is given by:

$$G_i(t) = c_{i0} + \sum_{j=1}^{N} c_{ij} S_j(X_j(t), \theta_j) \qquad (2.02)$$

where c_{i0} is the gene's basal production rate, c_{ij} is a constant, named the regulatory capacity between the regulating gene *j* and the regulated gene *i*, and S_j is a sigmoid function. The parameters for the sigmoid function are $\theta_j = \{r, M_j\}$, where *r* is the sigmoid's steepness and M_j is the average expression level for gene *j*. The sigmoid function is thus given by

$$S_j(X_j(t), \theta_j) = \frac{1}{1 + e^{-r(X_j(t) - M_j)}} \qquad (2.03)$$

The GRN inference method starts by finding a fixed number of regulators that minimizes the Akaike information criterion. Given this fixed number, an iterative algorithm is used to obtain the

most likely regulators for each gene. The candidates are 53 genes which are known to participate in the cell cycle (thus, this algorithm does not discover any regulations involving genes previously unknown to participate in the cell cycle GRN). Basically, the algorithm verifies which gene has the highest expression correlation with the regulated gene. This gene is selected as a regulator, and the corresponding parameter c_{ij} is determined, using a least-squares method. The newfound regulation is able to partially explain the regulated gene's expression. The algorithm is repeated, this time using the regulated gene's expression which could not be explained by the inferred regulation. The algorithm stops after the fixed number of regulators has been found.
H. Chen *et al* [10] present the regulators found, for regulated genes which are also known to be part of the cell cycle GRN. Most regulations are neither confirmed nor denied by biological literature. There is some incongruence between the source code provided by the authors and the method described in their article [10]. The method's robustness was tested in the following way:

- run the algorithm; use the inferred network and parameters to make an artificial simulation;
- use the new artificial expression time series as an input to the algorithm;
- run the algorithm again and compare the inferred regulators and parameters with the original ones.

We verified that both the new network architecture and the new parameters were very different from the ones originally obtained, so we consider this method not to be robust.

2.3.2. Genetic Network Analyzer

The Genetic Network Analyzer (GNA) is a tool for the qualitative simulation and analysis of GRNs, based on piecewise linear differential equations [4]. Gebert *et al* [9] give a detailed explanation of this particular formalism. The derivative for a gene's expression level is given by

$$\frac{dx_i}{dt} = f_i(\mathbf{x}(t)) - g_i(\mathbf{x}(t))x_i(t), \tag{2.04}$$

where $x_i(t)$ is the gene's expression level, $\boldsymbol{x(t)}$ is a vector containing all expression levels, f_i is the synthesis function and g_i is the degradation function. The synthesis function is given by

$$f_i(\mathbf{x}) = \sum_{l \in L_i} k_{il} b_{il}(x_l, \theta_{il}) . \qquad (2.05)$$

The function b_{il} is a step function, that is either given by

$$b_{il} = \begin{cases} 0, x_l < \theta_{il} \\ 1, x_l > \theta_{il} \end{cases} , \qquad (2.06)$$

in which case we write b_{il} as $s^+(x_l, \theta_{il})$, or by $(1 - s^+(x_l, \theta_{il}))$, in which case we write b_{il} as $s^-(x_l, \theta_{il})$. The function f_i is therefore a linear combination of step functions. The degradation function g_i has a similar definition. Given the nature of the function b_{il}, each function f_i has a small set of possible values (the number of distinct values f_i can display is bounded by $2^{|Li|}$, where L_i is the number of genes regulating gene i). The same happens for g_i. The following detail will be important later on: the threshold parameter θ_{il} is particular to a regulating gene l and a regulated gene i. If the same gene l is to regulate another gene j, then the corresponding threshold parameter, θ_{jl}, is not necessarily equal to the former. Thus, for a gene l, there is a set of threshold parameters θ_{jl} related to the set of genes that l regulates.

The equilibrium states for the gene's expression level are given by

$$\frac{dx_i}{dt} = 0 \Leftrightarrow f_i(\mathbf{x}(t)) - g_i(\mathbf{x}(t))x_i = 0, \Leftrightarrow x_i = \frac{f_i(\mathbf{x}(t))}{g_i(\mathbf{x}(t))}, \ g_i(\boldsymbol{x}) \neq 0. \qquad (2.07)$$

The motivation for working with a symbolic, qualitative model instead of with a numeric one is twofold: the symbolic model can summarize the dynamics of a set of numerical models; direct numerical data, especially for the threshold parameters θ_{il} and for the constants k_{il}, is seldom unavailable, being easier to obtain qualitative information such as an ordering of the threshold parameters [4]. In order to obtain a qualitative model, we first define an ordering for the threshold parameters θ_{il}, given a regulating gene l:

$$0 < \theta_l^1 < \ldots < \theta_l^{pl} < \max_l . \qquad (2.08)$$

P_l is the number of genes regulated by gene *l*. Then we define, for every possible combination of values between f_l and g_l, the equilibrium state for gene *l* (see equation 2.07):

$$\theta_l^a < \frac{f_l(\mathbf{x})}{g_l(\mathbf{x})} < \theta_l^b . \qquad (2.09)$$

What is mentioned in equation 2.09 is that, for each combination of values of f_i and g_i, the expression level for gene *l* will tend toward a value in the interval $]\,\theta^a{}_l\,,\theta^b{}_l\,[$. What is important to notice is that, while "on its way" to the equilibrium state, gene *l* may have to pass by an intermediate state. This may cause a regulated gene's expression level to change. If this change propagates back to gene *l*, it may change its dynamics and, as a consequence, change its equilibrium state. This constitutes the GRN's dynamic evolution.

The qualitative dynamics can be represented by a graph, where each node is a GRN state and each edge a transition between two states. A particularly interesting aspect of this kind of dynamics is that there is usually more than one path between two nodes, that is, given a source and destination states, there is usually more than one way to go from one to another. The conclusion from this observation is that one state may have more than one transition coming out from it. This happens for a simple reason: given each state, there is a set of genes that is not in equilibrium. Therefore, each gene in this set will head towards its respective equilibrium level, possibly passing by intermediate levels. There is no information regarding which are the genes that cross into new intermediate levels first. Therefore, all options need to be considered: what happens when the first gene in the set crosses into a new level before the others? What happens when all the genes cross into new levels at the same time? In general, if there are $n > 0$ genes whose expression is changing, we have to consider $2^n -1$ transitions. These transitions reflect different levels of synchronism in the GRN.

The GNA tool allows to configure, simulate and analyze a GRN using a qualitative piecewise differential equations model. In this technical report we show how to use GNA in order to simulate asynchronous dynamics in a Boolean network.

2.3.3. Modeling using Boolean networks

Boolean networks, applied to GRN modeling, were introduced in 1969 by S. Kauffman [6]. In the classical Boolean model, each gene's regulation program is completely defined by a Boolean truth table. In figure 2.3 we illustrate the concept with a simple example.

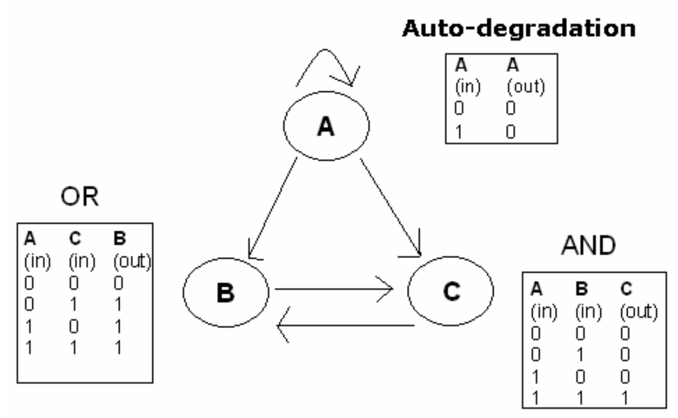

Fig. 2.3 – A simple boolean network.

One should notice that, in the classical model, the update of gene expression levels is synchronous (or parallel). This is in contrast with the qualitative model described in the previous section. However, there are Boolean approaches which relax this assumption. The motivation is that, in GRNs, one observes that each type of chemical reaction (transcription, translation, degradation, phosphorylation, and so on) takes its particular time. Besides, since biological systems are inherently stochastic, the order of occurrence of their processes is not deterministic. In Chaves *et al* [18], two types of asynchronism are tested, over a *Drosophila melanogaster* GRN previously modeled as a synchronous Boolean network. The authors obtain a set of constraints, regarding the relative duration of each chemical reaction, which allow for dynamics consistent with biological data. Another alternative to the classical model is the probabilistic Boolean network: a gene's regulatory program is described by more than one truth table; in each time point, one of the truth tables is selected to determine the gene expression level for the next instant, using a probability distribution. This kind of model tries to portray uncertainty on three levels – biological processes are stochastic, measurements convey errors and models often ignore or excessively approximate relevant variables and processes [17].

Boolean networks represent an extremely simplified version of genetic regulation processes. However, it is argued that these models may be useful in modeling large networks and in portraying large-scale, global effects, something that more complex formalisms fail to do. Today, it is known that the complexity of several biological systems translates into comparatively simple macro dynamics [21]. Therefore, it is important to search for models that ignore molecular details and focus on the system's global behavior. Boolean networks have already been successfully used to reproduce the qualitative dynamics of both wild type and mutant strains, on a pattern segmentation GRN of *Drosophila melanogaster* [19].

Part of this technical report describes a study on a Boolean model of the cell cycle [12]. We show that the model is capable of simulating several mutant strains dynamics; we extend the model in order to differentiate between mRNA and protein expression, which allows for more realistic dynamics; using the GNA tool, we study the dynamics relaxing the assumption of synchronism.

2.3.4. Modeling using piecewise linear equations

Coutinho *et al* [22] introduce a discrete time model with continuous variables. The expression of gene *i,* at time t + 1, is given by

$$x_i^{t+1} = ax_i^t + (1-a)\sum_{j\in I(i)} K_{ij} H(s_{ij}(x_j^t - T_{ij})) \quad , i = 1,...,N. \tag{2.10}$$

The constant $a \in [0,1[$ is the degradation term, equal to all N genes. *I(i)* is the set of genes regulating gene i. Each of those genes j has a regulation strength $K_{ij} > 0$; if, for each gene i, $\sum_{j\in I(i)} K_{ij}$ is normalized to 1, then, for each initial set of points in $\Re^N$, the system's orbit will eventually enter the N-dimensional [0,1] cube [22]. *H(x)* is the Heaviside function,

$$H(x) = \begin{cases} 0, x < 0 \\ 1, x \geq 0 \end{cases}, \tag{2.11}$$

$s_{ij} \in \{-1,1\}$ is the sign of the regulation from gene *j* to gene *i*, and $T_{ij} \in]0,1[$ is the threshold that defines, for the regulation from gene *j* to gene *i*, the frontier between the two possible Heaviside function values.

This particular formalism presents several interesting properties. For example, when the parameter *a* converges to 1, the attractors converge to those of the ordinary differential equation system

$$\frac{dx_i}{dt} = -x_i(t) + \sum_{j \in I(i)} K_{ij} H(s_{ij}(x_j(t) - T_{ij})) . \tag{2.12}$$

Coutinho *et al* also interpret the parameter *a* as a delay parameter [22].
The authors study the dynamics of several artificial networks, which, despite being simple in structure, exhibit non-trivial dynamics, namely periodic orbits whose period depends on the system's parameters. The dynamics, in this formalism, are clearly quantitative, in contrast to the previously described models.
We have applied a particular version of gradient descent to the networks studied by Coutinho *et al* [22], aiming to infer approximate parameters and reproducing the artificial time series given as inputs.

3. Analysis of a Boolean model

In this chapter, we analyze a Boolean model of the yeast cell cycle. First, we review published work; then, we obtain new results regarding mutant strains dynamics, separation between mRNA and protein expression and generalization of the dynamics in conditions of asynchronism.

3.1. Related Work

The yeast cell cycle has been previously modeled as a classical Boolean network by Li *et al* [12]. The authors present two models of the process, but most results pertain to only one of them. Figure 3.1 illustrates the proposed network.

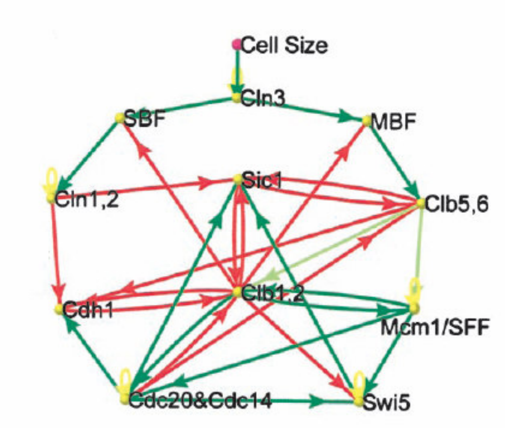

Fig. 3.1 – Model of the yeast cell cycle [12].

All nodes, with the exception of the "*Cell Size*" node, represent genes (there is no distinction between protein or mRNA). Green arrows represent positive regulation, red arrows represent negative regulation and yellow arrows represent auto-degradation. All genes share a similar regulation program, given by the following Boolean function:

$$S_i(t+1) = \begin{cases} 1, \sum_j a_{ij} S_j(t) > 0 \\ 0, \sum_j a_{ij} S_j(t) < 0 \\ S_i(t), \sum_j a_{ij} S_j(t) = 0 \end{cases} . \tag{3.1}$$

In equation 3.1, $S_i(t)$ is the state (0 or 1) of gene i at time t, and a_{ij} is the weight of the regulation from gene j to gene i (1 or -1). Intuitively, the functions $S_i(t)$ represent a democratic voting, in which the state does not change in case of a draw. This is clearly a coarse approximation to the true regulatory programs. In table 3.1 we present the general biological function of the protein corresponding to each modeled gene.

Gene	**Protein function**	**Gene**	**Protein function**
Cln3	Cyclin	Cdh1	Involved in the proteolysis of Clb2
Cln1,2	Cyclin	Cdc14	Phosphatase
Clb5,6	Cyclin	Swi5	Transcription factor
Clb1,2	Cyclin	SBF	Transcription factor
Sic1	Stechiometric inhibitor of Cdc28/Clb2 and Cdc28/Clb5	Mcm1	Transcription factor
Cdc20	Involved in the proteolysis of Clb2 and Clb5	MBF	Transcription factor

Tab. 3.1 – Biological function of each cell cycle protein [20].

In the literature related to cell cycle modeling, there is a very common approximation, even in more complex models, that is to ignore the chemical reactions between each cyclin and the Cdc28 kinase [20]. In the model we study, we use the same approximation, that is, we assume that Cdc28 concentration is in excess, and that the chemical reaction between Cdc28 and each cyclin is instantaneous. It is important to point out that the network in figure 3.1 contains arbitrary regulations, which are necessary for obtaining correct network dynamics. First, only some genes suffer auto-degradation. These are the genes which do not suffer any negative regulation from other genes. The absence of an auto-degradation term would prevent those genes from ever going from state 1 to state 0. This is the justification presented by Li *et al* [12]. However, the gene *Swi5*, despite being negatively regulated by gene *Clb1,2*, also degrades itself. This is inconsistent with the previous explanation. Nevertheless, our experiments suggest that *Swi5* auto-degradation is necessary for proper network dynamics. These examples show that the criterion for imposing some arrows in the network was not solely biological.

The network, when initialized to a state that corresponds to the beginning of the G1 phase, goes through a series of states that can be approximately identified to the sequence of events in the biological cell cycle described in chapter 2. See Li *et al* [12] for the full state sequence.
The authors analyze the network dynamics, beginning from each of its 2^{11} states. They observe that 1784 states, corresponding to 86,13% of the total number of states, converge to the same attractor. This attractor corresponds to the end of the cell cycle. The authors also point out that most states, before converging to the attractor, transit to one of the “cell cycle states” [12].
The network from figure 3.1 was compared to several random graphs with the same number of nodes and edges, in order to observe whether similar results regarding trajectory convergence could be observed by chance. It was observed that the probability of an attractor in a random network having a basin of attraction at least as big as 1784 states is 0.1034. Since 10% is a significant magnitude for a p-value, we conclude that the original result, about the basin of attraction of the state representing the end of the cell cycle, is not particularly strong. In order to quantify trajectory convergence, the authors measure the average traffic on the path from each state to its attractor. Given an edge, its traffic is defined as the number of trajectories that pass by that edge [12]. Since, in general, there is a lower trajectory convergence in random networks than in the cell cycle graph, the natural result is that the distribution of average traffic, in random networks, has a lower peak, and is denser around smaller average traffics, than the distribution in the cell cycle network. Finally, the authors perturb the system, by modifications such as removing an arrow or changing an arrow’s color. They observed that most changes did not significantly affect the system dynamics.

3.2. Mutant strains

3.2.1. Experimental procedure

A mutant strain can cause a range of effects in the cell cycle, depending on which gene is eliminated. Intuitively, the more important the gene is for the cell cycle, the greater the effects will be. While some of these effects are quantitative, and therefore difficult, if not impossible, to express in formalisms such as Boolean networks, others are of a qualitative nature and serve as a way to validate a hypothetical network. In particular, some mutant strains cause an arrest in the cell cycle. Our objective was to observe whether the cell cycle Boolean network of figure 3.1 was capable of correctly simulating some of these mutants. We simulated the network for a set of single (one eliminated gene) and double (two eliminated genes) mutant strains which are known to cause severe changes in the cell cycle.

3.2.2. Results

In tables 3.2 and 3.3 we summarize our results regarding single and double mutant strains.

Mutant	**Biological cell cycle**	**Model**	**Size of the attraction basin (% of all states) (if not mentioned, the attractor is the biggest in the system)**	**Number of attractors (the original system has 6)**
ΔCln3	Viable	Static	83,54	7
ΔSbf	Unviable	Unviable	22,95	6
ΔMbf	Viable	Unviable	31,74 **(*)**	5
ΔCln1,2	Viable, arrest in G1	Arrest in G1	21,29	12
ΔSic1	Viable,G1 lasts less	Arrest in M2, G1 lasts less	80,27	5
ΔClb5,6	Viable, S phase lasts longer	Arrest in S	33,59	10
ΔCdh1	Viable, degrades less Clb2	Viable, doesn't degrade less Clb2	89,26	4
ΔClb1,2	Viable, arrest in G2	Arrest in G2	62,5	8
ΔMcm1/SFF	Unviable	Viable	70,8	7
ΔCdc20/Cdc14	Unviable, arrest in metaphase	Arrest in M1	66,7	9
ΔSwi5	Viable	Arrest in M2	66,7	9

(*) – The attractor is only the second greatest of the system.

Tab. 3.2 – Simulating single mutant strains.

Mutant	**Biological cell cycle**	**Model**	**Size of the attraction basin (% of all states) (if not mentioned, the attractor is the biggest in the system)**	**Number of attractors (the original system has 6)**
ΔCln1 ΔSwi5	Viable	Arrest in M2	78,91	7
ΔCln1 ΔCdh1	Viable	Arrest in G1	18,75 **(*)**	6
ΔCln1 ΔClb5	Arrest in G1	Arrest in G1	60,16	12
ΔCln1 ΔSic1	Viable	Arrest in M2	78,91	7
ΔCdc20 ΔClb5	Arrest in metaphase	Arrest at the end of G1	27,93 **(*)**	11
ΔSic1 ΔCdh1	Arrest in mitosis	Arrest in M2	85,94	3
ΔSwi5 ΔCdh1	Arrest in telophase	Arrest in M2	73,63	5
ΔClb1 ΔCdh1	Viable	Arrest in a state with no parallel in the biological cell cycle	62,5	5

(*) – The attractor is only the second greatest of the system.

Tab. 3.3 – Simulating double mutant strains.

The symbols M1 and M2 refer to the prophase and metaphase processes, respectively. The column named "mutant" refers to the genes that were eliminated. The column named "biological cell cycle" summarizes the biological effects of the mutant strain. A viable mutant is one in which the organism (in this case, yeast) continues to grow, even if it cannot divide itself(in the cases where the cell cycle arrests). An unviable mutant does not allow the organism to survive. For example, the ΔClb1,2 mutant is viable and its cell cycle arrests in the G2 phase, while the mutant ΔCdc20/Cdc14 is unviable and arrests in the metaphase.
The dynamics of 8 out of the 19 tested mutants (42%) are in concordance with biological results. These mutants are ΔSbf, ΔCln1,2 , ΔCdh1, ΔClb1,2 , ΔCdc20/Cdc14, ΔCln1ΔClb5, ΔSic1ΔCdh1 and ΔSwi5ΔCdh1. In all of them, the "end of cell cycle" attractor is the one with the largest basin of attraction. The average number of attractors is 7.4, in comparison with the 6 attractors of the original system. We believe that incomplete modeling of the cell cycle (the network in figure 3.1 is a simplified skeleton of the process) and the use of overly simplistic Boolean functions are the two major reasons for not having achieved the desired results with other mutants.

3.3. Separating mRNA from protein

3.3.1. Experimental procedure

The arrows in the network of figure 3.1 represent more than one type of chemical reaction. For example, the transcription factor SBF regulates the rate of mRNA production for gene Cln1,2; Cln1,2 regulates not the gene Sic1, but its respective protein, via a phosphorylation reaction. This happens because each node has an ambiguous meaning – it means mRNA in some cases and protein in others. In order to obtain a less ambiguous relation between the graph components and their biological meaning, we have separated each node into two, one representing mRNA, the other representing protein. Arrows were changed accordingly. We repeated the mutant strains experiment for the new network.

3.3.2. Results

The mRNA-protein separation caused richer and more realistic dynamics for three genes: Sic1, Clb1,2 and Clb5,6:

- In figure 3.2 one can observe the differences in expression between Sic1 mRNA and protein, during a biological cell cycle [23].

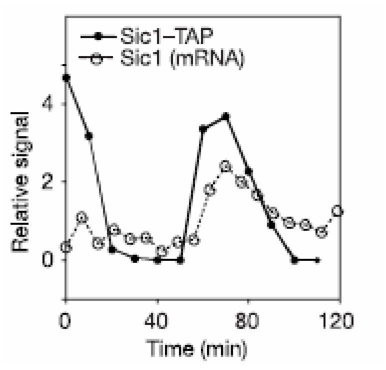

Fig. 3.2 – Protein (TAP) and mRNA expression, for Sic1, during the cell cycle [23].

It is important to notice that, in the G1 phase, protein expression is much higher that mRNA. Our new model allows expressing this difference, and obtaining dynamics qualitatively similar to figure 3.1;

- Clb1,2 degradation is sometimes due to post-translational regulation, not to a decrease in mRNA transcription. Our new model allows for this differentiation;

- The Clb5,6 protein is degraded by Cdc14/20, while its mRNA transcription rate is reduced by Sic1. With the new model, we obtain new expression profiles in which one can distinguish between lack of Clb5,6 mRNA or lack of its protein (this is important because Cdc14/20 and Sic1 are not acting on Clb5,6 at the same time).

These network improvements did not increase the number of mutant strains achieving satisfactory dynamics, but improved the realism of the dynamics of those which already presented good results.

3.4. Asynchronous dynamics

3.4.1. Experimental Procedure

An interesting question is: how will the Boolean network behave, if we relax the assumption of synchronism in gene expression update? As we have seen in section 2.3.2., the GNA tool is adequate for the simulation of asynchronous systems. In abstract, for a given state of the system, there is a set of genes whose expression has to change. What happens if a subset of those genes changes before the rest? GNA essentially evaluates all options: if, given a state, there are *n* genes whose expression must change, there are 2^n-1 possible transitions out of that state. We have analyzed the cell cycle network from figure 3.1 using GNA.

3.5.2. Results

We have observed that, from the initial state representing the beginning of the cell cycle, there are many paths which converge to the stable state representing the end of the process. This result suggests that the network is robust, not only in the classical sense (we observed that most states converge to the same attractor), but also in the sense that, even if genes are not synchronously updated, the system will probably still converge to the correct final state. This leads us to believe that, if the network was to be modeled using any particular kind of asynchronism [18], it would still possess correct dynamics. We obtain similar results for mutant strains. For those that originally presented correct synchronous dynamics, we observed that the correct final state is stable, that is, has no transitions coming out of it; we also noticed that this state has a large in-degree, which extends the conclusion of network robustness to the mutant strains.

GNA has a very limited capability of automatically analyzing the transition graph. On future studies, one should quantitatively analyze these generalized dynamics, and determine the number of trajectories that converge to states that correspond to cell cycle stages. One could also study which kinds of asynchronism preserve the properties that were observed in the synchronous network. For example, Li *et al* [12] refers that the network was tested for the case when transcriptional regulation processes have a different duration than phosphorylation processes, and that the same results were obtained. However, they fail to mention which actual values were used.

4. Analysis of a piecewise linear equations model

We study the problem of parameter inference in a GRN under a piecewise linear equations model, using expression data. The GRN topology is specified. The problem becomes one of finding the numerical values for the model parameters that generate dynamics as close as possible to the input expression data. We use a gradient descent algorithm, with momentum term and adaptive learning rate. We test the algorithm on several artificial networks.

4.1. Parameter estimation using gradient descent

The classical gradient descent (GD) algorithm is well-known among computer scientists. Consider a function with continuous first derivative $f\colon \Re^n \to \Re$. The GD algorithm's objective is to find a point $p \in \Re^n$ such that *f(p)* is a global optimum. A global optimum may be the function's global maximum or minimum, depending on whether we want to maximize or minimize the function. In practice, the GD algorithm finds a local optimum. Let the gradient of *f* at point ***p*** be

$$\nabla f(p) = (\frac{\partial f}{\partial p_1}, \ldots, \frac{\partial f}{\partial p_n}). \tag{4.01}$$

The algorithm relies on the following two observations: $\nabla f(p)$ indicates the direction of the steepest increase in *f* for point ***p***; if $\nabla f(p) = 0$, then ***p*** is a local optimizer of *f*.

The algorithm works in the following way: first, it generates a first point $\boldsymbol{p^0}$ and calculates $\nabla f(p^0)$. If the objective is to maximize the function, then the algorithm obtains a new point $\mathbf{p}^1 = \mathbf{p}^0 + \eta \nabla f(p^0)$, and calculates $\nabla f(p^1)$. The idea is to move in the direction of steepest increase. The parameter η is a scaling factor for the step size. The algorithm proceeds in this fashion until finding a point for which the gradient is 0, or until the gradient becomes less than a specified threshold. In the case where a minimum, instead of an optimum, is to be found, the parameter update equation would become $\mathbf{p}^1 = \mathbf{p}^0 - \eta \nabla f(p^0)$.

If η is sufficiently small, the GD algorithm converges to a local optimum. If it is excessively small, convergence will be slower. If η is too big, oscillation around a local optimum will occur instead of convergence.

There are several improvements on the classical GD method. We describe two of them – the momentum term and the adaptive learning rate. As for the momentum term, the idea is that, when we change the value of the parameter, we should take into account the last change that was made to it. That is, the parameter update equation ceases to be

$$p_i^j \leftarrow p_i^{j-1} - \eta \frac{\partial f}{\partial p_i}(\mathbf{p}^{j-1}) \tag{4.02}$$

and becomes

$$p_i^j \leftarrow p_i^{j-1} - \eta \frac{\partial f}{\partial p_i}(\mathbf{p}^{j-1}) + \alpha(p_i^{j-1} - p_i^{j-2})\,. \tag{4.03}$$

In equation 4.03, $\alpha \in [0,1]$ is the momentum constant. $\alpha = 0.9$ is considered to be a reasonable value [25]. For $\alpha = 0$ we obtain the classical method. Adding a momentum term accelerates convergence and makes the method find better local optima. As for the adaptive learning rate, the major motivation is the following: the constant η that guarantees a quick convergence varies throughout the parameter space; the corollary of this is that the use of the same η in the entire search procedure often leads to slow and poor convergence. The idea is to change η according to the success the algorithm is having. We use the following formulation, for when the objective is to minimize *f*[25]:

$$\Delta\eta = \begin{cases} a, \Delta f < 0, consistently\ * \\ -b\eta, \Delta f > 0 \\ 0, c.c. \end{cases} \tag{4.04}$$

*error diminished in the last *k* iterations.

If *f* is systematically decreasing, there is evidence that we are in a parameter region that allows for a greater learning rate; if *f* increases, then η is to big for the current parameter region. Notice that the update is conservative: η increases linearly but decreases geometrically. Whenever *f* increases, besides decreasing η, we should also undo the last parameter update and temporarily have α (the momentum term) equal to 0 [25].

4.2. Using a sigmoid function to model gene activation

A sigmoid is a continuous function of the form

$$S(x) = \frac{1}{1+e^{-x}}. \tag{4.05}$$

The derivative of a sigmoid is given by

$$\begin{aligned}
\frac{dS}{dx} &= \frac{d}{dx}\left(\frac{1}{1+e^{-x}}\right) = -(1+e^{-x})^{-2} * e^{-x} * -1 = \frac{1}{1+e^{-x}} * \frac{1}{1+e^{-x}} * e^{-x} = \\
&= \frac{1}{1+e^{-x}} * \frac{e^{-x}}{1+e^{-x}} = \frac{1}{1+e^{-x}} * \frac{1+e^{-x}-1}{1+e^{-x}} = \\
&= \frac{1}{1+e^{-x}} * \left(\frac{1+e^{-x}}{1+e^{-x}} - \frac{1}{1+e^{-x}}\right) = S(x) * (1-S(x))
\end{aligned} \tag{4.06}$$

It is usual to use, instead of the classical sigmoid of equation 4.05, a composite function $S_2(x,r) = S(r*x)$. Figure 4.1 shows the plot of $S_2(x,r)$, for r = 1,2 and 5.

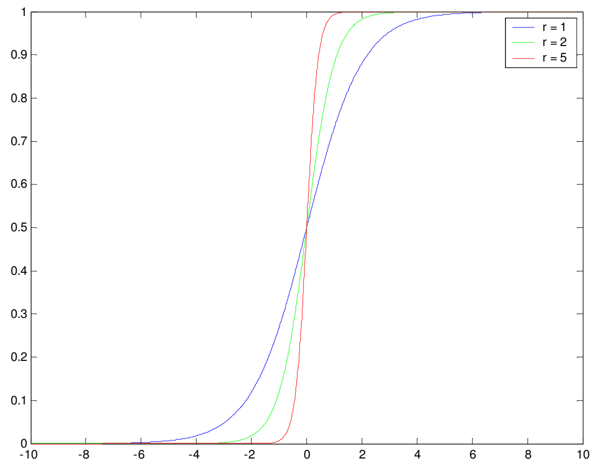

Fig. 4.1 – S_2(x,r), for r = 1,2 and 5.

It is important to notice that S_2 saturates when x is very low or very high. The steepness of the transition from a saturation area to another depends on the steepness parameter *r*. There is experimental evidence that mRNA production rates, as a function of transcription factor concentration, follow this type of function [4].

It is clear that, as $r \to \infty$, $S_2(x,r) \to H(x)$, that is, the sigmoid tends toward a Heaviside function. This is a very useful result, as we will see in the next section.

4.3. A sigmoid function as an approximation to a Heaviside function

In the piecewise linear equations model that we have described in section 2.3.4., the expression of a gene *i*, at instant *t+1*, as a function of its regulators' expression, is given by

$$x_i^{t+1} = ax_i^t + (1-a)\sum_{j\in I(i)} K_{ij} H(s_{ij}(x_j^t - T_{ij})) \ , i = 1,\ldots,N \tag{4.07}$$

This function is discontinuous, because it relies on Heaviside functions. As such, it cannot be used by a gradient descent algorithm. Our solution is to approximate a Heaviside function by a sigmoid. We use a low steepness for the initial sigmoid, so that the function parameters from 4.07 can more freely change. On each iteration we increase the steepness parameter. The idea is to, from a parameter set that leads to a certain error for a certain sigmoid steepness, find a parameter set that is slightly more adequate, for a slightly more adequate sigmoid steepness.

4.4. Quadratic error and partial derivatives

We intend to minimize the quadratic error over all patterns of gene expression. The quadratic error is given by

$$E = \sum_{g=1}^{G} \sum_{t=2}^{T} (x_g^t - O_g^t)^2 \ , \tag{4.08}$$

where *T* is the number of time points in the experiment, *G* is the number of genes in the network, x_g^t is the expression for gene *g* at time point *t* that we pretend to approximate, and O_g^t is the expression of gene *g* at time point *t*, according to the model at the current iteration. *E(g,t)* is the component in *E* that refers to gene *g* and time point *t*.

Using sigmoids instead of Heaviside functions, as mentioned in the previous section, and using the notation defined in the previous equation, expression of a gene *i* at time point *t+1* is given by

$$O_i^{t+1}(a, \theta_i) = a x_i^t + (1-a) \sum_{j \in I(i)} K_{ij} S(r * s_{ij}(x_j^t - T_{ij})) \tag{4.09}$$

where θ_i é the set of parameters that defines the regulation over gene *i*. We do not include parameter *a* in that set, because that parameter is of concern to the entire network.

We now need to calculate the partial derivatives for the quadratic error. We define S_{ij}^t as

$$S_{ij}^t = S(r * s_{ij}(x_j^t - T_{ij})) \, . \tag{4.10}$$

Using the previous definition, the partial derivatives considering only a gene and a time point are the following:

$$\frac{\partial E(g,t)}{\partial a} = (x_g^t - O_g^t)(-x_g^{t-1} + \sum_{j \in I(g)} k_{ij} S_{ij}^{t-1}) \quad , \tag{4.11}$$

$$\frac{\partial E(g,t)}{\partial k_{ij}} = \begin{cases} 0, i \neq g \\ (x_g^t - O_g^t)(-(1-a)S_{ij}^{t-1}), c.c. \end{cases}, \tag{4.11}$$

$$\frac{\partial E(g,t)}{\partial s_{ij}} = \begin{cases} 0, i \neq g \\ (x_g^t - O_g^t)(-(1-a)k_{ij}S_{ij}^{t-1}(1-S_{ij}^{t-1})r(x_j^{t-1} - T_{ij})), c.c. \end{cases} \tag{4.12}$$

and

$$\frac{\partial E(g,t)}{\partial T_{ij}} = \begin{cases} 0, i \neq g \\ (x_g^t - O_g^t)(1-a)k_{ij}S_{ij}^{t-1}(1-S_{ij}^{t-1})rs_{ij}, c.c. \end{cases}. \tag{4.13}$$

The complete partial derivatives are obtained by summing over all time points and genes. There are two technical details that should be pointed out. First, the parameters s_{ij} (the sign of the regulation), which should be integers ($s_{ij} \in \{-1,1\}$), are being modeled as continuous real variables ($s_{ij} \in [-1,1]$). Second, the gradient descent algorithm does not allow any parameter to exceed its definition interval. For example, a cannot be out of the interval [0,1].

4.5. Testing the algorithm on artificial networks

The algorithm was tested on the artificial networks named *self-inhibitor, negative-2, positive-2* and *feed-forward loop*. All the networks, except for the last one, were taken from Coutinho *et al* [22]. The *feed-forward loop* [13] was considered with the intention of observing whether the GD algorithm was capable of finding the right parameters for a network motif that frequently occurs in GRNs. All the networks, with the exception of the *feed-forward loop*, exhibit oscillatory behavior. Figures 4.2 – 4.5 illustrate the expression profiles we intend to approximate.

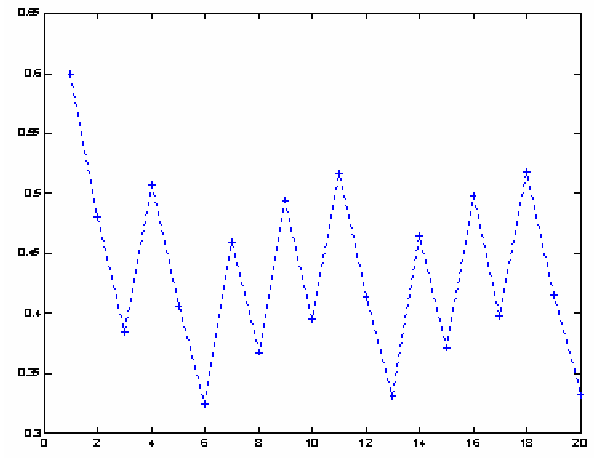

Fig. 4.2 – Gene expression in *self-inhibitor*.

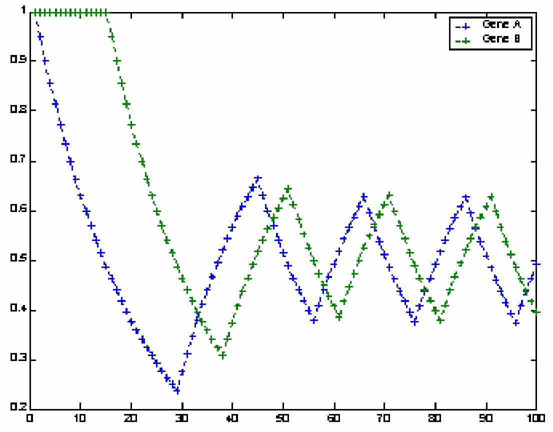

Fig. 4.3 – Gene expression in *negative-2*

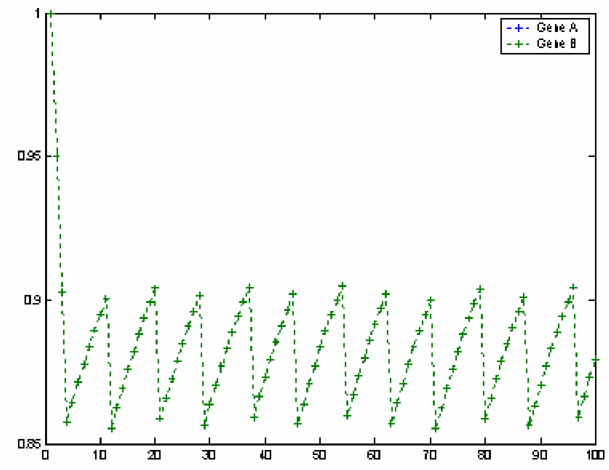

Fig. 4.4 – Gene expression in *positive-2*

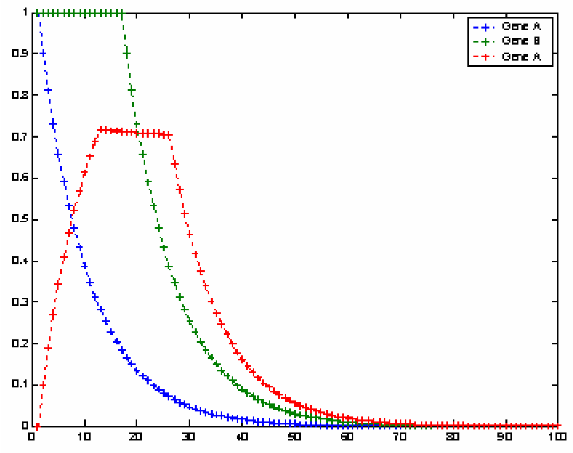

Fig. 4.5 – Gene expression in *feed-forward loop*

4.6. Results

4.6.1. Algorithm parameters

We ran our GD algorithm using the following parameters:

Sigmoids' initial steepness	0.1
Steepness increase after an iteration where the error decreases	0.1
Steepness increase after an iteration where the error does not decrease	1
Learning rate, η	0.001
Momentum constant	0.9
Increase in η, when the error diminishes systematically	0.001
Decrease in η, when the error increases	0.5
Number of consecutive iterations, in which the error decreases, that are necessary for increasing η.	2
Maximum number of iterations	3000
Number of times the algorithm is executed	10

Tab. 4.1 – Parameters for the gradient descent algorithm.

To generate expression data, we simulated each network for 100 consecutive time points. The GD algorithm, after running 10 times, returns the best set of parameters that was found.

4.6.2. Inferred network parameters and corresponding gene expression

In all test cases, the algorithm managed to attain parameters very close to the optimal solution. These parameters lead to gene expression patterns that very closely resembled the ones given as input. In figure 4.6 we show the difference between the original gene expression and the one obtained after parameter inference, for the *negative-2* network:

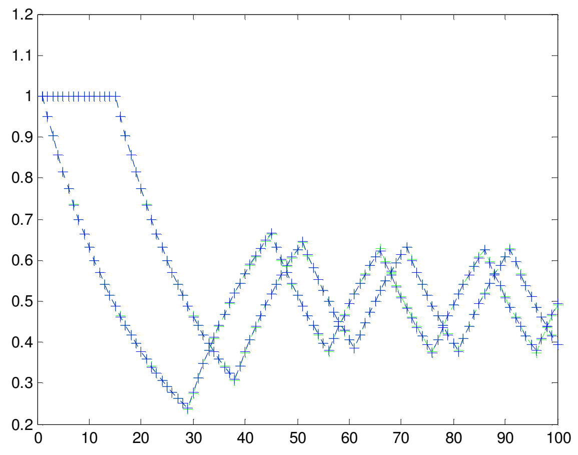

Fig. 4.6 – Difference between original and inferred gene expression, for the *negative-2* GRN.

In figure 4.6 there are four curves; the blue curves were given as input, while the green curves were obtained after parameter inference. The curves are so similar that they seem juxtaposed. The same occurred for the remaining networks. In tables 4.2 – 4.5 we compare original and inferred parameters for all networks. It can be seen that the inferred parameters are very close to the true ones. Another observation is that the s_{ij} parameters, despite being modeled as real variables, tend toward the original integer values.

Negative-2		
Parameter	**Original value**	**Inferred value**
a	0.95	0.95
$K_{A \rightarrow B}$	1	1.0004
$K_{B \rightarrow A}$	1	1.0013
$T_{A \rightarrow B}$	0.5	0.5023
$T_{B \rightarrow A}$	0.5	0.5057
$S_{A \rightarrow B}$	1	1
$S_{B \rightarrow A}$	-1	-1

Tab. 4.2 – Original and inferred parameters, in *negative-2*.

Self-inhibitor		
Parameter	**Original value**	**Inferred value**
a	0.8	0.7999
$K_{A \rightarrow A}$	1	1.0022
$T_{A \rightarrow A}$	0.4	0.4047
$S_{A \rightarrow A}$	1	1

Tab. 4.3 – Original and inferred parameters, in *self-inhibitor*.

Positive-2		
Parameter	**Original value**	**Inferred value**
a	0.95	0.95
$K_{A \rightarrow B}$	1	0.9962
$K_{B \rightarrow A}$	1	0.9962
$T_{A \rightarrow B}$	0.9	0.9003
$T_{B \rightarrow A}$	0.9	0.9003
$S_{A \rightarrow B}$	-1	-1
$S_{B \rightarrow A}$	-1	-1

Tab. 4.4 – Original and inferred parameters, in *positive-2*.

Feed-forward loop		
Parameter	**Original value**	**Inferred value**
a	0.9	0.8998
$K_{A \rightarrow B}$	1	1.0019
$K_{A \rightarrow C}$	0.3	0.2989
$K_{B \rightarrow C}$	0.7	0.6999
$T_{A \rightarrow B}$	0.2	0.1956
$T_{A \rightarrow C}$	0.3	0.2978
$T_{B \rightarrow C}$	0.4	0.4062
$S_{A \rightarrow B}$	1	1
$S_{A \rightarrow C}$	1	1
$S_{B \rightarrow C}$	1	1

Tab. 4.5 – Original and inferred parameters, in *feed-forward loop*.

5. Conclusion

In this technical report we described two computational approaches to the problem of modeling and inferring GRNs.

We conclude that the Boolean network formalism allows the modeling of qualitative and essential aspects of GRN dynamics, even when using general voting functions that, apart from distinguishing positive from negative regulations, do not reflect the different degrees of influence between genes. Applied to the yeast cell cycle, the Boolean network approach allowed for the correct simulation of the wild type and 8 mutant strains. Although these 8 strains correspond to only 42% of the total number of experimented mutant strains, we believe that, by extending the model network with more genes and using more realistic Boolean functions, the number of successful cases would increase. We show that separating mRNA from protein originates a semantically less ambiguous model, besides improving the realism of the dynamics. This separation is often ignored in GRN modeling literature. We find evidence that the network is robust to changes in synchronism. In the future, using a concrete type of asynchronism, as well as enriching the cell cycle model with more genes known to participate in the process, will clarify these aspects of the Boolean network.

We show that an improved version of the gradient descent algorithm is capable of inferring parameters from artificial expression data, given as input along with a network topology. The networks used for testing, although being simple in architecture, display non-trivial dynamics. As future research directions, we intend to model a biological stress-response GRN using the same piecewise linear equations formalism, and extend the learning algorithm so that it is capable of learning the expression of hidden variables (variables for which their expression is not provided as input).

6. References

[1] E.Segal *et al*, From signatures to models: understanding cancer using microarrays, *Nature Genetics,* Vol. 37, pp. S38 – S45, June 2005.

[2] D.G. Fraenkel, The top genes: on the distance from transcript to function in yeast glycolysis, *Current Opinion in Microbiology*, no 6, pp. 198-201, 2003.

[3] M. Levine e E.H. Davidson, Gene regulatory networks for development, *PNAS*, vol. 102, no 14, pp 4936 – 4942, 5th of April, 2005.

[4] H. de Jong *et al*, Genetic Network Analyzer: qualitative simulation of genetic regulatory networks, *Bioinformatics*, Vol. 19, no 3, pp. 336-344, 2003.

[5] P.T.Spellman *et al*, Comprehensive Identification of Cell Cycle-regulated Genes of the Yeast *Saccharomyces cerevisiae* by Microarray Hybridization, *Molecular Biology of the Cell*, Vol. 9, pp. 3273-3297, December, 1998.

[6] K. Murphy e S. Mian, *Modeling Gene Expression Data using Dynamic Bayesian Networks,* University of California, Berkeley.

[7] A new dynamic Bayesian network (DBN) approach for identifying gene regulatory networks from time course microarray data, *Bioinformatics*, vol. 21, no. 1, pp 71 - 79, 2004.

[8] B. Perrin *et al*, Gene networks inference using dynamic Bayesian networks, *Bioinformatics*, vol. 19, suppl.2, pp ii 138 – ii 148, 2003.

[9] J. Gebert *et al*, Modeling Gene Regulatory Networks with Piecewise Linear Differential Equations, *Elsevier Science*, 28th of October, 2004.

[10] H. Chen *et al*, Quantitative characterization of the transcriptional regulatory network in the yeast cell cycle, *Bioinformatics*, Vol. 20, no 12, pp. 1914 – 1927, December, 2004.

[11] E. Segal *et al*, Genome-wide discovery of transcriptional modules from DNA sequence and gene expression, *Bioinformatics*, vol. 19, suppl. 1, pp i 273 – i 282, 2003.

[12] F. Li *et al*, The yeast cell-cycle network is robustly designed, *PNAS*, Vol. 101, No. 14, pp 4781 – 4786, 6th of April, 2004.

[13] R. Milo *et al*, Network Motifs: Simple Building Blocks of Complex Networks, *Science*, Vol. 298, pp. 824 – 827, 25th of October, 2002.

[14] S. Mangan e U. Alon, Structure and function of the feed-forward loop network motif, *PNAS*, Vol. 100, No. 21, pp. 11980 – 11985, 14th of October, 2003.

[15] T. Mestl *et al*, A mathematical framework for describing and analyzing gene regulatory networks, *Journal of Theoretical Biology*, No. 176, pp. 291 – 300, 21st of September, 1995.

[16] S. Kauffman, Metabolic Stability and Epigenesis in Randomly Constructed Genetic Nets, *Journal of Theoretical Biology*, Vol. 22, pp. 437 – 467, 1969.

[17] I. Shmulevich *et al*, From Boolean to Probabilistic Boolean Networks as Models of Genetic Regulatory Networks, *Proceedings of the IEEE*, Vol. 90, No.11, pp. 1778 -1792, November, 2002.

[18] M. Chaves *et al*, Robustness and Fragility of Boolean Models for Genetic Regulatory Networks, *Journal of Theoretical Biology*, No. 235, pp. 431 – 449, 2005.

[19] R. Albert e H.G. Othmer, The topology of the regulatory interactions predicts the expression pattern of the *Drosophila* segment polarity genes, *Journal of Theoretical Biology*, No. 233, pp. 1 - 18, 2003.

[20] S. Bornholdt, Less is More in Modeling Large Genetic Networks, *Science*, Vol. 310, No. 5747, pp. 449 – 450, 21st of October, 2005.

[21] O. Brandman *et al*, Interlinked Fast and Slow Positive Feedback Loops Drive Reliable Cell Decisions, *Science*, Vol. 310, No. 5747, pp. 496 – 498, 21st of October, 2005.

[22] R. Coutinho *et al*, Discrete time piecewise affine models of genetic regulatory networks, *Journal of Mathematical Biology*, No. 52, pp. 524 – 570, 2006.

[23] S. Ghaemmaghami *et al*, Global analysis of protein expression in Yeast, *Nature*, Vol.425, pp. 737 – 741, 16th of October, 2003.

[24] K.C. Chen *et al*, Integrative analysis of cell cycle control in budding yeast, *Molecular Biology of the Cell*, No.15, pp. 3841 – 3862, 2004.

[25] J. Hertz *et al*, *Introduction to the Theory of Neural Computation*, Redwood City, CA, USA, Addison-Wesley, 1991.

[26] H. Anton *et al*, *Calculus*, *seventh edition*, New York, USA: John Wiley and Sons, Inc., 2002.