27

Identifying Gene Regulatory Networks from Gene Expression Data

Vladimir Filkov
University of California, Davis

27.1 Introduction

The ultimate goal of the genomic revolution is understanding the genetic causes behind phenotypic characteristics of organisms. Such an understanding would mean having a blueprint which specifies the exact ways in which genetic components, like genes and proteins, interact to make a complex living system. The availability of genome-wide gene expression technologies has made at least a part of this goal closer, that of identifying the interactions between genes in a living system, or *gene networks.* Well suited for both qualitative and quantitative level modeling and simulation, and thus embraced by both biologists and computational scientists, gene networks have the potential to elucidate the effect of the nature and topology of interactions on the systemic properties of organisms.

Gene networks can be modeled and simulated using various approaches [9, 17]. Once the model has been chosen, the parameters need to be fit to the data. Even the simplest network models are complex systems involving many parameters, and fitting them is a non-trivial process, known as *network inference*, *network identification*, or *reverse engineering.*

This chapter reviews the process of modeling and inference of gene networks from large-scale gene expression data, including key modeling properties of biological gene networks, general properties of combinatorial models, specifics of four different popular modeling frameworks, and methods for inference of gene networks under those models.

Our knowledge about gene networks, and cellular networks in general, although still limited, has grown in the past decade significantly, due mostly to advances in biotechnology. Some of those known and key properties of gene networks, which can aid in their understanding and modeling, especially with the discrete, combinatorial methods covered in this chapter, are described in Section 27.2. Some properties, like low average connectivity, or the nature of cis-trans interactions during transcription have been used repeatedly in modeling and inference of gene networks.

This chapter is de Some background on the nature of large-scale gene expression experiments together with very short description of methods used for the analysis of the observed data is given in Section 27.3. More detailed descriptions of both the technology and the data analysis methods can be found elsewhere in this book.

Before the actual methods are described in detail, general properties of modeling formalisms which both define and limit them are described in Section 27.4.

Those properties (synchronicity, stochasticity, etc.) are necessary considerations when modeling gene networks and define the resulting models to a large extent.

The four large classes of modeling formalisms covered in this chapter are graph theoretical models in Section 27.5, Bayesian networks in Sec. 27.6, Boolean networks in Sec. 27.7, and Linearized differential equation models 27.8. Together with the models and their properties inference methods and algorithms are presented from the most influential research articles in the area, together with pertinent results on both steady-state and time-course gene expression data. The relationship between model complexity and amount/type of data required versus the quality of the results is underlined.

At the end of the chapter the models and methods for inference are summarized and future directions toward better gene network inference are outlined.

This chapter is not comprehensive with respect to the different frameworks available for modeling gene networks, and more thorough reviews exist in that respect [17]. The emphasis here is on an integrated presentation of the models and the methods for network inference for them. Gene network modeling using *feedback control theory* is presented in another chapter of this book.

27.2 Gene Networks

27.2.1 Definition

Gene regulation is a general name for a number of sequential processes, the most well known and understood being transcription and translation, which control the level of a gene's expression, and ultimately result with specific quantity of a target protein.

A *gene regulation system* consists of genes, cis-elements, and regulators. The regulators are most often proteins, called *transcription factors*, but small molecules, like RNAs and metabolites, sometimes also participate in the overall regulation. The interactions and binding of regulators to cis-elements in the cis-region of genes controls the level of gene expression during transcription. The cis-regions serve to aggregate the input signals, mediated by the regulators, and thereby effect a very specific gene expression signal. *The genes, regulators, and the regulatory connections between them, together with an interpretation scheme form gene networks.*

Depending on the degree of abstraction and availability of empirical data, there are different levels of modeling of gene networks. Figure 27.1 shows a hypothetical gene network together with different levels at which it can be modeled. The particular modeling level depends on the biological knowledge and the available data, as well as the experiment goal, which can be as simple as hypothesis testing, or as complex as quantitative network modeling.

There are a number of gene regulatory networks known in great detail: the lysis/lysogeny cycle regulation of bacteriophage-λ [51], the endomesoderm development network in Sea Urchin [16], and the segment polarity network in Drosophila development [68, 5]. The first

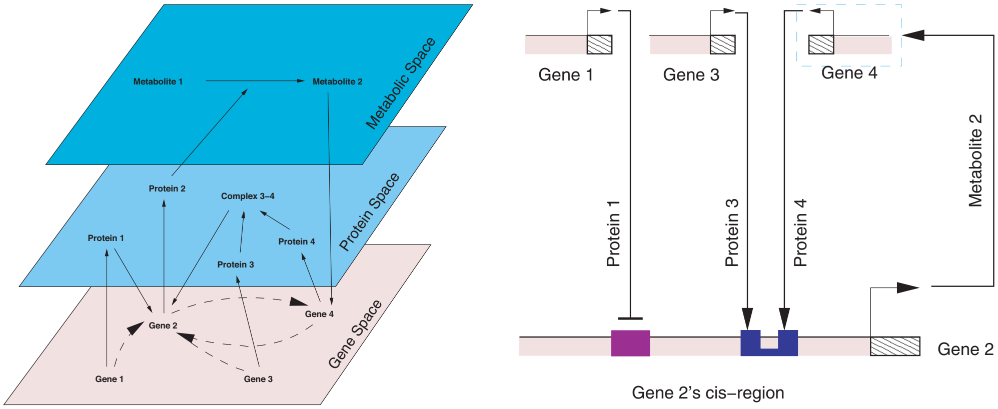


FIGURE 27.1: **(See color insert following page 20-4.)** A hypothetical gene network. Shown on the left (redrawn from [10]) are the multiple levels at which genes are regulated by other genes, proteins and metabolites. On the right is a useful abstraction subsuming all the interactions into ones between genes only. The cis-regions are shown next to the coding regions, which are marked with pattern fill and start at the bent arrows. The edges are marked with the name of the molecule that carries the interaction. Some reactions represent trans-factor – DNA binding, happen during transcription, and are localized on the cis-regions. In those cases the corresponding protein-specific binding sites, or cis-elements, on the cis-regions are shown (colored polygons). Otherwise, the interactions can take place during transcription or later (e.g. post-translational modifications) as may be the case with Metabolite 2 interacting with Gene 4. The nature of the interactions is inducing (arrow) or repressing (dull end).

two networks are mostly qualitative representations of the relationships among the genes, while the last involves quantitative models. Many other networks are available in some detail, laid out in databases which are publicly available, like KEGG [37] and EcoCyc [38]. Such detailed networks are extracted from the published work of many researchers working on individual links in the networks. With the advent of large-scale technologies in genomics things are becoming faster by orders of magnitude, and there is potential for automating the process of network discovery.

27.2.2 Biological Properties

Substantially more is known about gene regulation and networks today than a decade ago. Some of this knowledge can be used to effectively model gene networks. What follows is a collection of mostly empirical facts which, although few in number, are powerful modeling and design rules.

Topology

The topology of a network defines the connections between nodes, and it can be a starting point for modeling. One of the most important and powerful rules in gene network modeling is that their topology is sparse, i.e. there is a small constant number of edges per node, much smaller than the total number of nodes. This rule is a result of biological observations that genes are regulated by a small constant number of other genes, 2-4 in bacteria [43] and 5-10 in eukaryotes [6]. The sparseness property is often used to prune the search space during network inference, as described later.

More on connectedness, recent studies have shown that the frequency distribution of connectivity of nodes in biological (and other types of naturally arising) networks tends to be longer tailed than the normal distribution [36]. The appropriate distribution seems to belong to a class of power-law functions described by $P(k) = k^{-\gamma}$, where k is the degree of a vertex in the network graph, and γ is some network specific constant. Such networks are called *scale-free* and exhibit several important properties. The first is the emergence of *hubs*, or highly connected nodes, which would have been vanishingly unlikely to appear in a network with normally distributed node degrees. These hubs correspond to highly central nodes in the gene network, i.e. genes that do a large amount of the overall regulation. The second property is that through the hubs, the rest of the nodes are connected by, in the worst case, very short paths, yielding overall short longest paths between nodes (degree of separation, small-world networks).

Transcriptional Control

The cis-regions serve as aggregators of the effects of all transcription factors involved in gene regulation. Through protein-specific binding sites the cis-regions recruit and bring in proximity single or groups of TFs having specific regulatory properties, with the sole purpose of inducing precisely when, where, and at what rate a gene is to be expressed. The hard-coded binding sites, or cis-elements, in the regulatory regions of DNA sequences are in fact the code, or logic, by which input signals are chosen and processed. Very good quantitative examples of such logic processing units are, for example, the *endo16* system in Sea Urchin [15], and the process of lysis/lysogeny in bacteriophage-λ [51].

The range of effects (i.e. output) of the cis-processing logic on the input TF signals has been characterized for very few but telling examples. From them we learn that the cis-function is a multi-valued, complex function of the input concentrations even only for two inputs [55]. However the function becomes simpler, and can be decomposed into linear combinations of independent functional signal contributions, when the functional cis-elements are known (at least when performed over the same conditions, and for genes on the periphery of the network (i.e. ones without feedback input) [77, 76]. The traditional roles of individual or groups of TFs as inducers and suppressors (i.e. activators and inhibitors), with respect to a gene, have been based on the change they cause in expression, and are viable if their effects are strong enough and independent of other TF effects. Those roles have been refined recently by associating function to modules of cis-elements (or equivalently TFs) to include more operators, like enhancers, switches, and amplifiers, which form the repertoire of transcriptional control elements. Some of those have been used to engineer synthetic gene circuits with pre-specified functions [25].

Robustness

Real gene networks are very robust to fluctuations in their parameter values [68, 8], and there is strong indication that only specific topology choices can guarantee such robustness [68, 36]. Insensitivity to variations in molecular concentrations is particularly important during organism development, when things are happening in orchestrated cues around the whole organism at the same time, with the goal of producing the same body plan every time, but under a variety of different conditions [15]. It has also been argued that the process of evolution naturally converges to scale-free design in organized structures [52], and that the scale-freeness ensures the robustness of networks to random topology changes [36].

Noise

Noise is an integral part of gene networks, as they are emerging properties of biochemical reactions which are stochastic by nature [42]. Even small variations in the molecular concentrations during the process of translation can be passed along through the network [65]. The networks control the noise through feedback, although in some cases noise enhances some functional characteristics of the networks [67] and hence may have a role in network evolution. In some cases noise measurements can be used to learn the rates of signal propagation in networks too [48].

27.2.3 Utility

A gene network, at any level of modeling, is a blueprint for understanding the functional cooperativity among genes.

Individually, gene networks are succinct representations of the knowledge about the system studied. For example, they can be used for gene classification, based on the localization of the gene's influence on other genes and the others' influence on them. The regulatory interactions between genes can be studied at large-scales, allowing for example for understanding cascades of gene regulations (e.g during organism development). Another use can be in elucidating the connection between gene regulation and phenotype, again on a systemic scale.

In quantitative settings, they can be used to simulate various scenarios, and even predict future behavior of the system. As such they can serve as in-silico hypothesis generation and testing tools with the potential of running many experiments in a very short time on a desktop computer. Knowing the interacting components can help with identifying molecular targets for specific drugs, or drugs for specific targets. Such knowledge coupled with understanding of the network behavior, can lead to designing controlled systems with potential for producing disease-specific cures and personalized health care solutions.

Having gene networks of multiple organisms will allow for their comparison, yielding understanding of structural and functional changes of the networks with time, i.e. their evolution. Through such network comparisons the different effects of stimuli on organisms can be potentially attributed to differences in their networks.

27.3 Gene Expression: Data and Analysis

The amount of mRNA produced during transcription is a measure of how active or functional a gene is. Gene chips, or microarrays, are large-scale gene expression monitoring technologies, used to detect differences in mRNA levels of thousands of genes at a time, thus speeding up dramatically genome-level functional studies. Microarrays are used to identify the differential expression of genes between two experiments, typically test vs. control, and to identify similarly expressed genes over multiple experiments. Microarray data and analysis methods are described in detail in other chapters of this book. Here we only briefly describe the types of expression data and analysis methods used for gene network inference.

Microarray experiments for our purposes can be classified in two groups: time-course experiments, and perturbation experiments. The former are performed with the goal of observing the change of gene expression with time, and are used for understanding time-varying processes in the cell. The latter are performed to observe the effects of a change or treatment on the cell, and are used for understanding the genetic causes for observed differences between cell/tissue types, or responses of pathways to disruptions.

The processing pipeline of microarray data involves pre-processing the raw data to get a gene expression matrix, and then analyzing the matrix for differences and/or similarities of expression. The gene expression matrix, *GEM*, contains pre-processed expression values with genes in the rows, and experiments in the columns. Thus, each column corresponds to an array, or gene-chip, experiment, and it could contain multiple experiments if there were replicates. The experiments can be time points (for time-course experiments), or treatments (for perturbation experiments). Each row in the matrix represents a *gene expression profile*. An example hypothetical table is given in Table 27.1.

TABLE 27.1 Gene Expression Matrix

	Exp_1	Exp_2	$\cdots$	Exp_m
$Gene_1$	0.12	-0.3	$\cdots$	0.01
$Gene_2$	0.50	0.41		
$\vdots$	$\vdots$		$\ddots$	
$Gene_n$	-0.02	-0.07		

Gene-chips can hold probes for tens of thousands of genes, whereas the number of experiments, limited by resources like time and money, is much smaller, at most in the hundreds. Thus, the gene expression matrix is typically very narrow (i.e. number of genes, $n \gg$ number of experiments, m). This is known as the *dimensionality curse* and it is a serious problem in gene network inference. Namely, in any complex system, when trying to elucidate the interactions among the state variables it is best to have many more measurements than states, otherwise the system is largely under-constrained and can have many solutions. The inference methods described in Sections 27.5- 27.8 use different ways to address this problem.

Once normalized the data are analyzed for differential expression using statistical tests. The most robust tests allow for replicate measurements, yield a confidence level for each result, and correct for multiple hypothesis testing (i.e. testing if hundreds of genes are differentially expressed). Additionally, to identify functional relationships genes are often combined in groups based on similarities in their expression profiles using a variety of supervised or unsupervised clustering and/or classification techniques [59]. Most of those methods are good for identifying starting points for network inference.

A good source of microarray data is the Stanford Microarray Database (SMD) [29].

27.4 General Properties of Modeling Formalisms

The actual choice of a modeling formalism for a gene network will depend on the type and amount of data available, prior knowledge about the interactions in the network, nature of the questions one needs answered, area of formal training of the modeler, experimental and computational resources, and possibly other study- or organism-specific factors. Here, based on their general properties different models are classified in several groups, which can be used as selection criteria among them.

Physical vs. Combinatorial Models

The most detailed models of any complex dynamical system, like gene networks, are based on differential equations describing the quantitative relationships between the state variables in the system. Although such physical models can be used to run simulations and predict the future behavior of the system, in general any higher level organization is very difficult to obtain from the equations. But, identifying even simple features (e.g. if one variable is responsible for the behavior of another) may be complicated. In addition, physical models typically have many parameters, necessitating large number of experiments to fit them to the data. Some high-level analyses of dynamical systems, like steady state, have yielded promising results (S-systems by Savageau [53]), but lack inference methods.

On the other hand, combinatorial models start from higher level features of the system by defining the characteristics and features of interest, like the important observables, i.e. gene expression levels, and the nature of relationships, like causality for example. A typical representation for such models is a graph of nodes and edges between them from which many important high-level questions can be readily answered (see Section 27.5). Because of the higher level of modeling, the combinatorial models are most often qualitative, and effective methods for their learning or inference exist even for small number of observations (relative to the number of variables). Most models described in this chapter will be combinatorial, except for the linearized model of differential equations.

Dynamic vs. Static Models

Dynamic gene network models describe the change of gene expression in time. Typically, each node in the network is a function that has some inputs, which it aggregates, and produces output based on them. Most dynamic models are based on simplifications of differential rate equations for each node:

$$\frac{dx_i(t)}{dt} = f_i(x_{i_1}(t), x_{i_2}(t), \ldots) \tag{27.1}$$

where the x_i on the left is the observable concentration of gene expression, for gene i, the $x_{i_1}, x_{i_2}, \ldots$ on the right, are concentrations of molecules that anyhow influence x_i's expression, and $f_i(\cdot)$ is the rate function specifying the exact way in which the inputs influence x_i. Dynamic models tend to be more complete than static ones, in that they aim to characterize the interactions among the input signals, and offer quantitative predictions for the observable variables. However, they do require more input data in general because of the large number of parameters to fit, as well as some type of understanding of the nature of the aggregation function. This usually amounts to making simplifying assumptions, e.g. assuming that the interactions between regulators of a gene are limited to being additive. In addition, time is often discretized and the changes are assumed to be synchronous across all variables. Examples of dynamic models covered in this chapter are Boolean networks and linearized differential equations.

Static models do not have a time-component in them. This practically means that static models yield only topologies, or qualitative networks of interactions between the genes, often specifying the nature of the interactions. Static models can be revealing of the underlying combinatorial interactions among genes, a feature most often simplified away in the dynamic models, with the notable exception of the Boolean network model. Examples of static models are: Graph theoretic models, Bayesian networks, and Linear Additive Models (under some modeling assumptions).

The choice between these two modeling paradigms clearly depends on the type and amount of data and experimental setup available, and it often involves understanding and prioritizing the imperatives in the study, for example, the importance of the exact predictions vs. the nature of interactions.

Synchronous Models

Dynamic models are attractive because they offer quantitative predictions. However, many additional assumptions need to be made for the available inference methods to work. One such assumption is synchronous delivery of all signals to the targets or equivalently, updating the states of all genes at the same time. Large-scale gene expression measurements drive these models because all the observables are measured at the same times.

In synchronous models time is discretized to the intervals between consecutive observations. If these time intervals are small enough, then Eq. 27.1 can be approximated as:

$$\frac{x_i(t_{j+1}) - x_i(t_j)}{t_{j+1} - t_j} \approx f_i(x_{i_1}(t_j), x_{i_2}(t_j), \ldots) \tag{27.2}$$

where t_j and t_{j+1} are two consecutive observation times. Although clearly unrealistic when $t_{j+1} - t_j$ is not small, this approximation is the base for some of the most popular models for gene network inference from expression data, as described in section 27.4.

Finally, reality is obviously asynchronous. However there are no existing effective methods for inferring gene networks solely from expression data for any asynchronous model.

Deterministic vs. Stochastic Models

In deterministic models the expression states of the genes are either given by a formula or belong to a specific class. Measured at two different times or places, while keeping all other parameters the same, a gene's expression would be the same. The precision of the observed expression values, then, depends solely on the experimental setup and technological precision, and can be refined indefinitely with technological advances. The edges stand for relationships which, like the node states, are also deterministic.

Stochastic models, on the other hand, start from the assumption that gene expression values are described by random variables which follow probability distributions. The difference with the deterministic models is fundamental: randomness is modeled to be intrinsic to the observed processes, and thus all things being equal, a gene's expression on two different occasions may be different. Stochastic edges indicate probabilistic dependencies, and their absence may indicate independencies between nodes. They are not easy to interpret in practice.

Stochastic gene network models are especially appropriate for reconstructing gene networks from expression data because of the inherent noise present in them. A fairly general statistical way to accommodate imprecisions and fluctuations in the measurements is to assume that each observed quantity is drawn from an underlying set of values, or a distribution, that the observable variable may take. Then, assessing whether a gene is differentially expressed with respect to another turns into well studied problems of statistical hypothesis testing [59]. In addition to the distribution of a gene's expression one would also like to model the relationships between such distributions as indicators of possible causal relationships between them. One example of a stochastic model of gene networks are the Bayesian Networks, discussed in Section 27.6. Although likely more realistic, stochastic models are more difficult to learn and to interpret.

Expanding Known Networks

Often the goal is to find missing links in a partially known network, or expanding a network around a known core of interacting genes. Although, in general, not necessarily easier to solve than the problem of inferring a network from scratch, expanding a known network is typically a smaller problem and less data might be necessary. In other words, the same amount of data should yield better (i.e. more accurate) or simply more predictions if prior knowledge is available.

27.5 Graph Theoretical Models

Graph theoretical models (*GTMs*) are used mainly to describe the topology, or architecture, of a gene network. These models feature relationships between genes and possibly their nature, but not dynamics: the time component is not modeled at all and simulations cannot be performed. GTMs are particularly useful for knowledge representation as most of the current knowledge about gene networks is presented and stored in databases in a graph format.

GTMs belong to the group of *qualitative network models*, together with the Boolean Network models (Section 27.7), because they do not yield any quantitative predictions of gene expression in the system.

In GTMs gene networks are represented by a *graph* structure, $G(V,E)$, where $V = \{1,2,\ldots,n\}$ represent the gene regulatory elements, e.g. genes, proteins, etc., and $E = \{(i,j)|i,j \in V\}$ the interactions between them, e.g. activation, inhibition, causality, binding specificity, etc. Most often G is a simple graph and the edges represent relationships between pairs of nodes, although *hyperedges*, connecting three or more nodes at once, are sometimes appropriate. Edges can be directed, indicating that one (or more) nodes are precursors to other nodes. They can also be weighted, the weights indicating the strengths of the relationships. Either the nodes, or the edges, or both are sometimes labeled with the function, or nature of the relationship, i.e. activator, activation, inhibitor, inhibition, etc. (see Figure 27.1). The edges imply relationships which can be interpreted as temporal (e.g. causal relationship) or interactional (e.g. cis-trans specificity).

Many biologically pertinent questions about gene regulation and networks have direct counterparts in graph theory and can be answered using well established methods and algorithms on graphs. For example, the tasks of identifying highly interacting genes, resolving cascades of gene activity, comparing gene networks (or pathways) for similarity correspond to, respectively, finding high degree vertices, topological vertex ordering, and graph iso(homo)-morphisms in graphs.

Inferring gene networks under GTMs amounts to identifying the edges and their parameters (co-expression, regulation, causality) from given expression data. The type of data used directs the approaches taken in the inference. Typically, parsimonious arguments stemming from biological principles are used to restrict the resulting networks. Here, several different approaches are mentioned, using both time-series measurements and perturbation measurements of gene expression.

Inferring Co-expression Networks

Co-expression networks are graphs where edges connect highly co-expressed nodes (genes or gene clusters). Two genes are co-expressed if their expression profiles (rows in the expression matrix) are strongly correlated. The data can be any set of experiments on the genes, possibly a mix of time-series measurements and perturbation data. The goal in this approach is to obtain a graph where the nodes are clusters of genes with indistinguishable expression profiles, and the edges between such clusters are indicative of similarities between clusters.

The process of inference closely follows clustering: given are two-thresholds, within cluster, τ_w, and between cluster, τ_b, a gene expression matrix with n genes and m experiments, and a measure for pair-wise scoring of expression profiles (e.g. correlation coefficient),

$Score(i, j)$. These scores between pairs of genes are used to cluster them in the same cluster based on τ_w, using for example hierarchical clustering [35]. The genes in each cluster are considered indistinguishable for all practical purposes. The gene network is formed by placing edges between clusters of very similar average expression as determined by τ_b, while possibly averaging the profiles within clusters and using those as representative profiles.

The problem with this approach is that resulting edges in the network are difficult to interpret. They identify relationships which follow the specifics of the similarity function, but it is hard to justify any similarity function a priori. When the correlation coefficient is used, for example, the justification is that genes with visually similar expression patterns are likely coexpressed, and possibly regulating each other. When compared to known regulatory pairs such approaches have yielded prediction rates of no more than 20% [22].

Regulatory Interactions from Time-Series Data

Here, given time-series gene expression data the goal is to infer edges as potential regulation relationships between genes. Thus a directed edge (i, j) would imply that gene i regulates gene j. The idea is to consider time-course gene expression experiments and correlate sustained positive and negative changes in the expression levels while incorporating biological considerations.

Chen et al. [12] extended the co-expression networks model by considering a scoring function based on signal, or time-series similarity (instead of correlation at the second clustering step above). The scoring function identified putative causal regulation events of the type: a positive change in the expression of gene i followed by a positive change in the expression of gene j implies a potential regulation (or a joint co-regulation). Such *putative* relationships were consequently pruned by minimizing the overall number of regulators, and consistently labeling nodes as activators or inhibitors. Additional biological considerations were that regulators should either be activators or inhibitors, but not both, and that their number should be small, while the number of regulated nodes should be large. The authors showed that a number of theoretical problems of network inferenc on such labeled graphs are NP-complete, even for networks of very low in-degree (even only 2). By using local search heuristics on more general edge-weighted graphs they maximized the overall regulation while minimizing the number of regulators. Their results yielded networks with interesting biological properties. In a subsequent study they improved on their scoring function for detecting meaningful pairwise regulation signals between genes, and showed that they do significantly better than the correlation measure [22]. This pairwise scoring approach, however, is limited to resolving relationships between pairs of genes, whereas in real networks multiple genes can participate in the regulation.

In a similarly motivated, signal-based approach, Akutsu et al. [4] also considered only qualitative clues in time-series expression data to infer regulation. They define a regulatory relationship between genes to be consistent with the data only when all positive (or negative) changes in the expression of one gene consistently correspond to changes in expression of another gene. In general, their method amounts to solving linear inequalities using *linear programming* methods. The linear inequalities were obtained by estimating the rate (or change) of expression for each gene, from time measurements of gene expression, coupled with linearizing a rate equation 27.1 (as in Section 27.8). This method is comparable to the Linear Additive models' inference, and thus probably has the same limitations, see Sect. 27.8 for more on this. It is notable that their analysis is similar to the qualitative type of steady-state analysis of rate equations in the S-system model [53].

Causal Networks from Perturbation Experiments

Perturbing a gene in a gene network effects all genes downstream of it, but no others. For example, perturbing *Gene 1* in the network in Figure 27.1 can influence *Gene 2* and *Gene 4*, but not *Gene 3*. Thus, in principle, performing gene perturbations is a very good methodology for elucidating causal relationships among genes. The inference problem, then, is to find a network consistent with expression data resulting from genetic perturbations.

A perturbation of gene i in the gene network graph $G = (V, E)$ yields a set of differentially expressed genes $D_{iff}(i) \subseteq V$. This set is a subset of $\mathit{Reach}(i)$–the set of all genes reachable from i. The set $\mathit{Reach}(i)$ includes both the nodes adjacent to i, i.e. the set $\mathit{Adj}(i)$, and the nodes reachable but not adjacent to i in G. As examples of $\mathit{Adj}(i)$ and $\mathit{Reach}(i)$, in the network in Figure 27.1, $\mathit{Adj}(1) = \mathit{Adj}(3) = \{2\}$, $\mathit{Adj}(2) = \{4\}$, $\mathit{Adj}(4) = \{2\}$, and, $\mathit{Reach}(1) = \mathit{Reach}(2) = \mathit{Reach}(3) = \mathit{Reach}(4) = \{2, 4\}$. The graph $G_c = (V, E_c)$, where $E_c = \{(i, j), j \in D_{iff}(i)\}$, for all perturbed nodes i, specifies all perturbation effects, and represents a *causal gene network*. Note that in general G_c has more edges than the gene network graph G.

The inference problem can then be re-stated as: given $D_{iff}(i) \subseteq \mathit{Reach}(i)$ for all $1 \leq i \leq n$, retrieve the graph G. There are, in general, many solutions to this problem, since many different gene networks may yield the same differentially expressed genes under all perturbations. Additional biological assumptions on the nature or number of interactions can be used to resolve the best out of all the graphs consistent with the data.

Wagner [69] considered the case when perturbing a gene changes the expression of all the genes downstream from it, i.e. the observed changes yield the full transitive closure of G, or in other words $D_{iff}(i) = \mathit{Reach}(i)$. (The graph G^* with the same nodes as G and G_c and edges $E^* = \{(i, j), j \in \mathit{Reach}(i)\}$, for all i, is called the *transitive closure* of G (and G_c) [14].) Then, motivated by biological parsimony, he used an additional assumption that the biologically most plausible solution has the minimal number of edges of any graph having the same transitive closure. With that assumption, the inference problem above becomes the well-known problem in graph-theory of *transitive reduction*, which in general is NP-complete [1]. However, if there are no cycles in the graph, there is exactly one transitive reduction of G^*, and it can be obtained from the transitive closure very efficiently [1], also [69] (cyclic graphs can be reduced to acyclic by condensing the cycles into single nodes [69]). The required amount of measurements/experiments to resolve a network with this method is n, i.e. each gene must be perturbed and the effects measured (an experiment consists of measuring the expression of all n genes at a time, i.e. whole-genome microarray experiment). Thus, in the case of yeast, the number of experiments would be ~ 6000, which is still practically infeasible. Biologically, the parsimonious argument of the gene network having the minimal number of relationships is not so easy to justify, especially since for acyclic graphs this implies having no shortcuts (essentially no double paths between two nodes) which in reality is not true (e.g. feed-forward loops are essential in gene networks but because they contain shortcuts are excluded in this model [45]).

Limitations and Extensions

GTMs are very useful for knowledge representation but not simulation. The characteristic of these models is that they only resolve the topology of the networks, and are naturally amenable to inference using graph theoretical arguments. Existing inference methods use parsimonious assumptions about the nature of the networks to reduce the solution space and yield a single solution. The required amount of data points varies between models, and it is between $O(logn)$ for clustering based methods and $O(n)$ for perturbation based methods. GTMs are becoming increasingly important as recent studies indicate that the topology of networks is a determining factor in both re-engineering the network as well as understanding network and organism evolution.

27.6 Bayesian Networks

Bayesian networks are a class of *graphical probabilistic models*. They combine two very well developed mathematical areas: probability and graph theory. A Bayesian network consists of an annotated directed acyclic graph $G(X, E)$, where the nodes, $x_i \in X$, are random variables representing genes' expressions and the edges indicate the dependencies between the nodes. The random variables are drawn from conditional probability distributions $P(x_i|Pa(x_i))$, where $Pa(x_i)$ is the set of parents for each node. A Bayesian network implicitly encodes the *Markov Assumption* that given its parents, each variable is independent of its non-descendants. With this assumption each Bayesian network uniquely specifies a decomposition of the joint distribution over all variables down to the conditional distributions of the nodes:

$$P(x_1, x_2, \ldots, x_n) = \prod_{i=1}^{n} P(x_i|Pa(x_i)) \tag{27.3}$$

Besides the set of dependencies (children nodes depend on parent nodes) a Bayesian network implies a set of independencies too (see Figure 27.2). This probabilistic framework is very appealing for modeling causal relationships because one can query the joint probability distribution for the probabilities of events (represented by the nodes) given other events. From the joint distribution one can do inferences, and choose likely causalities. The complexity of such a distribution is exponential in the general case, but it is polynomial if the number of parents is bounded by a constant for all nodes.

Learning Bayesian Networks

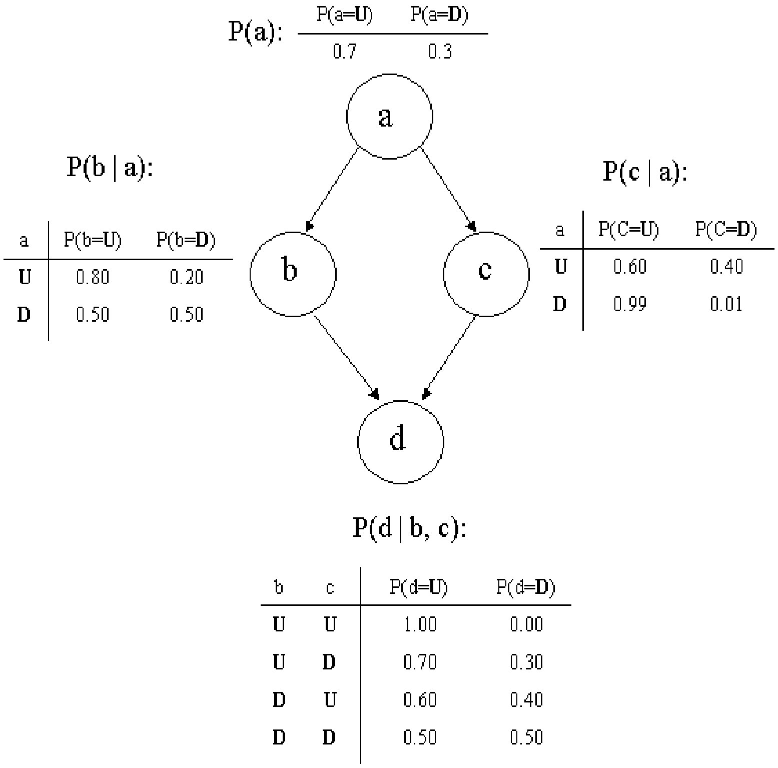
Given measurements of genome-wide gene expression data the goal is to learn candidate Bayesian networks that fit the data well. A survey of general methods for learning Bayesian networks is given in Heckerman et al. [32]. Here we give a very short overview and point the reader to studies in which Bayesian networks were used to analyze expression data.

Two Bayesian networks are defined to be equivalent, or indistinguishable, if they imply the same set of independencies [24]. From any given data most one can hope to learn is equivalence classes of networks, and not individual networks. The process of learning Bayesian networks from the data is essentially two-fold [23]:

- The first part is *model selection*: Given observed data find the best graph (or model) G of relationships between the variables.
- The second is *parameter fitting*: Given a graph G and observed data find the best conditional probabilities for each node.

Parameter fitting is the easier of the two in general. Given a graph model G, good candidates for the best conditional probability distributions can be found by *Maximum Likelihood Estimation* algorithms (when all nodes are known), or *Expectation Maximization* algorithms (when some nodes are hidden), both well known methods. For model selection, on the other hand, only simple heuristics are known, without solid convergence results, which amount to brute-force search among all graphs.

The gene network inference problem combines both of these: the space of all model graphs is searched, and each candidate model is scored. The highest scoring model is the



P(a):

P(a=**U**)	P(a=**D**)
0.7	0.3

P(b | a):

a	P(b=**U**)	P(b=**D**)
U	0.80	0.20
D	0.50	0.50

P(c | a):

a	P(C=**U**)	P(C=**D**)
U	0.60	0.40
D	0.99	0.01

P(d | b, c):

b	c	P(d=**U**)	P(d=**D**)
U	**U**	1.00	0.00
U	**D**	0.70	0.30
D	**U**	0.60	0.40
D	**D**	0.50	0.50

FIGURE 27.2: An Example Bayesian network where the genes can assume discrete states of Up and Down. The probability distributions at the nodes imply the dependencies of nodes only on their parent nodes. The joint probability can be computed directly from the graph, for any instance of the node values. Eg. $P(a = U, b = U, c = D, d = U) = P(a = U)P(b = U|a = U)P(c = D|a = U)P(d = U|b = U, c = D) = 0.7 * 0.8 * 0.4 * 0.7 = 16\%$. The implied independencies in this network are: $I(b; c|a)$ (i.e. b is independent of c given a), and $I(c; b|a)$ (i.e. c is independent of b given a).

best fitting network to the data.

Given a model, its Bayesian score is the posterior probability of the graph G, given the data D:

$$S(G : D) = logP(G|D) = log\frac{P(D|G)P(G)}{P(D)} = logP(D|G) + logP(G) + Const. \qquad (27.4)$$

The non-constant terms on the right hand side correspond to the two problems above. Namely, the first term averages the probability of the model G over all possible parametric assignments to it, and it corresponds to the maximum log-likelihood of G given D. The second term average the probability over all possible models G given the data and corresponds to model complexity. The third is a constant independent of the model. Logs are taken to achieve additive independence of the terms above which can be individually estimated.

Finding the best model amounts to optimizing $S(G : D)$, i.e. finding the model and parametric assignment with the best score. To do this efficiently, it is imperative to chose a scoring function which is decomposable to the individual score contributions of each node and for which there are provable guarantees that the highest scoring models are likelier to capture the real network, given the data, than other models. Examples of existing scoring functions with such properties are: the Bayesian Information Criterion (BIC), Akaike Information Criterion (AIC), and Bayesian Dirichlet equivalent (BDe). Even though because of their decomposition property these functions prune the search space, optimizing them over all models (i.e. graphs) is still NP-hard. Thus, heuristics like hill-climbing and simulated annealing are used to efficiently search the space of models by looking at neighboring graphs around a given graph, by adding and deleting edges, or reversing directions of edges.

In addition to the scoring function, other important choices have to be made to get from the observed data of gene expressions to the learned network. These include data discretization into few levels (e.g. -1,0,+1 for under-expression, no expression and over-expression) and choices for priors (e.g. linear, multinomial), and are often related.

In [30] different scoring schemes are illustrated and in [75] various scoring functions, discretizations, and heuristics are compared. The results, although from limited data studies, single out BIC as the score of choice, in conjunction with a three level discretization of the data, and a random-start hill-climbing heuristic for model selection.

The final issue is interpretation. Bayesian networks model the gene expression of each gene as a random variable having a distribution conditional on its parent genes. From that, one wants to infer causal relationships between genes. For that, another assumption, the *local Markov Rule* is needed, which says that variables are independent of all other nodes given their parents. With this assumption, directed edges in the learned equivalence class stand for causal relationships, with the direction indicated by the arrow. If in a class there are undirected edges, then the causality is unknown.

Practical Approaches and the Dimensionality Curse

In practice the available data suffers from the dimensionality curse (see Sec. 27.3) and many different Bayesian networks may fit the data just as well. To lower the number of high-scoring networks, simplifying assumptions regarding the topology of the graph or the nature of the interactions have been used. In addition, instead of trying to learn large-scale networks the focus has been on retrieving features consistently over-represented in high-scoring networks.

In [24] the authors use Bayesian networks to establish regulatory relationships between genes in yeast, based on time-series data of gene expression. To do that, they used several biologically plausible simplifying assumptions. The first one is that the nodes are of bounded in-degree, i.e. the networks are sparse. The second is that the parent and children nodes likely have similar expression patterns, and thus coexpressed pairs of genes might be in regulatory relationships. Both are realistic to a degree, and are meant to reduce the number of potential high-scoring networks. It was found that even with these assumptions the number of high-scoring networks was still high, and instead of full networks subgraphs, or robust features, in common to many high-scoring networks, were considered, like pair-wise relationships for example. Two separate prior distributions were used for the nature of the interactions of the edges at each node (i.e. the combined distributions): a linear Gaussian and a multinomial, requiring different data discretizations each, three levels for the first one and continuous for the second. The results, not surprising, were excellent for the linear prior and mediocre for the multinomial.

A subsequent study by the same group [49], applied Bayesian networks to perturbation gene expression data, with a similar but expanded goal: to identify regulatory relationships and in addition, to predict their nature of activation or inhibition. To do that, they incorporated perturbations into the Bayesian framework. Activations and inhibitions were modeled by requiring that for each change in a regulator's expression the regulated gene must exhibit a change too, much like Akutsu et al. [4] did in their qualitative network model study. The results were, again, conserved features over many runs of the algorithms, and the number of features considered was larger than in the previous study. The resolution of co-regulation was much better than that of correlation methods and yielded subnetworks of known networks automatically from the data.

Although no analysis of the required amount of data was given in either of these studies, the available data is very insufficient. In the first paper, this problem was circumvented by clustering of the data and considering clusters of genes, thus lowering the dimensionality of the original problem. In the second, the results were sub-networks, needing much less data than a genome-scale gene network, specifically those for which there was a lot of support in the expression data.

Extending Bayesian Networks

Bayesian networks offer an intuitive, probabilistic framework in which to model and reason about qualitative properties of gene networks. The gene expression data available is insufficient for full gene network learning, but conserved features over high-scoring models are biologically significant. The Bayesian Network formalism is easily extendable to describe additional properties of the gene networks, like the activatorin the second study above. This implies that other types of data, like regulator candidates, promoter sequence alignment, or TF-DNA binding data, can be used to refine the models, as recent studies have done [54, 7].

The problem with learning of Bayesian networks is combinatorial: if the graph model is not known then the space of all graph models has to be explored. But this space is super-exponential even for directed acyclic graphs and exploring it completely is impossible even with the fastest heuristics. Exploring hierarchical properties of DAGs can help in pruning the search space down to exponential size(from super-exponential), for which, with current technology, exact solutions can be obtained for small networks of $n < 30$ nodes [47]. A promising direction is to restrict the search space further by including additional biological knowledge as it becomes available.

In their basic form, Bayesian networks are qualitative models of acyclic gene networks. There have been efforts to extend them to both include cyclic networks and to model network dynamics by duplicating the nodes in the network. Namely, a state transition of a gene network from time t to time $t+1$ can be modeled by having two copies of the same Bayesian network, with additional edges between the two. Similar *unrolling* of the network can be used to model cycles. Learning such duplicated networks would require more data in general.

27.7 Boolean Networks

Boolean networks are a dynamic model of synchronous interactions between nodes in a network. They are the simplest network models that exhibit some of the biological and systemic properties of real gene networks. Because of the simplicity they are relatively easier to interpret biologically.

Boolean networks as a biological network modeling paradigm were first used by Kaufmann in the 1970s [39], where he considered directed graphs of n nodes and connectivity, or degree, per node of at most k, called *NK-Boolean networks*. He studied their organization and dynamics and among other things showed that highly-connected NK-networks behave differently than lowly connected ones. In particular, NK-networks of low per-node degree seem to exhibit several of the properties that real life biological systems exhibit, like periodic behavior and robustness to perturbation.

Model and Properties

For completeness, the following summary of Boolean logic is provided. A variable x that can assume only two states or values is called *Boolean.* The values are denoted usually as 0 and 1, and correspond to the logical values **true** and **false**. The logic operators **and**, **or**, and **not** are defined to correspond to the intuitive notion of truthfulness and composition of those operators. Thus, for example $x_1 \textbf{and} x_2 = \textbf{true}$ if and only if both x_1 and x_2 are **true**. A *Boolean function* is a function of Boolean variables connected by logic operators. For example, $f(x_1, x_2, x_3) = x_1$ **or** (**not** $(x_2$ **and** $x_3))$ is a Boolean function of three variables.

A *Boolean network* is a directed graph $G(X, E)$, where the nodes, $x_i \in X$, are Boolean variables. To each node, x_i, is associated a Boolean function, $b_i(x_{i_1}, x_{i_2}, \ldots, x_{i_l}), l \leq n, x_{i_j} \in X$, where the arguments are all and only the parent nodes of x_i in G. Together, at any given time, the states (values) of all nodes represent the *state* of the network, given by the vector $S(t) = (x_1(t), x_2(t), \ldots, x_n(t))$.

For gene networks the node variables correspond to levels of gene expression, discretized to either up or down. The Boolean functions at the nodes model the aggregated regulation effect of all their parent nodes.

The states of all nodes are updated at the same time (i.e. synchronously) according to their respective Boolean functions:

$$x_i(t+1) = b_i(x_{i_1}(t), x_{i_2}(t), ..., x_{i_l}(t)). \tag{27.5}$$

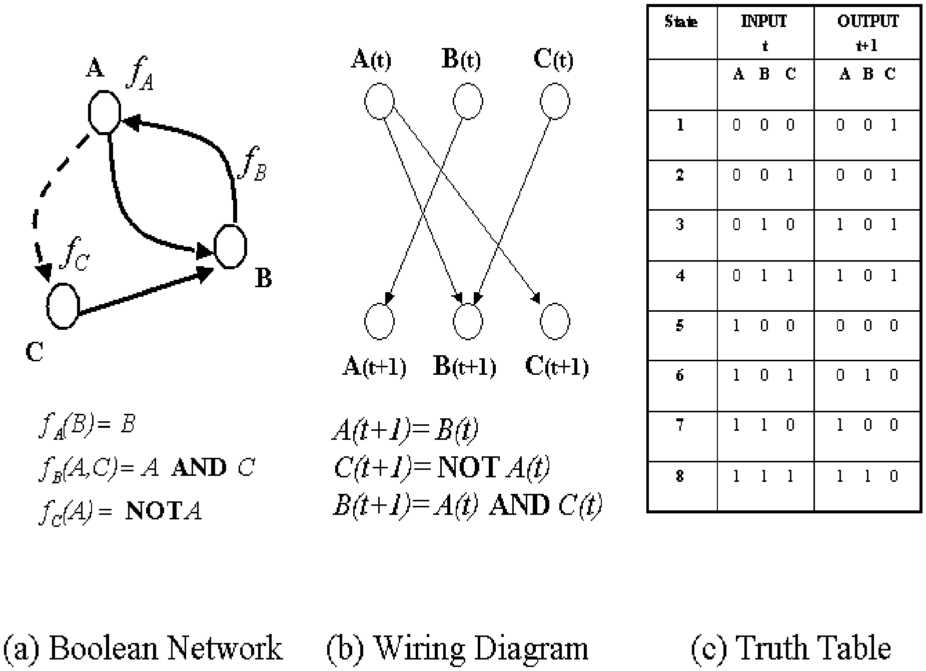
All states' transitions together correspond to a *state transition* of the network from $S(t)$ to the new network state, $S(t+1)$. A series of state transitions is called a trajectory, e.g. S_5, S_1, S_2 is a trajectory of length 3 in the network in Figure 27.3. Since there is a finite number of network states, all trajectories are periodic. The repeating part of the trajectories are called *attractors*, and can be one or more states long, e.g. S_2 is an attractor in the network in Figure 27.3. All the states leading to the same attractor are the *basin of attraction* [73].

The dynamic properties of Boolean networks make them attractive models of gene networks. Namely, they exhibit complex behavior, and are characterized with stable and reproducible attractor states, resembling many biological situations, like steady expression states. In addition, the range of behaviors of the system is completely known and analyzable (for smaller networks) and is much smaller than that of other dynamic models. In terms of topology, it has been shown that high connectivity yields chaotic behavior, whereas low connectivity leads to stable attractors, which again corresponds well to real biological networks [39].

Reverse Engineering

The goal in reverse engineering Boolean networks is to infer both the underlying topology (i.e. the edges in the graph) and the Boolean functions at the nodes from observed gene expression data.

The actual observed data can come from either time-course or perturbation gene expression experiments. With time-course data, measurements of the gene expressions at two consecutive time points simply correspond to two consecutive states of the network, $S(i)$ and $S(i+1)$. Perturbation data comes in pairs, which can be thought as the input/output



State	INPUT t	OUTPUT t+1
	A B C	A B C
1	0 0 0	0 0 1
2	0 0 1	0 0 1
3	0 1 0	1 0 1
4	0 1 1	1 0 1
5	1 0 0	0 0 0
6	1 0 1	0 1 0
7	1 1 0	1 0 0
8	1 1 1	1 1 0

FIGURE 27.3: An example Boolean network and three possible ways to represent it. The one on the left is a gene network modeled as a Boolean network, in the middle is a wiring diagram obviating the transitions between network states, and on the right is a truth table of all possible state transitions.

states of the network, I_i/O_i where the input state is the one before the perturbation and the output the one after it. In both cases, a natural experimental unit is a pair of consecutive observations of genome-wide gene expression, i.e. states of the network, or a *state transition pair*. The total amount of observed data, m, is the number of available state transition pairs. Perturbation input/output pairs can be assumed to be independent whereas time-course pairs are rarely so. The actual $0,1$ values are obtained by discretizing the observed continuous expression values, and has to be done with necessary care.

Given the observations of the states of a Boolean network, in general many networks will be found that are consistent with that data, and hence the solution network is ambiguous. There are several variants of the reverse engineering problem: (a) finding a network consistent with the data, (b) finding all networks consistent with the data, and (c) finding the "best" network consistent with the data (as per some pre-specified criteria). The first one is the simplest one and efficient algorithms exist for it, although the resulting network may be very different from the real one. Hence the second problem, which yields all possible solutions, including the real one, although the number of solutions may be very large. The third variant solves the problem of having too many correct solutions by introducing additional assumptions or modeling imperatives. Although very attractive computationally, sometimes the additional criteria may be oversimplifications.

Data Requirement

The reverse engineering problems are intimately connected to the amount of empirical data available. It is clear that, in general, by having more data points the inferred network will be less ambiguous. The amount of data needed to completely determine a unique network is known as the *data requirement problem* in network inference. The amount of data required depends on the sparseness of the underlying topology and the type of Boolean functions allowed. In the worst case, the deterministic inference algorithms need on the order of $m = 2^n$ transition pairs of data to infer a densely connected Boolean Network with general Boolean functions at the nodes [2]. They also take exponential time as the inference problem in this case is known to be NP-complete [4].

If, however, the in-degree of each node is at most a constant k then the data amount required to completely disambiguate a network drops significantly. The lower bound, or the minimum number of transition pairs needed (in the worst case) can be shown to be $\Omega(2^k + klogn)$ [19, 3]. For the expected number of experiments needed to completely disambiguate a network both randomized and deterministic theoretical results are known, although the latter only for a special case. Akutsu and collaborators have shown [3] that the expected number of pairs is proportional to $2^{2k}klogn$, with high probability, if the data is chosen uniformly at random. For a limited class of Boolean functions, called *linearly separable* and related to the linear models of Section 27.8, Hertz has shown [33] that $klog(n/k)$ pairs would suffice to identify all parameters. Empirical results show that the expected number of experiments needed is $O(logn)$, with the constant before the log being in the order of $2^k k$ [19, 3], although these studies assumed that the data consists of statistically independent state transition pairs, which is not generally true.

Inference Algorithms

The simplest exhaustive algorithm for inferring Boolean networks consistent with the data is to try out all Boolean functions $b_i(\cdot)$ of k variables (inputs) on all $\binom{n}{k}$ combinations of k out of n genes [3]. For a bounded in-degree network this algorithm works in polynomial time. If all possible assignments for the input state of the network are given (2^{2k} different values), then this algorithm uniquely determines a network consistent with the data. This is of course an extraordinary amount of data which is not available in practice. In [3] the authors show empirically on synthetic data that the expected number of input/output pairs is much smaller, and proportional to $logn$.

Algorithms that exploit similarity patterns in the data fair much better, on average. Liang et al. [41] used an information theoretic method to identify putative coregulation between genes by scoring the mutual information between gene profiles from the expression matrix. Starting from one regulator per node, their algorithm adds in-edges progressively, trying all the time to minimize the number of edges needed (i.e. variables in the Boolean function at that node) to explain the observed transition pairs. Although in the worst case the data requirement is exponential, their algorithm does much better in practice, demonstrating that $O(logn)$ experiments usually suffice. A theoretical analysis of the expected number of experiments required to disambiguate a gene network has not been given. Their algorithm worked well in practice for very small k.

The combinatorial approach of Ideker et al [34] also exploits co-expression among genes using steady-state data from gene perturbation experiments. Their algorithm identifies a putative set of regulators for each gene by identifying the differentially expressed genes between all pairs of network states (S_i, S_j), including the wildtype (or baseline) state, S_0. The network states, S_i, correspond to steady-states of gene expression of all the genes in the network following a single gene perturbation. To derive the network topology the authors utilize a parsimony argument, whereby the set of regulators in the network was postulated to be equal to the smallest set of nodes needed to explain the differentially expressed genes between the pairs of network states. The problem thus becomes the classical combinatorial optimization problem of *minimum set covering*, which is NP-complete in general. They solved small instances of it using standard branch and bound techniques. The solutions were graphs, or gene network topologies. To complete the network inference, i.e. to identify the Boolean functions at the nodes, they built truth tables from the input data and the inferred regulators for each node. This procedure does not yield a unique network in general. The authors proposed an information-theoretic approach for predicting the next experiment to perform which best disambiguated the inferred network, based on an information theoretic score of information content. Their results confirmed that the number of experiments needed to fully infer a Boolean network is proportional to $logn$, with double perturbation experiments having better resolving power on average than single perturbation ones.

Limitations and Extensions

Boolean networks make good models for biologically realistic systems because their dynamics resembles biological systems behavior and they are also simple enough to understand and analyze. However, these models are ultimately limited by their definition: they are Boolean and synchronous. In reality, of course, the levels of gene expression do not have only two states but can assume virtually continuous values. Thus discretization of the original data becomes a critical step in the inference, and often reducing the values to two states may not suffice. In addition, the updates of the network states in this model are synchronous, whereas biological networks are typically asynchronous. Finally, despite their simplicity, computationally only small nets can be reverse engineered with current state-of-the-art algorithms.

Boolean network models have been extended in various ways to make them more biologically realistic and computationally more tractable. With the availability of better data and models describing the cis-regulatory control and signal propagation through networks, a number of theoretical models, including *chain functions* [27, 28] and certain *Post classes* of Boolean functions [57], have been proposed to restrict the Boolean functions at the network nodes. In addition to offering more realistic network modeling, these approaches have the computational benefit of significantly pruning the solution space for inference.

Additionally, there have been approaches to introduce stochasticity to these models, through probabilistic Boolean networks [56], related to dynamic Bayesian networks, which further increase their realism.

27.8 Differential Equations Models and Linearization

Differential equations (DE) are the starting point for quantitative modeling of complex systems. DEs are continuous and deterministic modeling formalisms, capable of describing non-linear and emerging phenomena of complex dynamical systems.

DE models of gene networks are based on rate equations, quantifying the rate of change of gene expression as a function of the expressions of other genes (and possibly other quantities). The general form of the equations, one for each of n genes, is:

$$\frac{dx_i}{dt} = f_i(x_{i_1}, x_{i_2}, \ldots, x_{i_l}) \qquad (27.6)$$

where each x_j is a continuous function, representing the gene expression of gene j.

Each $f_i(\cdot)$ quantifies the combined effect of its arguments, or regulators, on x_i, and it subsumes all the biochemical effects of molecular interactions and degradation. $\{x_{i_1}, x_{i_2}, \ldots, x_{i_l}\}$, the set of arguments of $f_i(\cdot)$, is a subset of all gene expression functions, $\{x_1, x_2, \ldots, x_n\}$. In the gene network, $f_i(\cdot)$ can be thought of as the function at node i which processes the inputs, $x_{i_1}, x_{i_2}, \ldots, x_{i_l}$ and produces an output rate for gene i.

In addition to the variables, the $f_i(\cdot)$ will include many free parameters whose values must be determined from observed data. Given the $f_i(\cdot)$'s and all their parameters, the dynamics of the network can be approximated even if analytical solutions are unknown, by using various numerical differential equation solvers, or even more specifically, gene network simulator software. Thus, the functions $f_i(\cdot)$, with all parameters fitted, specify the gene network fully.

The specific forms of the node functions $f_i(\cdot)$ come out of biochemical considerations and modeling of the components of the genetic system. In general, these functions are non-linear because, among other things, in reality all concentrations get saturated at some point in time. These functions are usually approximated by sigmoid functions [17]. One such class of functions are the *squashing functions* [70], where $f_i(x(t)) = 1/(1 + e^{-(\alpha_j x(t) + \beta_j)})$. The constants α_j and β_j are gene specific and determine the rapidity of the gene response to the regulation. More complicated, non-linear functions, are also used; for example, Savageau's S-systems, which have some nice representational properties [53].

Identifying a gene network from observed data under this model means estimating (or fitting) the parameters in the functions $f_i(\cdot)$. In general the number of arguments, the functions $f_i(\cdot)$, and their parameters are not known. Given observed gene expression data, the first step to identifying the gene network is to guess or approximate the $f_i(\cdot)$'s. Since the identification process depends solely on the form of the $f_i(\cdot)$'s, the functions are typically linearized, as described below.

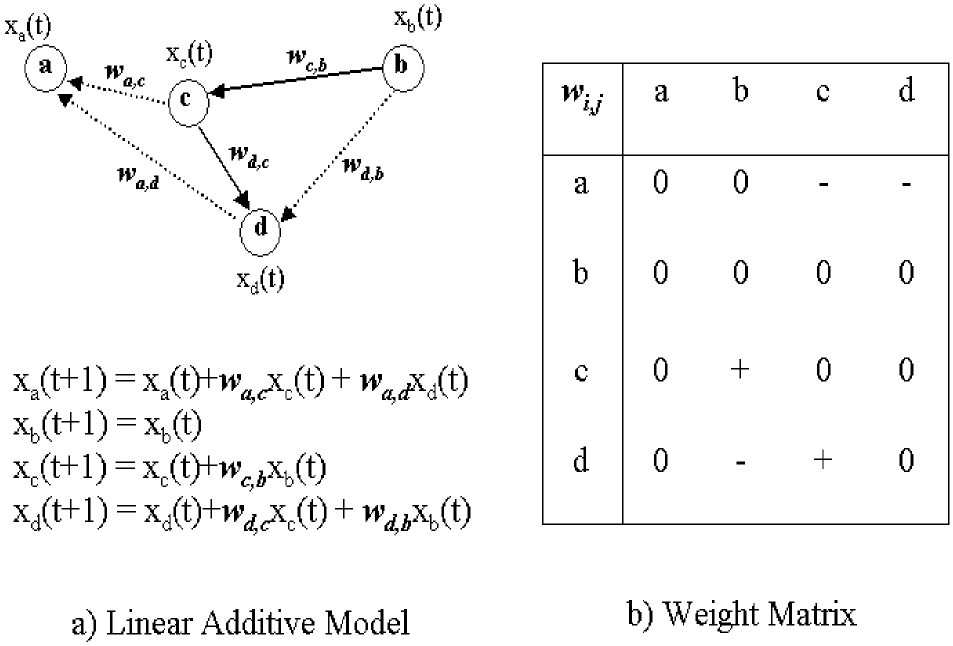
The question is how many observations are needed to identify the parameters in the differential equation system? For general differential equations, in a recent work Sontag showed that if the system has r parameters, then $2r + 1$ experiments suffice to identify them [58]. Although it is not immediately obvious how many parameters there are in a gene network, if we assume at most a constant number of parameters per argument in the rate functions, the total number of parameters in a dense network of n nodes is $O(n^2)$, and for a sparse one $O(n)$. A large scale gene expression monitoring technology allows for n observations at a time, which may further lower the number of sufficient experiments.

Linearized Additive Models (LAM)

The simplest interesting form that the $f_i(\cdot)$'s can take are linear additive functions, for which Eq. 27.6 becomes:

$$\frac{dx_i(t)}{dt} = ext_i(t) + w_{i1}x_1(t) + \ldots + w_{in}x_n(t) \qquad (27.7)$$

(with possibly some additional linear terms on the right hand side, indicating the degradation rate of gene i's mRNA or environmental effects, which can all be incorporated in the w_{ij} parameters, assuming their influence on x_i is linear [74]). The term $ext_i(t)$ indicates a (possible) controlled external influence on gene i, like a perturbation for example, and is directly observable.



$$x_a(t+1) = x_a(t) + w_{a,c}x_c(t) + w_{a,d}x_d(t)$$
$$x_b(t+1) = x_b(t)$$
$$x_c(t+1) = x_c(t) + w_{c,b}x_b(t)$$
$$x_d(t+1) = x_d(t) + w_{d,c}x_c(t) + w_{d,b}x_b(t)$$

a) Linear Additive Model

$w_{i,j}$	a	b	c	d
a	0	0	-	-
b	0	0	0	0
c	0	+	0	0
d	0	-	+	0

b) Weight Matrix

FIGURE 27.4: Example of a Linear Additive Model for a four node Gene network. The dashed lines are inhibiting relationships, and the full lines are inducing relationships. The weight matrix specifies the existence of relationships between genes and their nature and strength.

The justification for the linear additive model is three-fold. First of all, the w_{ij}'s intuitively relate to the regulatory effect of one gene, j, on another, i, and correspond to the strength of this effect. In the graph representation of the gene network, with genes at the nodes, and edges for each $w_{ij} \neq 0$, the parents of a node are its regulators. The biological interpretation for the weights is that if $w_{ij} > 0$ then gene j induces expression of gene i, and if $w_{ij} < 0$ then gene j represses transcription of gene i. For convenience, we assume $w_{ij} = 0$ if there is no edge from j to i. The second justification for the linearization is that in the immediate neighborhood of any n-dimensional point $(x_1, x_2, \ldots, x_n)$ the surface $f_i(x_1, x_2, \ldots, x_n)$ can be linearized to a plane tangent to it at that point, as in Eq. 27.7, where the coefficients are given by $w_{ij} = \partial f_i / \partial x_j$ [60]. Finally, when the system is observed in a steady-state, or equilibrium, where $dx_i/dt = 0$, and is brought to that steady state slowly, the linear approximation holds.

The reason for linearizing the original system is to turn it into a linear differential equation system, in which the parameters, w_{ij} can easily be fitted to the data using linear algebra methods. From each microarray experiment, be it a time-course, steady-state, or perturbation, one linear equation can be set up for each gene i. Then m such genome-wide microarray experiments would yield nm linear equations. All of those can be written succinctly in a matrix notation as follows.

For experiment l, let $\mathbf{x}^{(l)}$ and $\mathbf{ext}^{(l)}$ be column vectors consisting of the gene expression functions $x_1, \dots, x_n$, and the external influences on the individual genes, $ext_1, \dots, ext_n$ respectively. Then, let $\mathbf{X}_{n\times m} = (\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(m)})$ and $\mathbf{Ext}_{n\times m} = (\mathbf{ext}^{(1)}, \dots, \mathbf{ext}^{(m)})$, be the matrices consisting of the above vectors for all m experiments. Also, let $\mathbf{W}_{n\times n}$ be the matrix of all weights $w_{ij}, 1 \le i, j \le n$. Then, the system of equations becomes

$$\frac{d}{dt}\mathbf{X}_{n\times m} = \mathbf{W}_{n\times n}\mathbf{X}_{n\times m} + \mathbf{Ext}_{n\times m} \tag{27.8}$$

Here, $\mathbf{X}_{n\times m}$ is the gene expression matrix, consisting of observations of expression for each gene in each experiment, and $\mathbf{Ext}_{n\times m}$ is a matrix of observed external influences. The rate terms on the left hand side of Eq. 27.8, are either known, i.e. observed rates in the experiments, are equal to zero (in the steady state), or are approximated from the available data as $\Delta x_i/\Delta t$, where ideally Δt is very small (in practice that is not the case, and the approximations may be bad).

When dx_i/dt is approximated by $\Delta x_i/\Delta t = (x_i(t+1) - x_i(t))/\Delta t$, the behavior of the network is effectively discretized in time. Then, the next state of a gene's expression can be expressed as a function of the previous states of its regulators, i.e. $x_i(t+1) = \Delta t(x_i(t) + \sum_{j=1}^{n} w_{ij}x_j(t))$. Such models are also known as linear additive, or weight matrix models, and are particularly suited for modeling time-course gene expression experiments.

Gene network identification under LAM

Under the linearized differential equation model network identification amounts to finding a matrix $\mathbf{W}_{n\times n}$ which is consistent with Eq. 27.8, i.e. it is the best solution to that system.

In total, there are n^2 unknowns and nm equations in this system. The existence and uniqueness of solutions to the linear equation system depends on two parameters: m, the number of whole-genome microarray experiments, and $r(\le nm)$ the number of linearly independent equations in the system. Namely, if $r = n^2$ then the system has a unique solution. If $r > n^2$ then the system is over-constrained and there is no solution. However, in practice more than the sufficient number of observations is preferred for better results. A solution for an over-constrained system can be found by performing multiple regression of each gene on the other genes. If $r < n^2$ the system is under-constrained and there are infinitely many solutions, all of which can be expressed in terms of a subset of independent solutions. To address this dimensionality curse, many methods have been used to bridge the gap between r and n^2, resulting in both under-constrained and over-constrained systems.

In practice, the gene expression matrix is very long and narrow, that is, there are typically many more genes than experiments, so $m \ll n$ and thus $r \ll n^2$. Choosing a solution from the infinitely many plausible ones is a non-trivial task. Often a solution with special properties, or a pseudo-inverse is chosen. Such are for example the Moore-Penrose pseudo-inverse, and the Singular Value Decomposition (SVD), and although they rarely give the correct solution [74] they can be used as starting points for generating better solutions.

Additional biological assumptions are needed to prune the search space, of which the network-sparseness assumption has been the most popular one: each gene cannot be regulated by more than a fixed number k of other genes [70, 13, 74]. Chen et al. in [13] translated the problem of finding a solution $\mathbf{W}$ of Eq. 27.8 when the number of nonzero weights w_{ij} for any given i is at most a fixed constant k, into a combinatorial problem called Minimum Weight Solutions to Linear Equations, and showed that it is polynomially solvable in general, although they offered a computationally expensive algorithm.

Collins and collaborators in a series of papers used a system similar to Eq. 27.8 to model gene networks under the sparseness assumption, and offered several methods to identify networks from both time-course and perturbation data under such models. In [74], they used Singular Value Decomposition on time-course experiments to generate an initial solution and then refined it by using an optimization technique called *robust regression.* The solutions were much better than those from using SVD alone. Experimentally, the authors used very fine sampling times in this study which allowed them to approximate the dx_i/dt's very well. This method is particularly well suited for identification of larger networks. In [64] they use the same linear model, but with advanced gene perturbation (over-expression) technology (described in [25]), and measurements at steady-states. The goal there was different from above: based on previous perturbations they wanted to choose the next perturbation to best identify the still unknown parameters in the network. To that end, they used an algorithm which as the next best perturbation chooses genes which have either changed the least in the past perturbations or have most uncertain connections to other genes. However, as a consequence of the steady-state modeling their algorithm is unable to predict w_{ii} for any gene (which is the case in similar methods too, e.g. [18]). In subsequent work Collins and colleagues showed that the linear additive model with steady-state data of targeted perturbations allows for effective identification of both small [26] and large [21] gene networks.

As emphasized above, if only the observed data is used the system of linear equations is mostly under-constrained. However, under some assumptions, the gene expression matrix can be processed to yield an over-constrained system. To identify a small gene network of CNS development regulation in rat, D'Haeseleer and colleagues [20] used cubic interpolation between successive time points of gene expression observations, thereby increasing the total amount of data. They generated just as many interpolated points as needed for the linear system of equations to become over-constrained, which they consequently solved using a least squares optimization. The interpolation approach does well when the phenomena studied have been sampled finely enough to capture the essential changes in gene expression, which cannot be guaranteed in general. In a complementary approach, Someren et al. [66] used clustering of the time-course expression matrix to reduce the dimensionality of $\mathbf{W}$. Hierarchical (progressive) clustering was performed until the resulting linear system had the smallest error in explaining the whole data, i.e. was close to being over-constrained. Their approach drew a lot from the success of clustering in identifying coregulated clusters of genes through coexpression, but it has its limitations too: the resulting gene network is a network of gene clusters and not genes, and the interpretation is non-trivial.

Data Requirement

If the weight matrix is dense (i.e. the average connectivity per node in the network is $O(n)$, then $n+1$ arrays of all n genes are needed to solve the linear system, assuming the experiments are independent (which is not exactly true with time-series data). If instead the average connectivity per node is a fixed constant, k, as is the case in realistic networks, than it can be shown that the number of experiments needed is $O(klogn)$ [19, 74, 64].

Limitations and Extensions

Linear models yield good, realistic looking predictions when the expression measurements have been performed around a steady state or on a slow changing system. Otherwise, the rates of change of the genes' expressions cannot be estimated well. As with the other methods, the available data is insufficient for large-scale network inference, although the inference methods for this model scale well with the amount of data and should work for larger networks too.

A possible way to extend these models would be to make them hold not just in the vicinity of a steady-state point, but further away too. To do that second order relationships, i.e. $x_i x_j$ product terms, could be considered, although the data requirement may become prohibitive in that case.

27.9 Summary and Comparison of Models

The overall properties of the four models and inference methods are summarized in Table 27.2.

There have been several attempts to compare different network inference approaches based on how well they retrieve a known network [71, 72]. Unfortunately those approaches have compared different modeling strategies without any systematic method behind it. For example Boolean network reverse engineering methods were compared to Bayesian networks etc. The results are therefore not reliable, and by no means definite.

It is more reasonable to compare the efficacy of different network inference methods within the same modeling category. To that end well known real data sets should be used. In addition, results should be reported on much larger sets of artificial data satisfying prespecified statistical and biological properties, like precise knowledge of what is signal and what noise [44]. Finally, the results of any network inference from public data should be made fully available to the public for future comparison and reference. The trend so far has been of reporting predicted sub-parts of the overall networks, those that had some immediate positive reporting value.

TABLE 27.2 Summary of Model Properties and Inference Methods

	Graph	Bayes	Boolean	LAM
Nodes	Genes	Random Variables	Discretized Expression	Continuous Expression
Edges	Causality	Conditional Dependency	Arguments in Boolean Functions	Arguments in Regulatory Functions
Additional Parameters	Activation, Inhibition	Activation, Inhibition	Boolean Function	Weight Matrix
Properties	Static topology	Static topology, Stochastic	Dynamic, Biologically Realistic	Dynamic, Steady-state realistic
Inference	Biological Parsimony, Optimization	Bayes Scoring, ML, Optimization	Optimization, Information Theory	Linear Regression, Optimization
Data Requirement	$O(logn) - O(n)$	?	$O(2^k klogn)$	$O(knlogn)$

27.10 Future Directions and Conclusions

In this chapter were presented various models, modeling methodologies, and inference methods for reverse engineering gene networks from large-scale gene expression data. In addition to the computational methods, gene network inference involves considerations of type and quantity of data, prior biological knowledge, modeling framework, and experimental technologies. Ultimately, one has to first identify the modeling imperatives (e.g. obtaining the topology only may be enough for some applications but not for others), then compile all available knowledge about the network of interest, decide on what type of data/experiments to use, and only at the end choose the inference method.

The various inference problems presented in this chapter could be used as starting points and ideas to build on; many of them are rather naive although their solutions involve intricate theoretical arguments.

The main challenges in network reconstruction are data related. First of all, typically the currently available data is not sufficient for large-scale network reconstruction (dimensionality curse), and the formal statements of the models are very over-constrained. Second, the question is how much is in the data, i.e. do the experiments capture relevant events or not. For example, in time-series data if the sampling is too coarse the important biological events may happen in-between sampling points. Both of these data related issues will likely be resolved with technological advancements.

Even with ideal gene expression data the inference problems may be difficult to solve because of the large space of solutions. In general, including biological constraints, like the bounded indegree of a node, or restricted cis-functions, narrow down the huge space of possibilities to potentially a manageable one. Such a *model-based* modeling should become the standard as more and more is learned about gene networks.

It is very important to extend the gene network modeling to other levels of physical interactions as in Figure 27.1, especially since some gene–gene interactions cannot be recovered at all from the gene expression data. To do that other types of experiments are needed besides gene expression measurements. Utilizing different types of data together, or data integration, is becoming a hot area of research in functional genomics. The problem is how to put together information from heterogeneous empirical data.

The available large-scale genomic data includes DNA sequences (as most reliable), gene expression data, TF-DNA interaction data (Chromatin Immuno-Precipitation on a chip), protein expressions, and protein-protein interactions (e.g. yeast two-hybrid). Combining these diverse data could potentially reveal gene network interactions beyond what each can individually. For example, promoter DNA regions have been used together with expression data to identify cis-elements in co-regulated genes, starting from either the expression [63], the DNA promoter regions [11], or both [50, 40]. Gene coding DNA together with expression data have been very successfully combined to verify that conserved coding regions across organisms have a conserved function too [61]. TF-DNA interaction data together with expression data has been used to identify modules of functionally related genes [31]. Protein-protein interaction data has been used to refine gene networks estimated from expression data, in a Bayesian setting [46]. Many different types of data were used to identify most of the functional relationships between the genes in yeast [62]. Thus, having more and varied types of data would allow for better, more meaningful predictions of gene networks.

Conclusions

Gene network modeling and inference is a very young research area, mostly because current gene expression data is inadequate for most types of definitive analyses. In addition, most inference methods apply to the simplest models, Boolean networks, and even in those biological over-approximations the algorithms quickly become computationally intractable. In the future, the confluence of better data, through data integration, and powerful inference methods should bring about networks of predictive and comparative value yielding reliable and testable models of systemic gene regulation.

References

[1] A. Aho, M.R. Garey, and J.D. Ullman. The transitive reduction of a directed graph. *SIAM J. Comput.*, 1:131–7, 1972.

[2] T. Akutsu, S. Kuhara, O. Maruyama, and S. Miyano. Identification of gene regulatory networks by strategic gene disruptions and gene overexpressions under a boolean model. *Theor. Comput. Sci.*, 298:235–51, 2003.

[3] T. Akutsu, S. Miyano, and S. Kuhara. Identification of genetic networks from a small number of gene expression patterns under the boolean network model. In *Pac. Symp. Biocomputing*, volume 4, pages 17–28, 1999.

[4] T. Akutsu, S. Miyano, and S. Kuhara. Algorithms for inferring qualitative models of biological networks. In *Pac. Symp. Biocomputing*, volume 5, pages 290–301, 2000.

[5] R. Albert and H.G. Othmer. The topology of the regulatory interactions predicts the expression pattern of the drosophila segment polarity genes. *J. Theor. Biol.*, 223:1–18, 2003.

[6] M.I. Arnone and E.H. Davidson. The hardwiring of development: organization and function of genomic regulatory systems. *Development*, 124:1851–1864, 1997.

[7] Z. Bar-Joseph, G.K. Gerber, T.I. Lee, and N.J. Rinaldi *et al.* Computational discovery of gene modules and regulatory networks. *Nat. Biotech.*, 21(11):1337–1342, 2003.

[8] H. Bolouri and E.H. Davidson. Transcriptional regulatory cascades in development: Initial rates, not steady state, determine network kinetics. *Proc. Natl. Acad. Sci. USA*, 100(16):9371–9376, 2003.

[9] J.M. Bower and H. Bolouri, editors. *Computational modeling of genetic and biochemical networks*. MIT Press, 2001.

[10] P. Brazhnik, A. de la Fuente, and P. Mendes. Gene networks: how to put the function in genomics. *Trends Biotechnol.*, 20:467–472, 2002.

[11] H. Bussemaker, H. Li, and E. Siggia. Regulatory element detection using correlation with expression. *Nat. Genet.*, 27:167–71, 2001.

[12] T. Chen, V. Filkov, and S. Skiena. Identifying gene regulatory networks from experimental data. *Parallel Comput.*, 27(1-2):141–162, 2001.

[13] T. Chen, H.L. He, and G.M. Church. Modeling gene expression with differential equations. *Pac. Symp. Biocomputing*, 4:29–40, 1999.

[14] T.H. Cormen, C.E. Leiserson, and R.L. Rivest. *Intoduction to Algorithms*. MIT Press, 2nd edition, 2001.

[15] E.H. Davidson. *Genomic Regulatory Systems*. Academic Press, 2001.

[16] E.H. Davidson, J.P. Rast, P. Oliveri, and A. Ransick *et al.* A genomic regulatory network for development. *Science*, 295:1669–1678, 2002.

[17] H. de Jong. Modeling and simulation of genetic regulatory systems: a literature review. *J. Comp. Bio.*, 9(1):67–103, 2002.

[18] A. de la Fuente, P. Brazhnik, and P. Mendes. A quantitative method for reverse engineering gene networks from microarray experiments using regulatory strengths. In *2nd International Conference on Systems Biology*, pages 213–221, 2001.

[19] P. D'Haeseleer, S. Liang, and R. Somogyi. Genetic network inference: From co-expression clustering to reverse engineering. *Bioinformatics*, 16:707–26, 2000.

[20] P. D'Haeseleer, X. Wen, S. Fuhrman, and R. Somogyi. Linear modeling of mRNA expression levels during CNS development and injury. *Pac. Symp. Biocomputing*,

pages 41–52, 1999.

[21] D. di Bernardo, T.S. Gardner, and J.J. Collins. Robust identification of large genetic networks. In *Pac. Symp. Biocomputing*, volume 9, pages 486–497, 2004.

[22] V. Filkov, J. Zhi, and S. Skiena. Analysis techniques for microarray time-series data. *J. Comp. Bio.*, 9(2):317–330, 2002.

[23] N. Friedman. Inferring cellular networks using probabilistic graphical models. *Science*, 303:799–805, 2004.

[24] N. Friedman, M. Linial, I. Nachman, and D. Pe'er. Using bayesian networks to analyze expression data. *J. Comp. Bio.*, 7(6):601–620, 2000.

[25] T.S. Gardner, C.R. Cantor, and J.J. Collins. Construction of a genetic toggle switch in *escherichia coli*. *Nature*, 403:339–342, 2000.

[26] T.S. Gardner, D. di Bernardo, D. Lorenz, and J.J. Collins. Inferring genetic networks and identifying compound mode of action via expression profiling. *Science*, 301:102–105, 2003.

[27] I. Gat-Viks and R. Shamir. Chain functions and scoring functions in genetic networks. *Bioinformatics*, 19(Suppl 1):i108–i117, 2003.

[28] I. Gat-Viks, R. SHAMIR, R.M. KARP, and R. SHARAN. Reconstructing chain functions in genetic networks. In *Pac. Symp. Biocomputing*, volume 9, pages 498–509, 2004.

[29] J. Gollub, C.A. Ball, G. Binkley, and J. Demeter *et al.* The stanford microarray database: data access and quality assessment tools. *Nucleic Acids Res*, 31:94–6, 2003.

[30] A.J. Hartemink et al. Using graphical models and genomic expression data to statistically validate models of genetic regulatory networks. In *Pac. Symp. Biocomputing*, volume 6, pages 422–433, 2001.

[31] A.J. Hartemink, D.K. Gifford, T. Jaakkola, and R.A. Young. Combining location and expression data for principled discovery of genetic regulatory network models. In *Pac. Symp. Biocomputing*, volume 7, pages 437–449, 2002.

[32] D. Heckerman, D. Geiger, and D. Chickering. Learning bayesian networks: The combination of knowledge and statistical data. *Mach. Learn.*, 20:197–243, 1995.

[33] J. Hertz. Statistical issues in reverse engineering of genetic networks. *Pac. Symp. Biocomputing*, 1998. Poster.

[34] T. Ideker, V. Thorsson, and R. Karp. Discovery of regulatory interactions thrugh perturbation: inference and experimental design. In *Pac. Symp. Biocomputing*, volume 5, pages 302–313, 2000.

[35] A. Jain and R. Dubes. *Algorithms for Clustering Data*. Prentice-Hall, Englewood Cliffs NJ, 1988.

[36] H. Jeong, B. Tombor, R. Albert, and Z.N. Oltvai *et al.* The large-scale organization of metabolic networks. *Nature*, 407:651–4, 2000.

[37] M. Kanehisa and S. Goto. *kegg*: Kyoto encyclopedia of genes and genomes. nucleic acid. *Nucl. Acid Res.*, 28:27–30, 2000.

[38] P.D. Karp, M. Arnaud, J. Collado-Vides, and J. Ingraham *et al.* The *e. coli ecocyc* database: No longer just a metabolic pathway database. *ASM News*, 70:25–30, 2004.

[39] S.A. Kauffman. *The Origins of Order: Self Organization and Selection in Evolution*. Oxford University Press, 1993.

[40] M. Lapidot and Y. Pilpel. Comprehensive quantitative analyses of the effects of promoter sequence elements on mRNA transcription. *Nucleic Acids Res.*, 31(13):3824–3828, 2003.

[41] S. Liang, S. Fuhrman, and R. Somogyi. REVEAL, a general reverse engineering algorithm for inference of genetic network architectures. In *Pac. Symp. Biocomputing*,

volume 3, pages 18–29, 1998.

[42] H.H. McAdams and A. Arkin. Stochastic mechanisms in gene expression. *Proc Natl. Acad. Sci. USA*, 94:814–819, 1997.

[43] H.H. McAdams and A. Arkin. Simulation of prokaryotic genetic circuits. *Ann. Rev. Biophys. Biomolecul. Struct.*, 27:199–224, 1998.

[44] P. Mendes, W. Sha, and K. Ye. Artificial gene networks for objective comparison of analysis algorithms. *Bioinformatics*, 19:122–9, 2003.

[45] R. Milo, S. Shen-Orr, S. Itzkovitz, and N. Kashtan *et al.* Network motifs: Simple building blocks of complex networks. *Science*, 298:824–827, 2002.

[46] N. Nariai, S. Kim, S. Imoto, and S. Miyano. Using protein-protein interactions for refining gene networks estimated from microarray data by bayesian networks. In *Pac. Symp. Biocomputing*, pages 336–47, 2004.

[47] S. Ott, S. Imoto, and S. Miyano. Finding optimal models for small gene networks. In *Pac. Symp. Biocomputing*, volume 9, pages 557–567, 2004.

[48] J. Paulsson. Summing up the noise in gene networks. *Nature*, 427:415–418, 2004.

[49] D. Pe'er, A. Regev, G. Elidan, and N. Friedman. Inferring subnetworks from perturbed expression profiles. *Bioinformatics*, 17(Suppl 1):S215–S224, 2001.

[50] Y. Pilpel, P. Sudarsanam, and G. Church. Identifying regulatory networks by combinatorial analysis of promoter elements. *Nat. Genet.*, 29:153–159, 2001.

[51] M. Ptashne. *A Genetic Switch: Phage Lambda and Higher Organisms.* Cell Press and Blackwell Scientific, 2nd edition, 1992.

[52] E. Ravazs, A.L. Somera, D.A. Mongru, and Z.N. Oltvai *et al.* Hierarchical organization of modularity in metabolic networks. *Science*, 297:1551–5, 2002.

[53] M.A. Savageau and P. Sands. Completly uncoupled or perfectly coupled circuits for inducible gene regulation. In E.O. Voit, editor, *Canonical nonlinear modeling: S-system approach to understanding complexity.* Van Nostrand Reinhold, New York, 1990.

[54] E. Segal, M. Shapira, A. Regev, and D. Peer *et al.* Module networks: Identifying regulatory modules and their condition specific regulators from gene expression data. *Nat. Genet.*, 34(2):166–76, 2003.

[55] Y. Setty, A.E. Mayo, M.G. Surette, and U. Alon. Detailed map of a cis-regulatory input function. *Proc. Natl. Acad. Sci. USA*, 100:7702–7707, 2003.

[56] I. Shmulevich, E.R. Dougherty, S. Kim, and W. Zhang. Probabilistic boolean networks: A rule-based uncertainty model for gene regulatory networks. *Bioinformatics*, 18:261–74, 2002.

[57] I. Shmulevich, H. Lähdesmäki, E.R. Dougherty, and J. Astola *et al.* The role of certain post classes in boolean network models of genetic networks. *Proc. Natl. Acad. Sci. USA*, 100:10734–9, 2003.

[58] E.D. Sontag. For differential equations with r parameters, 2r+1 experiments are enough for identification. *J. Nonlinear Sci.*, 12:553–83, 2002.

[59] T. Speed, editor. *Statistical Analysis of Gene Expression Microarray Data.* CRC Press, 2003.

[60] J. Stark, D. Brewer, M. Barenco, and D. Tomescu *et al.* Reconstructing gene networks: what are the limits? *Biochem. Soc. Transact.*, 31:1519–25, 2003.

[61] J. Stuart, E. Segal, D. Koller, and S.K. Kim *et al.* A gene co-expression network for global discovery of conserved genetics modules. *Science*, 302:249–55, 2003.

[62] A. Tanay et al. Revealing modularity and organization in the yeast molecular network by integrated analysis of highly heterogeneous genomewide data. *Proc. Natl. Acad. Sci. USA*, 101(9):2981–2986, 2004.

[63] S. Tavazoie, J.D. Hughes, M.J. Campbell, and R.J. Cho *et al.* Systematic determina-

tion of genetic network architecture. *Nat. Genet.*, 22:281–5, 1999.

[64] J. Tegner, M.K. Yeung, J. Hasty, and J.J. Collins. Reverse engineering gene networks: integrating genetic perturbations with dynamical modeling. *Proc. Natl. Acad. Sci. USA*, 100(10):5944–5949, 2003.

[65] M. Thattai and A. van Oudenaarden. Intrinsic noise in gene regulatory networks. *Proc. Natl. Acad. Sci. USA*, 98(15):8614–8619, 2001.

[66] E.P. van Someren, L.F. Wessels, and M.J. Reinders. Linear modeling of genetic networks from experimental data. In *Proc. ISMB*, pages 355–366, 2000.

[67] J.M. Vilar, H.Y. Kueh, N. Barkai, and S. Leibler. Mechanisms of noise-resistance in genetic oscillators. *Proc. Natl. Acad. Sci. USA*, 99(9):5988–92, 2002.

[68] G. von Dassow, E. Meir, E.M. Munro, and G.M. Odell. The segment polarity network is a robust developmental module. *Nature*, 406:188–92, 2000.

[69] A. Wagner. How to reconstruct a large genetic network from n gene perturbation in n^2 easy steps. *Bioinformatics*, 17:1183–97, 2001.

[70] D.C. Weaver, C.T. Workman, and G.D. Stormo. Modeling regulatory networks with weight matrices. *Pac. Symp. Biocomputing*, 4:112–123, 1999.

[71] L.F. Wessels, E.P. van Someren, and M.J. Reinders. A comparison of genetic network models. In *Pac. Symp. Biocomputing*, volume 6, pages 508–519, 2001.

[72] F.C. Wimberly, C. Glymor, and J. Ramsey. Experiments on the accuracy of algorithms for inferring the structure of genetic regulatory networks from microarray expression levels. *IJCAI 2003 Workshop on Learning Graphical Models for Computational Genomics*, 2003.

[73] A. Wuensche. Genomic regulation modeled as a network with basins of attraction. In *Pac. Symp. Biocomputing*, volume 3, pages 89–102, 1998.

[74] M.K.S. Yeung, J. Tegnãr, and J.J. Collins. Reverse engineering gene networks using singular value decomposition and robust regression. *Proc. Natl. Acad. Sci. USA*, 99:6163–6168, 2002.

[75] J. Yu, A.V. Smith, P.P. Wang, and A.J. Hartemink *et al.* Using bayesian network inference algorithms to recover molecular genetic regulatory networks. In *International Conference on Systems Biology*, 2002.

[76] C.-H. Yuh, H. Bolouri, and E.H. Davidson. Genomic *cis*-regulatory logic: experimental and computational analysis of a sea urchin gene. *Science*, 279:1896–1902, 1998.

[77] C.-H. Yuh, J.G. Moore, and E.H. Davidson. Quantitative functional interrelations within the *cis*-regulatory system of the *s. purpuratus endo-16* gene. *Development*, 122:4045–4056, 1996.