

An evaluation of the feasibility of gene network
reconstruction using current methods

Frick, Don Andreas

April 24th, 2005

Abstract

The question of how gene networks may be reverse engineered from experimental data is one of the central questions in the field of genetics. Although many approaches to the problem have been made, few have tried to evaluate the difficulties and limits of possible solutions.

We use a specially designed network generation and simulation framework to evaluate the difficulty of reconstructing gene networks using different algorithms. The results are used to make a realistic estimate of the best reconstruction quality that can be expected when working with a limited amount of experimental data.

Contents

1	Introduction	2
1.1	Motivation	2
1.2	Goals and context of the project	2
1.3	Biological background and terminology	4
1.4	Previous studies	7
1.4.1	Modelling	7
1.4.2	Reconstruction	8
1.5	Graph theory terminology	9
2	Reconstruction methods	14
2.1	Overview	14
2.2	Digraph generation	15
2.3	Wagner reconstruction algorithm	15
2.4	Wille reconstruction algorithm	19
2.4.1	Adaptation to binary data	20
2.4.2	Binary implementation of Wille algorithm	21
2.5	Testing procedure	22
2.6	Wagner reconstruction and measurements	23
2.6.1	Variable error count	24
2.6.2	Variable node count	24
2.6.3	Variable edge count	27
2.7	Wille reconstruction and measurements	29
2.7.1	Variable node count	29
2.7.2	Variable number of expression data sets	32
2.7.3	Variable number of simulated updates per node	32
2.8	Wagner - Wille juxtaposition	32
3	Data requirements	36
3.1	Overview	36
3.2	Complexity	37
3.3	Testing procedure	38
3.4	Results	40
3.4.1	Variable node count	40
3.4.2	Variable edge density	42
4	Conclusion	44

Chapter 1

Introduction

1.1 Motivation

The field of genetic research has seen dramatic advances in the last decade. One of the hallmarks of this area of science is that researchers are gathering large amounts of experimental data, but lack well established methods to process and organize this information. For example, microarray experiments (Figure 1.1) yield data sets that cover a huge number of genes reacting to external conditions in different ways. Although it is now commonly accepted that these genes are organized in large networks of complex interactions that control and drive the vital processes of cells, there is no known method to reliably reverse engineer these interactions from experimental data. An accurate reconstruction of the gene interaction network in an organism could give scientists a much deeper understanding of the metabolic and regulatory processes that take place in cells, with far-reaching consequences for the fields of biology, genetics and medicine.

1.2 Goals and context of the project

There are many examples of studies that focus on the reverse engineering of a particular genetic pathway or the purpose of certain genes in an organism (e.g. [26], [8], [22], [30], [38]). Most of these studies employ algorithms that aid in the analysis of unmodified experimental data, but the focus is generally on an evaluation of the results and their implications from a biologist's point of view. While these reconstructions often have high practical relevance and are of immediate use to biologists, they are less concerned with a general solution to the question of how gene networks may be reverse engineered and simulated.

On the other hand, there are many papers that have investigated the problem of reconstructing a network from incomplete information from a predominantly mathematical perspective (e.g. [6], [28], [31], [41], [23], [3]). These studies are largely concerned with graph theoretical questions and algorithms. While these

approaches are sound in theory, problems may arise when they are applied to the imperfect and incomplete data gathered from experiments.

Finally, there is an abundance of papers studying the reverse engineering of network topology, but few examples of papers that concern themselves with the exact functions of the interactions - that is, to ask not only whether two genes interact with each other, but also to quantify how they do so.

We believe that there is a gap between biologically inclined studies and their mathematical counterparts. The purely mathematical problem of network reconstruction from incomplete data is extremely complicated and remains a field of intense research. It may prove helpful to consider the problem from a slightly different point of view, namely that of a researcher confronted with a real, but limited amount of data such as is available today. On the other hand, we do not wish to focus on a given example to the point where the question of network reconstruction in general is neglected.

This project has two primary objectives. The first is to apply two existing network reconstruction algorithms to sets of simulated networks to examine how and how well the reconstruction works. We will also study these algorithms on a more theoretical level, to find how they react to certain network constellations. By performing a set of simulations with varying conditions, we hope to expose some of the important practical problems that arise when reconstructing gene networks, and make some suggestions on how they may be solved.

In order to perform these reconstructions, we will introduce a gene network model and design a framework that can generate networks and simulate their dynamics. We will then adapt the chosen algorithms to our model to perform the reconstructions and evaluations.

The second goal is to estimate the practical limits to reconstruction attempts when one has a limited set of data available. Instead of assuming an unlimited amount of data or examining the behavior of an algorithm when faced with small

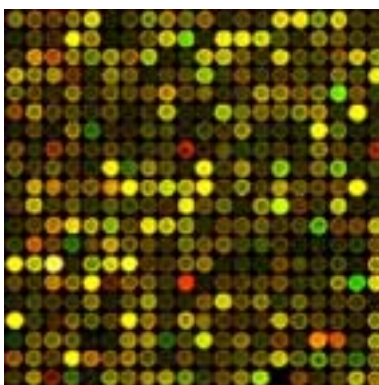


Figure 1.1: Results of a microarray experiment: Rows are genes, columns are external conditions.

amounts of errors and missing data, we will start with a number of measurements that can reasonably be expected from state of the art experiments and attempt to find the best reconstruction that is possible, independent of the algorithm that is used. We are not aware of any existing studies that gauge the practicality of network reconstruction in such a manner. To perform these estimates, we will make use of the model and framework that were introduced for the first objective.

- The first part of this thesis contains the background and motivation of the project, a brief description of some of the previous work that has been done in the field and some introductory information on genetics and graph theory.
- The second chapter focuses on an examination of two network topology reconstruction algorithms, with an explanation of how they function and a set of simulated results to demonstrate their characteristics.
- In the third part, we consider the more general question of how much information about a network can be gathered from a limited set of experiments.
- The final chapter contains a summary of results and the conclusions we have derived from them.

1.3 Biological background and terminology

This section will introduce some basic concepts and expressions from cell biology and genomics. For a comprehensive introductory text, see [4].

A *gene* is defined as “a DNA region that controls a discrete hereditary characteristic, usually corresponding to a single protein or RNA” (from [4]). Each gene in a living organism will have a varying level of phenotypical *expression*, which is essentially a measure of how active it is. Most genes, when activated, will synthesize proteins; these proteins can in turn influence the activity of other genes. This influence can take different forms; *activation* means that the products from one gene’s expression will enhance the expression levels of another gene. For instance, an enzyme produced from one gene may increase the production levels of an enzyme associated with another gene (Figure 1.2 A). The counterpart, *repression*, means that if one gene is active, some genes influenced by it will be less active. An example for this is a gene that produces a protein that blocks access to a particular DNA sequence, preventing another gene from being expressed (Figure 1.2 B).

A *transcription factor* is “a protein required to initiate or regulate transcription in eucaryotes” (quote from [4]). For our purposes, this means that a transcription factor will often have a comparatively high number of outgoing edges in the network, since it influences the expression levels of many other genes.

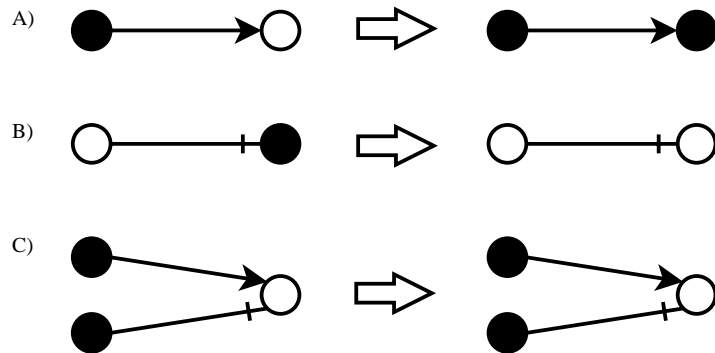


Figure 1.2: Illustration of activation, repression and a more complex interaction, active genes are black. Note that in case C, the activation and repression cancel each other out.

There are also complex interactions that involve the inputs of several genes. In these cases, a repressive effect can supersede an activation (as shown in Figure 1.2 C) or vice versa. The entire set of genes and their interactions can be modelled as a digraph (directed graph), where the genes are drawn as nodes and the interactions are represented by directed edges. A sample digraph is shown in Figure 1.3.

Such a set of genes and interactions is called a *genetic network*. In the digraph model, each node or gene is associated with a value that stands for its current expression level. The expression level of a given gene can be influenced by external factors, such as temperature and radiation, and by the expression of other genes as described above. Some different methods for simulating interactions between genes will be discussed in Section 1.4.

Microarray experiments provide the type of data that is most commonly used in studies attempting the reconstruction of gene networks. These experiments measure the expression levels of large numbers of genes in parallel, under varying external conditions and internal changes. For instance, one might examine how the expression levels of a set of genes change as the organism grows, or when it is exposed to direct sunlight, or kept in the dark. The types of data that are immediately useful for the reconstruction of gene networks are those from knockout or overexpression experiments and those from time series measurements.

A *knockout* experiment essentially removes a gene from an existing network, preventing it from ever being expressed. Doing so can help determine which genes are influenced by the knockout, since their expression value in that par-

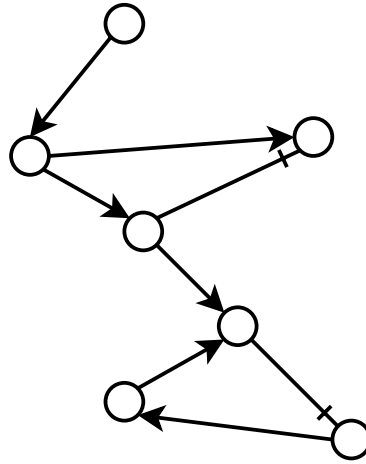


Figure 1.3: A digraph that represents a small gene network.

ticular experiment will often be different from the value in the unchanged case. *Overexpression* follows the same principle, but forces a gene to a high expression value regardless of inhibition and enhancement, which again will have an influence on some of the genes that are activated or repressed by it. Knockout and overexpression experiments are both expensive and difficult to perform, and will sometimes fail since they can seriously disrupt vital functions, which can kill the host organism.

Some complex gene interactions can only be brought to light by performing *double knockout* (or overexpression) experiments. These special cases will be examined in detail later on. The principle of the double knockout or overexpression is the same as in the single counterpart, except that two genes are altered at the same time. Double experiments are more difficult to execute, and carry an increased risk of causing damage to the host organism.

Once a knockout or overexpression has been performed, one can detect which genes are affected by the one that has been altered in the experiment. However, it is not clear whether these interactions are direct or indirect, that is, whether the change evident in the targets has propagated through other genes before taking effect. One problem of topology reconstruction then lies in finding out which genes are directly connected to each other.

As we shall see in the next chapter, this problem is simple if one is working with sufficient amounts of experimental data, but much more difficult in practice since available data is both scarce and unreliable. The problem of imposing the correct functions on each node is similar in that it can easily be solved given enough data, but is made difficult or even impossible by the limited number of

experiments that are normally available.

Time series experiments are used for the same purpose as knockout and overexpression experiments; to find out which genes affect each other. However, instead of directly altering the network and observing the changes, they rely on natural fluctuations of expression levels that happen over time, or in response to some external stimulus. Data gathered with this technique is slightly more difficult to analyze, since all genes and their changes must be observed simultaneously and the changes in expression levels are often less pronounced. The advantage is that in theory a much smaller number of experiments needs to be made to collect enough data for reconstruction. Instead of having to perform separate experiments for every gene, a comparatively small number of measurements can yield sufficient data to reconstruct the network topology, since each series of measurements can contain information on the behavior of large parts of the network.

1.4 Previous studies

A number of studies concerning the reconstruction of gene networks are collected and presented in [7] and [37]. Most of these reconstruction approaches are qualitative, in that they focus on determining whether or not a link exists between two given genes. If one wants to simulate the behavior of a network, one has to go one step further and include information on the strength and type of these interactions. Simulations, in turn, make it possible to gather data not immediately available from experiments and verify the accuracy of a model (see [9] and [8]).

1.4.1 Modelling

A large number of different network models have been introduced. In [7], the existing modelling approaches are separated into several categories: The first and largest includes all methods that use binary data and functions, with the expression levels of the genes being either high or low, and boolean functions used to simulate how genes react to their inputs. Boolean models have remained popular since their introduction by Kauffman in [17]; there are many models that are more detailed and more accurate, but the boolean approach is much simpler, and in many cases it provides for a sufficiently good model.

The second possibility is to use continuous data to represent the state of each gene. This is clearly more complex to design and takes more time to compute, but has the potential to deliver much better approximations of real biological networks. If the simulation is performed with discrete time steps, the states of both these network types can be updated in a synchronous or asynchronous manner, while differential equations allow simulations in continuous time. A well developed continuous data model is described in [9], [8] and [10].

There are several approaches that can be made when reconstructing a network with transition functions. One is to begin by rebuilding the topology,

which results in a simple directed graph, and then attach best fit functions on the edges to simulate the interactions between the genes. Another less often used option is to initially assume that all nodes are connected, and impose best fit functions where many of the variables have no influence on the output, effectively including information on the topology in the interaction functions themselves, as done in [34] and [36].

Most network models can be extended with stochastic behavior to simulate unexplained events and external influences on the biological system. An element of randomness can be introduced in several different ways: One of them assumes that any gene has a small chance of changing its expression level in either direction, regardless of input. This is particularly easy to implement in a boolean system, where the change corresponds to a flip from zero to one and vice versa. The rationalization for these random changes is that external influences (i.e. any effect not originating from the activity of another gene) can cause such seemingly unprovoked changes of state. The second random extension modifies the enhancing and inhibiting interactions between genes. These interactions can be made to operate in a probabilistic manner, with the result depending on both the activator or repressor genes and a random factor. This modelling method is closer to the behavior of a genetic network than one using purely deterministic functions. A summary of different statistical simulation methods and an example of a statistical model can be found in [21], and an inference method for a statistical model has been described in [15].

Finally, some researchers have constructed models using Bayesian networks, directed graphs with probabilistic functions associated with the edges (see [14], [28] and [39]).

1.4.2 Reconstruction

Most reconstruction methods were designed to fit a particular model, which makes it somewhat difficult to consider them on their own. In many cases, they use statistical methods to find patterns in expression data from which connections between genes may be inferred. Other attempts have been made using graph theoretical methods (e.g. [19], [24]) and genetic algorithms (e.g. [18], [33]).

While most reconstruction methods rely on mutant or time series experiments and the data derived from them, some take advantage of additional information. For instance, some gene interaction pathways of an organism may already be known and can thus be directly integrated into the model, as suggested in [26]. Some studies use qualitative knowledge of biological networks to achieve a more accurate reconstruction; an example would be the higher average outdegree associated with transcription factors as used by Qian in [29].

If for every gene in a network, one has determined the complete list of other genes that are affected by it, that network can be reconstructed with a high degree of accuracy, as suggested in [41] (this will be explained in detail in Chapter 2). However, such complete information will rarely be available, since for realistic networks with thousands of genes a huge number of experiments

would be required, which was shown in [2]. In addition to practical limits on the number of experiments, errors in the gathered data can cause problems for reconstruction attempts.

The reconstruction problem becomes more manageable when one is working with a small number of genes. One way to force smaller gene counts is to split a large network into smaller parts prior to reconstruction, as suggested in [13] and [27]. When one divides a network in this manner, one assumes that the network is composed of subnetworks (subgraphs) that are highly connected internally, but have a comparatively small number of connections to the outside. This not only reduces the computational complexity of the problem, but also corresponds to the predictions of biologists concerning the structure of these networks. However, the detection of these subnetworks is not necessarily a simple task, and would effectively require knowledge of the topological structure, the thing one is trying to achieve in the first place. In addition, errors that occur in the initial segmentation step can have a large impact on the quality of the final model.

1.5 Graph theory terminology

This section will introduce some basic terminology and concepts used when we discuss graph theory in Chapter 2. The descriptions herein are intended to be concise and illustrative; a mathematically precise and comprehensive treatment of these concepts may be found in [11].

A *graph* $G = \{N, E\}$ consists of a set of *nodes* (or *vertices*) N and a set of *edges* E . A graph can be represented as a drawing, where the nodes appear as circles and the edges are lines connecting the nodes. Each of the k elements of $N = \{n_1, n_2, \dots, n_{k-1}, n_k\}$ has a single identifier, which in our case will be an index between 1 and k , while each edge in $E = \{e_{ij}; i, j = (1 \dots k)\}$ is identified by the indices of a pair of nodes from N : The two end points of the edge.

Most of our discussion will focus on directed graphs, or *digraphs*, where each edge is assigned a direction. The first index of an edge e_{ij} is its initial node, while the second is the terminal node. The edge is said to be *directed* (or *pointing*) from n_i to n_j . This directionality is indicated by a marker at the end of the edge that is attached to the terminal node. Positive or activating edges are marked by an arrowhead, while negative or inhibiting edges have a crossbar. In the matrix representation, positive and negative edges are marked by positive and negative matrix values, respectively.

A *subgraph* is a graph $G' = \{N', E'\}$ where the sets of nodes and edges are subsets of the node and edge sets of another graph:

$$G = \{N, E\}; N' \subseteq N, E' \subseteq E$$

Other than the set of two lists $G = (N, E)$ and a drawing with lines and

edges, a graph may equivalently be described by a *connectivity matrix*, which has the form $M = \{m(i, j); i \in \{1 \dots k\} j \in \{1 \dots k\}\}$ where

$$m(i, j) = \begin{cases} 0 & \text{if there is no edge from } n_i \text{ to } n_j \\ 1 & \text{if there exists an edge from } n_i \text{ to } n_j \end{cases}$$

In an *undirected graph*, on the other hand, all edges are assumed to be bidirectional - if an edge e_{ij} exists, then its counterpart e_{ji} is automatically included. It follows that a matrix representing an undirected graph will always be symmetric.

A *path* is a non empty subgraph of the form

$$N = \{n_1, n_2, \dots, n_{k-1}, n_k\}; E = \{e_{12}, e_{23}, \dots, e_{(k-1)k}\},$$

where all the nodes are distinct. In other words, a series of directed edges that can be traversed from beginning to end. The number of edges in a path is its length.

The *adjacency list* is another way to represent a matrix. For each node n_i it lists the nodes n_j that are connected through an outgoing directed edge e_{ij} .

The *accessibility list* contains for each node n_i the list of all other nodes that can be reached by travelling along a path originating in n_i . The accessibility list can always be derived from the adjacency list, but the reverse is generally not true.

The *indegree* of a node n_i counts the number of directed edges e_{ji} that point into it, while the *outdegree* is the number of edges e_{ij} originating from that node.

Cycles are special cases of paths that start and end at the same node. A *loop* is a single edge e_{ii} that starts and ends at the same node. Many genes are self-regulating in a manner that corresponds to this construct, but since this effect is impossible to detect using the experiments and methods presented in this paper, we will assume that our graphs are free of them. The same goes for multiple parallel edges that share the same start and end nodes: $E = \{\dots, e_{ij}, e_{ij}, \dots\}$

In a *complete* graph or subgraph, all pairs of nodes $n_i, n_j; i \neq j$ are connected by edges e_{ij} and e_{ji} .

Strong Components are subgraphs where every node is in the accessibility list of all other nodes. It follows that for every element n_i there exists a directed path to all elements $n_j \in N$. Cycles are clearly a subset of strong components.

According to Watts [44], a *shortcut* is a single edge e_{ij} between two nodes n_i and n_j that are already connected by at least one path of length larger than one.

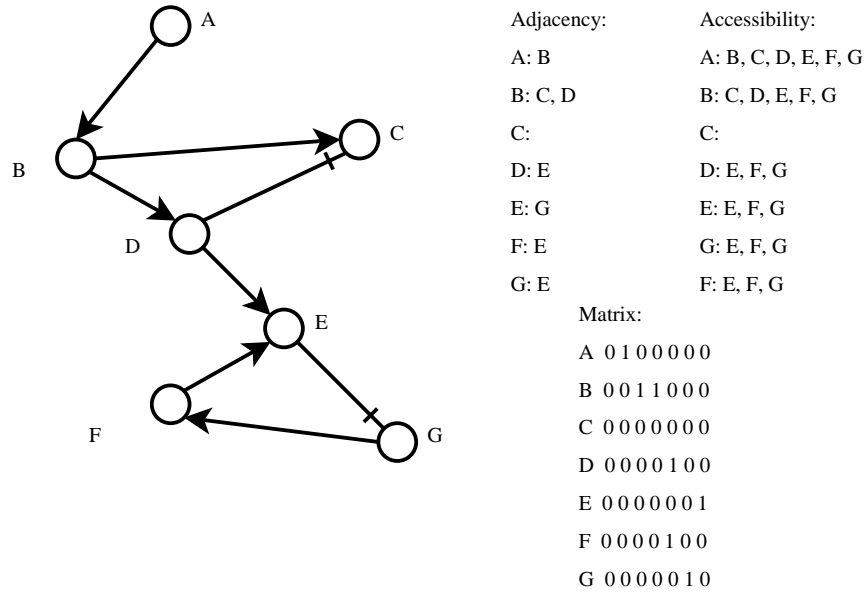


Figure 1.4: Different representations of the same graph.

All of the elements described so far describe the *topology* of a network. In order to simulate biological networks, we need to impose additional rules; in this paper, they will generally be referred to as *functions*.

The *state* of a network at a time t is a set of values $V = \{v_{1,t}, v_{2,t}, \dots, v_{k-1,t}, v_{k,t}\}$ where every value $v_{i,t}$ is associated with exactly one node n_i and vice versa. In our model, these values correspond to the expression values of a biological network. When simulating the behavior of a node over time, we need to be able to recalculate its value from the state of the rest of the network. This is done by assigning a *function*

$$v_{i,t+1} = f_i(v_{1,t}, v_{2,t}, \dots, v_{i-1,t}, v_{i+1,t}, \dots, v_{k-1,t}, v_{k,t})$$

to each node. When a graph topology $G = \{N, E\}$ is given, the function belonging to a node n_i is only dependent on the values of the nodes that are connected to n_i through incoming edges e_{ji} ; $j \neq i$.

A *probabilistic boolean network* assigns a set of multiple functions to each node, one of which is randomly selected every time the state of that node is updated.

According to [36], an input variable is considered *fictitious* if it has no effect on the output of the function. Therefore, if in a boolean function

$$f(x_1 \dots x_{k-1}, 1, x_{k+1} \dots x_n) = f(x_1 \dots x_{k-1}, 0, x_{k+1} \dots x_n)$$

the variable x_k is considered to be fictitious. The functions assigned to our nodes do not contain any fictitious variables as these would have been equivalent to a non-existent edge, making an accurate reconstruction impossible.

Chapter 2

Reconstruction methods

2.1 Overview

In this chapter we will examine two algorithms, proposed by Wagner [41] and Wille [45] for use in reconstructing gene networks. Our goal is to investigate how they work and evaluate their performance using data gained from simulations. A series of tests will be used to assess the characteristics of each algorithm.

To perform these tests, we defined a gene network model and designed a framework that generates random simulated networks. The data from these networks was then used to perform the reconstructions.

The network is represented by a directed graph with no loops $e_{i,i}$ that start and end at the same node. While such interactions are believed to exist in gene networks, they have no measurable effect on the types of data we derive during simulation. In addition, there may be only one instance of any given edge $e_{i,j}$ (i.e. no edge lists of the form $E = \{\dots e_{i,j}, e_{i,j}, \dots\}$).

We decided to use a binary data model for the process. Each node is assigned a boolean value to represent its state at any given point of time. If the gene's normalized expression value is higher than a certain threshold, we consider it to be activated (1), otherwise it is inactive (0). Each node also has a fixed state function that produce a single boolean value from the set of boolean states read from activating or repressing nodes.

Updates are performed in discrete time steps, either synchronously (where the states of all nodes change simultaneously) or asynchronously.

As suggested in [35], this model is simple but accurate enough to simulate basic network behavior. It is also the type of model used by the Wagner algorithm. We modified the algorithm by Wille, which was originally designed to work with continuous data, to function with our binary model.

2.2 Digraph generation

The directed graphs used in these tests were generated using the methods suggested in [25], similar to those used by Wagner in [40] and [41]. Each node is given randomly generated in- and outdegree values, the distributions of which are determined by the constants κ and τ . The degrees of the nodes have a power-law distribution, with the probability of a node having a degree of k being

$$p_k = k^{-\tau} e^{-k/\kappa}.$$

To calculate the in- or outdegree of a node, we generate a value with the distribution $e^{-k/\kappa}$:

$$k = 1 + \text{floor}(-\kappa * \ln(1 - r))$$

where r is a randomly generated variable in $[0, 1)$. This value is then accepted with a probability of $p_{acc} = k^{-\tau}$; if it is rejected, new values are generated until one is accepted.

Once in- and outdegree values have been generated for all nodes, the totals of both these values may have to be adjusted to be equal. In the next step, each node is considered to have a number of in- and outgoing stubs equal to its in- and outdegree. These stubs are connected with randomly selected partners, which completes generation of the digraph.

The mean and variation of in- and outdegrees can be adjusted by modifying κ and τ , but when using the method as presented above, the degree values cannot be smaller than 1. This results in graphs that always have cycles, which may not necessarily be correct from a biological perspective. Therefore, we performed most experiments with a minimal degree value of 0.

2.3 Wagner reconstruction algorithm

The first reconstruction method that we will examine was proposed by Andreas Wagner in [40] and [42]. As input, it requires the complete accessibility list of a graph, so for each node it must be given a list of all nodes that can be reached by following a path along the outgoing edges. The algorithm produces a graph that conforms to the given accessibility list while having the smallest number of edges; such a graph with a minimal edge count is also called *parsimonious*.

As shown in [41], for any given accessibility list ACC there exists exactly one most parsimonious graph that conforms to ACC . Wagner's algorithm is capable of finding this graph, provided that the accessibility list is error free and has been derived from an acyclic digraph.

A simple explanation of the algorithm's reconstruction procedure follows. It is initially assumed that the adjacency list of the network is identical to its accessibility list. As shown in Figure 2.1, this will introduce a large number of false edges, since indirect relations are complemented with all possible direct

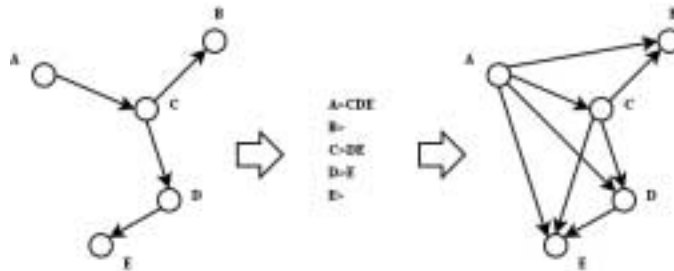


Figure 2.1: A sample digraph and the graph directly derived from its accessibility list.

connections. The algorithm then traverses the network, seeking the longest possible paths that can be used to fulfill the connectivity requirements of the accessibility list, and pruning all the connections that do not belong to these paths. If the graph is free of cycles, the only connections that need to be removed are *shortcuts*, edges between two nodes which are connected by at least one longer path of two or more edges (Figure 2.2). The result is the most parsimonious graph that fulfills the requirements of the accessibility list, and a very good approximation of the real network. Proofs and a more detailed explanation of the procedure can be found in [41].

If we assume that the given accessibility lists are complete and correct, there are only two situations where Wagner's algorithm will yield an incorrect result. The first is the aforementioned shortcut. In this case, the single edge path will always be removed when Wagner's algorithm rebuilds the most parsimonious graph. The reason for this behavior is simple: Before the algorithm begins pruning the initial graph that is derived from the accessibility list, every path of length larger than one is supplemented with a corresponding shortcut between the initial and terminal nodes. The basic assumption underlying Wagner's algorithm is that very few of these redundant connections actually exist. On the other hand, if such a shortcut does show up, it is impossible to distinguish from an indirect path just by looking at an accessibility list.

The second case is far more important in terms of possible errors: The basic version of Wagner's algorithm cannot determine whether any of the possible edges inside a strong component exist or not. To circumvent this problem, each subnetwork that is a strong component is represented by a single node. The only things known about these subnetworks is which nodes they contain, and that they are all connected to each other by at least one cycle; in effect, we must assume that they are completely connected subgraphs. Since a single cycle is enough to create such a situation, the reduction made in these cases is often a source of many false positive errors. The minimal number of edges for a cycle of n nodes is $n - 1$, but a strong component with the same nodes will have a total of n^2 edges.

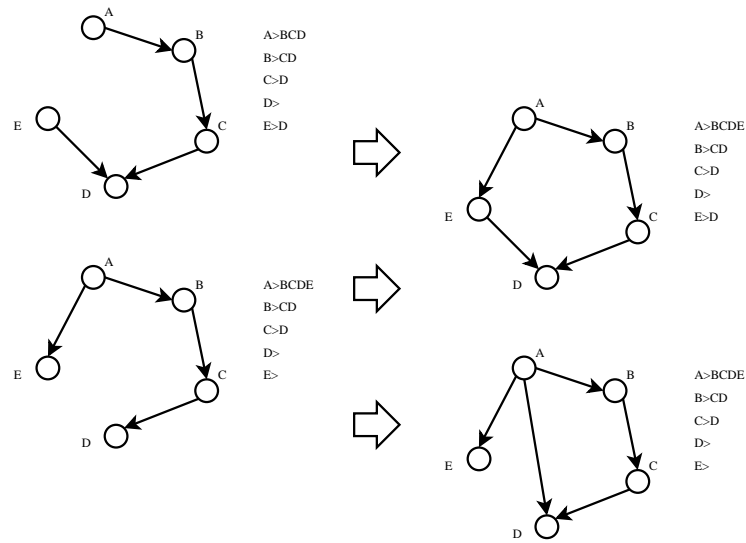


Figure 2.2: Shortcuts added to digraphs: Note that completing the two edge shortcut AED has an effect on the accessibility list, while the single edge shortcut AD does not.

In fact, it is impossible to reconstruct a strong component of three or more elements using only the information from the accessibility list or the data gained from single mutant experiments. The accessibility list is clearly of little use, since a strong component by definition consists of a group of genes that are all contained in each other's respective accessibility lists.

The ineffectuality of single mutant experiments can be explained as follows: Since a strong component must contain at least one cycle, every change that propagates through the subgraph can reach every one of the other nodes. This in turn means that every interaction that is observed (e.g. n_i has an activating effect on n_j) could have propagated directly or indirectly. Contrast this with a case of a non looped path, where mutations of nodes closer to the end do not affect the expression values of nodes near the beginning (Figure 2.3).

The only way to gather definite information that can yield additional insight in the topology of strong components is by performing double knockout or overexpression experiments. By altering the expression values of two genes in the same experiment, it is possible to isolate the relation between two particular genes, and thus find out whether they interact with each other directly (see Figure 2.4. However, even this method is limited to solving single cycles. If a strong component contains more than one cycle, one would have to mutate three or more genes at once to gain any useful information. However, the situation is not quite as hopeless as it may appear to be: All these observations only apply to a binary data model and reconstruction from accessibility lists.

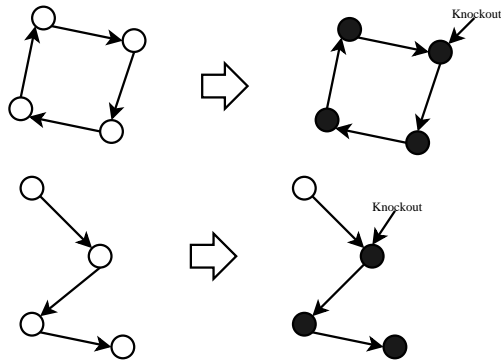


Figure 2.3: Effect of a strong component on expression data: In the first case the strong component makes it impossible to tell how the external change propagated. In the second case one can infer the order of influence.

Continuous expression values or other forms of data can be used to solve the strong component problem more efficiently.

A further thing to note about Wagner's algorithm is that it requires a large amount of data for an accurate reconstruction. In order to determine the full accessibility list of a set of genes, one has to perform several knockout and overexpression experiments for every one of them. Since the genomes of most known organisms contain thousands of genes, this is generally not feasible. The performance of the algorithm using limited or faulty data has been examined and found to be quite good in [42], but the ratio of false or missing data in those tests was less than 10%.

Two extensions to the algorithm have been proposed in [40]. The first takes into account the sign of each edge - whether it activates or represses its target - and uses this information to find shortcuts that have the opposite effect of a longer path and thus cannot safely be removed. Shortcuts that have the same sign as a parallel path are considered redundant and are deleted, just as before. The additional information required - whether a given interaction is activating or repressing - is trivial to derive from experimental data when determining the accessibility lists.

The second extension is used to get more detailed knowledge about the topology of strong components, using double mutant data as described above and shown in Figure 2.4. A large drawback of this method is that double mutant experiments have a higher cost associated with them, both in labor time and in increased risk of damaging the organism. In addition, the number of experiments required to solve a strong component increases with the square of the number of genes in that strong component, since every possible combination of two genes has to be tested out. If the strong component contains more than one cycle, the number of experiments increases even further.

In the reconstructions presented in this chapter, we have used the original

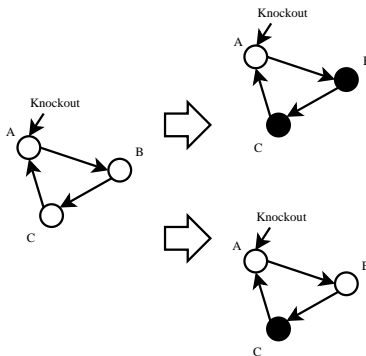


Figure 2.4: Method for reconstruction of a strong component. The single loop is broken by removing gene A, which allows us to observe the effects of altering gene B.

version of the Wagner algorithm, without the added extensions.

2.4 Wille reconstruction algorithm

The second algorithm used in this comparison was proposed by Wille in [45]. A first fundamental difference with the Wagner algorithm is that it operates on sets of expression values gained from microarray experiments. While a minimal number of expression value sets is required to get reasonable results, the algorithm was designed to operate on whatever amount of data is available.

The reconstruction method is based on graphical gaussian models and uses *partial correlation coefficients* to calculate the likelihood of an edge existing between two nodes, with the null hypothesis being that the nodes are not directly connected. Simply put, when considering any pair of nodes n_i and n_j , the algorithm determines whether the values of these nodes are strongly correlated in the given experimental data. It also considers all other nodes n_k ; $k \neq i, j$ to find out if a correlation can be explained with an indirect path running through n_k . If two nodes have highly correlated values that are probably not the result of an indirect connection (i.e. the null hypothesis is probably false), an edge is drawn between them.

Thus, the algorithm is capable of distinguishing between direct and indirect connections. It is theoretically capable of resolving strong components and nodes with multiple incoming or outgoing edges. However, the accuracy in all these cases depends on the type and amount of data that is supplied. In general, we have observed that nodes with multiple incoming or outgoing edges can pose a problem.

The sequence of edges in a path can be correctly identified since changes propagating along them will affect the levels of association between each pair

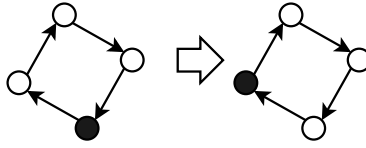


Figure 2.5: An anomaly propagating through a path or a strong component will tend to increase the correlation of directly connected nodes.

of genes; while all nodes in the chain will have highly correlated values, the correlations between those that are directly connected will be slightly higher than the correlations between indirectly linked nodes (Figure 2.5). A node with multiple inputs or outputs is a more difficult case; multiple inputs can result in complex functions where the effective correlation between two directly connected genes is significantly lower than usual. In a two input AND function, for example, the correlation between the separate inputs and the output is on average only 75%. Compare this with a single input NOT function, where in the absence of errors, the input and output values have an absolute correlation of 100%.

2.4.1 Adaptation to binary data

The algorithm was originally designed to operate using real, continuous data. Some modifications were necessary to allow the use of the binary data generated by our network model. In particular, the partial correlation coefficients cannot be calculated in the same manner as with continuous values since the assumption of the test data having a multivariate normal distribution no longer applies.

When using continuous expression values, the partial correlation coefficients $r_{ij|k}$ can easily be determined from the simple correlation coefficients r_{ij} by using the formula

$$r_{xy|z} = \frac{r_{xy} - r_{xz} * r_{yz}}{\sqrt{(1 - r_{xz}^2)(1 - r_{yz}^2)}}.$$

Our implementation uses *Fisher's exact test* to calculate the partial correlation coefficients from binary data. This test relies on a *contingency table* that counts the occurrences of every possible combinations of states that a group of three nodes can assume. A sample contingency table is shown in Table 2.1; since each of the three node can assume two states, there are 2^3 possible configurations. The value of 15 in the top left of the first table indicates that the three nodes assumed the states 000 in 15 of the observed experiments. The total sum of 47 across all elements of both tables is the number of all observations that have been made.

The basic version of Fisher's exact test only produces useful results with a large number of observations. As a rule of thumb, each state should be

15	1	2	14
3	2	1	9

Table 2.1: A sample contingency table. High values across diagonals (such as the 15 and 14 in this case) suggest a strong correlation between two nodes.

represented at least five times. This is not the case in the table shown in Figure 2.1, and was generally not the case in our simulated data sets. The networks we generated often had states that never occurred due to a particular constellation of functions and edges. In addition, the small number of experiments used for some of the simulations made it very likely that some of the elements of the contingency table would be too small.

To solve this problem, we used an *exact conditional test* to calculate the likelihood of the null hypothesis in each case. The result is similar to that of a Fisher’s exact test, but the procedure, while more complicated, is not affected by small sample sizes.

A detailed treatment of these concepts may be found in [12].

2.4.2 Binary implementation of Wille algorithm

The final procedure we used to apply Wille’s reconstruction method to binary expression data is as follows: For every combination of genes n_i, n_j ; $i \neq j$, we examine each possible third gene n_k and determine the maximum p - value

$$p_{ij,max} = \max(p_{ij|k}; k \neq i, j).$$

To calculate these values, we first derive the contingency tables from the expression data sets. We then determine the horizontal and vertical marginal values - the number of cases in which a node has a certain state. To continue the example shown in Table 2.1, the first vertical marginal value is $15 + 1 = 16$, meaning that node i was in the state 1 in 16 of the observed expression value sets. We then find all contingency tables that conform to these horizontal and vertical margins, and determine the probability of each of these tables, which is given by:

$$Pr(M|H_0) = \prod_{k=1}^L \frac{\prod_{i=1}^R n_{i+k}! \prod_{j=1}^C n_{+jk}!}{n_{++k}! \prod_{i=1}^R \prod_{j=1}^C m_{ijk}!}$$

where R is the number of rows, C the number of columns and L the number of lines in the contingency table. Expressions of the form n_{i+k} are contingency tables that have been summed up along the axis that has been marked by a “+” sign.

p_{ij} is then the sum of the probabilities of all permutated tables whose probability is smaller than that of the original table. $p_{ij,max}$ is simply the largest p_{ij} when all possible third nodes k are considered.

The $p_{ij,max}$ are then compared to a threshold value that is modified according to the *False Discovery Rate* procedure presented in [5]. This method, also known as the linear step up multiple comparison procedure, is designed to limit the number of false positive results. For convenience, we will rename the set of $p_{ij,max}$ - values to $p_1 \dots p_m$, where $m = N!$ is the number of possible $e_{i,j}$; $i \neq j$. The level α step up procedure is then performed as follows:

1. Put the set of p_l - values in ascending order: $p_1 \leq p_2 \leq \dots \leq p_m$.
2. Find the largest p_l that is below the sliding threshold $t_l = \frac{\alpha * l}{m}$
3. If such a $p_l < t_l$ exists, reject the null hypotheses $H_1^0, H_2^0, \dots, H_l^0$.

It is possible to begin comparing p - values from either end of the list; the results will not always be the same, but the difference is generally minimal.

If the null hypothesis is rejected for a given $p_{ij,max}$, an edge $e_{i,j}$ is inserted in the graph.

2.5 Testing procedure

For our experiments, we generated sets of random digraphs with varying edge and node counts, using the method described in Section 2.2. Each network was then used to generate a set of data of the form used by the one of the reconstruction algorithms - either a set of accessibility lists or a collection of expression values - to perform a reconstruction using the corresponding algorithm.

The accessibility lists used by the Wagner algorithm can directly be derived from the connectivity list of the digraph.

For the generation of the expression values, each node had to be assigned a binary function to determine its output from the values of its n input nodes. While one could select a random member from the set of all binary functions with n non-fictitious variables, we do not believe that doing so will lead to an appreciable increase in the accuracy of the model. Instead, we assumed the functions to be a combination of AND / OR statements, which also coincides with biological models where other combinations (like XOR) are rarely, if ever, found.

After constructing a network topology and imposing functions on it, the complete network was given a random starting state and asynchronously updated to generate output states for the Wille algorithm. A fixed total number of updates was given, and for each of these updates a random node was selected to have its state recalculated. On average, each node was updated three to five times, independent of the size of the network. According to our observations, that is sufficient for almost all networks to reach a stable state where performing further updates does not lead to a significant change in expression values or reconstruction accuracy.

After performing a set of simulations (the number of which depended on the experiment being made), we had an equal number of expression value sets which were then used to rebuild the network with the Wille algorithm.

A suggestion that immediately comes to mind would be to simulate time series data by performing step by step updates starting from a random state. Our experience shows that this does not work within the confines of the model that we are using. If one performs asynchronous updates, the separate changes are often too small and isolated to be observable, while nodes that by chance have identical states will be falsely correlated since they do not change. On the other hand, performing synchronous updates will cause many nodes to change states, but will also take the network into a quasi-stable state after very few steps; the changes after that point are at best oscillations between two very similar states. This can lead to extreme situations where all but one or two of the elements of a contingency table are zero, which severely reduces the reconstruction accuracy of the Wille algorithm.

It is for these reasons that we generated a random starting state for each new set of gene expression values. A large number of the final network states were still identical, but the frequent occurrence of a particular state is useful information for the Wille algorithm, and some anomalies from the starting state remain to the end. The reconstructions function well as long as the expression values contain a minimum amount of random aberrations.

The results of the reconstruction attempts were measured by their *sensitivity* and *specificity* scores, as in [19]. If $N_{original}$ is the number of edges in the original network, $N_{reconstructed}$ is the edge count of the reconstructed network and $N_{matches}$ the number of edges that are present in both networks, then the measures are calculated as

$$sensitivity = 100 * \frac{N_{matches}}{N_{original}}$$

and

$$specificity = 100 * \frac{N_{matches}}{N_{reconstructed}},$$

respectively.

The sensitivity represents the fraction of *false negative* errors, edges that exist but do not appear in the reconstructed model. The specificity on the other hand stands for the number of *false positive* errors, edges that appear in the reconstructed model but do not exist in the original network.

In almost every case, we made a set of 100 measurements for each data point. The box plot displays mark the top and bottom percentile of each distribution with a horizontal bar, and the 25th and 75th percentile with the edges of a notched box. The notch marks the median value of each measurement, and outliers are represented by crosses.

2.6 Wagner reconstruction and measurements

As can be seen in Figure 2.8, the algorithm results generally exhibit a very high sensitivity but have varying levels of specificity. The high sensitivity is due

to the algorithm's ability to detect all necessary edges, with the exception of shortcuts as explained above. Since a missed shortcut is the only way a false negative type error can occur, sensitivity values are generally above 90%. On the other hand, the low specificity levels are due to the algorithm's interpretation of strong components as complete subgraphs, which leads to a large number of false positives. Randomly generated graphs with lower average in- and outdegree values had a lower likelihood of containing cycles that cover many nodes, and thus were easier for the algorithm to reconstruct correctly. The requirement that each node have at least one incoming and outgoing edge has a particularly large negative effect on specificity scores, since it forcefully introduces a large number of cycles.

Clearly, the largest problems facing Wagner's reconstruction method are the high input data requirements and the inability to resolve cyclical formations without additional experiments. The more advanced version of Wagner's algorithm is able to solve cycles, but requires a very large number of double knockout experiments to be performed. We will explore these requirements in chapter 3.

The following subsections each explore a set of measurements where the effects of one specific parameter were examined. Unless noted otherwise, we used networks with 100 nodes and average in- and outdegrees of 1.

2.6.1 Variable error count

In this set of experiments, shown in Figure 2.6, we inserted a variable number of errors into the accessibility lists derived from the digraphs. The probability of errors ranges from 10% to 90%. We only inserted false negative type errors, since those were observed to be by far more common when deriving accessibility lists from expression data, as shall be shown in chapter 3.

The sensitivity plot shows a slight difference in slope, with the change occurring around the 40% error probability mark. Beyond this point, the sensitivity values show an almost linear degradation, with the mean values nearly mirroring the error probability. This suggests that no significant reconstruction is taking place anymore; any additional edges that are removed have a fixed likelihood of introducing a single false negative error, but have no other effect on the reconstructed network. The low specificity values in the 30% to 90% error region support this suggestion; the fraction of identified edges that actually exist in the original graph is roughly constant, reflecting the original ratio between the edge count of the adjacency list and the accessibility list.

In conclusion, the results suggest that no significant reconstruction with the Wagner algorithm can be expected if the likelihood of a false negative type error rises beyond 30%.

2.6.2 Variable node count

For these measurements, the results of which are shown in Figure 2.7, a set of random graphs with a varying number of nodes was generated. The other

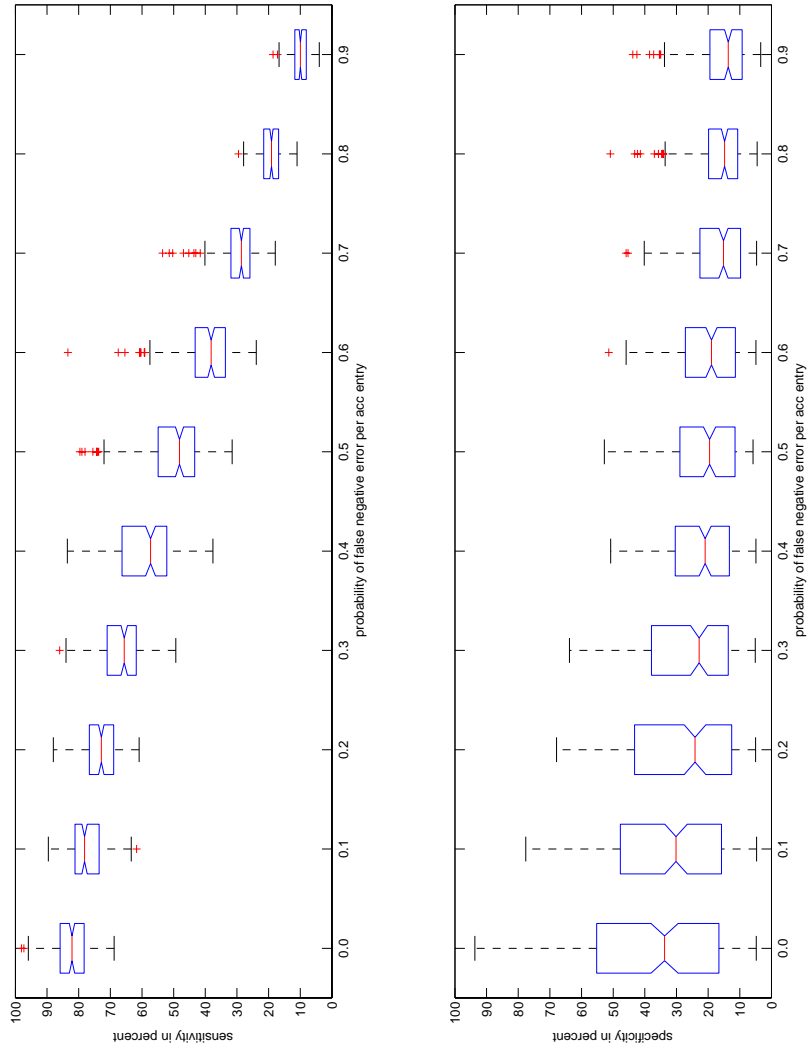


Figure 2.6: Sensitivity and specificity of Wagner algorithm when reconstructing graphs with varying probability of error in accessibility data.

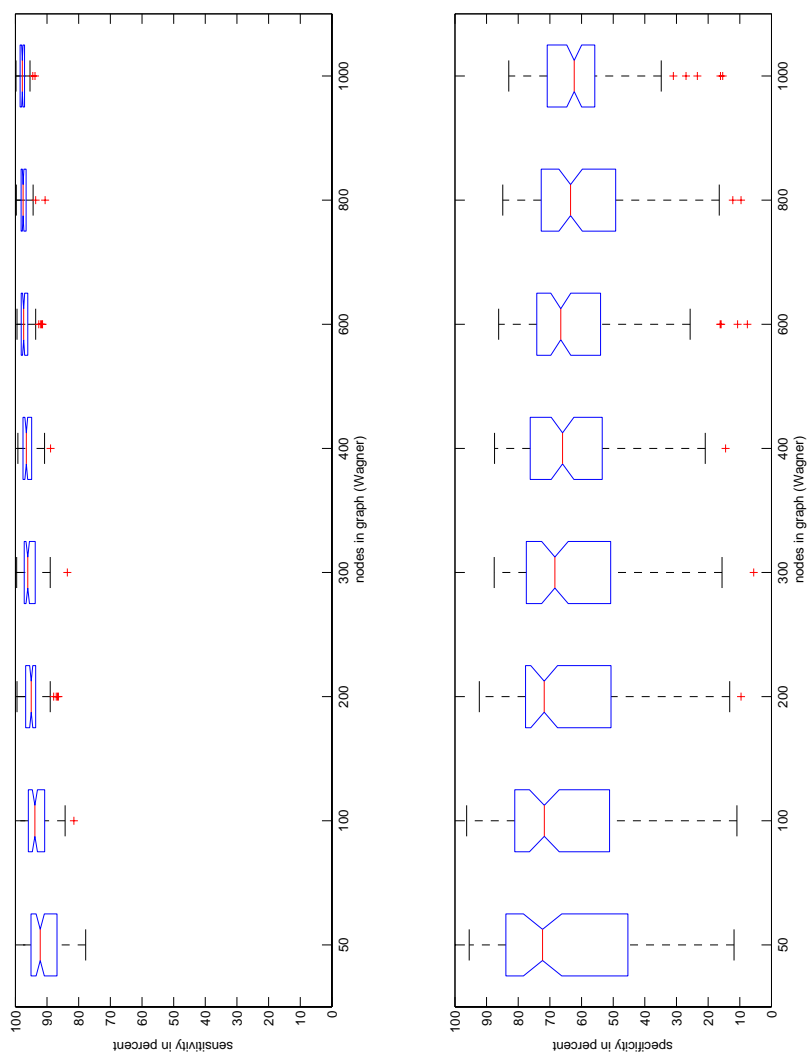


Figure 2.7: Sensitivity and specificity of Wagner algorithm when reconstructing graphs with varying node count.

parameters, mostly concerned with controlling the average number of edges per node, were kept constant throughout.

The sensitivity values are consistently high, and go up slightly with increasing network size. The reason for this probably lies in a lower likelihood for shortcut type connections. As the network grows larger, there are many more candidates for a single edge to connect to, so having two paths starting and ending at the same nodes becomes increasingly rare, which in turn reduces the false negative rate of the reconstruction.

Specificity generally shows large variations that get slightly smaller with larger node counts, which is due to the larger numbers reducing the likelihood of unusual configurations that would have caused strongly deviating values.

The mean specificity values remain relatively stable while decreasing slowly, perhaps due to the possibility of increasingly large strong components as the networks grows. However, the general stability of these values suggests that strong components do not become a significant problem.

We can conclude that network size does not have a large impact on the accuracy of reconstructions performed with Wagner’s algorithm. At this point, it should be noted that the algorithm executes very quickly - a network with a thousand nodes can be processed in a few seconds of CPU time.

2.6.3 Variable edge count

This set of measurements, shown in Figure 2.8, was performed with a fixed network size and a variable number of average edges per node. The last two measurements in this series, marked with a “+”, were made with graphs where each node had a minimum in- and outdegree of one. In all other cases, the minimum was zero.

The Wagner algorithm clearly performs better on sparse networks, since both sources of errors, shortcuts and strong components, are less likely to occur. The drop in specificity values, from an average of over 90% to around 10%, is due to the false positive errors strong components introduce; as the number of edges increases, the probability of having a strong component rises, and as mentioned earlier, the number of errors in a strong component with n elements can be as large as $n^2 - n + 1$.

This effect is particularly clear in the last two data points, where the minimum of one incoming and outgoing edge per node forces the generation of strong components that dramatically lower specificity values.

Sensitivity is generally high, but falls off slightly as edge density increases. This effect can be attributed to an increase of shortcut type edges. The slight increase in sensitivity as the average edge count increases to 2.8 and 3.4 and the two very high values at the end can be attributed to the strong component phenomenon: When a large part of the graph consists of strong components, each edge contained within them is automatically assumed to be detected correctly. As one can see from the specificity values, these numbers are somewhat deceptive; the results in these cases may not be very useful.

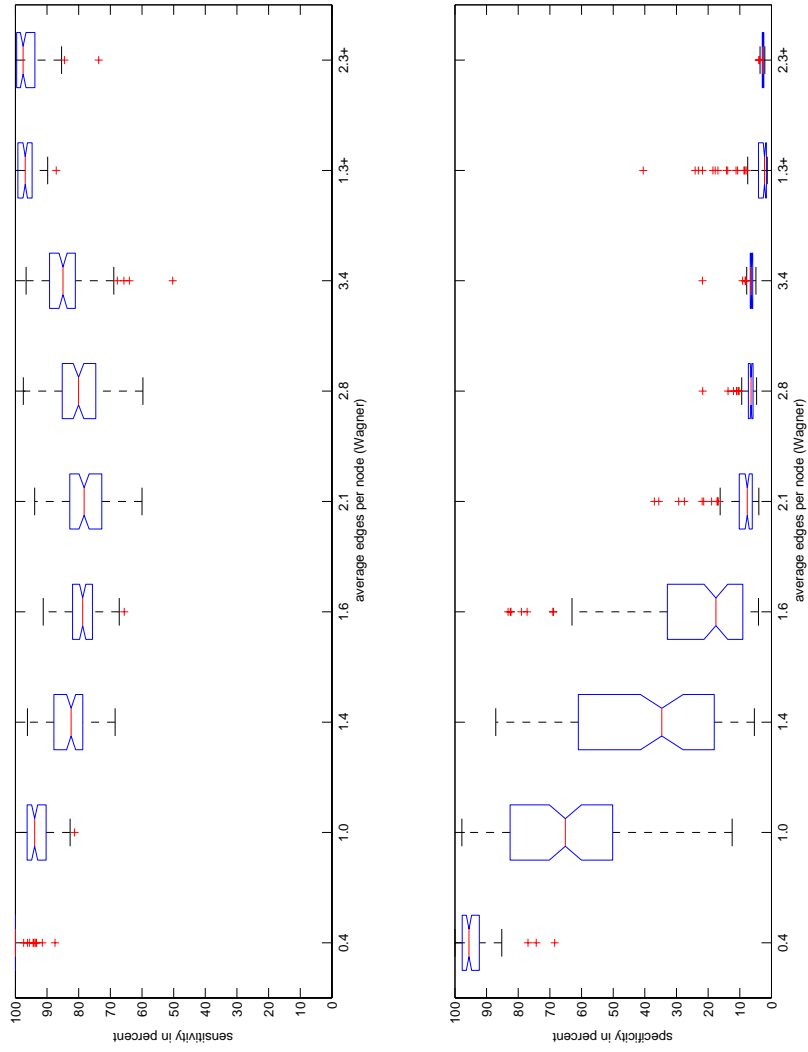


Figure 2.8: Sensitivity and specificity of Wagner algorithm when reconstructing graphs with varying average edge count per node.

The Wagner algorithm clearly performs best when confronted with sparse graphs, where the edge to node ratio is low. The presence of large strong components, which can be caused by a single loop that covers many nodes, is particularly detrimental to the accuracy of the results.

2.7 Wille reconstruction and measurements

Figures 2.10 and 2.9 are representative for reconstructions performed with Wille's method: The specificity is uniformly high, meaning that almost all results gained are correct.

Sensitivity results show more variation, which means that in many cases, an edge from the original graph will not appear in the reconstructed network. The reasons for this are twofold: Firstly, the algorithm is very strict in measuring correlations between the expression values of two genes. Thus, it will prefer false negative type errors to false positives by design.

The second reason that we have observed in experiments using small simulated networks is that nodes with several incoming or outgoing edges are a source of errors. As explained earlier, a node with several incoming edges will often have a much lower correlation with its input values than a single input node.

As before, we have performed a set of experiments with each one focusing on a particular parameter that is varied to observe its effect. Unless noted otherwise, the network had a size of ten nodes, each node had an average in- and outdegree of one, and the number of expression data sets generated was 100, with an average of three updates performed per node during simulation.

2.7.1 Variable node count

The results of these experiments, where Wille's algorithm was used to reconstruct networks of varying size, are shown in Figure 2.9. The number of edges per node stays constant throughout.

It would have been interesting to make measurements with much higher node counts; however, our version of the algorithm took a significant amount of time to execute, which limited the size of the networks we were able to work with. This is in no small part due to the need to calculate a large number of factorials when performing Fisher's exact test; the original version using continuous values may be significantly faster.

In the limited range we tested, we can observe a slight decrease in sensitivity for all but the smallest networks. The values do not change significantly after that, suggesting that the algorithm may not be particularly susceptible to changes in node count. Specificity stays at 100% throughout, with no visible changes.

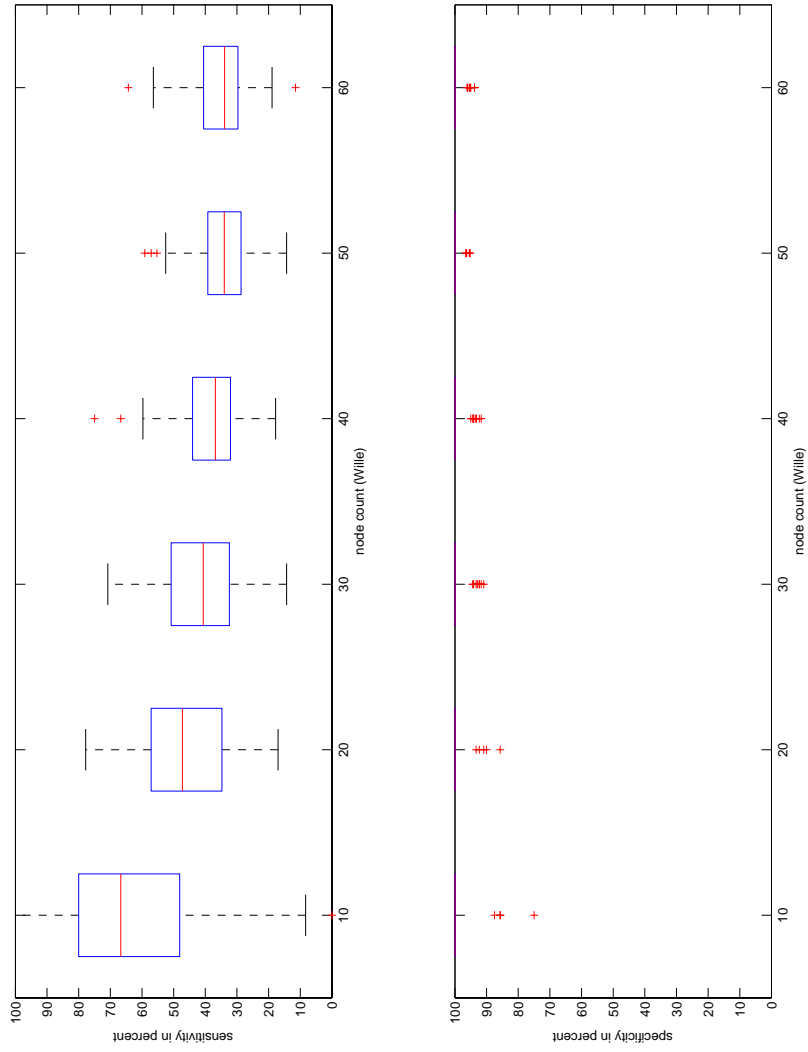


Figure 2.9: Sensitivity and specificity of Wille algorithm when reconstructing graphs with varying node count.

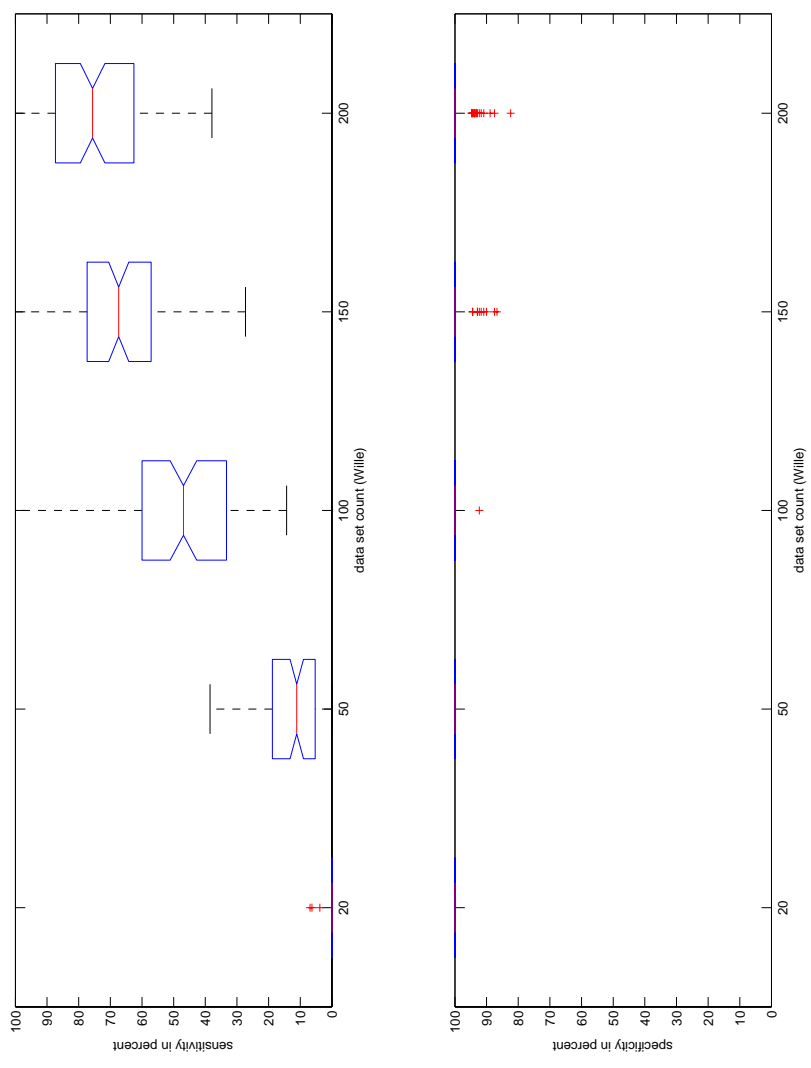


Figure 2.10: Sensitivity and specificity of Wille algorithm when reconstructing networks using a varying number of expression data sets.

2.7.2 Variable number of expression data sets

Figure 2.10 shows the results of experiments performed with a varying number of generated expression data sets. Network size and characteristics were constant. In other words, we were attempting to find out how Wille’s reconstruction algorithm responds to varying amounts of input data.

As with the network size, an increase in the number of data sets leads to much longer execution times. However, in this case a limit such as we used is somewhat justifiable, since the number of expression data measurements that can currently be performed is also strongly limited.

An increased number of experiments clearly improves the accuracy of the results. The requirement to perform a hundred or more experiments may seem restrictive, but this number is mitigated since our binary model introduces a considerable lack of information. We believe that similar accuracy levels could be attained with fewer experiments when using continuous data sets.

2.7.3 Variable number of simulated updates per node

In Figure 2.11, we have recorded the results of reconstructions performed on data sets where a variable average number of updates was performed on each node during simulation. The updates were performed in a random, asynchronous manner, but as mentioned earlier, networks showed a tendency to gravitate toward quasi-stable states quickly. This is mirrored in the sensitivity values, where no clear trends are visible beyond three updates per node.

It is interesting to note that all attempts to perform reconstruction on data that was generated through synchronous updates failed rather badly, with sensitivity values close to zero. Contrast that with the data at hand, where performing up to twenty updates per node has no detrimental impact on the accuracy of the results. We believe that the reasons for this behavior are twofold: First, the network state is far less likely to get stuck in a stable state that resembles a local minimum, where synchronous updates only cause a transition from A to B and back again. Secondly, the random nature of the asynchronous updates is clearly quite useful in this case, since it allows small inconsistencies to propagate through the network, which is the indicator the Wille algorithm uses to determine the order of a series of connected edges in a path. This type of random information is of course easier to obtain from sets of continuous valued expression data.

2.8 Wagner - Wille juxtaposition

Figure 2.12 shows the results of some networks that were reconstructed using first the Wagner and then Wille algorithm, with varying edge counts per node.

The results obtained with Wille’s reconstruction clearly have very high specificity values, even when the edge density rises to levels where many strong components must be expected. Sensitivity, on the other hand, is rather low. Wagner’s reconstruction has very good sensitivity values, but specificity quickly

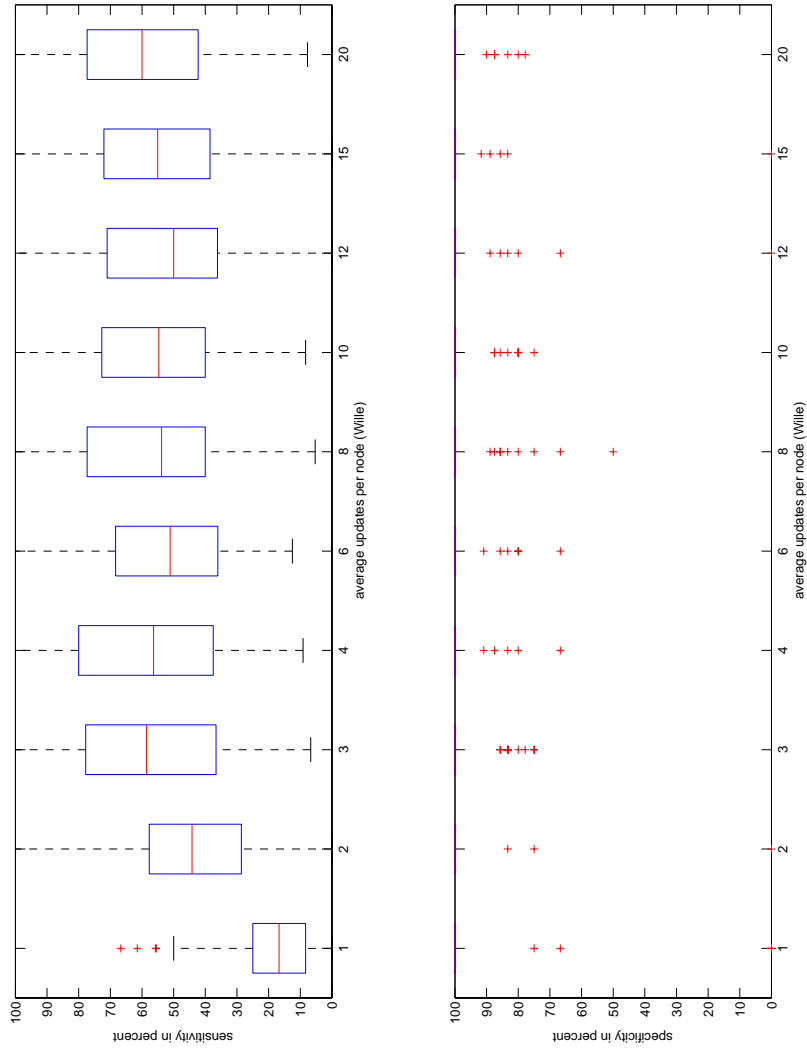


Figure 2.11: Sensitivity and specificity of Wille algorithm when reconstructing graphs with varying number of simulated updates per node.

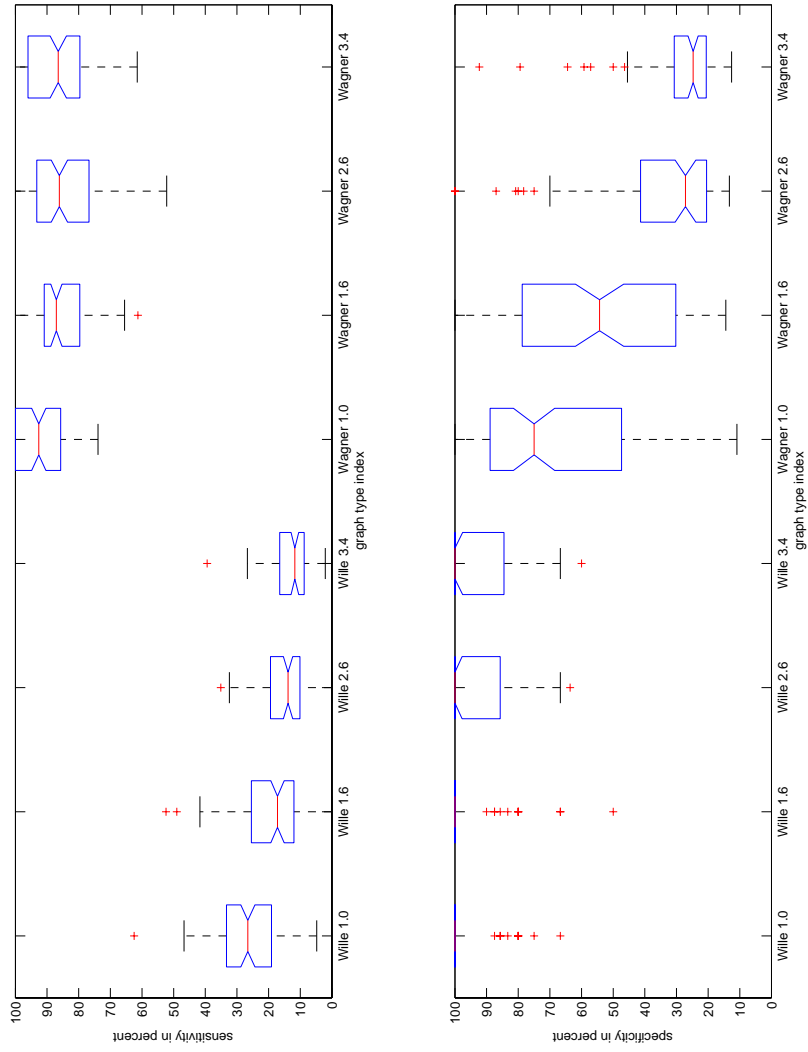


Figure 2.12: Sensitivity and specificity of Wagner and Wille algorithms when reconstructing graphs with varying edge counts.

degrades as edge numbers (and thus, the likelihood of strong components) increase.

There are several reasons why no direct comparison should be made between the algorithms examined in this chapter. Wagner's reconstruction method relies on the full and correct accessibility list of the network being available although it does have some tolerance to errors in the data. Wille's algorithm on the other hand operates on raw gene expression data gained from experiments, which in most cases contain far too little information to reconstruct a complete accessibility list. In addition, our experiments assume that this data has been binarized, which causes a further loss of information. Finally, Wille's algorithm reconstructs a network model with undirected edges, which makes it difficult to compare the results to those of Wagner, which have directed edges.

However, the experiment does offer some perspective on the results shown in this chapter; the two algorithms show large differences in many areas. Wagner's works very well on networks of almost arbitrary size, but suffers with high edge counts and has a tendency to include false positive results. Wille's reconstruction method operates well on highly connected networks and can operate on raw expression data, but requires large amounts of CPU time to process large networks and will often ignore some edges completely if they are not strongly represented in the given data.

Chapter 3

Data requirements

3.1 Overview

In this chapter, we will consider the following questions: How many gene expression experiments must one perform to get a reasonably good reconstruction of a gene network? And if we are limited to a given number of experiments, what is the best reconstruction one can expect to make?

To answer these questions, we will make use of the model and framework from the previous chapter. A similar boolean model was used by Akutsu and Kuhara, who have determined the upper and lower bounds on the number of experiments required for perfect reconstruction of gene regulatory networks in [1]. Their results are summarized in Table 3.1.

The results are rather discouraging since the genes in a full network number in the thousands, while the number of experimental data sets available are typically counted in tens or hundreds. However, these are strictly theoretical worst case results; the goal of this chapter is to examine some scenarios that fall between the extremes.

Constraint	Lower bound	Upper bound
No constraint	$\Omega(2^{(n-1)/2})$	$O(n2^{n-1})$
Indegree $\leq D$	$\Omega(n^D)$	$O(n^{2D})$
Indegree $\leq D$ & all genes are AND-nodes (OR-nodes)	$\Omega(n^D)$	$O(n^{D+1})$
Indegree $\leq D$ & Acyclic	$\Omega(n^D)$	$O(n^D)$
Indegree ≤ 2 & all genes are AND-nodes (OR-nodes) & no inactivating edges	$\Omega(n^2)$	$O(n^2)$

Table 3.1: Worst case number of experiments required for perfect reconstruction of networks with n genes [1]

3.2 Complexity

When calculating the minimal number of experiments needed for an accurate reconstruction, there are two basic factors that come into play: The complexity of the network, and the usefulness of the experiments that are made. Akutsu and Kuhara consider the cases of easy and difficult to construct networks, but assume that the experiments will always be of a worst case type. That is, of all possible experiments that can be made, the next one will always be the one that adds the least possible information to the recreation of the network.

An alternate point of view is that one can have knowledge of either the *presence* or the *absence* of a given interaction. In the beginning, nothing is known of any of the edges. As one performs experiments and deduces interactions from them, knowledge of these direct interactions is added to the model. However, while the presence of an edge can often be inferred from the results of a single experiment, knowledge of the absence of an edge generally requires a much larger number of tests.

For example, a single mutant experiment can yield very strong evidence for the existence of a particular edge $e_{i,j}$ if a change in node n_i evokes a change in n_j while the states of all other nodes remain the same. But to disprove the existence of said edge in our binary model, one would have to observe the results of every possible combination of input values, where every node except for the end point of the edge $e_{i,j}$ must be considered to be an input. In the worst case, the node n_j must be assumed to have an input function of the form

$$v_{j,t+1} = f_j(v_{1,t}, v_{2,t}, \dots, v_{j-1,t}, v_{j+1,t}, \dots, v_{N-1,t}, v_{N,t}),$$

where its expression value is influenced by all other nodes in the network. To conclusively prove that the parameter $v_{i,t}$ has no influence on the output value $v_{j,t}$, all 2^{N-1} expression value combinations of an N node network must be tested individually. This corresponds to the “worst possible experiment” problem explained above.

The problem can be mitigated to a certain degree if one uses *combinatorial design* testing (see [20] and [32]) to limit the number of experiments that need to be made. If one assumes that the indegree of any given node cannot exceed a given value D , one can construct a series of tests that contain all possible combinations of states for every possible set of D nodes from the network. This results in a much smaller set of experiments that need to be performed. The main disadvantage is that this method cannot detect functions with more than D variables; however, it is suggested in [43] and [16] that the number of connections per gene follows a power-law distribution, so this is probably the case for the majority of nodes in real gene networks.

The sparsity of real world networks is also beneficial in a more direct way since we can expect good results from focusing on the presence of edges while ignoring the proof of absence aspect.

Clearly, what has been said in the last few paragraphs also applies to the reconstruction of the functions of individual genes. It is for this reason that

we have not discussed the reverse engineering of activation functions. A binary input function is not fully defined unless all possible combinations of input values have been tested out. If we are given a set of binary expression values, the functions will almost always be over- or underdefined. Thus, while it is possible to find a function that is the closest possible fit to the given data, one cannot expect to perform any extrapolation.

Most of the problems just described are artifacts of the binary data model that we are using and are mitigated when one performs reconstructions using continuous data. Since a continuous expression value contains much more information than a binary one, it should be possible to achieve similar levels of accuracy with a lower experiment count. For example, it is probably easier to determine the nonexistence of a certain edge with continuous expression data, since it is possible to observe state changes that would be too small to cause a transition from one binary state to another.

The number of required experiments can be lowered further by making use of previously gained knowledge of the network during the reconstruction, and by fine tuning the reconstruction method with heuristics that take advantage of the idiosyncrasies of gene networks (e.g. transcription factors in [29]).

Ideally, an algorithm would be able to suggest an experiment that contributes a large amount of information to a network reconstruction (see [46]). Our observations suggest that the usefulness of a single mutant or overexpression experiment depends heavily on the state of the network that it is performed on. The same experiment can yield more or less information depending on the initial states of the genes that are being observed (Figure 3.1). This means that a suggestion for a good experiment must indicate both the gene to perturb, and the initial state of the network.

3.3 Testing procedure

The approach made by Akutsu and Kuhara is primarily concerned with theoretical limits. Our goal is to explore values between the extremes, in the hope that the results will provide a useful yardstick for researchers who are designing reconstruction algorithms that work with limited amounts of data.

The network model we use here is identical to the one from the previous chapter: A digraph with no loops or multiple edges, with a binary function assigned to each node with incoming edges and a single binary variable to represent the state of each node. Network simulation was performed with random asynchronous updates.

The goal was to find the average reconstruction accuracy when one has to work with a limited number of expression data sets. The first step, then, was to generate this limited number of data sets. Each of the data sets was assumed to be a single mutant or overexpression experiment, i.e. a network where one of the nodes is locked into a state of 0 or 1.

A simulated network without alterations will tend to gravitate toward one of very few stable states, similar to local minima. Simulating external influences

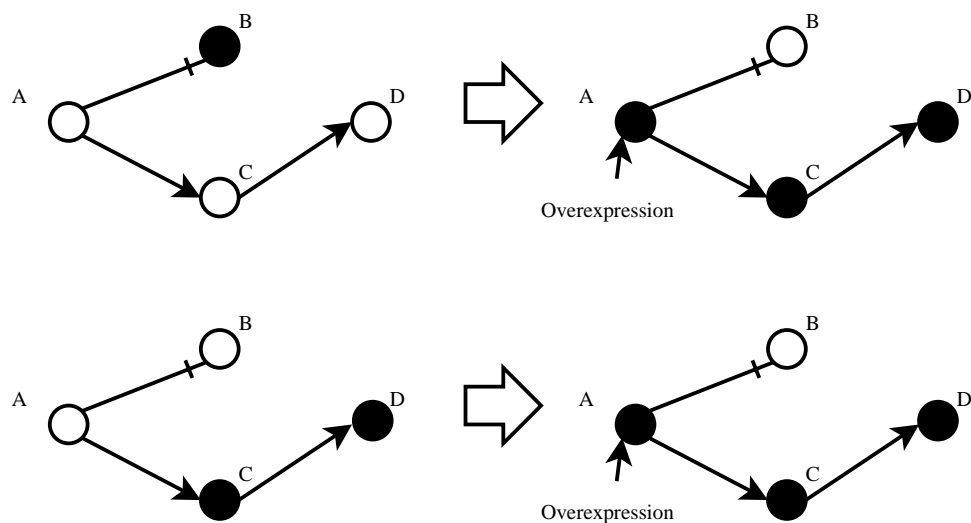


Figure 3.1: Top: A favorable initial state, where an overexpression experiment reveals many interactions. Bottom: The same network, with an unfavorable starting state that yields no information.

with random changes to the states of individual nodes does not prevent this from happening. Thus, in the confines of our model, overexpression and knockout experiments are the best way to generate data containing useful information.

One can alter two or more genes in a single experiment to gain additional information about a network, but the procedure is expensive and the pool of possible gene combinations is often prohibitively large. Since we wanted to put a strict limit on the number of experiments that were performed and did not even come close to exhausting the number of possible single mutant cases, we decided to work without multiple mutant data.

Each simulated single mutant or overexpression experiment was performed starting from a stable initial state. The network was then updated, with one node kept fixed in its altered state. This procedure is an approximation of how experiments are performed on real networks, and should be sufficiently accurate as long as the starting states correspond to the stable states of the real network.

The final state was then compared to the starting point; any nodes whose expression values had changed were put in the accessibility list of the mutated or overexpressed gene (Figure 3.2). By combining the information from several such experiments, a partial accessibility list representation of the graph can be constructed.

Why an accessibility list and not a finished digraph? The reason for this was mentioned in the previous chapter: It is impossible to reliably determine the inner structure of a strong component using only data from single mutant

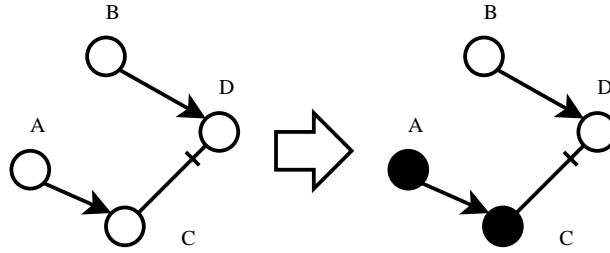


Figure 3.2: Reconstruction of accessibility list by performing an overexpression experiment on gene A. Since the state of C is changed, the edge $e_{A,C}$ can be inferred. There is no evidence of $e_{C,D}$.

or overexpression experiments. In many cases a subset of the interactions inside a strong component can be detected, but there is no guarantee that all of them can be identified correctly.

However, in the absence of strong components, a complete accessibility list yields enough information for a nearly perfect reconstruction of the original digraph using the Wagner algorithm. The largest source of errors for the Wagner algorithm - the strong component - is a moot point in this case. Furthermore, we have seen that the sensitivity of reconstructions performed with the Wagner algorithm deteriorates almost proportionally with the ratio of false negative type errors in the accessibility lists. We therefore suggest that, when using only single mutant or overexpression data, the accuracy with which the accessibility lists can be reconstructed is a good approximation of the best possible reconstruction of the original digraph.

3.4 Results

Graphs 3.3 and 3.4 show sensitivity and specificity ratios of accessibility list reconstructions using single mutant or overexpression data. Each single measurement was performed as described in Section 3.3; one data point in the graph represents the mean value of 100 independent measurements.

In both plots, the specificity values quickly approach 100% and remain steady after that. False positives are clearly only a problem with very small experiment counts. It is for this reason that we focused on false negative type errors when examining the error tolerance of the Wagner reconstruction algorithm in the previous chapter.

3.4.1 Variable node count

Figure 3.3 shows measurements using graphs with varying node counts and an average of two edges per node. As the size of the graph increases, sensitivity

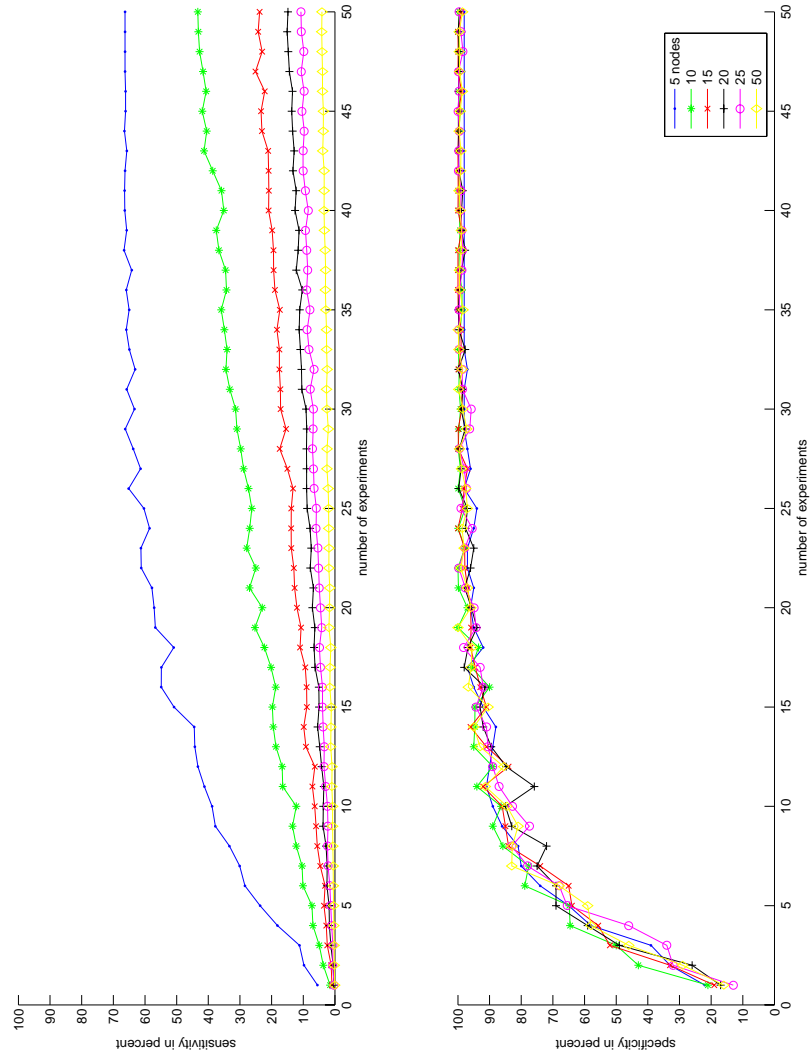


Figure 3.3: Sensitivity and specificity of accessibility list reconstruction, variable experiment count.

values drop dramatically; This is a consequence of the increased number of edges that comes with the larger network. The main conclusion that can be made is that the single mutant or overexpression experiments we performed only serve to reveal a relatively small, fixed number of interactions that does not grow with the network. For big networks, a much larger amount of data must be collected for an accurate reconstruction.

The plot of the smallest networks yield some additional information. After a certain point, adding more experimental information to the pool does not lead to a noticeable increase in sensitivity. Beyond this point, the probability that a new experiment will add relevant information is very low.

Sensitivity values top out around 60% in the best case. The remaining 40% of the edges are difficult, but not necessarily impossible to detect. Most of them are in strong components; since these subgraphs consist of nodes that are accessible from all other nodes in the subgraph, they introduce a large number of indirect interactions that cannot always be found with single mutant or overexpression experiments.

Another cause for false negatives is that some network areas may gravitate toward stable states that make it particularly unlikely that a given interaction will be found (see Figure 3.1). If none of the stable states of the network make it possible to detect a particular edge, the only way to find it with our single experiments is to force the network into a different, possibly unstable starting state, perhaps by introducing external influences.

3.4.2 Variable edge density

Figure 3.4 contains the results of measurements made with networks containing ten nodes and a variable number of average edges per node. The sensitivity values show little variation with differing edge densities. It appears that the number of edges detected with each experiment is roughly proportional to the overall edge density.

It is hard to tell how the increased number of strong components in the denser graphs affects the sensitivity values. The sensitivity values are clearly lower for the denser graphs, but the effect is much weaker than expected. One explanation for this is that at high edge counts, the possible edges inside a strong component are more likely to be filled out, which makes them easier to detect using single mutant or overexpression type experiments.

Interestingly, these incomplete reconstructions of the accessibility lists may sometimes lead to more accurate final graphs than the complete versions. Direct interactions have a good chance of being detected by our method, which is desirable. On the other hand, indirect connections have a good chance of being ignored, which is actually beneficial in this case since strong components often introduce a huge number of them, regardless of how many edges actually exist. A fully connected strong component has the same accessibility list as one that contains only a single loop. The missing entries of the accessibility list will reduce the quality of the Wagner reconstruction, but not as much as the false positives caused by a large strong component.

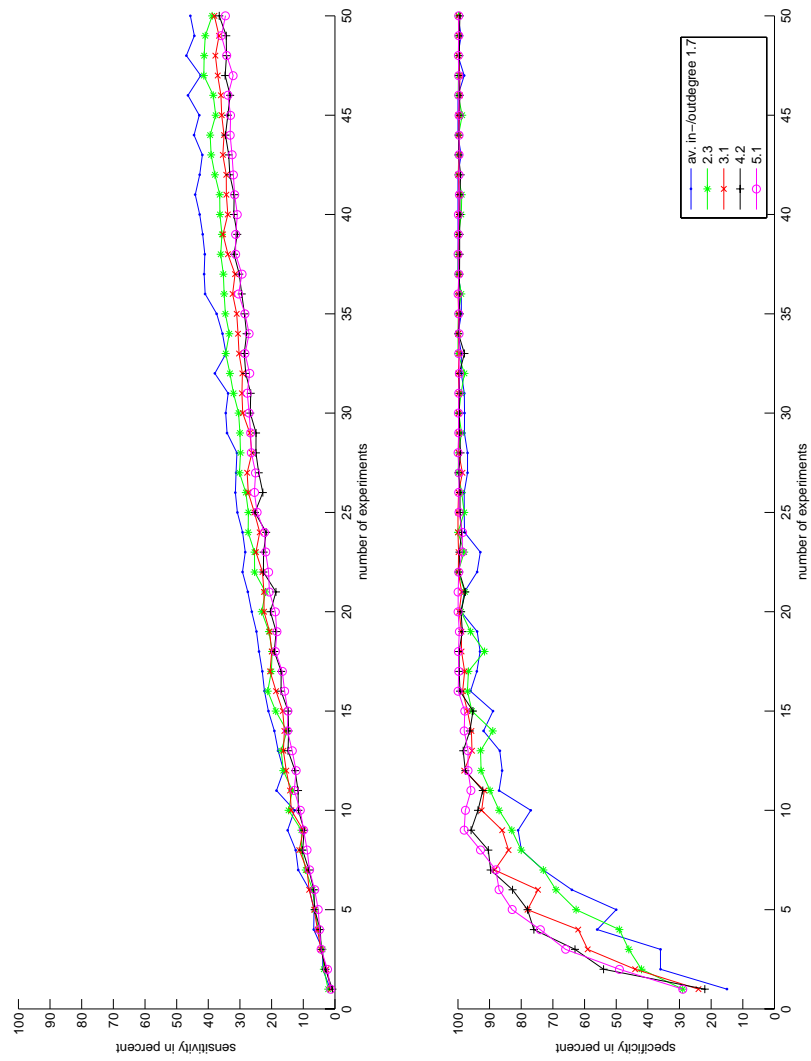


Figure 3.4: Sensitivity and specificity of accessibility list reconstruction of networks with varying experiment count.

Chapter 4

Conclusion

We have introduced a binary network model and designed a simulation framework that generates random networks and their expression data. Using this framework, we have examined the properties of two gene network reconstruction algorithms, one of which was changed to accommodate our data model.

The results suggest that the two algorithms complement each other in their strong and weak points. They also show that the accuracy of any reconstruction attempt depends heavily on the features of the network.

Using our model, we went on to approximate the quality of reconstruction that can be expected when one is working with a given, limited amount of information. The sensitivity of these reconstructions is low, especially for networks with a medium or large numbers of nodes. Although these values can probably be improved by making use of heuristic methods and previously gained knowledge, there is a clear limit to what can be achieved with a given set of expression values.

In [19], Kyoda and Morohashi suggest that a model using continuous values is superior to the more commonly used boolean type. We agree with this sentiment, not least because it might yield more accurate information using tests similar to the ones we performed here. The pure boolean model introduces many artifacts and degenerate cases that should not be a problem when working with real expression data.

Acknowledgments

I would like to extend thanks to my supervisors, Amela Prelic and Stefan Bleuler, and Professor Eckart Zitzler for their patience and advice. I would also like to thank Anja Wille, for a large part of this thesis would not have been possible without the help she offered.

List of Tables

2.1	Sample contingency table	21
3.1	Worst case reconstruction requirements, by Akutsu	36

List of Figures

1.1	Sample microarray	3
1.2	Basic gene interactions	5
1.3	Sample digraph	6
1.4	Graph representations	11
2.1	Digraph rebuilt from accessibility list	16
2.2	Shortcuts in graphs	17
2.3	Effects of strong components	18
2.4	Reconstruction of strong component	19
2.5	Wille special cases	20
2.6	Wagner sensitivity and specificity, error probability	25
2.7	Wagner sensitivity and specificity, node count	26
2.8	Wagner sensitivity and specificity, edge count	28
2.9	Wille sensitivity and specificity, node count	30
2.10	Wille sensitivity and specificity, number of expression data sets	31
2.11	Wille sensitivity and specificity, updates per node	33
2.12	Comparison of Wagner and Wille reconstruction results	34
3.1	Favorable and unfavorable initial states for experiments	39
3.2	Accessibility list reconstruction from experiment	40
3.3	Accessibility list reconstruction, variable node count	41
3.4	Accessibility list reconstruction, variable experiment count	43

Bibliography

- [1] T. Akutsu, S. Kuhara, O. Maruyama, and S. Miyano, *Identification of gene regulatory networks by strategic gene disruptions and gene overexpressions*, Proceedings of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms (1998), 695–702.
- [2] Tatsuya Akutsu, Satoru Kuhara, Osamu Maruyama, and Satoru Miyano, *A system for identifying genetic networks from gene expression patterns produced by gene disruptions and overexpressions*, Genome Inform. (1998).
- [3] Tatsuya Akutsu, Satoru Miyano, and Satoru Kuhara, *Inferring qualitative relations in genetic networks and metabolic pathways*, Bioinformatics **16** (2000), no. 8, 727–734.
- [4] Bruce Alberts, Alexander Johnson, Julian Lewis, Martin Raff, Keith Roberts, and Peter Walter, *Molecular biology of the cell*, 4 ed., Garland Science, 2002.
- [5] Y. Benjamini and Y. Hochberg, *Controlling the false discovery rate: A practical and powerful approach to multiple testing*, J. Roy Stat Soc (1995).
- [6] Tianjiao Chu, Clark Glymour, Richard Scheines, and Peter Spirtes, *A statistical problem for inference to regulatory structure from associations of gene expression measurements with microarrays*, Bioinformatics **19** (2003), no. 9, 1147–1152.
- [7] Hidde de Jong, *Modeling and simulation of genetic regulatory systems: a literature review*, Journal of computational biology **9** (2002), no. 1, 67–103.
- [8] Hidde de Jong, Johannes Geiselmann, Grégory Batt, Céline Hernandez, and Michel Page, *Qualitative simulation of the initiation of sporulation in bacillus subtilis*, Bulletin of mathematical biology **66** (2004), 261–299.
- [9] Hidde de Jong, Johannes Geiselmann, Céline Hernandez, and Michel Page, *Genetic network analyzer: qualitative simulation of genetic regulatory networks*, Bioinformatics **19** (2003), no. 3, 336–344.
- [10] Hidde de Jong, Jean-Luc Gouzé, Céline Hernandez, Michel Page, Tewfik Sari, and Johannes Geiselmann, *Qualitative simulation of genetic regulatory*

networks using piecewise-linear models, Bulletin of Mathematical Biology **66** (2004), 301–340.

- [11] Reinhard Diestel, *Graph theory*, 2 ed., Springer Verlag, 2000.
- [12] D. Edwards, *Introduction to graphical modelling*, 2e ed., Springer Verlag, 2000.
- [13] Ronaldo F. Hashimoto, Seungchan Kim, Ilya Shmulevich, Wei Zhang, Micheal L. Bittner, and Edward R. Dougherty, *Growing genetic regulatory networks from seed genes*, Bioinformatics **20** (2004), no. 8, 1241–1247.
- [14] Dirk Husmeier, *Sensitivity and specificity of inferring genetic regulatory interactions from microarray experiments with dynamic bayesian networks*, Bioinformatics **19** (2003), no. 17, 2271–2282.
- [15] Ivan Iossifov, Michael Krauthammer, Carol Friedman, Vasileios Hatzivas-siloglou, Joel S. Bader, Kevin P. White, and Andrey Rzhetsky, *Probabilistic inference of molecular networks from noisy data sources*, Bioinformatics **20** (2004), no. 8, 1205–1213.
- [16] H. Jeong, B. Tombor, R. Albert, Z.N. Oltvai, and A.-L. Barabási, *The large-scale organization of metabolic networks*, Nature **407** (2000), 651–654.
- [17] S.A. Kauffman, *Metabolic stability and epigenesis in randomly constructed genetic nets*, Journal of theoretical Biology (1969).
- [18] Shinich Kikuchi, Daisuke Tominaga, Masanori Arita, Katsutoshi Takahashi, and Masaru Tomita, *Dynamic modeling of genetic networks using genetic algorithm and s-system*, Bioinformatics **19** (2003), no. 5, 643–650.
- [19] Koji M. Kyoda, Mineo Morohashi, Shuichi Onami, and Hiroaki Kitano, *A gene network inference method from continuous-value gene expression data of wild-type and mutants*, Genome Informatics **11** (2000), 196–204.
- [20] L.V. Lejay, D.E. Shasha, P.M. Palenchar, A.Y. Kouranov, A.A. Cruikshank, M.F. Chou, and G.M. Coruzzi, *Adaptive combinatorial design to explore large experimental spaces: approach and validation*, Systems Biology (2004).
- [21] Linyong Mao and Haluk Resat, *Probabilistic representation of gene regulatory networks*, Bioinformatics **20** (2004), no. 14, 2258–2269.
- [22] Luis Mendoza and Elena R. Alvarez-Buylla, *Dynamics of the genetic regulatory network for arabidopsis thaliana flower morphogenesis*, Journal of theoretical Biology **193** (1998), 307–319.
- [23] Thomas Mestl, Erik Plahte, and Stig W. Omholt, *A mathematical framework for describing and analysing gene regulatory networks*, Journal of theoretical Biology **176** (1995), 291–300.

- [24] Akihiro Nakaya, Susumu Goto, and Minoru Kanehisa, *Extraction of correlated gene clusters by multiple graph comparison*, *Genome Informatics* **12** (2001), 44–53.
- [25] M.E.J. Newman, S.H. Strogatz, and D.H. Watts, *Random graphs with arbitrary degree distributions and their applications*, *Physical Review E* **64** (2001), no. 2.
- [26] Irene M. Ong, Jeremy D. Glasner, and David Page, *Modelling regulatory pathways in e. coli from time series expression profiles*, *Bioinformatics* **18** (2002), no. Suppl. 1, S241–S248.
- [27] Dana Pe’er, Aviv Regev, Gal Elidan, and Nir Friedman, *Inferring sub-networks from perturbed expression profiles*, *Bioinformatics* **17** (2001), no. Suppl. 1, S215–S224.
- [28] Bruno-Edouard Perrin, Liva Ralaivola, Aurélien Mazurie, Samuele Bottani, Jacques Mallet, and Florence d’Alché Buc, *Gene networks inference using dynamic bayesian networks*, *Bioinformatics* **19** (2003), ii138–ii148.
- [29] Jiang Qian, Jimmy Lin, Nicholas M. Luscombe, Haiyuan Yu, and Mark Gerstein, *Prediction of regulatory networks: genome-wide identification of transcription factor targets from gene expression data*, *Bioinformatics* **19** (2003), no. 15, 1917–1926.
- [30] Dmitry A. Rodionov, Inna Dubchak, Adam Arkin, Eric Alm, and Mikhail S. Gelfand, *Reconstruction of regulatory and metabolic pathways in metal-reducing δ -proteobacteria*, *Genome Biology* **5** (2004), no. 11.
- [31] Ilya Shmulevich, Edward R. Dougherty, Seungchan Kim, and Wei Zhang, *Probabilistic boolean networks: a rule-based uncertainty model for gene regulatory networks*, *Bioinformatics* **18** (2002), no. 2, 261–274.
- [32] Dennis E. Shasha, Andrei Y. Kouranov, Laurence V. Lejay, Micheal F. Chou, and Gloria M. Coruzzi, *Using combinatorial design to study regulation by multiple input signals. a tool for parsimony in the post-genomics era*, *Plant Physiology* **127** (2001), 1590–1594.
- [33] Ando Shin and Hitoshi Iba, *Construction of genetic network using evolutionary algorithm and combined fitness function*, *Genome Informatics* **14** (2003), 94–103.
- [34] I. Shmulevich, E.R. Dougherty, and W. Zhang, *From boolean to probabilistic boolean networks as models of genetic regulatory networks*, *Proceedings of the IEEE* **90** (2002), no. 11, 1778–1792.
- [35] I. Shmulevich and W. Zhang, *Binary analysis and optimization-based normalization of gene expression data*, *Bioinformatics* **18** (2002), no. 4.

- [36] I. Shmuulevich, A. Saarinen, O. Yli-Harja, and J. Astola, *Computational and statistical approaches to genomics*, Kluwer Academic Publishers, 2002.
- [37] Paul Smolen, Douglas A. Baxter, and John H. Byrne, *Modeling transcriptional control in gene networks—methods, recent results and future directions*, Bulletin of Mathematical Biology **62** (2000), 247–292.
- [38] Zhengchang Su, Phuongan Dam, Xin Chen, Victor Olman, Tao Jiang, Brian Palenik, and Ying Xu, *Computational inference of regulatory pathways in microbes: an application to phosphorus assimilation pathways in *synechococcus sp. wh8102**, Genome Informatics **14** (2003), 3–13.
- [39] Yoshinori Tamada, SunYong Kim, Hideo Bannai, Seiya Imoto, Kousuke Tashiro, Satoru Kuhara, and Satoru Miyano, *Estimating gene networks from gene expression data by combining bayesian network model with promoter element detection*, Bioinformatics **10** (2003), no. Suppl. 2, ii227–ii236.
- [40] Susannah G. Tringe, Andreas Wagner, and Stephanie W. Ruby, *Enriching for direct regulatory targets in perturbed gene-expression profiles*, Genome Biology **5** (2004), no. 4.
- [41] Andreas Wagner, *How to reconstruct a large genetic network from n gene perturbations in fewer than n^2 easy steps*, Bioinformatics **17** (2001), no. 12, 1183–1197.
- [42] Andreas Wagner, *Reconstructing pathways in large genetic networks from genetic perturbations*, Journal of Computational Biology **11** (2004), no. 1.
- [43] Andreas Wagner and David Fell, *The small world inside large metabolic networks*, Working papers, Santa Fe Institute, July 2000, available at <http://ideas.repec.org/p/wop/safiw/00-07-041.html>.
- [44] D.J. Watts, *The structure and dynamics of small world networks*, Ph.D. thesis, Cornell University, 1997.
- [45] Anja Wille and Philip Zimmermann, *Sparse graphical gaussian modeling of the isoprenoid gene network in *arabidopsis thaliana**, Genome Biology **5** (2004).
- [46] Changwon Yoo and Gregory F. Cooper, *An evaluation of a system that recommends microarray experiments to perform to discover gene-regulation pathways*, Artificial Intelligence in Medicine **31** (2004), 169–182.