Evolutionary Reverse Engineering of Gene Networks

THÈSE N^{O} 4503 (2009)

PRÉSENTÉE LE 9 OCTOBRE 2009

À LA FACULTÉ SCIENCES ET TECHNIQUES DE L'INGÉNIEUR

LABORATOIRE DE SYSTÈMES INTELLIGENTS

PROGRAMME DOCTORAL EN INFORMATIQUE, COMMUNICATIONS ET INFORMATION

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR

Daniel MARBACH

acceptée sur proposition du jury:

Prof. M. Hasler, président du jury
Prof. D. Floreano, directeur de thèse
Prof. S. Bergmann, rapporteur
Prof. F. Naef, rapporteur
Prof. G. A. Stolovitzky, rapporteur

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Suisse
2009

Für Leonardo

Acknowledgments

First, I would like to thank Prof. Dario Floreano, my thesis director, and Claudio Mattiussi, who was a great mentor to me. For my Masters degree, I had specialized in artificial intelligence and robotics. Dario and Claudio encouraged me to discover a new field, systems biology, which is now my great passion. I am particularly grateful to Dario for giving me freedom in my research and for his full support, trust, and encouragement during these four years. I also thank Dario for creating such a great team and stimulating atmosphere at his laboratory. Claudio's guidance and advice were crucial for this project, invaluable for my professional experience, and decisive for my interpretation of probability...

I would like to acknowledge support from the Swiss National Science Foundation (grant 200021–112060) and thank EPFL for providing an outstanding research environment.

I thank Prof. Felix Naef, Prof. Sven Bergmann, Prof. Gustavo Stolovitzky, and Prof. Martin Hasler for having accepted to be members of the thesis committee, and for their time and interest in my work.

I am truly thankful to Prof. Sven Bergmann and Prof. Gustavo Stolovitzky for being very helpful, friendly, and open to collaboration. The many interesting and instructive discussions with Sven strongly influenced my view on systems biology. I owe much to Gustavo for organizing DREAM, for entrusting me with providing the *in silico* challenges, and for fruitful collaboration over the past years.

A big "Thank you!" goes to my colleagues and friends from the Laboratory of Intelligent Systems (even you, hardware people). Their honest feedback and critical comments were invaluable for improving my work. I would like to thank especially Peter Dürr and Claudio Mattiussi for sharing the beauties and worries

of AGE; Thomas Schaffter for working through nights and weekends on GNW; Sara Mitri, my most solicited proofreader and first assistant in the kitchen; Sara Mitri, Peter Dürr, Claudio Mattiussi, and Steffen Wischmann for proofreading parts of this thesis; and all members of the LIS for making it four unforgettable years.

I would like to give special thanks to my brother Fred, my second most solicited proofreader, for many enlightening discussions and for priceless feedback and encouragement. Thanks also to Denise Choinière for proofreading several manuscripts and for her unbounded motivation and friendship.

My biggest thanks go to my parents, Monique and Peter, for their love and unconditional support throughout all these years. Loving thanks also to my other brother, Christian, for making sure that I have a cultural and social life besides my Ph.D. I would also like to thank my friends from Bern for all the turkey diners and parties—spending less time with you was the only downside of these years in Lausanne.

Finally, I thank my wife Melusina for her love, for her understanding, and for being the best partner I could have wished for. I am also thankful to our son Leonardo for sleeping quietly—most of the time—despite (or maybe thanks to) the humming of my computer as I was writing this thesis.

Lausanne, October 2009

Abstract

The expression of genes is controlled by regulatory networks, which perform fundamental information processing and control mechanisms in a cell. Unraveling and modelling these networks will be indispensable to gain a systems-level understanding of biological organisms and genetically related diseases. In this thesis, we present an evolutionary reverse engineering method, which allows to simultaneously infer both the wirings and nonlinear dynamical models of gene regulatory networks from gene expression data. The proposed method reconstructs gene networks by mimicking the natural evolutionary process that constructed them. This is achieved by modelling both the way in which gene networks are encoded in the biological genome, and the different types of mutations and recombinations that drive their evolution, using an artificial genome called Analog Genetic Encoding (AGE). Since AGE mimics the evolutionary forces and constraints that shape biological gene networks, the reconstruction is naturally guided towards biologically plausible solutions. Consequently, the search space is explored more efficiently, and the networks are recovered more reliably, than with alternative methods. We have confirmed the state-of-the-art performance of AGE both *in vivo* (on real gene networks) and *in silico* (on simulated networks). In particular, AGE achieved winning performance in the *in vivo* gene network inference inference challenge of the 2nd DREAM (Dialogue on Reverse Engineering Assessment and Methods) conference, which consisted in predicting the structure of a synthetic-biology gene network in *Saccharomyces cerevisiae* from time-series data.

In vivo performance assessment of network-inference methods is problematic because it is in general not possible to systematically validate predictions, except for few well-characterized gene networks. Consequently, *in silico* benchmarks

are essential to understand the performance of network-inference methods. We have developed tools to generate biologically plausible *in silico* gene networks, which allow realistic performance assessment of network-inference methods. In contrast to previous *in silico* benchmarks, we generate network structures by extracting modules from known gene networks of model organisms, instead of using random graphs. Furthermore, we simulate network dynamics using more realistic kinetic models, which include both mRNA and proteins. We have implemented this framework in an open-source Java tool called *GeneNetWeaver* (GNW). Using GNW we have generated benchmarks for community-wide challenges of the 3rd and 4th DREAM conference (the DREAM *in silico* network challenges). Here, we assess the performance of 29 network-inference methods, which have been applied independently by participating teams of the DREAM3 challenge. Performance profiling on individual network motifs reveals that current inference methods are affected, to various degrees, by three types of systematic prediction errors. We find that these errors are induced by inaccurate prior assumptions of prevalent gene-network models. The evolutionary reverse engineering approach, which would have ranked 3rd in this challenge, can be used with a wide range of nonlinear models. It could thus provide the necessary framework for the development of models that better approximate different types of gene regulation, thereby enabling ever more accurate reconstruction of gene networks.

Keywords: Analog Genetic Encoding (AGE), evolutionary algorithms, gene regulatory networks, pathway inference, reverse engineering, systems biology

Zusammenfassung

Genetische Netzwerke kontrollieren die Aktivierung von Genen und erfüllen damit grundlegende Informationsverarbeitungs- und Kontrollaufgaben in einer Zelle. Das Entschlüsseln und Modellieren dieser Netzwerke, das sogenannte Reverse Engineering, ist eine Voraussetzung zum besseren Verständnis biologischer Organismen und genetisch bedingter Krankheiten auf Systemebene. In dieser Dissertation präsentieren wir eine evolutionäre Reverse Engineering Methode, die es ermöglicht gleichzeitig Verbindungsmuster und nichtlineare dynamische Modelle von genetischen Netzwerken aus Genexpressionsdaten zu rekonstruieren. Die vorgeschlagene Methode rekonstruiert genetische Netzwerke durch Nachahmung jenes natürlichen evolutionären Prozesses, der diese ursprünglich konstruierte. Dies wird mit einem künstlichen Genom, dem sogenannten Analog Genetic Encoding (AGE), erreicht. AGE modelliert sowohl die Art und Weise in welcher Gennetzwerke im biologischen Genom codiert sind, als auch die Mutationen und Rekombinationen, die ihre Evolution vorantreiben. Da AGE die evolutionären Einflüsse und Einschränkungen, welche biologische genetische Netzwerke formen, nachahmt, wird die Rekonstruktion auf natürliche Weise zu biologisch plausiblen Lösungen geführt. Folglich wird der Suchraum effizienter erkundet und die Netzwerke werden zuverlässiger rekonstruiert als mit anderen Methoden. In experimentellen Vergleichen zeigt AGE sowohl *in vivo* (bei der Rekonstruktion echter genetischer Netzwerke) also auch *in silico* (bei der Rekonstruktion simulierter genetischer Netzwerke) hervorragende Resultate. Insbesondere erreichte AGE die beste Leistung im *in vivo* Reverse Engineering Challenge der zweiten DREAM (Dialogue on Reverse Engineering Assessment and Methods) Konferenz. Dieser bestand daraus, ein genetisches Netzwerk von *Saccharomyces Cerevisiae* aus Genexpressionsdaten zu

rekonstruieren.

Die Leistungsbewertung von Reverse Engineering Methoden *in vivo* ist problematisch, da es, ausser für einige wenige gut charakterisierte genetische Netzwerke, nicht möglich ist, die Prognosen systematisch zu verifizieren. Folglich sind *in silico* Tests unumgänglich, um die Leistung von Reverse Engineering Methoden zu beurteilen. Wir haben Tools zur Erzeugung von biologisch plausiblen *in silico* Netzwerken entwickelt, welche eine realistische *in silico* Leistungsbewertung ermöglichen. Im Gegensatz zu früheren *in silico* Tests generieren wir Netzwerkstrukturen, indem wir Module aus bekannten Netzwerken von Modellorganismen extrahieren, statt willkürlich generierte Strukturen zu verwenden. Zudem wird die Netzwerkdynamik mit realistischeren kinetischen Modellen simuliert, welche mRNA und Proteine berücksichtigen. Wir haben dieses Framework im open-source Javatool *GeneNetWeaver* (GNW) implementiert. GNW wurde zur Erzeugung von Benchmark-Tests für die sogenannten "DREAM *in silico* Challenges" der dritten und vierten DREAM Konferenz eingesetzt. In dieser Dissertation bewerten wir die Leistung von 29 Reverse Engineering Methoden, welche von den Teilnehmern des DREAM3 Challenges unabhängig voneinander angewandt wurden. Eine Analyse der Rekonstruktion von elementaren Verbindungsmustern der Netzwerke zeigt, dass gegenwärtige Reverse Engineering Methoden durch drei Arten von systematischen Prädiktionsfehlern beeinträchtigt werden. Wir stellen fest, dass diese Fehler durch inexakte Prämissen von gängigen Reverse Engineering Methoden bedingt sind. Unsere evolutionäre Methode, welche in diesem Challenge das drittbeste Resultat erzielt hätte, kann mit einem breiten Spektrum von nichtlinearen Modellen verwendet werden. Sie könnte deshalb das notwendige Fundament für die Entwicklung von Modellen mit einer besseren Approximation der verschiedenen Arten von Genregulation darstellen und damit eine genauere Rekonstruktion von Gennetzwerken ermöglichen.

Stichwörter: Analog Genetic Encoding (AGE), evolutionäre Algorithmen, genetische Netzwerke, Inferenz, Reverse Engineering, Systembiologie

Preface

One of the exciting aspects of systems biology is that it is a highly interdisciplinary field. During my thesis, I had the chance to interact with researchers from very different backgrounds, which was an extremely enriching experience. However, I have also learned that the gap between researchers from different fields can be wider than one would think. For example, a computer scientist may not know what a gene network is, and a biologist may not know what reverse engineering is. After one of my initial presentations, where I may not have paid sufficient attention to this "gap", a colleague with a background in experimental biology told me: "Wow, this is really great work, very interesting—I didn't understand a thing, though".

Therefore, I wrote this thesis keeping in mind an audience with very different backgrounds. This came at the cost of simplifying some of the descriptions. Whenever possible, I prefer common terms over (potentially more accurate) technical terms. Also, I often use simple diagrams and natural language to explain algorithms, instead of relying on more formal specifications or pseudocode, which may not be familiar to biologists. For readers with little experience in molecular biology, the relevant notions are explained throughout the thesis.

Contents

Acknowledgments **i**

Abstract **iii**

Zusammenfassung **v**

Preface **vii**

Contents **ix**

1 Introduction **1**
1.1 Introduction 2
1.2 Reverse engineering gene networks 3
1.2.1 Gene regulatory networks 3
1.2.2 Reverse engineering 5
1.3 Original contribution 9
1.4 Organization of the thesis 11

2 Evolutionary reverse engineering of gene networks **13**
2.1 Introduction 14
2.2 Evolutionary reverse engineering with AGE 17
2.2.1 Artificial genome 17
2.2.2 Mutation and recombination 22
2.2.3 Evolutionary algorithm 25
2.3 Application to an *in silico* test case 28
2.3.1 The test case 28

2.3.2 Setup of AGE for the test case . . . 30
2.3.3 Results . . . 31
2.4 Conclusion . . . 33

3 Reverse engineering an *in vivo* benchmark network in yeast 37
3.1 Introduction . . . 38
3.2 The log-sigmoid gene network model . . . 39
3.3 Benchmark networks and data . . . 44
3.3.1 Synthetic-biology benchmark . . . 44
3.3.2 *In silico* five-gene repressilator . . . 47
3.4 Reverse engineering method . . . 47
3.4.1 Setup of AGE . . . 47
3.4.2 Predictions and confidence levels . . . 48
3.5 Results . . . 49
3.5.1 Evaluating the accuracy of predictions . . . 49
3.5.2 *In silico* benchmark results . . . 51
3.5.3 Synthetic-biology benchmark results . . . 52
3.6 Conclusion . . . 56

4 Generating realistic *in silico* benchmark networks 59
4.1 Introduction . . . 60
4.2 Results . . . 62
4.2.1 Module extraction from global interaction networks . . . 62
4.2.2 Topological modules correlate with functional modules . . . 64
4.2.3 Module extraction preserves structural properties . . . 67
4.3 Discussion . . . 71
4.4 Methods . . . 72
4.4.1 Subnetwork extraction . . . 72
4.4.2 Identification of enriched functional annotations . . . 74
4.4.3 Network motif significance profiles . . . 74

5 *In silico* performance assessment in a community-wide experiment 75
5.1 Introduction . . . 76
5.2 Framework for assessment of inference methods . . . 78
5.2.1 Generating realistic *in silico* benchmarks . . . 78
5.2.2 The DREAM *in silico* network challenge . . . 80

5.2.3 Evaluating the performance of network inference methods 81
5.3 Results . . . 83
5.3.1 Performance assessment of 29 network inference methods . 83
5.3.2 Network-motif analysis reveals three types of systematic prediction errors . . . 87
5.3.3 Most inference methods fail to accurately predict combinatorial regulation . . . 92
5.3.4 Performance of AGE on the DREAM3 benchmarks . . . 95
5.4 Discussion . . . 96
5.4.1 A word of caution . . . 96
5.4.2 Systematic prediction errors are induced by inaccurate prior assumptions . . . 98
5.4.3 Conclusion . . . 99
5.5 Methods . . . 100
5.5.1 Gene network model . . . 100
5.5.2 Simulation of gene expression data . . . 102
5.5.3 Evaluation of predictions . . . 103
5.5.4 Network-motif analysis . . . 103

6 Wisdom of crowds in gene network inference 107
6.1 Introduction . . . 108
6.1.1 Ensemble methods . . . 109
6.1.2 A probabilistic formalization . . . 110
6.2 Methods for combining ensembles of inferred networks . . . 112
6.2.1 Selecting the "best" network from the ensemble . . . 112
6.2.2 Analysis of the posterior weight distributions . . . 112
6.2.3 Majority voting on the network structure . . . 113
6.2.4 Signed voting on the network structure . . . 114
6.3 Ensemble predictions from multiple runs of AGE . . . 115
6.3.1 Constructing the ensembles . . . 115
6.3.2 Combining the ensembles and evaluating the predictions . 115
6.3.3 Ensemble voting outperforms individual networks in the presence of noise . . . 116
6.3.4 Ensemble voting achieves best performance in the DREAM2 synthetic-biology network challenge . . . 120

6.4 Ensemble predictions from variations of the data 123
6.5 Predictions from a community of methods 124
6.5.1 Forming community predictions 126
6.5.2 Community predictions are more robust than individual predictions 128
6.6 Conclusion 130

7 Discussion and outlook **133**
7.1 Accomplished work 134
7.1.1 Evolutionary reverse engineering of gene networks 134
7.1.2 Critical performance assessment of inference methods . . . 135
7.2 Outlook 136
7.2.1 Current limitations of AGE 136
7.2.2 Reverse engineering heterogeneous networks 137
7.2.3 Realistic *in silico* performance assessment: DREAM or reality? 139
7.3 Conclusion 140

A Supplementary information of Chapter 2 **143**
A.1 Implementation of the implicit encoding 143
A.1.1 The interaction map 144

B Supplementary information of Chapter 3 **147**
B.1 Elementary two-dimensional input functions 147
B.2 Five-gene repressilator 151

Bibliography **153**

Curriculum vitæ **167**

1 Introduction

Synopsis

In this chapter, we frame the problem of gene-network reverse engineering. We discuss advantages and limitations of state-of-the-art methods for gene-network reverse engineering, and summarize how this thesis contributes to overcome some of these limitations. Finally, we give an overview of the structure of the thesis and situate the subject of each chapter in the global context.

1.1 Introduction

Over the past decades, molecular biology has been extremely successful at identifying the functional components of cells. The genomes of many organisms have been sequenced and, at least for model organisms, the majority of genes, RNA transcripts, proteins, metabolites, etc., are known and have been catalogued in dedicated databases. However, despite extensive knowledge of individual parts, we are far from understanding how cells work, and how their functioning could be easily manipulated or fixed in the case of disease (Cardelli, 2005). Lazebnik (2002) illustrates this apparent paradox with a humorous touch by describing how biologists would try to understand and repair a radio:

> We would eventually find how to open the radios and will find objects of various shape, color, and size. We would describe and classify them into families according to their appearance. We would describe a family of square metal objects, a family of round brightly colored objects with two legs, round-shaped objects with three legs and so on. [...] Inspired by these findings, an army of biologists will apply the knockout approach to investigate the role of each and every component. [...] Eventually, all components will be cataloged, connections between them will be described, and the consequences of removing each component or their combinations will be documented. This will be the time when the question, previously obscured by the excitement of productive research, would have to be asked: Can the information that we accumulated help us to repair the radio? (Lazebnik, 2002)

Clearly, characterizing individual components and connections is valuable, but not sufficient to understand the functioning of a radio, or of a biological cell. Instead, to gain a systems-level understanding, we also need to examine how the components dynamically interact during operation (Kitano, 2002). Or, as Sauer et al. (2007) put it:

> The reductionist approach has successfully identified most of the components and many interactions but, unfortunately, offers no convincing concepts and methods to comprehend how system properties emerge. [...] the pluralism of causes and effects in biological networks is better addressed by observing, through quantitative mea-

> sures, multiple components simultaneously, and by rigorous data integration with mathematical models. (Sauer et al., 2007)

Though this may not be a new insight, the necessary technology to quantitatively measure activities of cellular components and to enable a systems approach in biology has only recently become available, thus giving rise to the field of *systems biology* (Kitano, 2002; Pennisi, 2003).

The work described in this thesis can be situated in the field of computational systems biology. It is our goal to develop computational tools to unravel and model biological systems based on quantitative experimental data. Specifically, we consider the problem of unraveling and modeling gene regulatory networks. In the rest of this chapter, we will frame this problem and outline the contributions of this thesis in more detail.

1.2 Reverse engineering gene networks

1.2.1 Gene regulatory networks

The expression (activity) of genes in a biological cell is controlled by gene regulatory networks (also called *genetic* regulatory networks, or simply *gene networks*). Gene regulatory networks are composed of a collection of genes, which may interact through their RNA and protein products as illustrated in Figure 1.1. For example, a gene may code for a protein that can bind to the DNA (a so-called *transcription factor*), thereby activating or inhibiting the expression of other genes (transcriptional regulation).

Besides transcriptional regulation, there are a variety of other important mechanisms that are employed by cells to regulate the expression of genes, working at different levels and time scales. At the level of RNA, the transcript of a gene may be processed in different ways (Ladd and Cooper, 2002) or silenced by microRNAs (Ambros and Chen, 2007), for instance. Proteins may also interact with each other or with signaling molecules, thereby activating, suppressing, or modifying their functionality. Gene regulatory networks are complex circuits of such diverse regulatory interactions.

Depending on the external environment and internal state of a cell, different genes need to be expressed more or less strongly to ensure its "optimal" functioning. For example, the *lac* operon (an operon is a sequence of several

genes that are transcribed together) contains genes that allow *Escherichia coli* to use the sugar lactose for growth. However, *E. coli* prefers the sugar glucose, if available. Thus, the cell uses signals indicating the levels of these two sugars as inputs to control the expression of the *lac* operon in such a way to optimize the cell growth rate (Setty et al., 2003; Dekel and Alon, 2005). In multicellular organisms, gene regulatory networks also control processes such as development, growth, and differentiation of cells (Yuh et al., 1998). For example, since all cells of a multicellular organism have the same genome, different cell types are defined through differential regulation of genes. Gene regulatory networks are thus the basic control and computational systems of cells (Bray, 1995), and

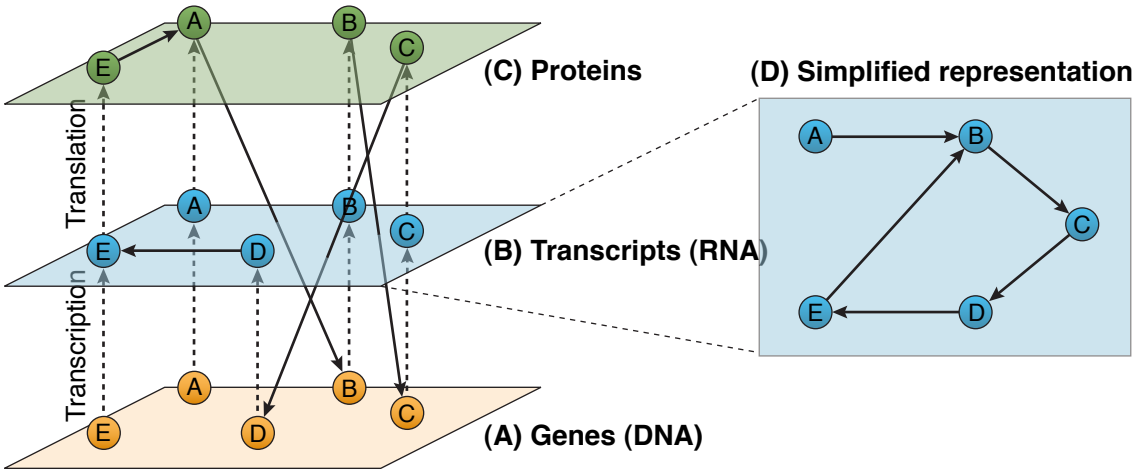


Figure 1.1: Illustrative example of a gene regulatory network. **(A)** Genes are transcribed to mRNA. The transcription rate may be regulated by proteins (incoming arrows). **(B)** mRNA transcripts are processed and corresponding proteins are synthesized (translation). Some RNAs have a regulatory function and are not translated, e.g., transcript D is a microRNA that silences transcript E. **(C)** Regulatory proteins (transcription factors) bind to the DNA and enhance or inhibit the expression of genes. Proteins also interact among each other, e.g., protein E binds to protein A, thereby changing the regulatory effect of protein A on gene B. **(D)** Simplified representation commonly used in gene network inference when only mRNA levels are measured. Each gene is represented by a single node that represents its mRNA transcript. The links are regulatory influences between the transcripts. Note that protein-protein interactions can not be observed at this level. Thus, the interaction between proteins A and E would be observed only indirectly via its regulatory effect on transcript B, leading to links A→B and E→B in the simplified representation.

understanding them is a necessary first step to understand cells at a system level.

Gene networks are often represented in (even more) simplified form as graphs, where each gene is represented by a single node, and the links are regulatory influences between the genes. In gene-network reverse engineering (see next section), gene expression levels are often measured in terms of mRNA concentrations. In this case, the nodes are associated with the mRNAs of the genes, and the regulatory influences can be thought of as the "projection" of the different types of regulatory interactions onto the "RNA space", as shown in Figure 1.1D.

We have given here only a very brief and simplified description of gene regulatory networks, as excellent textbooks and review articles exist on this topic (Bolouri and Davidson, 2002; Bower and Bolouri, 2004; Browning and Busby, 2004). Throughout this thesis, we will discuss several examples of gene networks and their properties in more detail.

1.2.2 Reverse engineering

Reverse engineering can be defined as the process of figuring out the design of a system by studying its structure, function, and operation. The goal of reverse engineering is typically to understand the target system to the point where it can be rebuilt (copied) or reengineered (modified).

Reverse engineering has a long history in traditional engineering disciplines (Ljung, 1999), mostly to understand competitor's products. For example, during the second world war, the Soviets reverse engineered the American B-29 strategic bomber bolt-by-bolt, after three such planes had to do emergency landings in the USSR after missions over Japan. Fridlyander, a Russian engineer who worked on reverse engineering the aluminum alloys of the B-29, writes in his memoir:

> We were US allies, and the American pilots expected a warm welcome. However, I.V. Stalin interned the pilots without informing the United States and ordered a replica of the B-29 to be made; any changes could be introduced only upon his authorization. [...] There were very many difficulties, especially with regard to 30-meter plates for the wings, but nobody dared to appeal to Stalin. In three years, Soviet metallurgists and designers managed to manufacture 850 Tu-4 aircraft, which were replicas of the B-29 (such rates are impossible in the early 21st century). (Fridlyander, 2008)

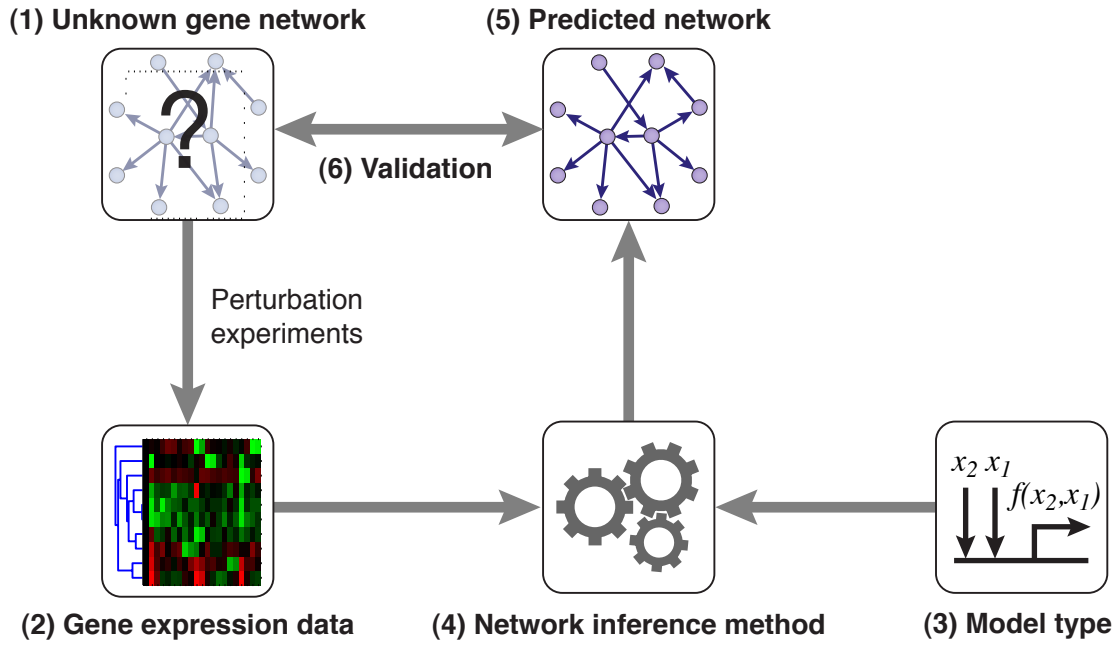


Figure 1.2: Reverse engineering of gene networks. **(1)** Gene network of unknown structure (the so-called *target network*). **(2)** Gene expression levels are measured after applying different types of perturbations to the network. **(3)** A modeling framework (model type) for the gene network needs to be defined. **(4)** The inference method predicts one or several networks that are consistent with the available gene expression data. **(5)** Depending on the model type, only the structure or also a quantitative model of the network can be inferred. **(6)** The predicted gene network is validated with additional experiments.

Fortunately, though bombers may be built at a lower rate than under Stalin, biological systems are being reverse engineered at an unprecedented rate. Note that according to the definition given above, the whole field of systems biology may be considered reverse engineering (Csete and Doyle, 2002). However, the term is often used in a more restrictive sense, to designate methods that unravel (infer) biological networks based on measurements of the activity levels of the network components. The goal of these reverse engineering methods (also called *inference methods*) is to identify the regulatory links of the network, and possibly to infer a quantitative model that can be used to predict the network dynamics. Practical applications of biological network inference may have a strong impact on biotech and pharmaceutical industries, potentially setting the stage for rational redesign of living systems and predictive, model-based drug design (Gardner et al., 2003).

Figure 1.2 illustrates the principle of gene-network reverse engineering. Technologies to assay gene expression levels in terms of mRNA or protein concentrations are advancing at a fast pace. Using oligonucleotide chips or quantitative PCR for instance, it is possible to probe a set of genes of interest that are part of an uncharacterized gene network (the *target network*) under different conditions. Two types of gene expression data are often used: time series and/or steady state. Time series data is more difficult and expensive to obtain, but may contain more information as it shows the dynamics of the network. Steady state data shows the state of the network at a time where it is assumed to be in equilibrium. Time series and/or steady state gene expression levels are typically measured after applying different perturbations to the network. In the following chapters, we will discuss different types of perturbations and technologies to measure gene expression levels in more detail.

The choice of a suitable inference method depends both on the type of available gene expression data, and on the type of model used to describe the target network. Here, we give only a brief overview of the three main approaches currently used in gene-network reverse engineering. The relevant state of the art will be discussed more extensively in the introduction of each chapter.

Statistical approaches

If a gene A regulates a gene B, their expression profiles tend to be correlated (statistically dependent). Statistical approaches predict regulatory interactions between genes based on measures of similarity of their gene expression profiles, for example conditional correlation (Rice et al., 2005) or partial correlation (de la Fuente et al., 2004; Baralla et al., 2009).

A more general measure of similarity than correlation is mutual information. Inferring regulatory interactions based on mutual information was first proposed by Butte et al. (2000). Recently, several methods were proposed that extend this approach by using different strategies to statistically evaluate whether the given values of mutual information are significant, and whether they correspond to causal interactions (Basso et al., 2005; Faith et al., 2007; Watkinson et al., 2009). An excellent introduction to information-theoretic approaches in gene network inference is given in the review of Anastassiou (2007).

In contrast to the approaches described in the next sections, statistical methods do *not* require the definition of a detailed probabilistic or dynamical model

of the gene network. This is an advantage, because the choice of the "right" model type (step 3 in Figure 1.2) can be difficult in practice, as we will see later in this thesis. Furthermore, statistical methods are scalable to large networks with thousands of genes, and they require less gene expression data than other reverse engineering methods. This comes at the cost that only the network structure is being inferred, and not a quantitative model that could be used to simulate the network and predict its response for different conditions, for instance.

Bayesian networks

Bayesian networks are probabilistic graphical models. Within this framework, gene expression levels are represented as random variables X_i, which are the nodes of the network. The edges of the network represent direct dependencies between the random variables, i.e., direct regulatory interactions. Each gene i has an associated conditional probability distribution $P(X_i|R_i)$, where $R_i = \{X_j, X_k, ...\}$ is the set of regulators of gene i (also called the parents of X_i in the Bayesian network). Thus, inference of Bayesian networks amounts to finding the direct dependencies between the genes and the corresponding conditional probability distributions that are most likely, given the measured gene expression data.

Originally proposed by Friedman et al. (2000), Bayesian networks have become a very popular approach for gene network inference (Friedman, 2004; Airoldi, 2007; Needham et al., 2007). A major limitation is that Bayesian networks are acyclic graphs, i.e., they can't represent feedback loops in the networks. This limitation can be overcome by using dynamic Bayesian networks (Nachman et al., 2004), however, these in turn have the limitation that they can only be used with time series data.

Dynamical models

A large class of inference methods model gene regulatory networks as dynamical systems, typically using ordinary differential equations (ODEs). Within this framework, the dynamics of gene i are described by the equation

$$\frac{dx_i}{dt} = f(\mathbf{x}) \tag{1.1}$$

where x_i is the expression level of gene i and $\mathbf{x}$ is the state vector containing the expression levels of all genes. There are two main types of inference methods for dynamical models, methods based on linear models (i.e., the function $f(\cdot)$ is linear) and methods based on nonlinear models.

The linear model, which is a first-order approximation of gene expression dynamics, is the most widely used dynamical model for gene network inference (D'Haeseleer et al., 1999; de la Fuente et al., 2002; Kholodenko et al., 2002; Yeung et al., 2002; Gardner et al., 2003; Brazhnik, 2005). Its main advantage is that it can be inferred analytically using multivariate regression and other standard techniques of system identification (Ljung, 1999). However, gene regulation is known to be strongly nonlinear. Hence, the linearization is generally only valid in a small regime, i.e., close to a specific steady state.

Some nonlinear models (those that are linear in the parameters) can also be inferred using multivariate regression (Weaver, 1999; Chen et al., 2005; Gupta et al., 2005; Bonneau et al., 2006). However, in general, nonlinear models can't be inferred analytically and numerical optimization methods have to be employed (Mjolsness et al., 1991; Reinitz and Sharp, 1995; Moles et al., 2003; Jaeger et al., 2004b; Perkins et al., 2006; Bongard and Lipson, 2007). The evolutionary reverse engineering method proposed in this thesis aims at inference of such nonlinear models. Thus, we will discuss reverse engineering methods based on dynamical models in more detail in the following chapters.

1.3 Original contribution

The contribution of this thesis is two-fold. First, we present novel approaches and methods for **modeling and reverse engineering of gene networks**. Second, we introduce a framework for ***in silico* performance assessment of reverse engineering methods**. Specifically, the main contributions of this thesis are

- an evolutionary reverse engineering method for gene networks, which embeds prior biological knowledge in the reconstruction process by mimicking the way in which gene networks are thought to evolve in nature (Chapter 2);

- validation of the state-of-the-art performance of the evolutionary reverse engineering method on *in silico* and *in vivo* benchmarks of an international

reverse engineering challenge (Chapters 3 and 5);

- a log-sigmoid model of gene networks that provides a better approximation to different types of gene regulation than existing models used in gene network inference (Chapter 3);

- tools to generate realistic *in silico* benchmarks for performance assessment of inference methods and their application to create a community-wide reverse engineering challenge (the DREAM3 *in silico* challenge, Chapters 4 and 5);

- methods to evaluate the performance and identify systematic errors of reverse engineering methods (Chapter 5)

- insight into the capabilities and limitations of prevalent reverse engineering methods through analysis of 29 methods that were applied by participating teams of the DREAM3 *in silico* challenge (Chapter 5);

- an exploration of *ensemble methods* for gene network inference, which improve the accuracy of predictions by combining multiple inferred networks, instead of relying on individual inferred networks (Chapter 6).

Author contributions

At the beginning of each chapter, the publication(s) on which it is based are indicated. All authors of these publications have contributed to the design of the methods and experiments of the respective chapter, and have discussed the results and implications. I have implemented the methods and performed the experiments, with the following exceptions. Robert Prill has implemented and performed the overall performance evaluation of predictions in the DREAM3 *in silico* challenge, whereas I have implemented and performed the network-motif performance evaluation (Chapter 5). Gustavo Stolovitzky has organized the DREAM3 challenge (Chapter 5). Thomas Schaffter has significantly contributed to the implementation of a Java tool for the generation of *in silico* benchmarks (Chapters 4 and 5), and has performed the community analysis described in Section 6.5.2.

1.4 Organization of the thesis

The dissertation is organized in seven chapters, including the introduction (Chapter 1) and the conclusion (Chapter 7). At the beginning of each chapter, we indicate the publication(s) on which it is based, and we include a synopsis, which situates the chapter in the context of the thesis and summarizes its content.

- **Chapter 2** ***Evolutionary reverse engineering of gene networks*** This chapter introduces the evolutionary reverse engineering approach and its realization using Analog Genetic Encoding (AGE). We demonstrate the application of AGE on an *in silico* test case, which shows that—given sufficient gene expression data—a near-perfect reconstruction of the target network can be obtained.

- **Chapter 3** ***Reverse engineering an in-vivo benchmark network in yeast*** After the promising results on the *in silico* test case of Chapter 2, we confirm the state-of-the-art performance of AGE on an *in vivo* benchmark that was released as an international gene network inference challenge of the DREAM2 conference. Furthermore, we introduce a log-sigmoid model, and show that it provides a better approximation to different types of transcriptional regulation than a standard sigmoid model.

- **Chapter 4** ***Generating realistic in-silico benchmark networks*** In this chapter, we present a method for generating biologically plausible *in silico* networks, which allow realistic performance assessment of network-inference algorithms. The method is based on the extraction of modules from known gene networks. Using the yeast transcriptional regulatory network as a test case, we show that extracted modules have a biologically plausible connectivity because they preserve functional and structural properties of the original network.

- **Chapter 5** ***In-silico performance assessment in a community-wide experiment*** We present an *in silico* benchmark suite that we released as a reverse engineering challenge for the DREAM3 conference. We assess the performance of 29 methods, which have been independently applied by different teams. Performance profiling on individual network motifs reveals that inference methods are affected, to various degrees, by three types

of systematic prediction errors. The competitive performance of AGE is confirmed.

- **Chapter 6** ***Wisdom of crowds in gene network inference*** In this chapter, we consider the problem of combining the information contained within multiple inferred networks in order to (1) make more accurate network predictions and (2) estimate the reliability of these predictions. The potential of considering ensembles of networks, rather than individual inferred networks, is demonstrated by showing how ensemble approaches achieve the best performance both in the DREAM2 and DREAM3 challenges.

- **Appendices** The two appendices contain supplementary information for Chapters 2 and 3.

2 Evolutionary reverse engineering of gene networks

This chapter is based on the following publications:

- Marbach, D., Mattiussi, C., and Floreano, D. (2007). Bio-mimetic evolutionary reverse engineering of genetic regulatory networks. In *Proceedings of the 5th European Conference on Evolutionary Computation, Machine Learning and Data Mining in Bioinformatics (EvoBIO 2007)*, pp. 155–165.
- Mattiussi, C., Marbach, D., Dürr, P., and Floreano, D. (2008). The age of analog networks. *AI Magazine*, 29(3) pp. 63–76.
- Marbach, D., Mattiussi, C., and Floreano, D. (2009). Replaying the evolutionary tape: Biomimetic reverse engineering of gene networks. *Annals of the New York Academy of Sciences*, 1158 pp. 234–245.

Synopsis

In this chapter, we present a new approach for reverse engineering gene regulatory networks, which consists of using a reconstruction process that is similar to the evolutionary process that created these networks. The aim is to integrate prior knowledge into the reverse engineering procedure, thus biasing the search towards biologically plausible solutions. To this end, we propose an evolutionary method based on an artificial genome called Analog Genetic Encoding (AGE), which abstracts and mimics the natural evolution of gene regulatory networks. We demonstrate the application of AGE on an in-silico test case. The results of this chapter indicate that, given sufficient gene expression data, AGE provides a perfect reconstruction of the target network.

2.1 Introduction

Conceptually, there are three basic entities involved in the reverse engineering procedure (Ljung, 1999). In the case of gene network reverse engineering, these entities are: 1) A dataset of gene expression measurements; 2) A mathematical model of gene regulation; 3) A search method that can find, within the framework of the model, the networks that are most probable given the dataset and possibly some prior knowledge. These three aspects must be balanced for effective reverse engineering.

The quantity and quality of available data strongly influences the choice of a suitable model type and reverse engineering method. For example, microarrays simultaneously assess the expression of thousands of genes. This results in an extremely high-dimensional search space. For such large-scale reverse engineering, statistical methods (Basso et al., 2005; Faith et al., 2007) or regression techniques relying on relatively simple dynamical models based on first-order approximations of gene expression dynamics (Gupta et al., 2005; di Bernardo et al., 2005) are typically used.

In this thesis, we aim at inference of small networks comprising only dozens of genes (i.e., small modules of the complete gene network of an organism), but using more accurate and biologically plausible, nonlinear gene network models. Microarray data is not well suited for this purpose. More selective and accurate measurement of the expression level of single genes is possible using quantitative Polymerase Chain Reaction (q-PCR, see next chapter) or fluorescent transcriptional reporters, for instance. These technologies are advancing at a fast pace. For example, Zaslaver et al. (2006) have constructed a library of transcriptional reporters for the majority of promoters in *E.coli*, which allows the observation of gene expression rates *in vivo* with a temporal resolution of minutes. As both the quantity and quality of the available gene expression data improve, we can aim at a more faithful reconstruction of gene networks than what is possible with the statistical methods and linear models mentioned above.

More sophisticated, nonlinear model types require the conception of adequate reverse engineering algorithms that can navigate the more intricate search space. Here, we propose to take inspiration from the evolutionary mechanisms that have led to the emergence of complex, nonlinear gene regulatory networks in nature, to design reverse engineering methods for these same networks. We

present a biomimetic reverse engineering method that uses *in silico* evolution to explore the search space in a similar way as gene networks are thought to evolve in nature. In contrast to standard evolutionary or genetic algorithms (Bäck et al., 2000; Floreano and Mattiussi, 2008), we model the way in which gene networks are encoded in the biological genome, and the different types of mutations and recombinations that drive their evolution, using a biomimetic artificial genome called Analog Genetic Encoding (AGE) (Mattiussi, 2005; Mattiussi and Floreano, 2007).

AGE is compatible with a wide range of nonlinear gene network models, and it permits simultaneous inference of both the network structure and the numerical parameter values of these models. More abstract, artificial genomes inspired from gene regulatory networks have previously been used, mainly for studying evolutionary network dynamics (Reil, 1999; Bongard, 2002; Watson et al., 2004; Hintze and Adami, 2008). Unlike AGE, these artificial genomes were not designed as tools for the evolution of dynamical models with real-valued parameters, and they are not suited for reverse engineering of gene networks.

Making effective use of prior knowledge (prior information) is crucial in any inference problem (Jaynes, 1984), but it is particularly important in gene network inference (van Someren et al., 2003). This is because gene expression data is typically noisy and limited, leaving the inference problem underdetermined. In other words, typically many different networks are consistent with the available gene expression data, and prior knowledge must be used to choose the most plausible among them. Since gene networks are known to be sparsely connected, most reverse engineering methods use explicit constraints to favor sparse networks in the reconstruction process. This is typically done by imposing a maximum number of regulators per gene (Gardner et al., 2003), iteratively setting weak connections to zero (Wahde and Hertz, 2001), choosing the sparsest among all possible solutions (Yeung et al., 2002; Gupta et al., 2005), or formulating explicit constraints that punish connections (van Someren et al., 2003; Kimura et al., 2005; Bonneau et al., 2006). However, the target network does not necessarily correspond to the most parsimonious solution, and posing explicit constraints that limit or punish connections is problematic because *a priori*, one cannot be sure how densely connected an unknown gene network is.

Here, we advocate a new approach for embedding prior knowledge in a reverse engineering method. Instead of formulating *ad hoc* constraints, we employ

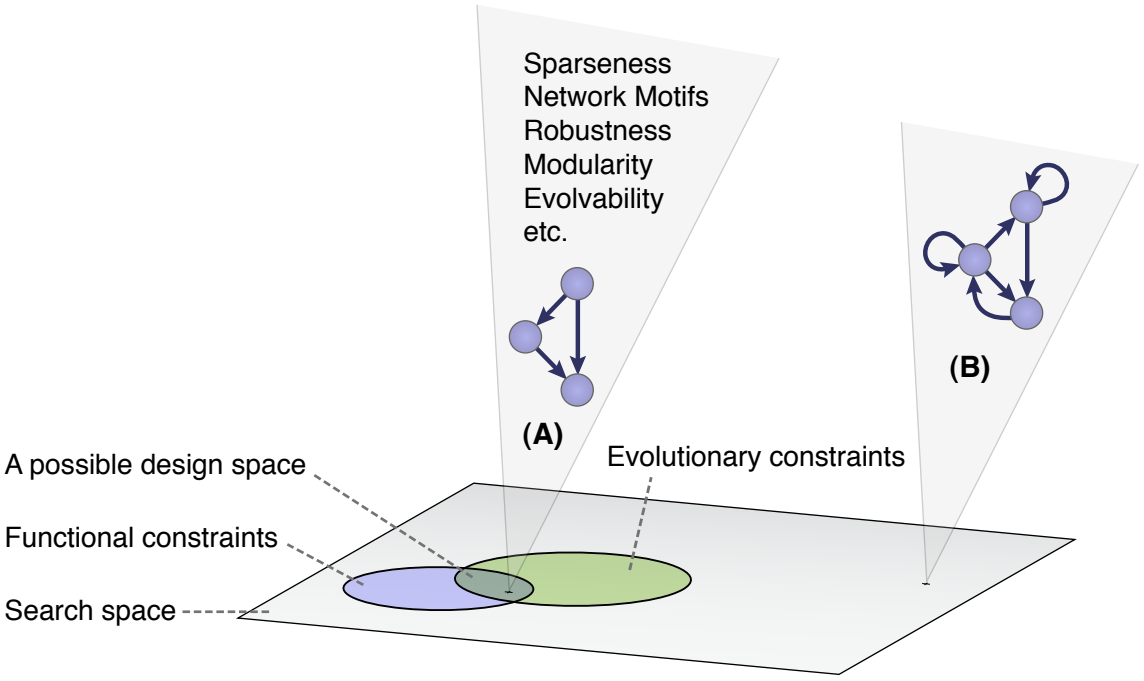


Figure 2.1: Evolutionary and functional constraints shape the *design space* of gene networks (Kitano, 2007), which is characterized by "design principles" such as sparseness, robustness, modularity, etc. (Alon, 2007a). In other words, from the space of all possible networks (here represented as a plane), only a subset are likely to be found by natural evolution. For example, the connectivity pattern (A) may be easier to create by natural evolution than the connectivity pattern (B). It would thus be more likely to occur in a natural gene network. By "replaying the evolutionary tape", the biomimetic reverse engineering approach aims at reproducing the evolutionary constraints of biological networks, thereby partly biasing the search towards nature's design space.

a search method that bears close similarity with the *design method* of the reverse engineering target.[1] In the case of gene regulatory networks, the design method is an evolutionary process that—like any other design method—implies specific constraints on how the search space is explored. By reproducing, at a certain level of abstraction, the evolutionary constraints of biological gene networks, the reverse engineering process can be biased towards biologically plausible solutions, thereby improving the accuracy of predictions (Figure 2.1).

In the next section, we describe the evolutionary reverse engineering method based on AGE. We proceed by demonstrating its application using an *in silico*

[1]We use the term *design method* to refer to the process that created the target system without implying that there is an intelligent designer involved…

test case. The results reported in this chapter indicate that, given enough gene expression data, AGE provides a perfect reconstruction of the target network. In the following chapters, we will consider the more common scenario where the data is not sufficient to uniquely determine the target network.

2.2 Evolutionary reverse engineering with AGE

The first step in the reverse engineering process generally consists in the choice of a gene network model type (e.g., the sigmoid model that will be introduced in Section 2.3.1). As mentioned above, AGE is not constrained to a specific model type, but can be used with a large class of nonlinear models termed *analog networks* (Mattiussi et al., 2008). An analog network is composed of a collection of devices connected by links of different strengths. Here, devices are genes and links correspond to regulatory interactions[2]. Without limiting ourselves to a specific model type, we assume that the gene network can be modeled as a nonlinear dynamical system, where genes are characterized by a vector of internal parameters $\mathbf{p}$ (e.g., decay rate, maximum transcription rate, etc.), and regulatory links have a single parameter called weight w (the interaction strength). Within this framework, reverse engineering requires the specification of the network structure (size, topology) and the specification of the numerical values of all gene parameter vectors $\mathbf{p}$ and connection weights w.

Using AGE, we encode both the network structure and the values of all numerical parameters in a biomimetic artificial genome similarly to the way biological gene networks are encoded in the genome in nature. The reverse engineering process then amounts to the artificial evolution of gene networks that best match the available gene expression data, as described in the following sections.

2.2.1 Artificial genome

We will now describe how networks are represented (encoded) in the AGE genome. We would like to stress that the goal is not building a detailed model of the workings of gene networks, but abstracting only key features believed to be

[2]In this thesis, we consider the simplest case where all devices are of the same type, but AGE can also handle heterogeneous networks (Mattiussi et al., 2008). We plan to use several device types in the future, e.g., for more accurate inference of gene-protein networks (see Section 7.2.2).

important in their evolution. Specifically, we model three fundamental aspects of the biological encoding, which are illustrated in Figure 2.2 and described in detail below. Note that conventional genomes used in genetic algorithms typically only capture the first of these three aspects.

1. **The genome is a sequence of nucleotides**. The AGE genome is constituted by one or several chromosomes. A chromosome is a finite and nonempty sequence of characters drawn from a genetic alphabet. The genetic alphabet of the biological genome contains four letters, namely the nucleotide bases adenine (A), guanine (G), cytosine (C), and thymine (T). *In silico*, we are not limited by the substrate of DNA and are thus free to use another genetic alphabet. In the experiments reported here, we use a 26 letter alphabet consisting of the characters (`A`–`Z`).[3]

2. **Genes can be located anywhere in the genome**. In nature, the beginning and the end of genes are marked by signals encoded in the DNA (promoters and terminators). Analogously, we use special nucleotide patterns (`GN` and `TE`) termed *tokens* to delimit genes in the artificial genome as illustrated in Figure 2.2. Consequently, genes may be located anywhere in the genome. Sequences that are not part of a gene are non-coding.

   Since the position of genes in the genome is not fixed, they may be moved or copied to other positions. In other words, the genome does not have a fixed structure, but can be reorganized by mutations, as described in the next section. In particular, the number of genes (i.e., the size of the gene network), but also the number of chromosomes and their length can change in evolutionary time.

3. **Implicit encoding of regulatory interactions**. In a cell, the potential regulatory interaction between two genes A and B is not encoded *explicitly* in

[3]Note that the exact number of letters in the genetic alphabet is not a critical choice. A discussion of the advantages and disadvantages of large versus small genetic alphabets is given by Mattiussi (2005). Here, we just note that the same information can be encoded in a shorter genome sequence when using a larger genetic alphabet, which has practical and computational advantages. Furthermore, by using a different alphabet we clearly distinguish the artificial genome from real genome sequences and emphasize that we are not trying to build a model that is as faithful as possible to the biological encoding, but rather a model that abstracts only the functionally relevant concepts.

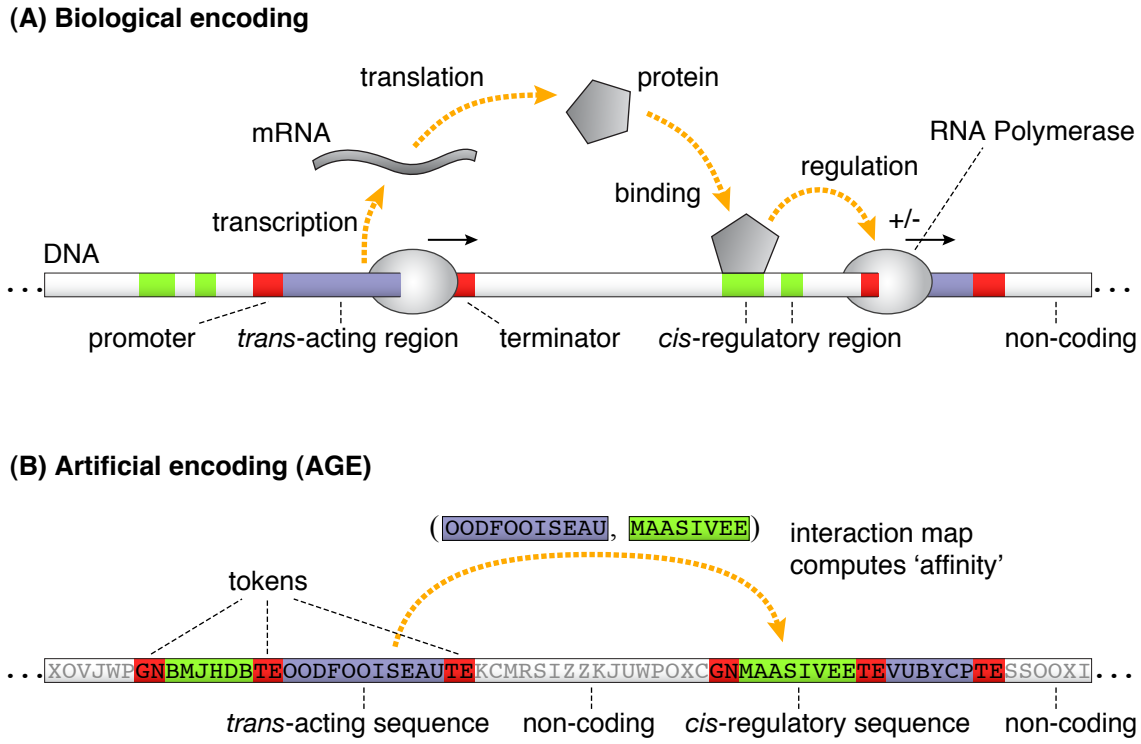


Figure 2.2: Implicit encoding of genetic interactions in the biological and the artificial genome. **(A)** In a cell, transcriptional regulatory interaction between two genes is the result of a biochemical process that depends among other things on the protein-coding region (*trans*-acting region) of the first gene, which defines the characteristics of the regulatory protein, and the *cis*-regulatory region of the second gene, which contains the potential binding sites for the regulatory protein. **(B)** AGE abstracts the following three aspects of the biological encoding: 1. *The genome is a sequence of nucleotides*. The artificial genome is constituted by one or more chromosomes, which are sequences of characters (`A`–`Z`). 2. *Genes can be located anywhere in the genome*. The beginning and the end of genes are marked by special motifs called *tokens* (`GN` and `TE`) analogous to promoters and terminators of biological genes. 3. *Implicit encoding of regulatory interactions*. The potential regulatory interaction between two genes is not encoded *explicitly* in the genome. Instead, genes have a *cis*-regulatory sequence and a *trans*-acting sequence, which may interact via an interaction map that computes an "affinity" (interaction strength). The interaction map abstracts the complex biochemical processes of transcription and translation illustrated on top.

the genome, but follows *implicitly* from a biochemical process that depends among other things on: (1) the protein-coding region (*trans*-acting region) of gene A, which defines the characteristics of protein A, and (2) the *cis*-regulatory region of gene B, which contains the potential binding sites for the regulatory protein (Figure 2.2A; note that this is a simplified representation of transcriptional regulation, other mechanisms of gene regulation will be discussed in Chapter 7). Thus, the strength of the interaction is implicitly encoded by the respective *cis*- and *trans*-acting sequences. Hypothetically, if we had unlimited understanding of all involved steps (transcription, translation, protein folding, etc.), we could define a quantitative model $\mathcal{I}(s_{\text{trans}}, s_{\text{cis}})$, which we call *interaction map*, that would predict the regulatory effect of gene A on gene B directly from the respective protein-coding sequence s_{trans} and *cis*-regulatory sequence s_{cis}.

Similarly, artificial genes in AGE have *cis*-regulatory sequences and *trans*-acting sequences, which implicitly encode the regulatory interactions and their strength (see Figure 2.2B). The weight w_{ij}, which measures how strong gene j regulates gene i, is decoded by an interaction map I that computes an "affinity" between the two sequences: $w_{ij} = I(s_{\text{trans},j}, s_{\text{cis},i})$. The interaction map I abstracts the complex, biological interaction map $\mathcal{I}$. It is based on the local alignment score of the two sequences. Figuratively speaking, the closer the match between two subsequences ("binding sites") of s_{cis} and s_{trans}, the stronger the interaction. For details, see Appendix A.

In summary, decoding of the AGE genome involves the identification of valid genes, which must be correctly delimited by the corresponding tokens, and the subsequent application of the interaction map to all pairs of *cis*- and *trans*-acting sequences. The interaction strength between two sequences may be zero, in which case there is no regulatory link between the two genes. Thus, the size of the decoded network is given by the number of genes in the genome, and the topology and weights w follow from the computed interaction strengths.

The exact implementation of the interaction map, described in Appendix A, is not critical. What matters is the implicit nature of the encoding, i.e., the fact that the N^2 possible connections (where N is the network size) are encoded implicitly in only N *cis*- and *trans*-acting elements. One of the consequences of the implicit encoding is that a *single* mutation in a protein-coding or *cis*-regulatory sequence may affect zero, one, or several regulatory interactions simultaneously.

In contrast, in an explicit (direct) encoding a single mutation typically affects only one characteristic of the network. This and other features discussed below make the implicit encoding a more evolvable representation of networks than conventional, direct encodings.

Encoding additional numerical parameters of genes

In the example of Figure 2.2B, genes are composed of two sequences s_{cis} and s_{trans}, which implicitly encode the network structure and the weights w of the regulatory interactions. We will now describe how additional numerical parameters $\mathbf{p}$ of genes (e.g., the decay rate and the maximum transcription rate of the model described in Section 2.3.1), are encoded in AGE.

The number of parameters depends on the type of gene network model used. To encode P parameters, genes must have P additional sequences delimited by tokens `TE`. For example, when using a model that has two numerical parameters p_1 and p_2 per gene, a valid gene would be composed of the following sequences s (see Figure 2.3 for an illustration)

$$\text{gene} \;:=\; \texttt{GN}\ s_{\text{cis}}\ \texttt{TE}\ s_{\text{trans}}\ \texttt{TE}\ s_{\text{p,1}}\ \texttt{TE}\ s_{\text{p,2}}\ \texttt{TE} \tag{2.1}$$

The values of the parameters are decoded from the sequences using a mapping $p_i = J(s_{\text{p},i})$. The mapping is based on Center of Mass Encoding (CoME), which is a self-adaptive, variable length encoding for real-valued parameters (Mattiussi et al., 2007).

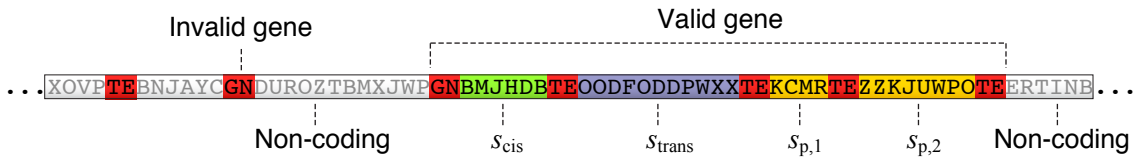


Figure 2.3: Syntax of a gene with two numerical parameters. In order to encode numerical parameters p in addition to the weights of the regulatory interactions, genes must have an additional sequence $s_{\text{p},i}$ for every parameter i. Thus, a valid gene with two gene parameters has four sequences separated by tokens `TE`. The token `GN` to the left of the valid gene is not followed by the necessary four tokens `TE`, thus it is not coding.

2.2.2 Mutation and recombination

One of the key features of AGE is the possibility to apply a wide range of biologically inspired genetic operators (mutations and recombinations) that are known to play important roles in the evolution and complexification of natural gene regulatory networks. From the point of view of the genetic operators, the tokens that delimit the genes have no special meaning—there is no distinction between tokens, coding, and non-coding genome fragments.

1. **Single-nucleotide mutations**

   - *Nucleotide substitution*. Each nucleotide of the genome has probability p_{n} to be substituted with a random character from the genetic alphabet.
   - *Nucleotide deletion*. Each nucleotide is removed from the genome with probability p_{n}.
   - *Nucleotide insertion*. At any given position in the genome, a random character from the genetic alphabet is inserted with probability p_{n}.

2. **Mutations that affect chromosome fragments**

   - *Fragment deletion*. For each chromosome, with probability p_{f}, the sequence between two randomly chosen positions is deleted.
   - *Fragment duplication*. For each chromosome, with probability p_{f}, the sequence between two randomly chosen positions is copied and inserted at a random position in the genome (the duplicated fragment can be inserted in the same or in a different chromosome).
   - *Fragment transposition*. For each chromosome, with probability p_{f}, the sequence between two randomly chosen positions is removed and inserted at a random position in the genome (in the same or in a different chromosome).

3. **Mutations that affect entire chromosomes**

   - *Chromosome duplication*. Each chromosome is duplicated with probability p_{c}.
   - *Chromosome deletion*. Each chromosome is deleted with probability p_{c}.

- *Crossover*. Crossover is performed with probability $p_{\text{crossover}}$ when producing offspring from two parents (see next section). We use the crossover operator described by Mattiussi (2005), which ensures that only homologous chromosome fragments are exchanged. Crossover is only performed if the two parents have the same number of chromosomes. If this is the case, chromosomes are paired in the order as they appear in the genome. For every pair, a tentative crossover point is randomly selected in the first chromosome. A subsequence from the neighborhood of this point is then used as a template to search for a homologous crossover point in the second chromosome (Mattiussi, 2005). If a homologous crossover point can be established, the corresponding chromosome fragments are swapped to form the recombined chromosomes.

4. **Mutations that affect the entire genome**

- *Genome duplication*. The genome is duplicated with probability p_{g}.

Some examples of these mutations, and their effects at the network level, are shown in Figure 2.4. Since the genetic operators are applied probabilistically to randomly chosen parts of the genome, they can invalidate genes (e.g., through invalidation of a token) and transform the corresponding fragments into non-coding genome, which may play the role of an evolutionarily useful repository of genetic fragments. On the other hand, new genes can be created, for example through the appearance of new tokens or the duplication of a genome fragment.

In the experiments reported in this thesis, we used the mutation probabilities given in Table 2.1. AGE genomes are typically composed of a handful of chromosomes and have a total length of several thousand nucleotides. With the given mutation probabilities, there are thus on average several single-nucleotide mutations per reproduced genome. In contrast, the more disruptive mutations that affect larger chromosome fragments, entire chromosomes, or the entire genome, have a low probability of occurrence in a given genome. We found that the exact values of these parameters do not critically affect the results, as long as the disruptive macro-mutations have a low chance to occur.

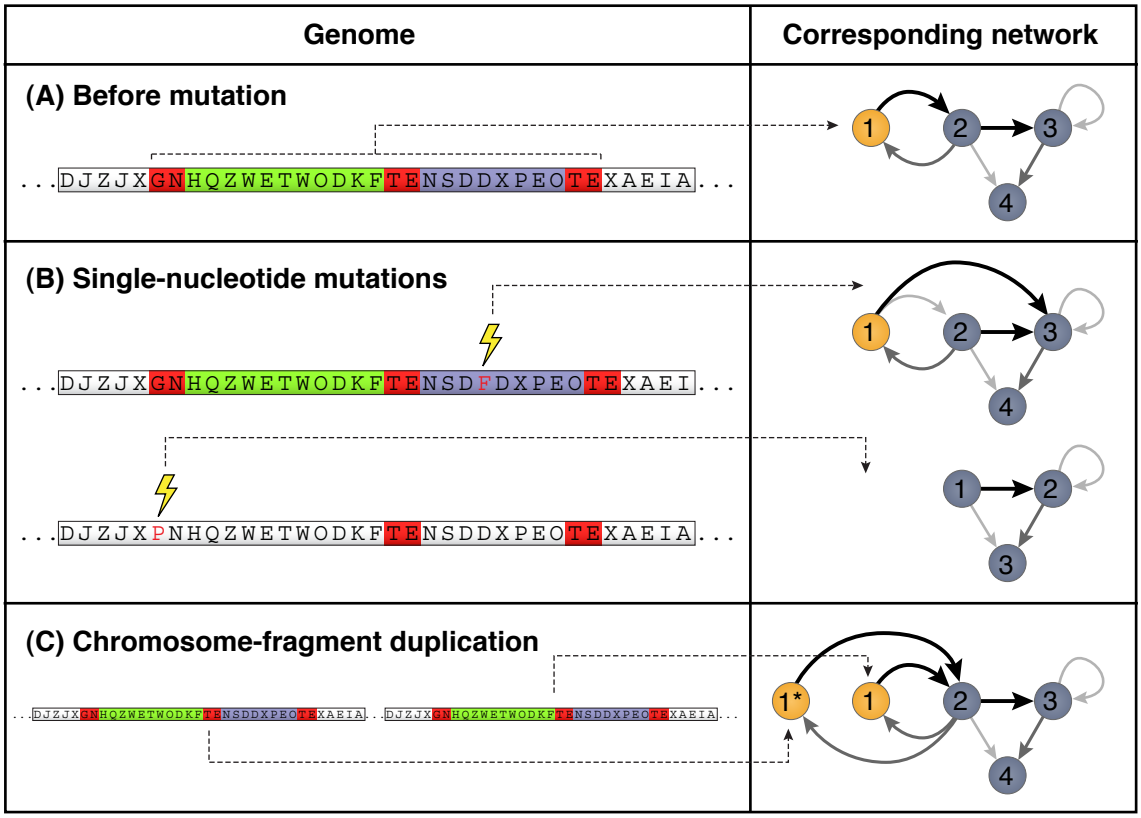


Figure 2.4: Examples of mutations and their effects on the network. **(A)** On the left, we show a possible subsequence of an AGE chromosome defining a gene. On the right, this gene is highlighted in the complete decoded network. The darkness of the links indicates the interaction strength. **(B)** Two examples of single-nucleotide mutations. In the first example, a character is inserted in the *trans*-acting sequence of the gene, potentially affecting its out-going connections. In the example shown here, a new regulatory link is created (1⟶3), and the strength of an existing link is decreased (1⟶2). The second example shows a nucleotide substitution that happens to invalidate the token GN that marked the beginning of the gene. Consequently, the sequence is now non-coding and the gene is deleted from the network. **(C)** An example of a chromosome fragment duplication that contains the highlighted gene. As a result, the gene is duplicated in the network. Note that the copied gene inherits the interactions of the original gene.

Table 2.1: Mutation probabilities used in the experiments of this thesis.

Mutation type	Parameter	Value
Single nucleotides	p_{n}	0.001
Chromosome fragments	p_{f}	0.01
Entire chromosomes	p_{c}	0.001
The entire genome	p_{g}	0.001
Crossover	$p_{\mathrm{crossover}}$	0.5

2.2.3 Evolutionary algorithm

The biomimetic genome, complemented with the bio-inspired mutation and recombination operators described above, allows us to evolve gene networks *in silico* according to a given fitness criterion (the *fitness function*). Apart from the bio-inspired genotype and genetic operators, the process of artificial evolution is similar to a standard genetic algorithm (Bäck et al., 2000). It consists of the following steps, which are depicted in Figure 2.5.

0. **Random initialization**. Evolution starts from a population of randomly created genomes. In the experiments reported here, genomes were initialized with a random number of two to four chromosomes. Each chromosome is initialized with a random character sequence of length 100. To accelerate the initial phase of evolution, we seed the genomes with randomly created genes. A gene can be created by inserting a token `GN` at a random position, followed by the required number of tokens `TE` to define a valid gene (cf. Figure 2.3). We insert the tokens `TE` with a spacing of 20 characters (i.e., the sequences of the gene that they define have an initial length of 20). The length of these sequences, as well as the number of chromosomes and their length, are typically quickly adapted by evolution. The choices for the random initialization described above are thus not critical, they only affect the initial phase of evolution and not the final results.[4]

[4]Note that we purposely initialize the genomes at short length to avoid making them bigger than needed, i.e., we let evolution grow them to the "right" size (the length of the genomes typically grew to a few thousand nucleotides in the experiments reported here).

1. **Selection**. From the population of N genomes, which have each an associated fitness value (fitness evaluation is described below), M genomes are selected for reproduction. Genomes with a better fitness are selected with higher probability. Specifically, we use tournament selection (Bäck et al., 2000), which consists in first choosing k genomes from the population at random, and then selecting for reproduction the genome with the best fitness among these k genomes. The parameter k is the so-called *tournament size*, it can be used to adjust the selection pressure (the bigger the tournament, the stronger the selection pressure). Tournament selection is repeated M times to select M genomes for reproduction (note that the same genome can be selected multiple times).

2. **Crossover and mutation**. The M selected "parent" genomes are paired to produce new "offspring" genomes. From each pair, O offspring genomes are created by applying the crossover operator described in the previous section (if no crossover is used, the genomes are simply copied). Finally, the different types of mutation operators are applied to each offspring genome.

3. **Fitness evaluation**. Next, the fitness of all genomes is evaluated. To evaluate the fitness of a genome, we first decode the gene network as described in Section 2.2.1. Remember that the decoded network is a nonlinear dynamical model, i.e., a system of ordinary differential equations.[5] The fitness is a measure of how well this model reproduces the available gene expression data. The better the data fit, the better the fitness value assigned to the genome (an example is given in the next section).

4. **Replacement**. Finally, the genomes of the previous generation are replaced with the newly produced offspring genomes. We use so-called *elitism*, which means that the genome with the best fitness of the population (the elite) is protected from replacement. In the experiments reported here, we choose the population size N, the number of parents M, and the number of offspring per parent O, so that $N = M \cdot O + 1$. In other words, at every generation the entire population, except for the elite, is replaced with newly produced genomes.

[5]Note that the AGE genome and the decoded network model are two completely separate entities that do not interact. In contrast, in a biological cell the genome is part of the gene network, and the dynamics of the network can affect its state, e.g., through chromatin modifications.

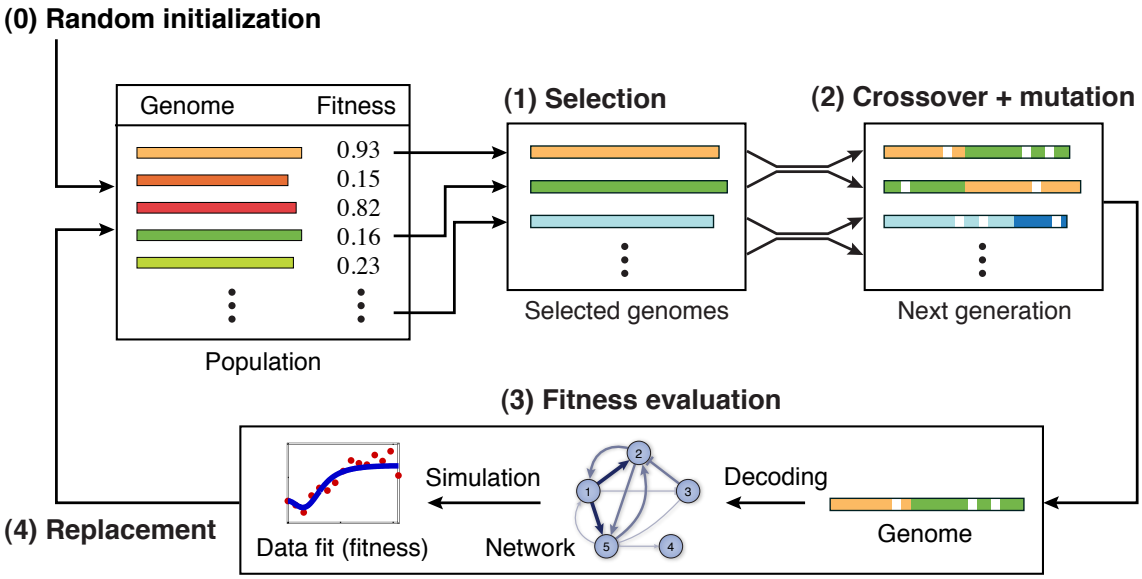


Figure 2.5: The evolutionary algorithm. **(0)** Evolution starts from a population of randomly initialized genomes. **(1)** Genomes with high fitness are selected for reproduction. **(2)** New "offspring" genomes are produced from the selected "parent" genomes by applying crossover and mutation operators. **(3)** The fitness of the offspring genomes is evaluated. This involves decoding the gene networks from the AGE genome and measuring how well they reproduce the available gene expression data. The better the data fit, the better the fitness. **(4)** Genomes of the previous generation are replaced by the newly produced offspring.

The results presented in this thesis were obtained with the parameters of the evolutionary algorithm set as specified in Table 2.2. The choice of these parameters is not critical. They were determined heuristically based on a series of test runs.

In summary, the evolutionary reverse engineering method consists in evolving, *in silico*, gene networks with an increasingly better fit to the gene expression data. However, note that the primary goal of reverse engineering is not fitting the data, but inferring the structure of the unknown target network. For now, we simply assume that the network that best matches the data is the most plausible reconstruction of the unknown target gene network. In the following chapters, we will discuss this assumption in more detail, and consider the case where many different networks are consistent with the available data.

Table 2.2: Parameters of the evolutionary algorithm.

Parameter		Value
Population size	N	101
Number of parents	M	50
Number of offsprings per parent	O	2
Tournament size	k	2

2.3 Application to an *in silico* test case

When applying a novel reverse engineering technique directly to biological data, performance evaluation is difficult because the target network is in general unknown. Thus, we first tested AGE using gene expression data generated in simulation from an *in silico* target gene network. Subsequently the inferred networks were compared with the target network in order to validate the results. This is a standard approach to assess the performance of reverse engineering methods (Mendes et al., 2003), its advantages and limitations will be discussed in detail in Chapter 5.

The purpose of this section is to demonstrate the application of the evolutionary reverse engineering framework on a practical example. For this purpose, we use a standard gene network model and noise-free data. In the following chapters, we will discuss different types of gene network models and analyze the performance of AGE in more realistic settings.

2.3.1 The test case

Gene Network Model

We demonstrate the application of AGE using a standard *sigmoid model* (Reinitz and Sharp, 1995; Weaver, 1999; Wahde et al., 2001; Jaeger et al., 2004b) defined by the system of state equations

$$\frac{dx_i}{dt} = m_i \cdot \sigma\Big(\sum_{j=1}^{N} w_{ij} x_j\Big) - \delta_i x_i \tag{2.2}$$

where the variable x_i is the expression level of gene i, the parameter m_i is the maximum transcription rate, δ_i is the degradation rate, w_{ij} represents the regula-

tory influence of gene j on gene i, and N is the number of genes in the network. The weight w_{ij} is positive for enhancing, negative for repressing, and zero for absent interactions. The so-called *input function*, which computes the level of activation of the gene as a function of its inputs, is a sigmoid

$$\sigma(z) = \frac{1}{1+e^{-z}} \tag{2.3}$$

In this section, we use steady-state expression levels as input data for the inverse problem, though AGE can as well be applied to time series data (see Chapter 3). At steady state, the state equations become a set of algebraic equations

$$\frac{dx_i}{dt} = 0 = p_i \cdot \sigma\Big(\sum_{j \in R_i} w_{ij} x_j\Big) - x_i \qquad \text{with } p_i = m_i/\delta_i \tag{2.4}$$

In silico target network and expression data

We employ the topology of a nine-gene network of *Escherichia coli* (*E. coli*) described by Gardner et al. (2003) as a test case. We refer to this topology as *SOS network*, because it is part of the so-called SOS pathway involved in DNA damage response.

There is no quantitative model of the SOS network available in the literature. Hence, numerical parameter values for the weights w_{ij} and the parameters p_i of the steady state equations (2.4) introduced above were sampled randomly.[6] The signs of the weights were set according to the SOS network topology (positive for enhancers and negative for repressors). The resulting *in silico* target gene networks are *random targets* with a realistic topology, which is a biologically more plausible approach than random generation of both topology and parameters (see Chapter 4).

Synthetic gene expression data was obtained by applying a perturbation to the *in silico* target network and computing the steady state expression levels of all genes.[7] This process was repeated for different perturbations to gather the necessary gene expression data for reverse engineering. We simulated two different

[6] Weights w_{ij} were initialized uniformly in the range $[0.15, 1.5]$ and parameters p_i in the range $[0.5, 2]$. These ranges were selected empirically with the goal to obtain rich nonlinear dynamics in the target networks, i.e., so that on average the total regulatory input for the sigmoid input functions was neither completely saturated nor constrained to a very small, almost linear regime.

[7] We compute the steady states numerically using Powell's method of the GNU Scientific Library (GSL, http://www.gnu.org/software/gsl).

types of perturbations that are commonly used for gene network inference: gene knockouts (silencing of a particular gene), and gene over-expression, which consists in artificially boosting the transcription rate of a gene. A gene knockout can be simulated by setting the rate parameter m_i to zero. Consequently, the expression level x_i of this gene at steady state will be zero. Over-expression is simulated by doubling the parameter m_i of the affected gene.

For the experiments reported below we generated expression data from the *in silico* SOS network for the wild type (unperturbed network) and for 9 knockout and 9 over-expression experiments (knockout and over-expression of every gene).

2.3.2 Setup of AGE for the test case

As mentioned before, one of the advantages of the evolutionary reverse engineering method with AGE is that it can be easily used with different types of dynamical models and gene expression data. In order to set up AGE for a specific model and data type, one has to:

1. **Specify the number of real-valued parameters of the gene model** that have to be encoded in the AGE genes in addition to the weights. Here, we use the sigmoid model, which has only one such parameter per gene (p_i in Equation 2.4).

2. **Define the fitness function** that is used to evaluate the decoded networks. Here, we use a least squares optimization criterion. Thus, the goal of the evolutionary algorithm is to *minimize* the following fitness function

$$f(\hat{\mathbf{X}}) = \sum_{i=1}^{M} \sum_{j=1}^{N} (x_{ij} - \hat{x}_{ij})^2 \tag{2.5}$$

   where $\mathbf{X}$ is the synthetic gene expression data generated from the target network (element x_{ij} corresponds to the expression level of gene j in experiment i) and $\hat{\mathbf{X}}$ are the corresponding expression levels in the inferred network. M denotes the number of different perturbation experiments and N is the number of genes.

With this setup, we evolve sigmoid gene network models that minimize the square error with the *in silico* produced gene expression data of the test case.

2.3.3 Results

The results of a batch of ten reverse engineering runs are shown in Figure 2.6. For each run, we record the fitness of the best individual at every generation of the evolutionary algorithm. Naturally, in addition to a good fit of the expression data, the structure of the inferred networks should match the target gene network. In a real biological application, the structure of the target network is unknown, but in the *in silico* test case employed here the accuracy of the inferred network models can be measured. To this end, we use the mean square error $E(\hat{\theta})$ of all parameters $\hat{\theta}$ of the reverse engineered network (including all weights w_{ij} and gene parameters p_i), compared to the true parameter values θ of the *in silico* target gene network

$$E(\hat{\theta}) = \frac{1}{K} \cdot \sum_{l=1}^{K} (\theta_l - \hat{\theta}_l)^2 \tag{2.6}$$

where θ_l denotes the l-th element of parameter vector θ, and K is the total number of parameters. We refer to $E(\hat{\theta})$ as *prediction error* of the inferred network. In addition we also count the number of *false positives* and *false negatives*.[8]

The ten runs shown in Figure 2.6 were executed for $100,000$ generations of the evolutionary algorithm. All ten runs achieved a fitness below 0.1. Since fitness is a *sum* of square errors, the individual expression levels were fitted extremely accurately with a relative error in the order of 1%. As the reverse engineering algorithm optimizes the fitness, the prediction error of the inferred networks decreases. Four out of ten runs inferred the SOS network with high precision (final prediction error between 0.02 and 0.03).[9] In other words, these runs closely matched the structure, weights, and gene parameters of the target network. The other runs converged at prediction errors of about 0.1.

In a set of reverse engineering runs, one would like to choose the inferred network with the lowest prediction error $E(\hat{\theta})$. However, since $E(\hat{\theta})$ is unknown in a real application, this is not possible. Hence, we choose the inferred network

[8] We count as *false positive* when a target weight $w_{ij} = 0$ and the absolute value of the inferred weight $|\hat{w}_{ij}| > 0.1$; a *false negative* occurs when $w_{ij} \neq 0$ and $|\hat{w}_{ij}| < 0.1$.

[9] Further analysis indicates that the accuracy achieved by the best runs corresponds to a lower bound given by the discretization of the search space due to the quantization of the parameters and weights.

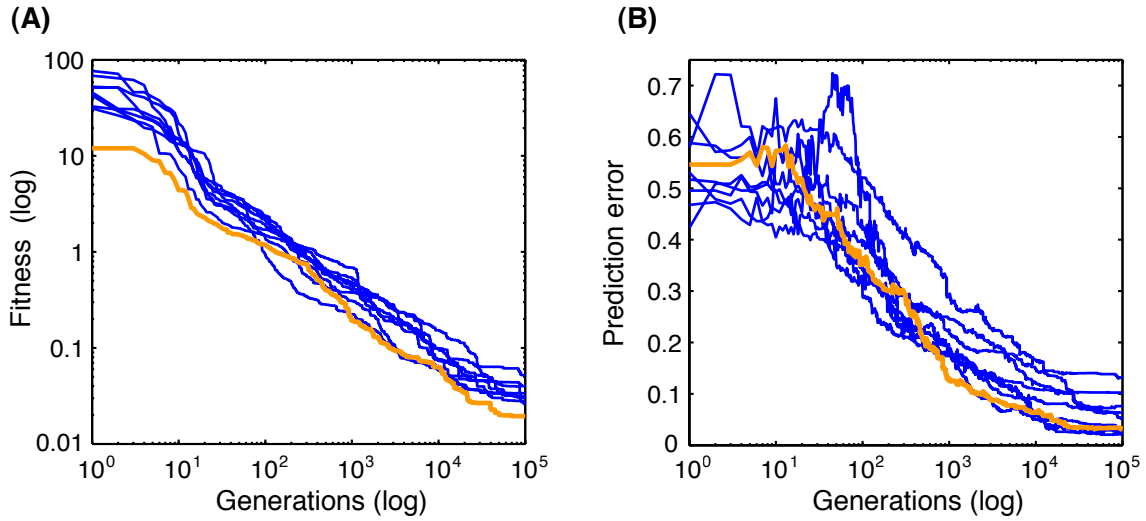


Figure 2.6: Reverse engineering the SOS network—ten runs. **(A)** The fitness $f(\hat{\mathbf{X}})$ of the best individual of each run. **(B)** The corresponding prediction error $E(\hat{\theta})$ of the individuals plotted on the left. As the fitness is optimized (i.e., minimized), the inferred networks match the structure and parameters of the target network with increasing accuracy (the prediction error goes down). The run with the best final fitness is highlighted.

with the best fitness as the most plausible reconstruction of the target network.[10] Here, the best run (see Figure 2.6) achieves a fitness of 0.02 and the corresponding network has a very low prediction error of 0.03 with only one false positive and one false negative out of a total of 81 possible connections (see Figure 2.7).

In additional experiments, using different random initializations of the SOS network's parameter values, we have obtained the same quality of results. AGE infers networks with an excellent data fit in every run. Roughly 40% of the runs also achieve very low prediction errors (i.e., they correctly infer the target, having only few false positives and false negatives). Simpler networks, for example cascades of size six, were inferred correctly in every run. In addition, we have have also tested a gradient descent method. As expected, gradient descent was not successful—even when restarted many times—because it prematurely converged to local optima with a poor data fit and high prediction error.

[10] In a real application, one should not just consider the inferred network with the best fitness—which merely corresponds to the network with the highest likelihood—but analyze all well-scoring (i.e., probable) inferred networks (see Chapter 6).

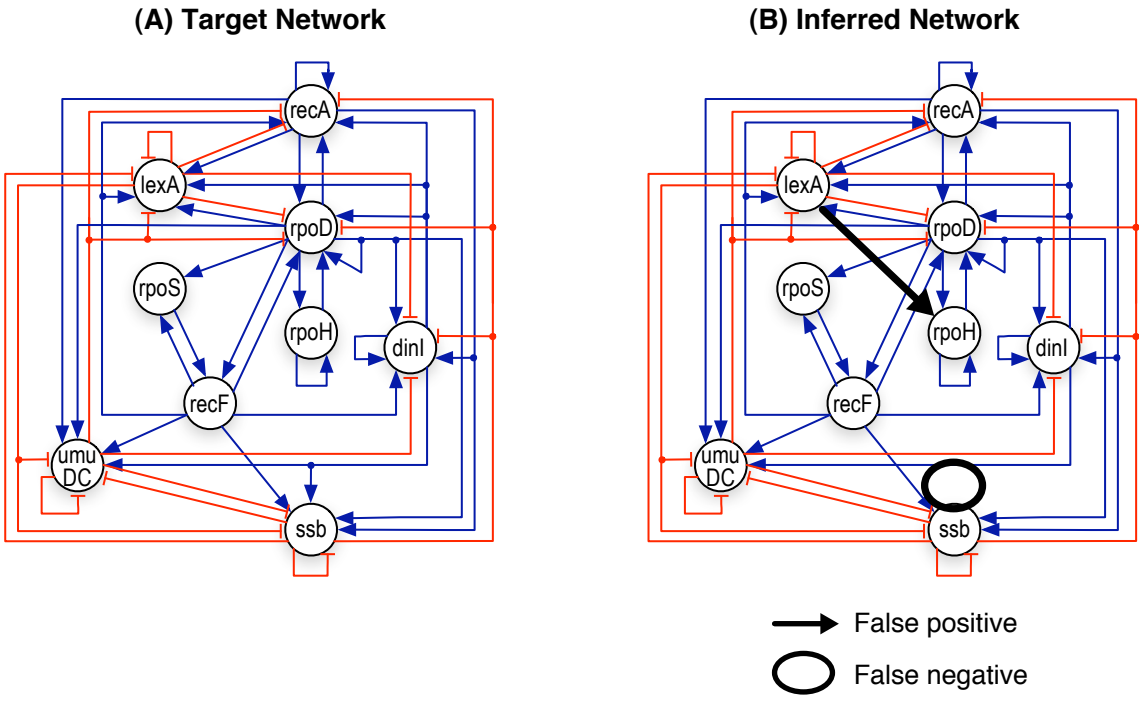


Figure 2.7: The inferred SOS network. **(A)** The structure of the *E.coli* SOS network (Gardner et al., 2003). Arrows are enhancing, T-ends denote inhibitory interactions. **(B)** The inferred structure by the run with the best final fitness. The predicted structure is correct except for one false positive (bold) and one false negative (encircled).

2.4 Conclusion

In this chapter, we have presented a novel approach for reverse engineering of gene regulatory networks, which consists of using a reconstruction process that is similar to the evolutionary process that created these networks. By taking inspiration from the mechanisms that enable the evolution of complex gene regulatory networks in nature, we have designed an evolutionary reverse engineering method based on a novel, biomimetic artificial genome (AGE).

The aim of the evolutionary approach is to effectively integrate prior knowledge in the reverse engineering procedure. Because biological networks are generally sparse, most state-of-the-art inference methods include an explicit bias towards sparse networks (Weaver, 1999; Wahde et al., 2001; Gardner et al., 2003; Tegner et al., 2003; Gupta et al., 2005), for example by constraining the maximum number of connections per gene. Van Someren et al. (2003) have proposed to incorporate prior knowledge by formulating constraints for additional features of biological gene networks—e.g., stability, modularity, and robustness—thus lead-

ing to a multi-criterion optimization problem. However, the formulation and weighting of *ad hoc* constraints is difficult in practice. The biomimetic approach circumvents this problem by embedding prior knowledge at a more fundamental level, namely by using a search method that bears close similarity with the evolutionary process (the "design method") that created the reverse engineering target. Thus, although the selection pressure that acts on biological gene networks is different from the fitness function used in our evolutionary algorithm, the exploration of the search space is similar. The goal is to reproduce—at a certain level of abstraction—evolutionary constraints of the biological genome, thereby biasing the reverse engineering process towards biologically plausible solutions and improving the accuracy of predictions.

For example, the biomimetic genome implies a bias towards sparse networks because—as in biological gene networks—regulatory interactions need to be actively evolved through creation of appropriate motifs ("binding sites") in the *cis*- and *trans*-acting sequences. Links tend to be pruned by random mutations and only the links under selective pressure (i.e., those that contribute positively to the fitness) are maintained. Thus, given the choice between a highly connected and a sparse network that fit the data equally well, the biomimetic algorithm reconstructs the sparse network with a higher probability—consistent with our prior knowledge that biological networks are generally sparse.

Conventional evolutionary algorithms have been previously used for gene network inference (Wahde and Hertz, 2001; Iba and Mimura, 2002; Kikuchi et al., 2003; Moles et al., 2003; Spieth et al., 2004; Deng et al., 2005; Kimura et al., 2005; Bongard and Lipson, 2007). Corne (2004) have argued that genetic algorithms may be particularly well suited for reverse engineering biological networks, which are themselves a product of an evolutionary process. However, these methods are based on *direct encodings*, where the genome is typically a fixed-length vector of real-valued parameters. This type of encoding is fundamentally different from the *implicit encoding* of gene regulatory networks in the biological genome, and can thus not be expected to reproduce its evolutionary constraints.

The approach proposed here extends the evolutionary algorithm with a bio-inspired artificial genome, which mimics the implicit encoding of natural gene networks. The implicit encoding represents the many possible regulatory links of the network not explicitly in the genome, but implicitly using only one *cis*-

regulatory and one *trans*-acting (protein-coding) sequence per gene. This has the advantage of reducing the number of elements that must be encoded in the genome with respect to a direct encoding. Moreover, with an implicit encoding a single mutation can have several, non-trivial effects on the network structure, which implies that the search space is explored in a very different way compared to a direct encoding. In particular, AGE permits mutation of single nucleotides, but also rearrangement, duplication, and deletion of larger chromosome fragments or entire chromosomes. It is also worth pointing out that AGE has a very high neutrality in the search space, i.e., many mutations have no immediate effect on the network (they are neutral). This is because the implicit encoding of the regulatory interactions with the interaction map, as well as the encoding of the additional numerical parameters with Center of Mass Encoding (CoME), are both highly redundant (many different character sequences produce the same numeric value). Neutrality permits continuing the exploration of the genotype space even when the algorithm is unable to find a network with a better fitness than those of the current population. In this way the probability of a stalling of the search is reduced (Huynen et al., 1996). In summary, the above-mentioned features make AGE a more evolvable genetic representation for networks than conventional encodings.

AGE can be used with a large class of nonlinear gene network models, and it allows simultaneous inference of both the network structure and numerical parameter values of these models. In fact, AGE has proven to be a powerful search method for the (re)construction of nonlinear dynamical networks in a range of applications besides reverse engineering of gene networks. For example, instead of using dynamical models of genes, we can use models of electronic components to evolve analog electronic circuits (Mattiussi, 2005; Mattiussi and Floreano, 2007). AGE has also been applied to evolve different types of artificial neural networks (Dürr et al., 2006, 2008, 2009). In all these applications, AGE achieved state-of-the-art performance when compared to the best domain-specific methods documented in the literature.

In this chapter, we have demonstrated the application of AGE on an *in silico* test case based on the structure of a nine-gene network of *E. coli* (the SOS network). Despite the intricate wiring of the SOS network, which includes many interlocked feedback loops, the evolutionary reverse engineering method correctly inferred both the network structure and the numerical parameter values

of the nonlinear model from steady-state gene expression data alone. Compared with time-series data, steady state data is easier to obtain experimentally and its information content is potentially higher (samples in a time-series are not independent). Recently, several methods have been proposed for inference of linear models from steady-state perturbation data (Gardner et al., 2003; Brazhnik, 2005; Kholodenko et al., 2002; de la Fuente et al., 2002). Here, we demonstrate that steady state data is also suitable for inference of *nonlinear* models using AGE.

The gene expression data of our test case was noise-free and simulated using the same model type that was also used for the reverse engineering (the sigmoid model). Thus, the results of this chapter indicate that AGE provides a near-perfect reconstruction of the target network, *given data of sufficient quality and the "right" model type*. Testing a reverse engineering method under these conditions is important, even though they are not met in real biological applications, because it shows whether the method is able to effectively navigate the enormous search space and find the needle (the true network structure) in the haystack.[11] However, it is clear that this only corresponds to a necessary first step in performance assessment. In the following chapters, we will analyze the performance of AGE in more realistic settings.

[11]Considering just the network structure and not the numerical parameter values, N genes have N^2 possible regulatory links and 2^{N^2} possible network structures (each link can be present or not). The SOS network is thus one out of about 10^{24} possible wirings of the nine genes. However, note that due to the high neutrality of the AGE genotype space, there are effectively many needles (many different genomes that correspond to the single true network structure), in the haystack.

3 Reverse engineering an *in vivo* benchmark network in yeast

This chapter is based on the following publication:

- Marbach, D., Mattiussi, C., and Floreano, D. (2009). Replaying the evolutionary tape: Biomimetic reverse engineering of gene networks. *Annals of the New York Academy of Sciences*, 1158:234–245.

Synopsis

In this chapter, we assess the performance of AGE on an in-vivo benchmark that was released as an international gene-network inference challenge within the DREAM2 conference. The goal of this challenge was to infer the structure of a five-gene network, which was unknown to the participants, from time-series gene expression data. The network was in fact a synthetic-biology gene network that had been constructed in yeast, hence its structure was known to the organizers and the submitted predictions could be systematically evaluated. Before applying AGE to the DREAM2 challenge, we considered the choice of the gene-network model type more carefully, and conceived a principled approach to compare candidate models. This has led us to design a new log-sigmoid model, which provides a better approximation to different types of transcriptional regulation than a standard sigmoid model. AGE, combined with the log-sigmoid model, achieved the best performance in the DREAM2 five-gene network challenge.

3.1 Introduction

In silico performance assessment of network-inference methods can give insights into their strengths and weaknesses, as demonstrated in Chapter 5. However, the biological mechanisms involved in gene regulation are so complex that *in silico* benchmarks—despite our efforts to make them as realistic as possible—can never replace careful characterization of performance *in vivo*. Unfortunately, *in vivo* performance assessment is extremely difficult, because network predictions can in general not be systematically validated. Even in a best-case scenario, only a very small subsample of predictions are typically validated using sound experimental assays (Stolovitzky et al., 2009). To make matters worse, the few hand-picked predictions that are selected for validation are usually those that were made with highest confidence, and may thus not be representative for the complete set of predictions.

To sidestep these difficulties, Cantone et al. (2009) propose to use synthetic-biology gene networks for *in vivo* benchmarking of network-inference methods.[1] Synthetic biology allows the construction of small gene networks in living organisms (Becskei and Serrano, 2000; Elowitz and Leibler, 2000; Gardner et al., 2000; Ellis et al., 2009). In contrast to the endogenous (original) gene network of the organism, the structure of an engineered synthetic-biology network is perfectly known. Thus, if a network-inference method is tested on a synthetic-biology network, the predictions can be systematically validated, enabling rigorous performance assessment *in vivo*. Cantone et al. (2009) have constructed a five-gene network in the yeast *Saccharomyces cerevisiae* specifically for this purpose. The network is called IRMA (*In vivo* Reverse engineering and Modelling Assessment).

Cantone et al. provided two time series from the IRMA network as a reverse engineering challenge for the second DREAM conference (DREAM2) (Stolovitzky et al., 2009). This challenge became known as the *DREAM2 five-gene network challenge*. The goal of the challenge was to predict the structure of the network from the provided time-series data. The true structure of the network was unknown to the participants prior to submission of their predictions (in fact, participants didn't even know that the network was synthetic and that it was in yeast). We

[1] To avoid confusion between *in silico* and *in vivo* synthetic networks, I consistently call the former *in silico networks* and the latter *synthetic-biology networks*.

participated in the DREAM2 five-gene network challenge to allow direct comparison of AGE with other methods on a blinded benchmark, i.e., in the realistic situation where the true network is not known in advance and methods can thus not be "tuned" to the specific benchmark.

As discussed in the previous chapters, the choice of the gene-network model type is a crucial step in reverse engineering. Genes can be combinatorially regulated in different ways, i.e., they can have different *cis*-regulatory input functions (the input function of a gene describes the combined effect of its regulators on the transcription rate). Different models, such as the linear model (D'Haeseleer et al., 1999; Gardner et al., 2003; Tegner et al., 2003), the log-linear model (Liao et al., 2003; di Bernardo et al., 2005; Gupta et al., 2005), the sigmoid model (Mjolsness et al., 1991; Reinitz and Sharp, 1995; Weaver, 1999; Wahde et al., 2001; Perkins et al., 2006), or S-Systems (Voit and Savageau, 1987; Kimura et al., 2005), approximate different types of input functions more or less well. For example, models based on additive terms would be expected to better approximate independent regulation (OR-type input functions) than synergistic regulation (AND-type input functions), whereas models based on multiplicative terms may better approximate synergistic than independent regulation. However, we can't know which type of interactions occur in a given network *a priori*. Thus, it would be desirable to have a gene network model that approximates all of them equally well. We have designed a novel gene network model, called the log-sigmoid model, which fulfills this requirement.

In the next section, we introduce the log-sigmoid model. We show that, in contrast to the standard sigmoid model, it approximates different types of synergistic and independent input functions equally well. We proceed by describing an *in silico* benchmark that we use to assess the performance of AGE at different levels of noise. Finally, we discuss the synthetic-biology benchmark of the DREAM challenge, where AGE combined with the log-sigmoid model achieved winning performance.

3.2 The log-sigmoid gene network model

Hill-type transcription kinetics are often used when building a quantitative model of a well studied promoter bottom-up (Bower and Bolouri, 2004; Alon, 2007a). This type of models are also called *thermodynamical models*, because the kinet-

ics of transcription factor binding are derived from thermodynamics. The level of detail and complexity of thermodynamical models is far beyond the scope of gene network reverse engineering, which must rely on more simple, phenomenological models. Such simplified models approximate the real gene expression dynamics with generic functions, thereby abstracting implementation details of the network components and focusing on their functionality (Reinitz and Sharp, 1995).

The following sigmoid model is arguably the most widely used nonlinear model in gene-network reverse engineering (Mjolsness et al., 1991; Reinitz and Sharp, 1995; Weaver, 1999; Wahde et al., 2001; Jaeger et al., 2004b; Perkins et al., 2006).

$$\frac{dx_i}{dt} = m_i \cdot \sigma\Big(\sum_{j=1}^{N} w_{ij} x_j + b_i\Big) - \delta_i x_i \tag{3.1}$$

This model was already introduced in the previous chapter, it is repeated here to allow direct comparison with the log-sigmoid model below. The state variable x_i is the expression level of gene i, the parameter m_i is the maximum transcription rate, b_i is a bias that relates to the basal transcription rate, λ_i is the degradation rate, w_{ij} represents the regulatory influence of gene j on gene i (positive for enhancers, negative for repressors, and zero for no interaction), and N is the number of genes in the network. The so-called activation or input function $\sigma(\cdot)$ is a sigmoid (different sigmoids are used in the literature, the one given here being the most common)

$$\sigma(z) = \frac{1}{1+e^{-z}} \tag{3.2}$$

The underlying assumptions and limitations of this type of phenomenological modeling are well discussed in the pioneering work of Reinitz and Sharp (1995).

Here, we propose a log-sigmoid model that is identical to the standard sigmoid model, except that the logarithm of the inputs is used

$$\frac{dx_i}{dt} = m_i \cdot \sigma\Big(\sum_{j=1}^{N} w_{ij} \log(x_j) + b_i\Big) - \lambda_i x_i \tag{3.3a}$$

$$= m_i \cdot \frac{\prod_{j=1}^{N} \left(x_j/k_i\right)^{w_{ij}}}{1+\prod_{j=1}^{N} \left(x_j/k_i\right)^{w_{ij}}} - \lambda_i x_i \tag{3.3b}$$

The second formulation (3.3b) is equivalent to the first one, it is obtained simply by rewriting (3.3a) and substituting b_i with a different parameter k_i. The reader acquainted with biochemistry may notice the similarity of the input function in

(3.3b) with a Hill function. Indeed, if there is only one input, it is identical to a Hill function. Thus, the parameter k_i may be loosely interpreted as a dissociation constant and the weights w_{ij} as Hill coefficients. Note that the log-sigmoid model has the same number of parameters as the standard sigmoid model. As for the standard sigmoid model, positive weights w_{ij} correspond to enhancing interactions, negative weights correspond to repressing interactions, and zero weights mean that there is no interaction.

Unfortunately, the choice of a particular phenomenological model type is often based on hand-waving arguments. Here, we propose to choose the model type based on objective criteria by comparing candidate models systematically on diverse types of input functions. To this end, we have defined eight elementary types of two-dimensional input functions, which correspond to the eight logic functions that can be performed on two inputs (AND, NAND, OR, NOR, IMPLIES, NIMPLIES, EQUAL, and XOR). These elementary two-input functions are shown in Figure 3.1A. They can all be realized by biological genes (Buchler et al., 2003) and were defined using a thermodynamical model, as described in Appendix B.

As mentioned above, the thermodynamical model is too complicated to be used for network-inference and must thus be approximated using phenomenological models.[2] We have compared how well the eight elementary two-input functions are approximated by the standard sigmoid model and the log-sigmoid model. The input functions of these two models are (rewritten here for two inputs, cf. Equations 3.1 and 3.3b)

$$f_{\text{sigmoid}}(x_1, x_2) = \sigma\Big(w_1 x_1 + w_2 x_2 + b\Big) \tag{3.4}$$

$$f_{\log-\text{sigmoid}}(x_1, x_2) = \frac{\big(x_1/k\big)^{w_1}\big(x_2/k\big)^{w_2}}{1+\big(x_1/k\big)^{w_1}\big(x_2/k\big)^{w_2}} \tag{3.5}$$

The best possible fits of the two models to the eight elementary input functions described above are shown in Figure 3.1.[3] As expected, the standard sigmoid

[2]For a gene with M inputs, the thermodynamical model has in the order of 2^M free parameters, compared to only in the order of M free parameters for the phenomenological models typically used in gene network inference.

[3]Least-square fits were obtained with nonlinear minimization in Matlab, using the trust-region reflective Newton method of Coleman and Li (1996). Diverse initial conditions converged to the same best-fit solutions.

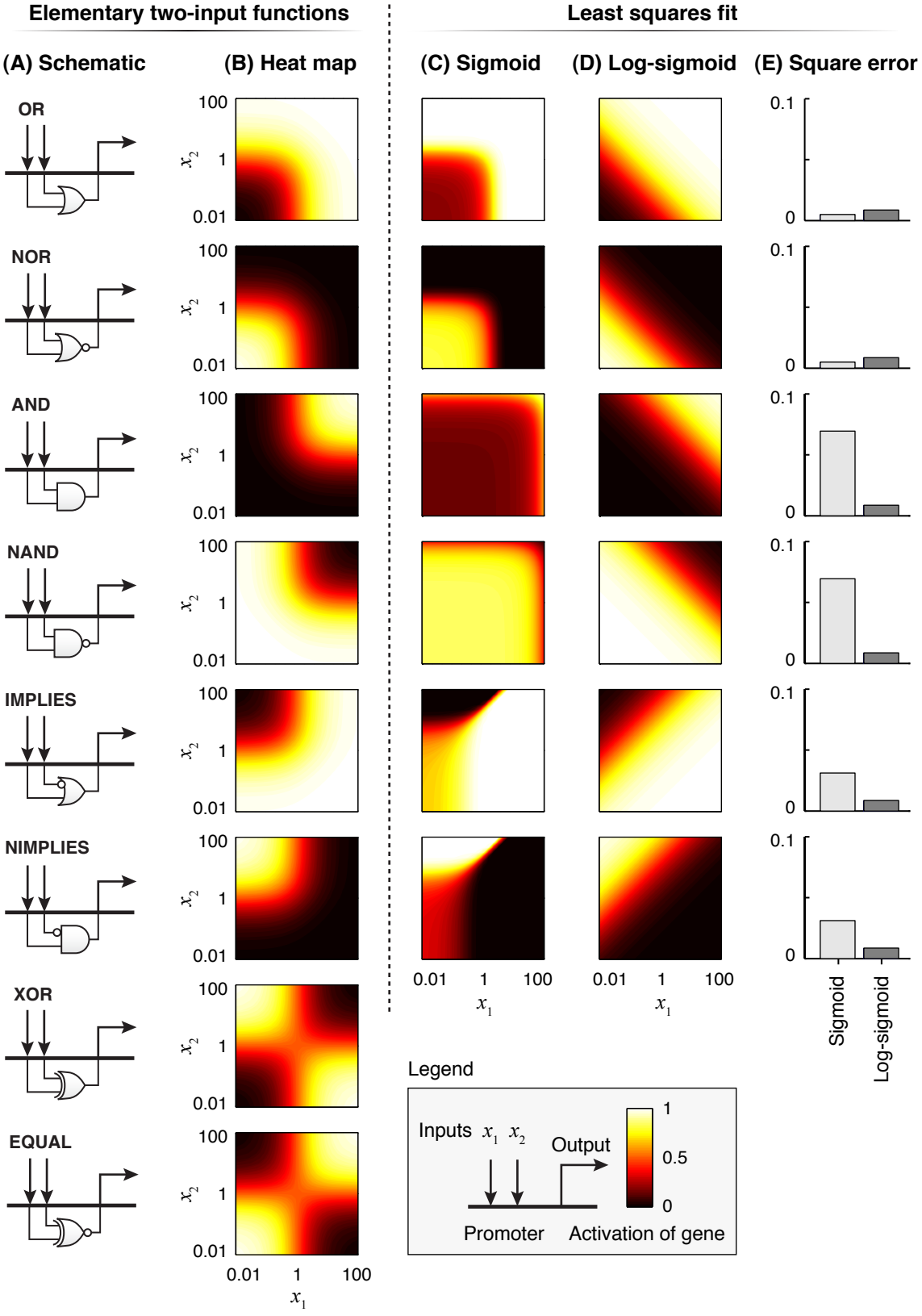


Figure 3.1: Caption on next page.

Figure 3.1: Comparison of the standard sigmoid and the log-sigmoid model on elementary two-input functions. **(A)** Schematic representation of the logic computation realized by the eight elementary two-input functions. **(B)** Heat maps of the elementary two-input functions. The x- and y-axis correspond to the two regulatory inputs of the gene. The color indicates the output, i.e., the relative activation of the gene (see legend). At an activation of zero (black), the gene is shut off, at an activation of one (white), it is maximally activated (the transcription rate is maximal). **(C-D)** Least squares fits of the standard sigmoid and the log-sigmoid model to the elementary two-input functions. **(E)** Mean square error of the fits. The standard sigmoid model approximates only OR- and NOR-type functions well. The log-sigmoid model provides a reasonable, phenomenological approximation for all linearly separable functions. Both models fail on non-linearly separable functions (EQUAL and XOR, results not shown).

model performs well only on OR- and NOR-type functions (independent regulation). In contrast, the log-sigmoid model provides a reasonable phenomenological approximation for both independent and synergistic regulation. Both models fail to express non-linearly separable functions (EQUAL and XOR). This is not a big drawback, because we expect such functions to occur rarely, if at all, in gene regulatory networks.[4] These observations extend to gene input functions with more than two regulators (results not shown).

The comparison with the elementary two-input functions helps to understand how the phenomenological models approximate different types of combinatorial gene regulation. However, biological input functions are likely to be intermediates between the elementary logic functions considered above. As a real biological example, we consider the input function of the *E. coli lac*-operon (an operon is a sequence of several genes that are transcribed together, i.e., whose mRNA is synthesized in one piece). The input function of the *lac* operon has been experimentally mapped out by measuring its activation with a GFP-plasmid system for different levels of its two inputs (Setty et al., 2003; Mayo et al., 2006). The two inputs are the signaling molecule cyclic AMP (cAMP) and the artificial inducer isopropyl β-D-thiogalactoside (IPTG). These molecules regulate the promoter activation indirectly via the two transcription factors cAMP

[4]Whereas OR-type and AND-type functions can be easily realized by biological promoters (Mayo et al., 2006), the non-linearly separable functions would be more difficult to evolve. In fact, we are not aware of an example of an EQUAL or XOR function realized at the level of a single promoter.

receptor protein (CRP) and LacI. cAMP binds to and activates CRP, which in turn activates transcription. IPTG binds to and deactivates the repressor LacI.

The input function of the *lac* operon and the fits of the two phenomenological models are shown in Figure 3.2. In contrast to the sigmoid model, the log-sigmoid model provides a reasonable qualitative approximation, though it can't describe the four characteristic plateaus. Note that more detailed kinetic models, which could fit the four plateaus more accurately (Setty et al., 2003; Mayo et al., 2006), are too complicated to be used for reverse engineering of gene networks.

In summary, the following points make the log-sigmoid model an interesting choice:

- The log-sigmoid model is identical to the standard sigmoid model, except that the logarithm of the inputs is used. This makes it compatible with many reverse engineering methods originally developed for the standard sigmoid model.
- In contrast to the standard sigmoid model, it approximates independent and synergistic combinatorial regulation of genes equally well.
- It is similar to a Hill-type model, the parameters k_i can be loosely interpreted as a dissociation constants and the weights w_{ij} as Hill coefficients.
- Some types of gene expression data are naturally treated in log space, as discussed in the next sections.

3.3 Benchmark networks and data

3.3.1 Synthetic-biology benchmark

The design of the IRMA synthetic-biology network provided by Cantone et al. (2009) for the DREAM2 challenge is illustrated in Figure 3.3. The network has been constructed in the yeast *Saccharomyces cerevisiae*. It consists of the five genes *SWI5*, *GAL80*, *ASH1*, *CBF1*, and *GAL4*, which code for the transcription factors Swi5, Gal80, Ash1, Cbf1, and Gal4. The network has a variety of different regulatory interactions, including transcriptional regulation and one protein-protein interaction (between Gal80 and Gal4), thus capturing some features of large eukaryotic gene networks at a small scale. Despite its small size, it has an intricate

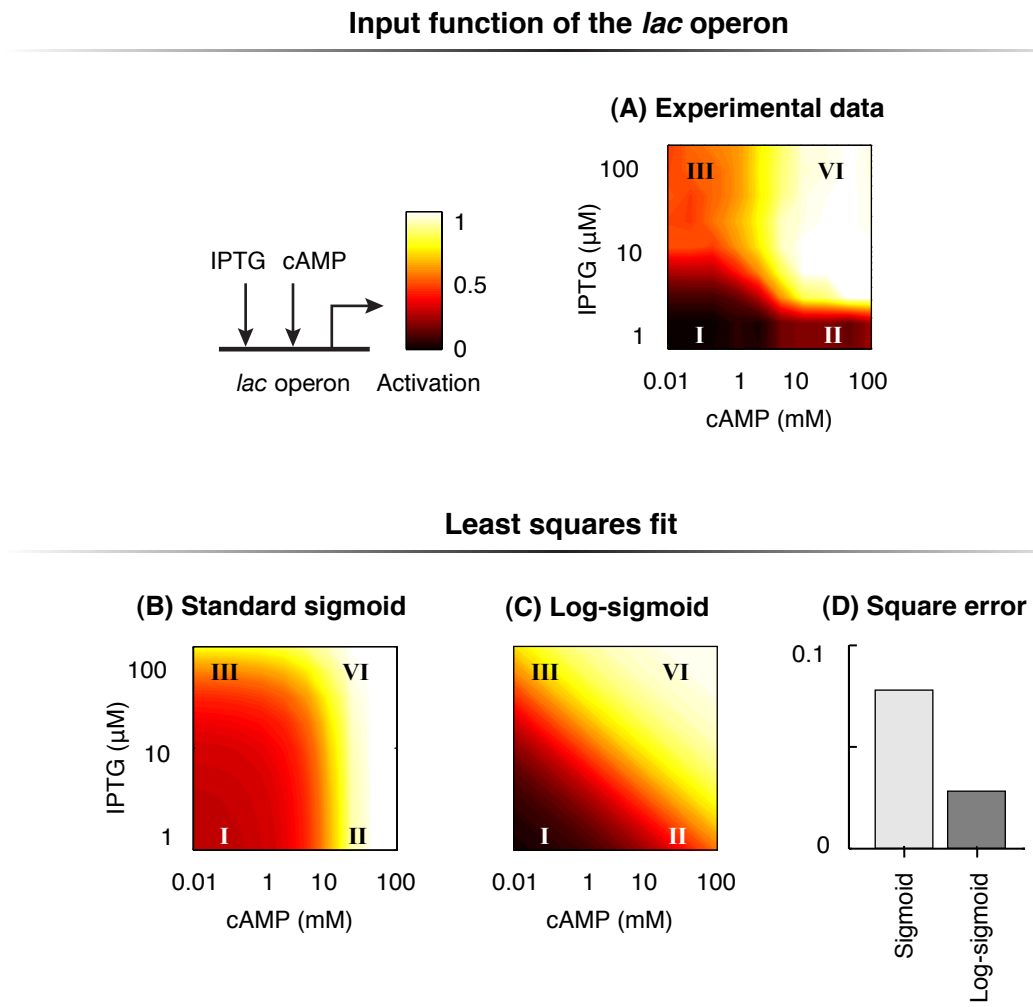


Figure 3.2: Comparison of the standard sigmoid and the log-sigmoid model on the input function of the *E. coli lac*-operon. **(A)** Activation of the *lac* operon measured as a function of cAMP and IPTG concentrations. The input function has four characteristic plateaus I-IV. **(B-D)** Least squares fits of the standard sigmoid and the log-sigmoid model to the *lac* input function. The standard sigmoid model fails to capture the logic of the *lac* operon. The fit with the log-sigmoid model is an intermediate between an AND-type and OR-type function. Consistent with the experimental data, promoter activity is zero in region I, below half-maximal activation in region II, above half-maximal activation in region III, and maximal in region IV.

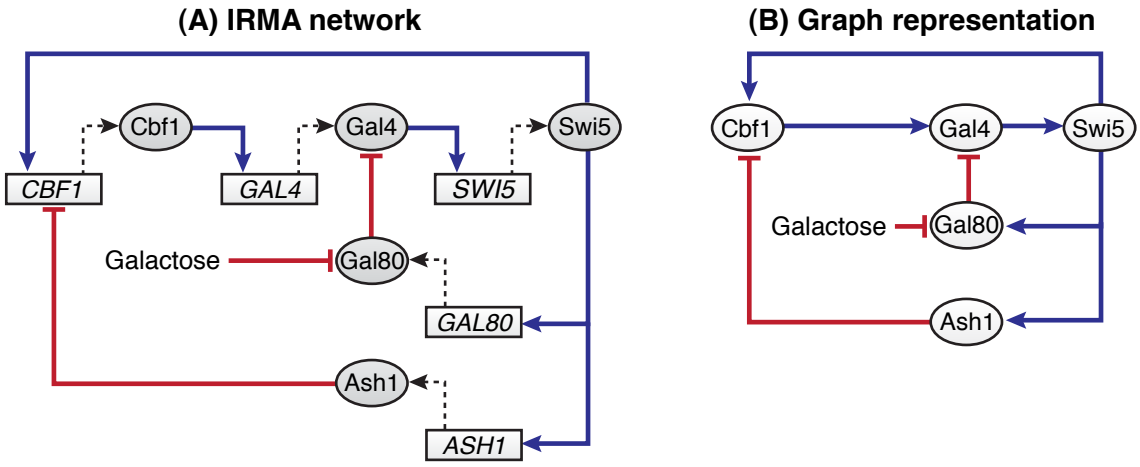


Figure 3.3: IRMA synthetic-biology benchmark network. **(A)** Genes are represented by rectangles, proteins by circles. Arrows are enhancing and T-ends inhibitory interactions. The network has one protein-protein interaction between Gal80 and Gal4. **(B)** Graph representation of the network. Figure adapted from Stolovitzky et al. (2009).

structure with a positive feedback loop (*GAL4* → *SWI5* → *CBF1* → *GAL4*) and two negative feedback loops (*GAL4* → *SWI5* → *GAL80* → *GAL4* and *GAL4* → *SWI5* → *ASH1* → *CBF1* → *GAL4*), potentially allowing for interesting nonlinear dynamics. The IRMA network was designed so as to be completely independent from the endogenous yeast gene network, i.e., it is only negligibly affected by other genes of yeast (Cantone et al., 2009).

When the yeast is cultured in a glucose medium, the IRMA synthetic-biology network is shut off because the Gal80 repressor binds to and inactivates Gal4. The network can be "switched on" by shifting the cells from glucose to galactose, which inhibits the Gal80 repressor, thereby activating Gal4 and the other genes of the network. For the DREAM2 challenge, Cantone et al. recorded two time-series of the network after such a "switch-on" shift by measuring the expression level of the five genes at different time points (Stolovitzky et al., 2009). Expression levels were measured using quantitative real-time polymerase chain reaction (q-PCR). The first time series consists of 15 samples, taken at intervals of 3 hours, and the second time series of 11 samples, taken at intervals of 5 hours. As mentioned above, only the two time series were provided to the participants of the DREAM2 challenge, and we neither knew the underlying network nor the nature of the perturbation that was applied to the network.

3.3.2 *In silico* five-gene repressilator

As a further test case, we have constructed an *in silico* gene network of the same size as the synthetic-biology network of the DREAM2 challenge. The structure this network is a loop of five inhibitory connections. We call this network *five-gene repressilator* because of its similarity with the so-called repressilator of Elowitz and Leibler (2000) (a loop of three inhibitory connections). We chose this structure because it is the most simple possible wiring that supports rich temporal dynamics of the network, including oscillations (Elowitz and Leibler, 2000) (a method to generate network structures for *in silico* benchmarks in a more principled way will be introduced in the next chapter). The dynamical model of the five-gene repressilator is based on the log-simgoid model, it is included in Appendix B.

We used the five-gene repressilator to generate two time series with the same number of samples as the DREAM challenge dataset described above. Different levels of log-normal noise were added to the simulated data. We assume log-normal noise on the data because microarrays and q-PCR assess gene expression on a logarithmic scale. Hence, the measurement error is expected to be approximately log-normal. Furthermore, there is experimental evidence that biological noise in gene regulation may also be approximated by a log-normal distribution (Rosenfeld et al., 2005). The simulated gene expression datasets from the five-gene repressilator are available upon request.

3.4 Reverse engineering method

We used AGE with the log-sigmoid model for the network inference. In this section, we describe the setup of AGE for the experiments reported in this chapter, including the fitness function used to evaluate the networks. Finally, we explain how we analyzed the inferred networks to estimate confidence levels for regulatory link predictions, as required for the DREAM challenges.

3.4.1 Setup of AGE

As described in the previous chapter, preparing AGE for a reverse engineering experiment amounts to specifying the number of gene parameters and the fitness function that is used to evaluate the evolved networks. Here, we use the

log-sigmoid model. Thus, genes have three additional parameters besides the weights w_{ij}, namely the maximum transcription rate m_i, the parameter k_i, and the degradation rate λ_i (cf. Equation 3.3b).

The evolved gene networks are evaluated according to how well they reproduce the measured data. Let $\hat{x}_i^k(t)$ denote the estimated gene expression level of gene i at time point t of the k'th time series, obtained by simulating the evolved gene network,[5] and $z_i^k(t)$ the corresponding logarithmic expression level of the measured target dataset (i.e., the negative q-PCR log-expression ratio). The fitness f of the evolved network is then given by the sum of squares error

$$f = \sum_k \sum_i \sum_t (z_i^k(t) - \hat{z}_i^k(t))^2 \qquad \text{with} \quad \hat{z}_i^k(t) = \log_2(\hat{x}_i^k(t)) \tag{3.6}$$

Note that the square error is defined on a logarithmic scale, consistent with our assumption of log-normal noise (see previous section). In other words, we fit the networks to the original q-PCR log-expression ratios, without first transforming the data to a linear scale.[6]

3.4.2 Predictions and confidence levels

It is clear that with the noisy and relatively small datasets available, it is impossible to infer regulatory links with 100% certainty. Within this context, the goal of reverse engineering is not identifying a single "true" network, but rather making a set of predictions of regulatory links, which can have different confidence levels. Such a list of predictions is the official format in which reverse engineering results are to be submitted to the DREAM challenge.

If the reverse engineering problem is underdetermined by the available data, multiple runs of a stochastic inference method (as the evolutionary method pro-

[5]The gene networks are simulated by integrating the system of differential equations (3.3b). The measured expression levels at the first time point are used as initial conditions. We also tried to infer the initial conditions by encoding them as additional parameters of the genes in AGE, however, this had no effect on the accuracy of the inferred networks. Thus, we kept to the more simple solution of taking directly the first time points as initial conditions, even though this may be less robust to noise. Numerical integration is done using the Runge-Kutta-Fehlberg (4,5) method of the GNU Scientific Library (GSL, http://www.gnu.org/software/gsl).

[6]Using the *in silico* benchmark, we have confirmed that fitting the data on a logarithmic scale allows a more faithful reconstruction of the network in the presence of log-normal noise than fitting the data on a linear scale, as expected.

posed here) typically converge to different networks. From N runs, we thus expect to get a set of N different inferred networks. Analyzing such an ensemble of inferred networks to make predictions and assign confidence levels is not trivial. For now, we simply define the confidence level of a regulatory link as the fraction of times that it was present in the set of inferred networks. Enhancing and inhibitory links are counted separately, i.e., the predictions are signed. A detailed discussion of different methods to extract predictions and confidence levels from ensembles of inferred networks is the focus of Chapter 6.

3.5 Results

3.5.1 Evaluating the accuracy of predictions

We use the scoring metrics proposed by the organizers of the DREAM2 challenge to measure the quality of the network predictions. Here, we give only a brief description of these scoring metrics, a detailed discussion is available in Stolovitzky et al. (2009).

Network predictions are represented as lists of predicted edges with their assigned confidence levels, constructed in decreasing order of confidence from the most reliable to the least reliable prediction. The quality of such a list of edge predictions is defined as the area under the precision-versus-recall (PR) curve, a standard metric used in the field of machine learning (Davis and Goadrich, 2006).[7] The precision and the recall of the first k predictions of the list are defined as

$$\text{precision}_k = TP_k/k \tag{3.7}$$

$$\text{recall}_k = TP_k/P \tag{3.8}$$

where TP_k is the number of correct predictions (true positives) up to prediction k, and P is the total number of true links (positives) in the target network. PR curves are drawn by incrementing k from the first until the last element of the

[7] PR curves are closely related to the more widely known receiver operating characteristic (ROC) curves. However, since gene networks are sparsely connected, there are many more negatives (links that are not part of the true network) than positives (links of the true network), i.e., the dataset is highly skewed. In this situation, PR curves are more informative than ROC curves (Davis and Goadrich, 2006).

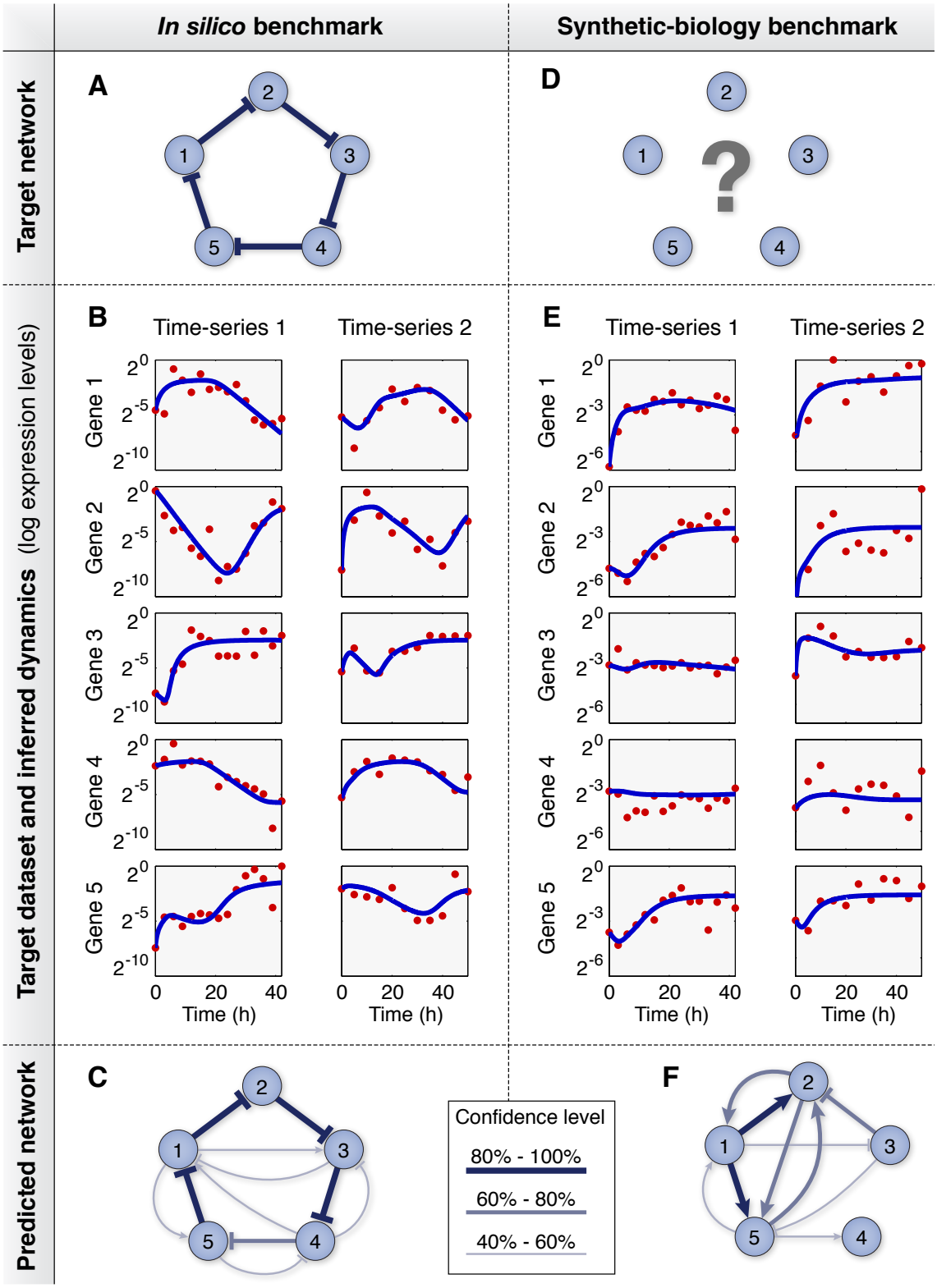


Figure 3.4: Caption on next page.

ranked list of predictions. The area under the PR curve is computed as described in Davis and Goadrich (2006).

The DREAM2 challenge had different categories for signed and unsigned predictions. For signed predictions, participants were asked to submit two separate lists, one for the predicted excitatory interactions and one for the predicted inhibitory interactions. The area under the PR curve was computed for both lists separately (Stolovitzky et al., 2009).

3.5.2 *In silico* benchmark results

We first applied the reverse engineering method to the *in silico* data from the five-gene repressilator at different levels of noise. The results obtained from a batch of 25 runs on the dataset with log-normal noise of standard deviation 1.0 are shown in Figure 3.4. All runs seemed to fit the noisy data reasonably well—the best run with a mean square error of 0.76 (the corresponding data fit is shown in Figure 3.4B) and the worst run with a mean square error of 1.04. Note that there is no overfitting to the noise, i.e., the dynamics of the inferred networks are close to the "true" noise-free dynamics of the repressilator (result not shown). Four out of the five inhibitory links of the target network were correctly inferred in over 95% of the runs. The fifth link was also correctly predicted, though with a

Figure 3.4: (A-C) *In silico* benchmark. **(A)** The structure of the *in silico* target network (the five-gene repressilator described in Section 3.3) is a loop of five inhibitory connections. **(B)** Normalized gene expression levels—plotted on a logarithmic scale—for the two time series. The points are the noisy data used as input for the reverse engineering method. The lines are the time courses of the inferred network with the lowest square error (best fitness). They fit the input data without overfitting to noise. **(C)** The regulatory interactions of the target network were correctly predicted with high confidence levels. Arrows are enhancing, T-ends denote inhibitory interactions. **(D-F) Synthetic-biology benchmark of the DREAM2 challenge**. **(D)** The topology of the *in vivo* target network was not disclosed prior to submission of our predictions. **(E)** The points are the dataset of the DREAM2 challenge, i.e., the experimentally measured expression levels (negative q-PCR log expression ratios). The lines are the time courses of the inferred network with the lowest square error. **(F)** Compared to the *in silico* benchmark, the predicted regulatory interactions have lower confidence levels, which indicated that they would be less accurate already before the true structure of the network was disclosed.

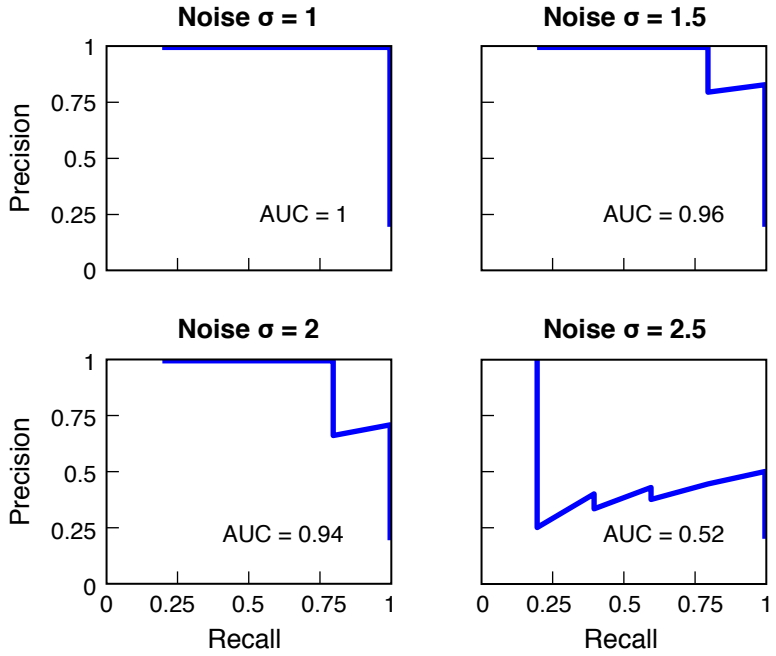


Figure 3.5: PR curves of the *in silico* benchmark at different levels of noise. The quality of the predictions is measured by the area under the curve (AUC). With log-normal noise of standard deviation σ = 1, AGE achieves a perfect network prediction (AUC = 1). Up to standard deviation σ = 2, the network predictions are still near perfect. For noise with standard deviation σ > 2, the performance degrades rapidly.

lower confidence level of 76%. All other links had confidence levels below 60% (Figure 3.4C). When adding more noise to the dataset (standard deviations 1.5 and 2.0), the network predictions were still near perfect: four of the five true links were correctly identified and put at the top of the list of edge predictions. With noise of standard deviation 2.5, the predictions were not accurate anymore. Thus, the performance seems to be quite resistant to noise up to a critical threshold (here standard deviation 2.0), after which it degrades rapidly. The PR curves at the different levels of noise are shown in Figure 3.5.

3.5.3 Synthetic-biology benchmark results

We launched 50 runs of the evolutionary reverse engineering method on the synthetic-biology benchmark of the DREAM2 challenge. The inferred networks fit the q-PCR data with a mean square error between 0.36 (best run) and 0.50 (worst run). The data fit of the best run is shown in Figure 3.4E. The predictions

Table 3.1: DREAM2 challenge rankings for excitatory, inhibitory, and undirected unsigned link predictions. The identities of teams have not been revealed, except for the best performers. There were nine teams in total (the numbers correspond to their identifiers on the DREAM website).

Excitatory links	
Team	AUC
Fuente et al.	0.72
AGE	**0.43**
Team 110	0.41
Team 40	0.41
Team 107	0.35
Team 60	0.17
Team 119	0.17

Inhibitory links	
Team	AUC
AGE	**0.14**
Team 107	0.13
Fuente et al.	0.12
Team 110	0.11
Team 40	0.06
Team 60	0.06
Team 119	0.06

Undirected unsigned	
Team	AUC
AGE	**0.87**
Team 40	0.79
Team 80	0.78
Team 110	0.48

of the regulatory interactions (Figure 3.4F), in particular those of the inhibitory links, have lower confidence levels than those of the *in silico* benchmark. Thus, already before the true structure of the DREAM2 network was disclosed, the estimated confidence levels indicated (correctly, as we will see below) that the predictions would be less accurate. Estimating confidence levels correctly is important in real biological applications to guide further experimental design. In this case, we would have concluded that a costly validation of the network prediction would be premature, and instead further gene expression data should be obtained from additional experiments to infer the network more reliably. Evidently, we didn't have the possibility to request additional data in the DREAM2 challenge, and we submitted our predictions for validation despite their relatively low confidence levels.

Predictions for excitatory and inhibitory interactions were evaluated separately by the DREAM organizers. In order to compare our approach with reverse engineering methods that produce undirected and unsigned network predictions, we removed the signs and directionality of our predictions (as described in Chapter 6) and participated in these categories of the DREAM challenge as well. The rankings of all teams for the different types of predictions, along with

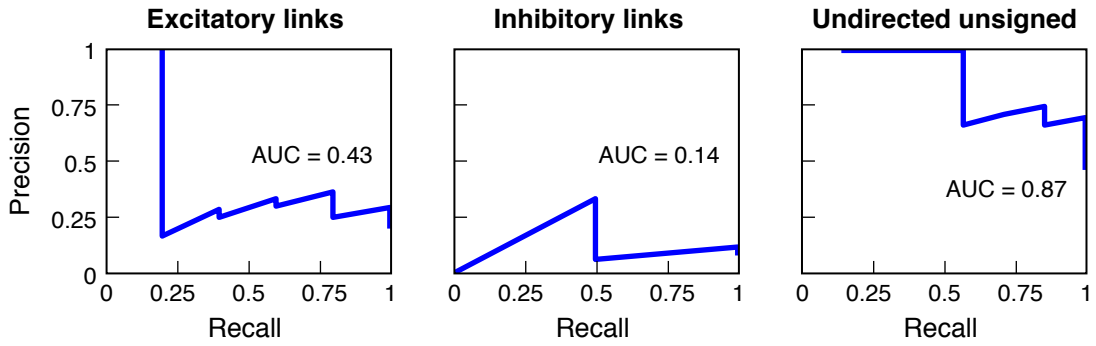


Figure 3.6: PR curves of the DREAM2 challenge predictions. The plots show the PR curves for excitatory, inhibitory, and undirected-unsigned link predictions obtained with AGE.

their area under the curve (AUC) scores, are given in Table 3.1. The corresponding PR curves are shown for AGE in Figure 3.6.

Overall, the evolutionary method based on AGE has a very competitive performance, ranking second for excitatory, first for inhibitory, and first for directed-unsigned link predictions. The DREAM organizers have statistically analyzed predictions by comparing them to a null hypothesis of random network predictions (Stolovitzky et al., 2009). The prediction of highest statistical significance of the challenge was obtained in the undirected unsigned category by AGE (the first four most confident edge predictions were all correct). Here, we focus our discussion on the directed signed predictions, which are the primary output of AGE. The other categories are discussed in Chapter 6. Additional information, including the areas under the ROC curves, are available on the DREAM website.[8]

Despite the competitiveness of our results, it has to be stressed that the accuracy of the predictions of all participating teams (including us) is rather low. According to the statistical test against random network predictions mentioned above, the predictions of inhibitory links were not significant at a level of 5% (Stolovitzky et al., 2009). However, note that our method assigned low confidence levels to these predictions, i.e., it correctly signaled that inhibitory links were difficult to identify from the provided data (Figure 3.4F). Even though the results of the other categories are significant according to this test, overall, the inferred networks are not close to the true topology.

[8]DREAM2 results: http://wiki.c2b2.columbia.edu/dream, we are team 55.

The method of Baralla, Mentzen, and de la Fuente achieved the most accurate predictions of excitatory links (Table 3.1) (Baralla et al., 2009). Their method is actually very similar in spirit to AGE:

- Both methods are based on nonlinear dynamical models. De la Fuente et al. use so-called S-systems, a model that may approximate—similar to the log-sigmoid model—different types of nonlinear dynamics (Voit and Savageau, 1987).

- Both methods fit the nonlinear models to the data using evolutionary algorithms. De la Fuente et al. use standard evolutionary algorithms of the biochemical simulation and parameter estimation software COPASI (Hoops et al., 2006), whereas we use AGE.

- Both de la Fuente et al. and us use an *ensemble approach* (see Chapter 6) to estimate confidence levels from a set of inferred networks. De la Fuente et al. ran their method on different variations of the data provided in the challenge: the first time series only, the second time series only, and both time series combined. These three combinations were fitted both on a linear scale and on a logarithmic scale, yielding six predictions of the network that were averaged to produce the final prediction. We only used one of these six possible variations (both time series combined, fitted on a logarithmic scale), but we run the evolutionary algorithm multiple times and combined the predictions from the different runs. In agreement with our results of Chapter 6, which show that the ensemble approach significantly improved our predictions, de la Fuente et al. confirmed that averaging over the six variations of the dataset significantly improved theirs. For example, when using only the variation that we used (both time series on a logarithmic scale), they obtained a similar AUC score (0.42) than AGE (0.43) for the excitatory interactions (Baralla et al., 2009) (see Chapter 6 for details).

We do not know the details of the methods used by the other participants of the challenge, as they remain anonymous. However, the organizers of the challenge conducted a survey among the participants to extract some lessons also from the strategies that performed sub-optimally, and they concluded:

> The best performers [de la Fuente et al. and AGE] modeled the five-gene network system *using the knowledge that the system consisted of a*

> *gene regulatory network*. Even though the details are different both best performers used kinetic models to represent the evolving system and found the best models that fit the data. The poorly performing methods, instead, used Bayes network inference and regression models and attempted to use probability theory to capture a causal kinetics that was probably better modeled by rate equations. (Stolovitzky et al., 2009) [emphasis added]

In summary, making use of the prior knowledge that the reverse engineering target is a gene network was the distinguishing feature of the best-performing methods,[9] which highlights the importance of devising effective strategies for this purpose. Here, we have proposed two such strategies, one for the design of the dynamical model and one for the design of the search method. Namely, the dynamical model was designed by considering different types of input functions that are known to occur in gene networks, and the search method was designed by mimicking the mechanisms that enable the evolution of complex gene regulatory networks in nature.

3.6 Conclusion

In this chapter, we described the application of AGE to a reverse engineering challenge of the DREAM2 conference, which was based on a synthetic-biology network in yeast. This has led us to consider more carefully the problem of how to choose between different phenomenological model types. To this end, we have proposed a principled approach, which consists in systematically comparing the performance of candidate models on different types of independent and synergistic transcriptional regulation. We have introduced a novel log-sigmoid model, which provides a better approximation to these different types of regulation than a standard sigmoid model. This was confirmed using experimental data from the *E. coli lac*-operon.

The reverse engineering benchmarks considered here are underdetermined by the available data. Hence, individual runs of the evolutionary algorithm converge to different networks that fit the data approximately equally well. Yet, we

[9] A third team also integrated prior knowledge, however, using a linear model and regression (Parisi et al., 2009).

have shown that by considering the set of inferred networks obtained from multiple runs, the network topology of the *in silico* benchmark could be successfully predicted despite high levels of noise in the gene expression data.

The evolutionary reverse engineering method based on AGE was the best performer of the DREAM2 five-gene-network challenge, obtaining the result of highest statistical significance and performing competitive in all categories. However, the accuracy of the predictions submitted by the participants in this challenge (including us) is not satisfactory.

The fact that several state-of-the-art reverse engineering methods, applied by different participating teams, have failed to predict the true topology with reasonable accuracy gives strong reason to believe that the network is not identifiable from the provided dataset. Even though the two time series of the dataset were obtained after application of the same initial perturbation (a shift from a glucose to a galactose medium), the dynamics are quite different in the two cases (Figure 3.4E). A possible explanation for the discrepancies between the two time series may be that the synthetic-biology network, unlike real biological networks, has not evolved to be robust to noise (Yokobayashi et al., 2002). However, an underlying assumption of deterministic models (used by de la Fuente et al. and us) is that the target network has a certain robustness towards noise, so that stochastic fluctuations are kept small. This assumption is reasonable for many biological networks, which have evolved to give reproducible responses to stimuli despite noise in gene expression and external conditions (Kitano, 2004). However, this assumption may be unfounded for synthetic-biology networks, which are extremely difficult to synthesize so as to perform reliably (Yokobayashi et al., 2002; Ellis et al., 2009). We believe that a high level of noise in the synthetic-biology network, which reduces the information content of the data and makes common deterministic models inadequate, is the most likely explanation for the low prediction accuracy of all participating teams.

These observations point to a possible pitfall of the synthetic-biology benchmarking approach, namely, the fact that a synthetic network has been constructed in a living cell does not necessarily imply that it is a biologically plausible network. Above, we have discussed the issue of robustness. A more obvious question is actually whether the structure of the network is biologically plausible. For example, the IRMA network provided by Cantone et al. (2009) for the DREAM challenge is certainly not representative for prokaryotic gene net-

works, which have a hierarchical structure with very few feedback loops at the transcriptional level (Ma et al., 2004b). The question whether the three tightly interlocked feedback loops of the IRMA network are a common wiring pattern in eukaryotic gene networks has not been considered by Cantone et al. (2009). In the next chapter, we address the problem of how to generate realistic network structures for synthetic benchmark networks. We use these network structures to generate *in silico* networks, however, the same approach may also be used to guide the design of biologically more plausible synthetic-biology networks.

4 Generating realistic *in silico* benchmark networks

This chapter is based on the following publication:

- Marbach, D., Schaffter, T., Mattiussi, C. and Floreano, D. (2009) Generating Realistic *In Silico* Gene Networks for Performance Assessment of Reverse Engineering Methods. *Journal of Computational Biology*, 16(2) pp. 229-239.

Synopsis

Reverse engineering methods are typically first tested on simulated data from in-silico networks, for systematic and efficient performance assessment, before an application to real biological networks. In this chapter, we present a method for generating biologically plausible in-silico networks, which allow realistic performance assessment of network inference algorithms. Instead of using random graph models, which are known to only partly capture the structural properties of biological networks, we generate network structures by extracting modules from known biological interaction networks. Using the yeast transcriptional regulatory network as a test case, we show that extracted modules have a biologically plausible connectivity because they preserve functional and structural properties of the original network. In the next chapter, we will describe how we applied this approach to generate the benchmark networks for the gene-network inference challenge of the DREAM3 conference.

4.1 Introduction

As discussed in the previous chapter, it is difficult to evaluate the performance of gene network inference methods *in vivo*, because it is seldom possible to systematically validate the predictions of unknown interactions (Figure 4.1A). We have seen that an elegant approach to circumvent this problem is using as benchmarks data from *in vivo* synthetic-biology gene networks. Unfortunately, only very small gene networks can be constructed with the current technology, and it is difficult to make them perform reliably. In fact, the IRMA network described in the previous chapter is currently the only available synthetic-biology

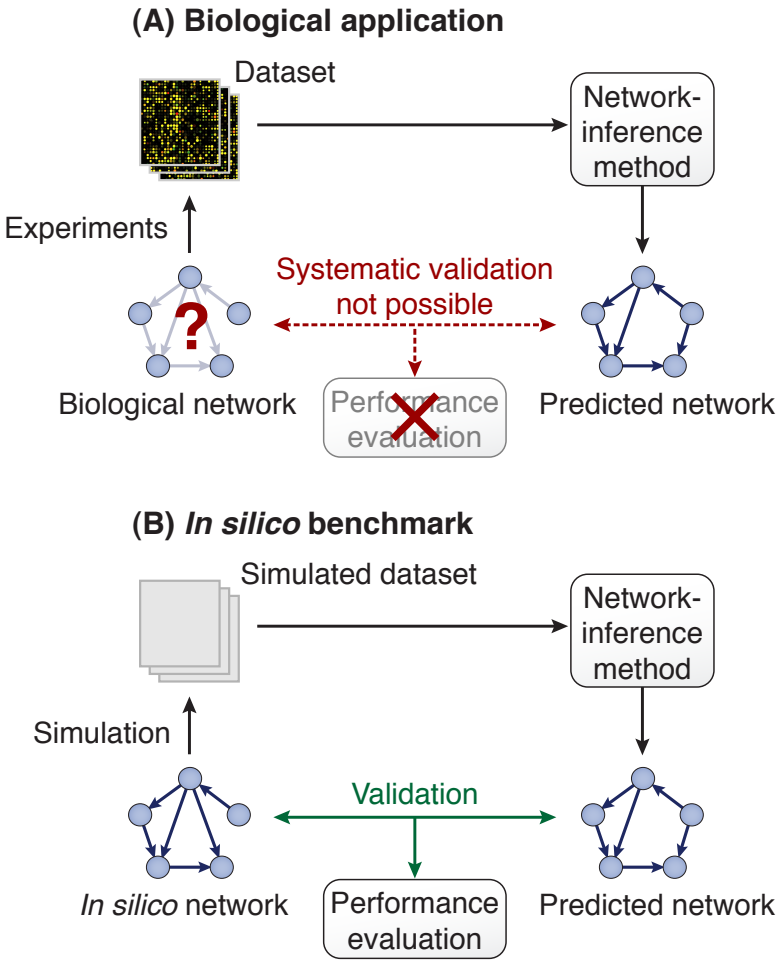


Figure 4.1: Validation strategies for network inference methods. **(A)** The "true" network structure of biological gene networks is in general unknown or only partly known, which hinders systematic performance evaluation. **(B)** Since the structures of *in silico* networks are known, predictions can be validated.

benchmark. Consequently, simulated data from *in silico* gene networks is often the only possibility for systematic performance assessment (Figure 4.1B). In simulation, all aspects of the networks and experiments are under full control. This allows characterization of reverse engineering methods for different types of data and levels of noise. In addition to performance assessment, *in silico* studies are of great relevance for optimal experimental design for subsequent real biological applications (Tegner et al., 2003).

However, results are only meaningful if the *in silico* benchmarks are biologically plausible. Creating such benchmarks involves: (1) generating realistic gene network structures, and (2) generating realistic data from these networks using adequate dynamical models. In this chapter, we consider the first problem, that is, how to generate network topologies with the same structural properties as real gene networks.

Apart from manual design of small benchmark networks (Zak et al., 2003; Tegner et al., 2003; Kremling et al., 2004), Erdös–Rényi and scale-free (Albert–Barabási) random graph models are currently the predominant approaches for generating *in silico* gene network structures (Mendes et al., 2003; van den Bulcke et al., 2006; Wildenhain and Crampin, 2006; Camillo et al., 2009; Haynes and Brent, 2009). However, random graphs capture only few of the structural properties of real biological gene networks (van den Bulcke et al., 2006). For example, scale-free random graphs approximate the power-law degree distribution of biological gene regulatory networks, but do not model other important properties such as modularity (Ravasz et al., 2002) or the occurrence of network motifs, which are statistically overrepresented circuit elements (Shen-Orr et al., 2002). Instead of constructing more complex random graph models, which would be difficult to justify and might wrongly favor some reverse engineering algorithms over others, we believe that the fairest way to compare reverse engineering methods is based on real biological network structures. Nowadays, rough drafts of the complete gene regulatory network of model organisms are available in dedicated databases (we call such networks *global interaction networks*). Global interaction networks can be used as "templates" for generating realistic network structures. Rice et al. (2005) have generated a single *in silico* network from the structure of the global *E. coli* transcription network. In order to generate multiple networks, which is necessary for collecting statistics on the performance of reverse engineering methods, van den Bulcke et al. (2006) extract

random subnetworks from global interaction networks.

We argue that global interaction networks should be sampled in a biologically meaningful way for generating plausible benchmarks. The approach introduced in this chapter is based on the extraction of modules, i.e., groups of genes that are more highly connected than expected in a random network. We show that topological modules extracted with the method described here correlate with functional modules of the global interaction network. Thus, the obtained network structures are realistic targets for reverse engineering, given that in a real application, one typically tries to reverse engineer the topology of a set of functionally related (and not randomly selected) genes.

4.2 Results

4.2.1 Module extraction from global interaction networks

We have devised a method to generate *in silico* network structures by extracting modules from a given global interaction network (the so-called *source network*). The modular subnetwork extraction method, or short *module extraction method*, starts from a seed node that is selected randomly among the nodes of the source network. From this seed, a subnetwork is grown by iteratively adding nodes to it until a desired size is reached. At each step, from all neighbors of the subnetwork, we select the node that leads to the highest modularity Q, where

$$\begin{aligned} Q = &(\text{number of edges within the subnetwork}) \\ &- (\text{expected number of such edges in a randomized graph}) \end{aligned}$$

The method outlined above, and described in detail in *Methods* (Section 4.4), can be applied repeatedly to extract different subnetworks of a desired size M from a source network of size $N \gg M$ by selecting different seed nodes.

Before applying the module extraction method to real biological networks, we demonstrate it on the hierarchical scale-free network model of Ravasz et al. (2002), which has a scale-free topology with embedded modularity similar to many biological networks. The network consists of a repeated four-node-motif, which is hierarchically grouped into clusters. As shown in Figure 4.2, the module extraction method tends to first add nodes from the four-node-motif of the seed, then it expands to other four-node-motifs of the same cluster, and only

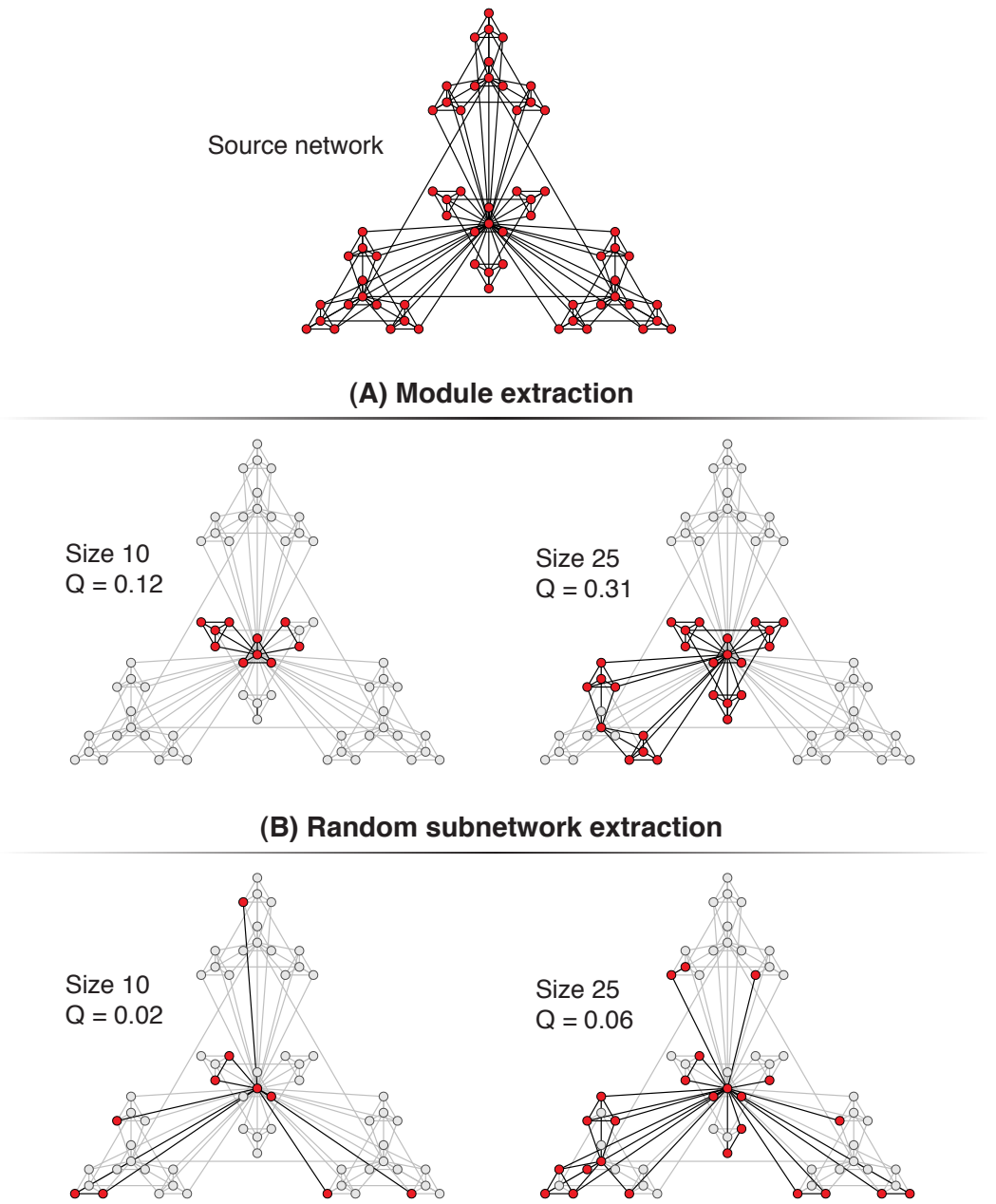


Figure 4.2: Module extraction vs. random sampling of a hierarchical scale-free network. Starting from the central hub (the seed node), subnetworks of size 10 and 25 are extracted. **(A)** Module extraction leads to subnetworks of high modularity Q. Most importantly, these subnetworks have the same functional building blocks and structural properties as the source network. **(B)** Randomly extracted subnetworks have a low modularity Q and very different structural properties than the source network.

if the desired size has not yet been reached it will start to include nodes from another cluster.

The example in Figure 4.2 illustrates how subnetworks obtained with module extraction are more representative samples of the source network structure than randomly extracted subnetworks (random subnetwork extraction is described in *Methods*). Similar to the source network, the subnetworks resulting from module extraction are built from four-node-motifs that are organized in a hierarchical modular structure. In contrast, the randomly sampled subnetworks do not share these structural properties with the source network.

It has been suggested that network motifs correspond to functional units of biological networks (Alon, 2007b). Assuming that the four-node-motifs in the artificial source network considered here also correspond to functional units (organized hierarchically into higher-level functional modules), the method described above allows the extraction of different combinations of such functional units and modules. In the following sections, we show that module extraction indeed permits sampling of functional units and modules in real gene regulatory networks.

4.2.2 Topological modules correlate with functional modules in the yeast gene network

As a test source network, we used the transcriptional regulatory network of the yeast *Saccharomyces cerevisiae* as described by Balaji et al. (2006). With 4,441 genes and 12,873 interactions, this network is one of the most comprehensive drafts of a eukaryotic transcriptional regulatory network so far.

We used module extraction and random subnetwork extraction to generate subnetworks of sizes 25, 100, and 400 from the yeast gene network. For each size, 20 subnetworks were generated starting from different randomly chosen seed nodes. We confirmed that the subnetworks obtained with the module extraction method described here indeed correspond to topological modules (densely interconnected subnetworks) of the gene network, as their modularity Q is positive and significantly higher than for randomly extracted subnetworks (Figure 4.3). For small networks of size 25 the difference of the mean modularity Q compared to randomly extracted subnetworks is smaller than for large subnetworks, but it is still statistically significant (p-value $< 10^{-6}$ using the Wilcoxon-Mann-

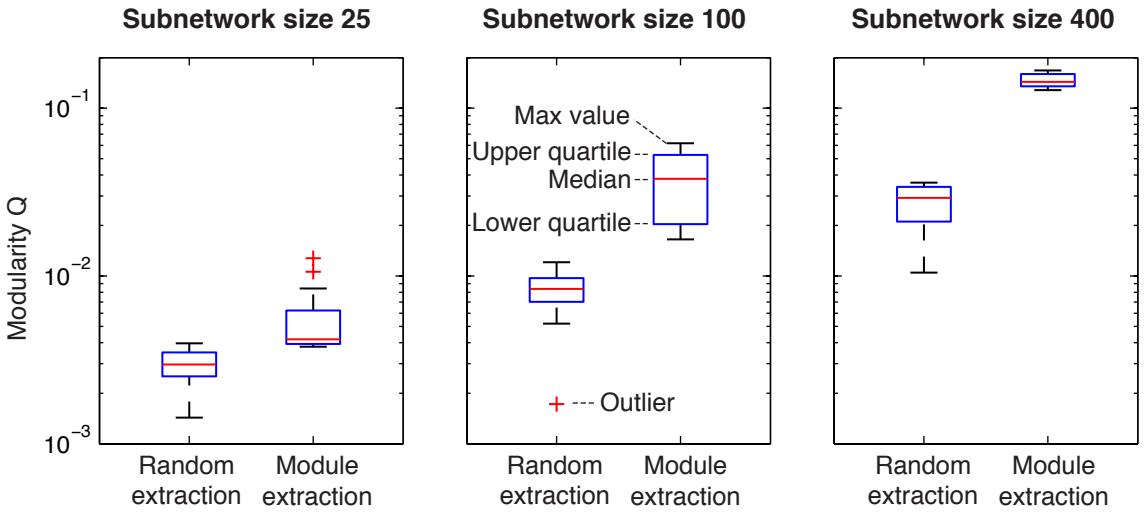


Figure 4.3: Boxplots for Q-values of subnetworks obtained with module extraction and random subnetwork extraction from the yeast gene network. As expected, module extraction leads to subnetworks with significantly higher modularity Q.

Whitney rank-sum test).

To check whether these topological modules also correspond to functional modules, i.e., groups of genes that have a related function, we considered the Gene Ontology (GO) functional annotation of the genes in the Saccharomyces Genome Database (Hong et al., 2008). For a given subset of genes (a subnetwork), we identified all GO terms that are enriched (statistically overrepresented), i.e., that occur more frequently than expected compared to their background frequency in the complete set of all genes (see *Methods*).

With module extraction, we find a high total number of enriched GO terms: 111, 282, and 444 for the 20 subnetworks of sizes 25, 100, and 400, respectively. For random subnetwork extraction, the total number of enriched GO terms is much lower: only 10, 9, and 112 for the same three network sizes. As can be seen in Figure 4.4A from the distribution of the p-values of these GO terms, module extraction leads not only to a higher number of enriched GO terms, but the enrichment also tends to be more significant.

In the previous paragraph we have looked at the *total* number of enriched GO terms of the 20 subnetworks for the three sizes. Let's now consider the number of such terms *per subnetwork*. The median number of terms per subnetwork is significantly higher for module extraction than for random subnetwork extraction (p-value $< 10^{-3}$ using the Wilcoxon-Mann-Whitney rank-sum test),

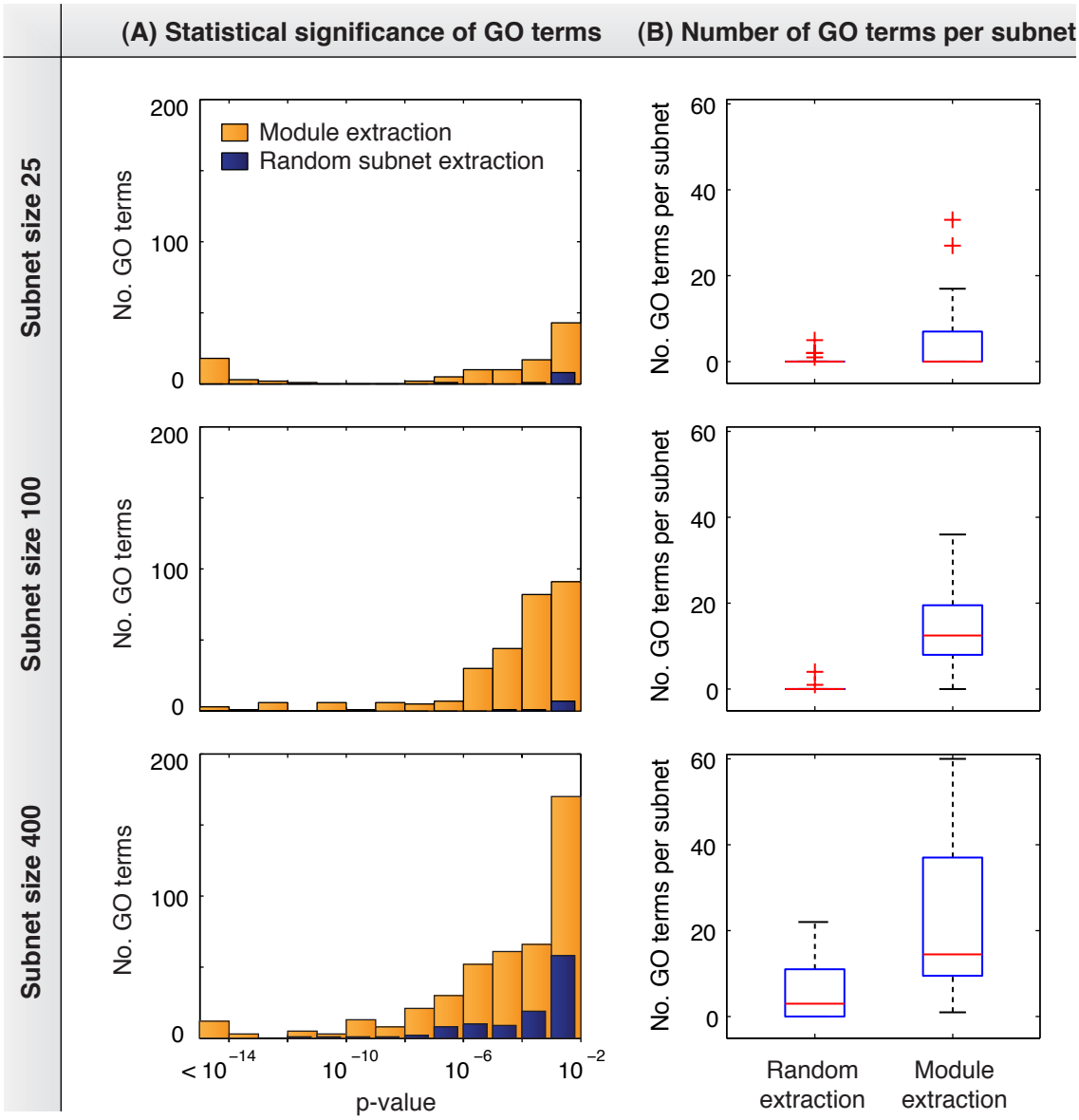


Figure 4.4: Enriched functional annotations (GO terms) in subnetworks obtained with module extraction and random subnetwork extraction from the yeast gene network. **(A)** Histograms with the total number of GO terms of 20 subnets of sizes 25, 100, and 400. Module extraction leads to a higher number of enriched GO terms, and the enrichment tends to be more significant (lower p-values). **(B)** Boxplots (see Figure 4.3 for legend) of the number of enriched GO terms per subnetwork. The median number of enriched terms is significantly higher with module extraction, except for small subnetworks of size 25.

except for subnetworks of size 25, where both medians are zero (Figure 4.4B). Remember that for small subnetworks of size 25 the modularities Q obtained with module extraction are not very high and only slightly superior to those of random subnetworks (Figure 4.3). Thus, it is not surprising that for this network size, there are also fewer enriched GO terms than for larger subnetworks.

In summary, our results demonstrate that module extraction is a more biologically meaningful way of sampling gene networks than random subnetwork extraction (especially for medium and large subnetworks). Moreover, they confirm the hypothesis that topological modules correspond to functional entities of gene regulatory networks, as treated in more detail in *Discussion* (Section 4.3). As an example, Figure 4.5 shows the structure and function of two extracted modules.

4.2.3 Module extraction preserves structural properties of the yeast gene network

After having studied the functionality of extracted subnetworks, we turned our attention to their structural properties. First, we considered the degree distributions[1]. We found that both modularly and randomly extracted subnetworks have a very similar degree distribution as the complete yeast gene network: the Pearson correlation between the degree distribution of the complete network and the mean degree distribution of the 20 subnetworks of size 400 is 0.92 for module extraction, and it is 0.93 for random subnetwork extraction (Figure 4.6). Thus, both strategies yield network structures with a biologically plausible degree distribution.

In the previous section we have shown that subnetworks obtained with module extraction correlate with functional modules, whereas randomly extracted subnetworks do not. It has been hypothesized that network motifs (statistically overrepresented sub-circuits) are functional building blocks of gene networks (Alon, 2007b). Thus, we would expect to find these motifs in the subnetworks obtained with module extraction, and not in the randomly sampled subnetworks.

To verify this, we compared the subnetworks based on their *triad significance profile* (Milo et al., 2004), which indicates for each three-node-motif the degree

[1]The degree distribution $P(k)$ is defined as the fraction of genes that have k connections (degree k).

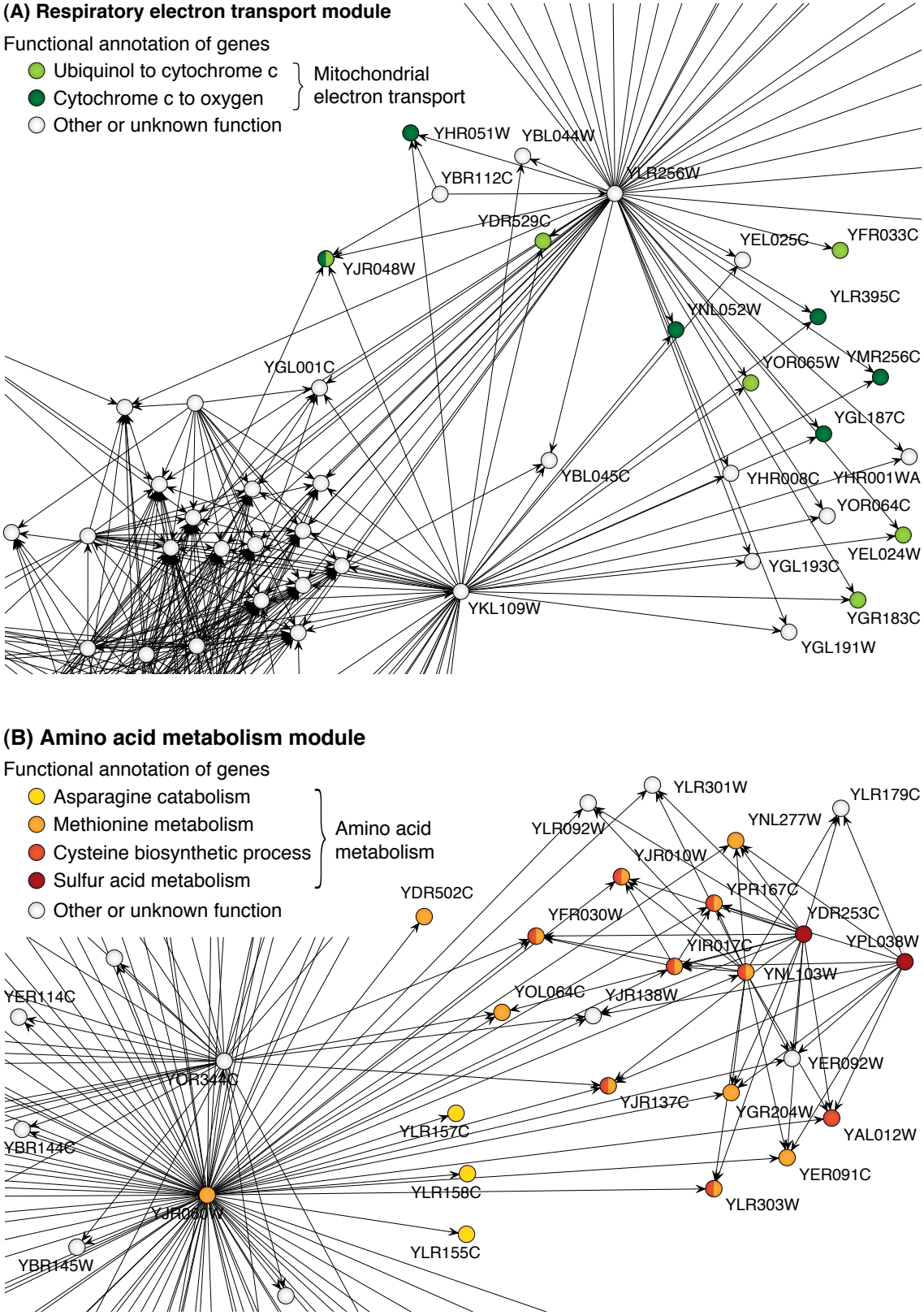


Figure 4.5: Caption on next page.

Figure 4.5: Two example subnetworks of size 100 obtained with module extraction from the yeast gene network. We have zoomed in on two groups of functionally related genes (colored circles), illustrating two different types of modules. The functional annotation of genes is taken from the Saccharomyces Genome Database (Hong et al., 2008) and p-values are calculated as described in *Methods*. **(A)** In the first subnetwork, genes related to mitochondrial electron transport are enriched with a p-value $< 10^{-10}$. The structure of this module can be described as a set of co-regulated genes. **(B)** The module of the second subnetwork is a more complex regulatory circuit related to amino acid metabolism (p-value $< 10^{-9}$). Note that in both subnetworks there are additional functional modules, which are not shown here.

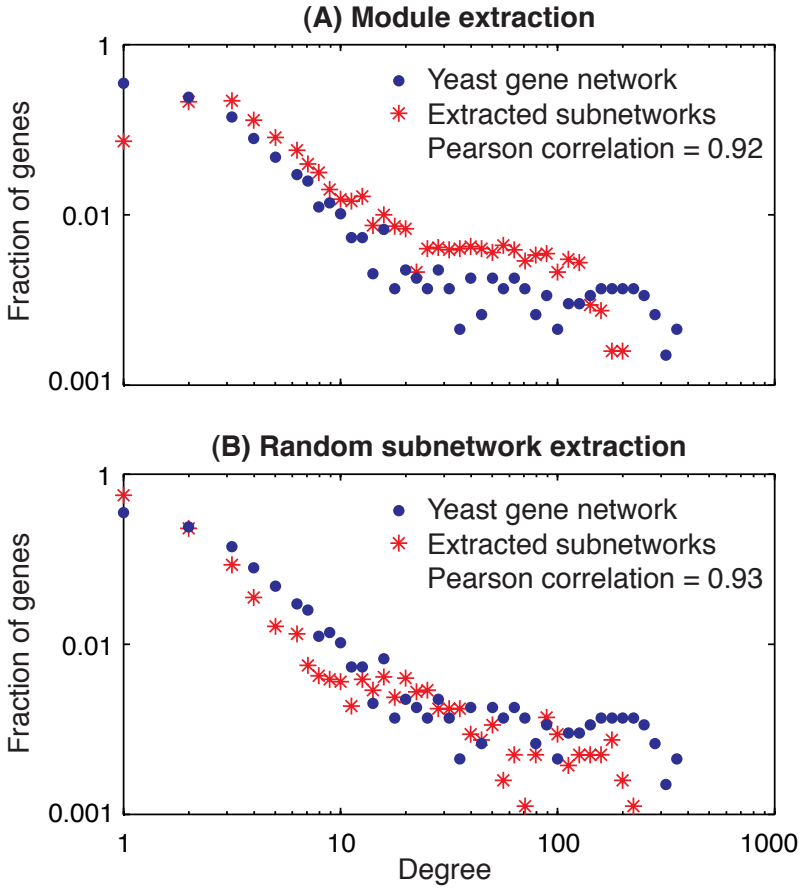


Figure 4.6: Degree distributions of modularly and randomly extracted subnetworks. The degree distribution of the complete yeast gene network compared to the mean degree distribution of 20 subnetworks of size 400 obtained with **(A)** module extraction and **(B)** random subnetwork extraction. Both strategies lead to subnetworks with similar degree distributions as the complete network.

to which it is statistically over or underrepresented (see *Methods*). As shown in Figure 4.7, the significance profile of the complete gene network is indeed very similar to the mean significance profile of extracted modules (it falls within the range of one standard deviation). In contrast, randomly extracted subnetworks have a completely different profile.

In summary, we have confirmed that subnetworks obtained with module extraction have a biologically plausible connectivity because they preserve functional and structural properties of gene networks such as degree distribution and network motifs. Incidentally, our finding that functional modules preserve network motifs, whereas non-functional subnetworks (random subnetworks) do not, supports the hypothesis that network motifs are functional building blocks of modules in gene regulatory networks.

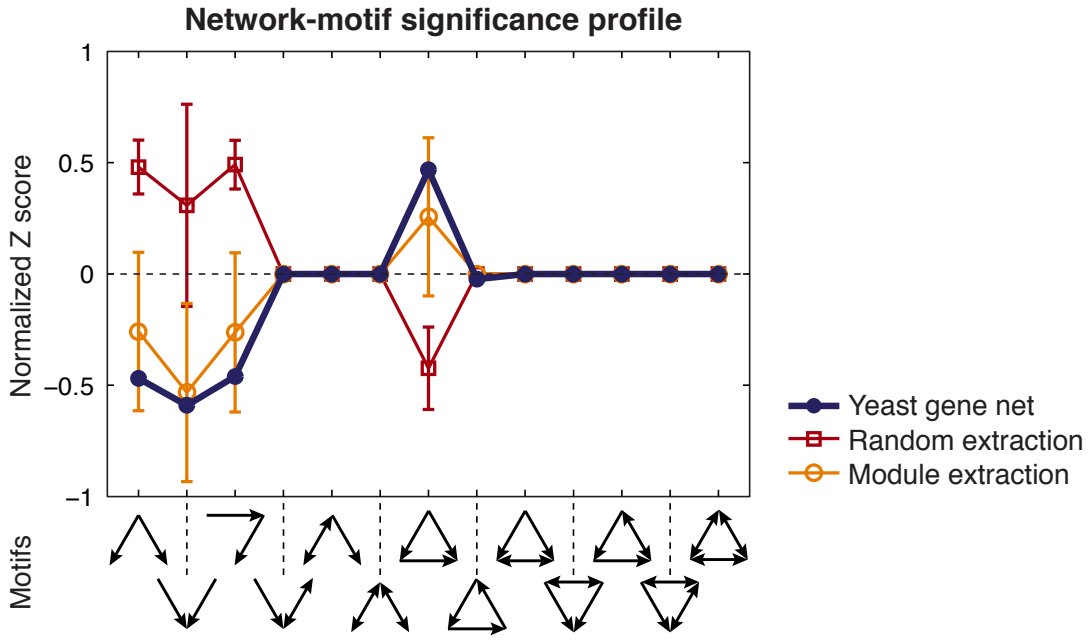


Figure 4.7: Network-motif profiles of modularly and randomly extracted subnetworks. The thick line is the triad significance profile (SP) of the complete yeast gene network. The thin lines correspond to the the mean SP of 20 subnetworks of size 400 obtained with module extraction and random subnetwork extraction. The error bars indicate the standard deviation. The SP of the complete network falls within the range of one standard deviation from the mean SP of modularly extracted subnetworks. In contrast, the SP of randomly extracted subnetworks is opposite to the SP of the complete network.

4.3 Discussion

We have presented a method that permits the generation of biologically meaningful network structures for performance assessment of reverse engineering methods. The method is based on the extraction of topological modules from global interaction networks. In this chapter we have focused on gene regulatory networks, but the same approach could be used for other types of cellular networks.

Using the yeast transcriptional regulatory network as a test case, we have shown that topological modules extracted with the method described here have a high number of enriched functional annotations, indicating that they correlate with functional modules of this network. Furthermore, extracted modules preserve structural properties of the original network, such as degree distribution and network motifs. We conclude that subnetwork extraction is a biologically meaningful way of sampling gene networks both from a functional and structural point of view.

The approach described in this article was originally motivated by the hypothesis that topological modules in gene networks coincide with functional modules. Such a separation of functions into more or less structurally isolated modules is thought to favor network evolvability and robustness (Hartwell et al., 1999). Indeed, it has been shown that topological modules in the *E. coli* transcriptional regulatory network can be assigned specific functions (Resendis-Antonio et al., 2005; Ma et al., 2004a). Our results from the yeast network indicate that this may also be true for eukaryotic transcriptional regulatory networks, which have an increased number of interconnections and cannot be as clearly decomposed into distinct topological modules.

An example application of module extraction is the generation of realistic benchmark networks for the DREAM challenges. For the DREAM2 *in silico* network challenges (Stolovitzky et al., 2009), Mendes et al. (2003) provided one benchmark network with a random Erdös–Rényi structure and one with a random scale-free structure. The goal of the challenge was to infer the structure of these networks (which was not disclosed to the participants) from simulated gene expression data. Interestingly, the best-performing method of this challenge (Gardner et al., 2003) had a much better performance on the Erdös-Rényi network than on the scale-free network. Indeed, the performance of reverse en-

gineering methods can be very sensitive to the type of network structure that they are applied to (Rice et al., 2005; Wildenhain and Crampin, 2006; Mukherjee and Speed, 2008). We have confirmed this by showing that the performance of reverse engineering methods depends strongly on the frequency of the different motifs that occur in the target network (see next chapter). Thus, for a fair comparison of methods in the DREAM challenge, it is crucial that the benchmark networks have a realistic structure. We have applied the module extraction method to provide the *in silico* network challenges of DREAM3, which will be described in the following chapter.

Here, we have focused on the generation of realistic gene network structures. The design of dynamical models and initialization of numerical parameters to obtain biologically plausible network dynamics are equally important challenges (Nykter et al., 2006; Roy et al., 2008). We have developed a framework for this purpose (see next chapter), but module extraction can be employed with any dynamical modeling approach of choice. Note that the structure of biological networks often confers robustness towards perturbations and variations in kinetic parameters (von Dassow et al., 2000). Thus, network structures obtained with module extraction may actually facilitate the initialization of models with biologically plausible network dynamics and enable the design of truly realistic *in silico* reverse engineering benchmarks.

4.4 Methods

4.4.1 Subnetwork extraction

The module extraction method grows a subnetwork of desired size M starting from a seed node, which can be selected randomly or manually from the source network. The procedure starts with a subnetwork containing only the seed node. Additional nodes are added iteratively until the subnetwork reaches the desired size.

Nodes are selected for addition as follows. First, the set of all neighbors of the subnetwork is constructed (a neighbor is a node of the source network that is connected by a direct link to at least one node of the subnetwork). Second, we compute for each neighbor the modularity Q of the subnetwork after adding only this neighbor to the subnetwork. Finally, we select the neighbor

that obtained the highest modularity Q for addition to the subnetwork (if several neighbors obtained the same modularity, we randomly choose one of them).

The modularity Q is defined as the number of edges within the subnetwork minus the expected number of such edges in a randomized network with the same degree sequence (Newman and Girvan, 2004; Newman, 2006b)

$$Q = \frac{1}{4m}\mathbf{s}^T\mathbf{B}\mathbf{s} \tag{4.1}$$

where m is the total number of edges in the network, $\mathbf{s}$ is the index vector defining the subnetwork ($s_i = 1$ if node i is part of the subnetwork, $s_i = -1$ if not), and $\mathbf{B}$ is the so-called *modularity matrix* with elements $B_{ij} = A_{ij} - P_{ij}$. A_{ij} is the actual number of edges between node i and j, and $P_{ij} = k_i k_j / 2m$ is the expected number of edges in a randomized network (k_i being the degree of node i).

There exist methods to find a globally optimal decomposition of a complex network into a set of modules (Newman, 2006a). Here, our goal is not the identification of optimal modules, but the extraction of *diverse* subnetworks of *prespecified size* and reasonably high (not necessarily globally optimal) modularity Q. Classical modularity detection algorithms are not well suited for this purpose.

Note that neighboring seeds may converge to identical or very similar (overlapping) subnetworks. The diversity of subnetworks can be increased by adding some randomness to the module extraction: instead of always selecting the neighbor that leads to the highest modularity Q, one can randomly select among the top k percent of all neighbors. For $k = 100\%$, this amounts to the random neighbor addition strategy used by van den Bulcke et al. (2006). Varying k between 0% and 100% allows for tuning of the sampling strategy from pure module extraction to random subnetwork extraction.

Apart from the case of neighboring seeds mentioned in the previous paragraph, module extraction from different random seed nodes typically leads to very diverse subnetworks. In principle, every extracted subnetwork may be considered a realistic network structure for a reverse engineering benchmark because they all correspond to modules of a real biological network. In practice, one may have additional criteria and discard certain types of modules. For example, for the benchmark networks of DREAM3 we did not include subnetworks that only contained a global regulator and its direct targets, because this is not a very challenging network structure for a reverse engineering benchmark.

4.4.2 Identification of enriched functional annotations

We identify enriched functional annotations in a subset of genes (a subnetwork) using the GO::TermFinder tool (Boyle et al., 2004). GO::TermFinder calculates p-values to determine whether any GO term occurs more frequently in the subset of genes than expected by chance. The p-value of a term corresponds to the probability of obtaining an equal or greater frequency of this term when randomly selecting genes from the background set of genes (in our case, the background set is the set of all genes of the network). Bonferroni correction is used for multiple hypothesis testing.

4.4.3 Network motif significance profiles

The triad significance profile (SP) indicates for each type of three-node-subgraph whether it is over or underrepresented in a given network. The statistical significance of triad i is measured by its Z score

$$Z_i = \frac{N\text{real}_i - \langle N\text{rand}_i \rangle}{\text{std}(N\text{rand}_i)} \tag{4.2}$$

where $N\text{real}_i$ is the number of times the triad occurs in the network. $\langle N\text{rand}_i \rangle$ and $\text{std}(N\text{rand}_i)$ are the mean and standard deviation of its occurrences in an ensemble of randomized networks with the same degree sequence. The SP corresponds to the normalized Z vector. For details, refer to Milo et al. (2004).

5 *In silico* performance assessment in a community-wide experiment

This chapter is based on the following manuscript:

- Marbach, D., Prill, R. J., Schaffter, T., Mattiussi, C., Floreano, D., and Stolovitzky, G. Critical assessment of methods for gene network inference. *In preparation*.

Synopsis

Due to the difficulties of evaluating the performance of gene-network inference methods discussed in the previous chapters, their comparative performance remains poorly understood. Here, we introduce a framework for critical performance assessment of network inference methods. We assess the performance of 29 methods, which have been independently applied by different teams to the same in-silico benchmark suite. Performance profiling on individual network motifs reveals that inference methods are affected, to various degrees, by three types of systematic prediction errors. In particular, all but the best-performing method failed to accurately infer multiple regulatory inputs (combinatorial regulation) of genes. The results of this community-wide experiment give unique insight into the capabilities and limitations of current network inference methods. Furthermore, they confirm the competitive performance of AGE, which ranks third out of the 29 applied methods.

5.1 Introduction

Spurred by advances in experimental technology over the past decade, a plethora of modeling approaches and methods for gene network inference have been developed. However, the problem of rigorously assessing the performance of these methods has received little attention until recently (Stolovitzky et al., 2007). Consequently, we lack a clear understanding of the capabilities, limitations, and comparative performance of reverse engineering methods.

To foster a concerted effort to address this issue, the DREAM (Dialogue on Reverse Engineering Assessment and Methods) project was initiated (Stolovitzky et al., 2007, 2009). One of the key aims of DREAM is the development of community-wide challenges for objective assessment of methods for inference of biological networks. Similar efforts have been highly successful in the field of protein structure prediction (Moult et al., 2007). However, the design of such benchmarks for biological network inference is problematic. On the one hand, well-known networks cannot be used because their identity is not easily hidden from the participants to create "blinded" challenges. On the other hand, there is not yet a gold-standard experiment for establishing the ground truth (the true network structure) for unknown *in vivo* networks, as discussed in Chapter 3. Consequently, *in silico* benchmarks (i.e., simulated networks and data) remain the predominant approach for performance assessment of reverse engineering methods: in simulation, the ground truth is known and predictions can be systematically and efficiently validated (Mendes et al., 2003), as discussed in the previous chapter.

Here, we describe a gene-network reverse engineering challenge that we organized within the DREAM3 conference, held at the Broad Institute of MIT and Harvard in October 2008. The challenge is based on a series of *in silico* networks (Figure 5.1), which we created using a new approach for the generation of biologically plausible network structures and dynamics. Instead of using random graphs, we generate network structures by extracting modules from known gene networks of model organisms, as described in the previous chapter. Here, we introduce in addition a dynamical model that is biologically more plausible than previously used models in gene-network reverse engineering benchmarks (Mendes et al., 2003; van den Bulcke et al., 2006; Roy et al., 2008; Camillo et al., 2009; Haynes and Brent, 2009). This framework allows reverse engineering meth-

ods to be tested *in silico* on networks with similar types of structural properties and regulatory dynamics as occur in biological gene networks.

With 29 participating teams, the DREAM3 *in silico* challenge has become by far the most widely used gene-network reverse engineering benchmark in the community. The participants have submitted almost 400 network predictions, which we have evaluated in a double-blind manner (Figure 5.1).

The Java tool—called GeneNetWeaver (GNW)—that was used to generate the *in silico* benchmarks and to assess the predictions is available open-source at: `http://gnw.sourceforge.net`.

Here, we analyze the performance of all 29 applied methods. First, we as-

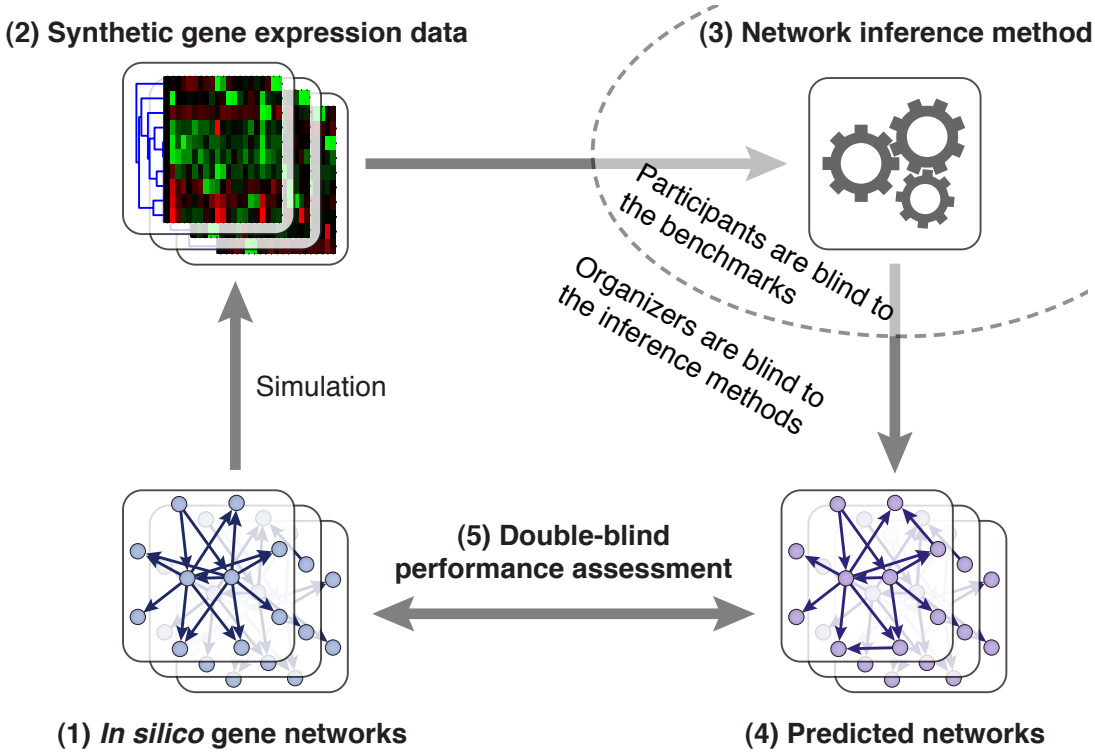


Figure 5.1: Double-blind performance assessment of reverse engineering methods. **Steps 1-2:** From a set of *in silico* benchmark networks (the so-called *gold standards*), steady-state and time-series gene expression data was generated and provided as a community-wide reverse engineering challenge. **Steps 3-4:** Participating teams were asked to predict the structure of the benchmark networks from this data. They were blind to the true structure of these networks. **Step 5:** We evaluated the submitted predictions, being blind to the inference methods that produced them. This allowed for a double-blind performance assessment.

sess the overall (global) prediction accuracy of the different network inference methods. We proceed by analyzing their performance profiles on local connectivity patterns (network motifs), which reveals their strengths and weaknesses in predicting different types of sub-circuits of the networks. Finally, we assess the performance of AGE on this benchmark suite.

5.2 Framework for critical assessment of network inference methods

For *critical assessment* of gene-network inference methods, it is essential that

- the benchmarks are blinded,
- the inference methods are evaluated on several networks to assess the statistical significance of results,
- in addition to the overall performance, the types of prediction errors and the relative strengths and weaknesses of the different methods are evaluated.

The framework that we propose for this purpose consists of: (1) tools for generating realistic *in silico* benchmarks, (2) an annual community-wide network inference challenge, and (3) tools for evaluating the performance and analyzing the prediction errors of the applied inference methods. In the following three sections, we briefly describe each of these elements.

5.2.1 Generating realistic *in silico* benchmarks

To assess the performance of gene-network inference methods *in silico*, it is essential that the benchmarks are biologically plausible. This involves generating realistic topologies for the benchmark networks, generating the corresponding kinetic models, and using these models to produce synthetic gene expression data by simulating different biological experiments.

We produce realistic network structures by extracting modules from known biological interaction networks. In the previous chapter, we have described this method and shown that it produces network topologies that preserve functional and structural properties of the original network.

Next, we generate kinetic models for the gene networks based on the extracted network topologies, as described in *Methods*. The kinetic models include both transcription and translation. Instead of using phenomenological terms, the input functions[1] of the genes are derived from thermodynamics (Ackers et al., 1982), which allows us to model the rich types of transcriptional regulatory logic that occur in biological gene networks (cf. Figure 3.1 A-B). In particular, both independent ("additive") and synergistic ("multiplicative") interactions occur in the networks.

Finally, we use the *in silico* gene networks to produce different types of steady-state and time-series gene expression data that are commonly used for gene network inference. For the DREAM3 *in silico* challenges described below, we generated the following types of data:

- **Wild-type steady state**. The steady-state gene expression levels of the unperturbed network.

- **Gene knockouts**. The steady-state levels of single-gene knockout (deletion) experiments. An independent knockout is provided for every gene of the network. A knockout is simulated by setting the transcription rate of this gene to zero. *In vivo*, such experiments are commonly done by deleting genes, manipulating their promoter sequences, or using RNA interference, for instance (Cantone et al., 2009).

- **Gene knockdowns**. The steady-state levels of single-gene knockdown experiments. A knockdown of every gene of the network is simulated. Knockdowns are obtained by reducing the transcription rate of the corresponding gene by half. The same techniques as mentioned above for the knockouts can be used to perform knockdowns *in vivo*. E.g., for heterozygous organisms, knockdowns can be obtained by deleting one of the two copies of a gene.

- **Time series**. The time series data shows how the network recovers from multifactorial perturbations. In contrast to the targeted *genetic* perturbations described above (knockout or knockdown of one gene at a time), a multifactorial perturbation affects the expression level of many genes at the

[1]The input function of a gene describes the combined effect of its regulators on the transcription rate, as described in Chapter 3.

same time. For example, this could be a *physical* or *chemical* perturbation applied to the cells (e.g., heat shock or a change of the growth medium), which would—via regulatory mechanisms not explicitly modeled here—increase or decrease the expression level of numerous genes by different amounts. At the first measured point of the time series (t=0), the network is in the perturbed state. At this time point the perturbation is removed, so the trajectories show how the gene expression levels go back from the perturbed to the wild-type state.

A more detailed description of the different types of experiments and gene expression data is given in *Methods*.

5.2.2 The DREAM *in silico* network challenge

Using the tools described in the previous section, we generated *in silico* benchmark suites that we released as community-wide challenges within the DREAM project. Here, we focus on the first edition, the so-called DREAM3 *in silico* network challenge, which was held within the DREAM3 conference. A description of the next edition (the DREAM4 *in silico* network challenge), for which teams are invited to submit predictions by October 2009, is available on the DREAM website (`http://wiki.c2b2.columbia.edu/dream`).

The DREAM3 challenge had three separate sub-challenges with networks of 10, 50, and 100 genes, respectively. For each size, five *in silico* benchmark networks were generated as described in the previous section. Two are based on topologies extracted from an *E. coli* transcriptional regulatory network (Shen-Orr et al., 2002), and three are based on topologies from a yeast genetic interaction network (Reguly et al., 2006). For each network, we generated steady-state and time-series gene expression data as described in the previous section.

The gene expression data was provided to the participants in the form of normalized mRNA concentrations with Gaussian noise (we did not simulate any particular measurement technology, see *Methods*). The protein concentrations were not provided. From the given gene expression datasets, participants were asked to predict the underlying networks, which were unknown to them. An archive containing the 15 benchmarks of the challenge is available on our website (`http://gnw.sourceforge.net`). In addition to the provided gene expression datasets and the topologies of the benchmark networks, the archive

also contains supplementary information such as the noise-free datasets, the protein concentrations, and the kinetic models of the gene networks. The kinetic models can be loaded with GNW to generate additional datasets, for example with other perturbations or types of noise than those used in the challenge.

5.2.3 Evaluating the performance of network inference methods

In the DREAM3 challenge, we exclusively evaluated the ability of inference methods to predict the presence of regulatory interactions between genes (some inference methods may predict additional aspects of the networks, which were not evaluated). Participants were asked to submit network predictions in the form of ranked lists of predicted edges. Each prediction was a list of edges ordered according to the confidence of the predictions, so that the first entry corresponds to the edge predicted with highest confidence (Figure 5.2B). A discussion of this format is given by Stolovitzky et al. (2009).

For networks of size 10, 50, and 100, the length of a complete list of predictions is 90, 2450, and 9900 edges (the number of possible edges in the network without autoregulatory interactions, which were not asked to be predicted). According to this format, a perfect network prediction is a list of all possible edges of the network, ordered such that the true edges of the target network are at the top of the list. We statistically evaluate predictions by computing a p-value indicating the probability that a random list of edge predictions would be of the same or better quality (see Figure 5.2 and *Methods*).

As mentioned in the previous section, each sub-challenge has five networks. To participate, teams were required to submit a prediction for each of the five networks. We computed an "overall p-value" by taking the mean of the p-values of the five network predictions. The final score used to assess the performance is the negative log-transformed overall p-value: for example, an overall p-value of 10^{-2} gives a score of 2, an overall p-value of 10^{-3} gives a score of 3. Thus, larger scores indicate smaller p-values, hence better predictions (see *Methods* for details).

This approach to measure the overall accuracy of network predictions will be applied in the next section to evaluate the performance of inference methods in the DREAM3 challenge. In Section 5.3.2, we will then introduce an approach to analyze the different types of prediction errors that network inference methods make.

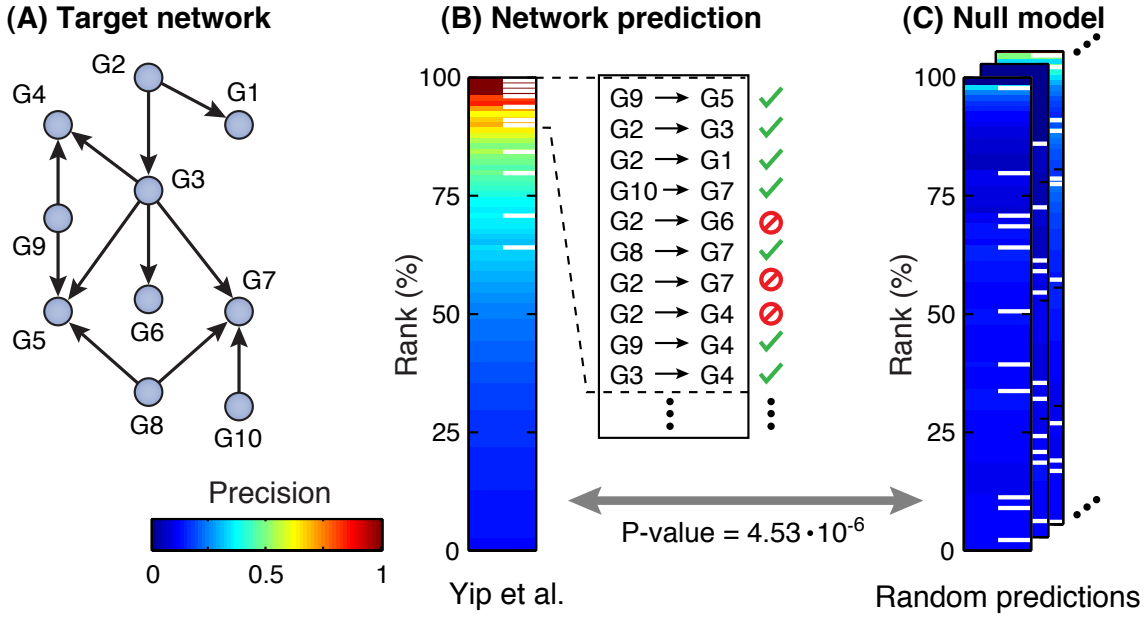


Figure 5.2: Example illustrating the evaluation of network predictions. **(A)** The true topology of the target network (the example shown here is the first network of the DREAM3 challenge). **(B)** Example of a submitted network prediction (it is the prediction of the best-performing team). The format is a ranked list of predicted edges, represented here by the vertical colored bar. The white stripes indicate the true edges of the target network. A perfect prediction would have all white stripes at the top of the list. The inset shows the first ten predicted edges: the top four are correct, followed by an incorrect prediction, etc. The color indicates the *precision* at that point in the list. Until the first four predictions, the precision is 1 (4 correct predictions out of 4 predictions). After the first ten predictions, the precision is 0.7 (7 correct predictions out of 10 predictions). **(C)** The network prediction is evaluated by computing a p-value that indicates its statistical significance compared to random network predictions.

5.3 Results

5.3.1 Performance assessment of 29 network inference methods

In total, 29 teams participated in the challenge. The majority of teams submitted predictions for all three network sizes (10, 50, and 100 genes): the corresponding sub-challenges had 29, 27, and 22 participants, respectively, which makes a total of 390 submitted network predictions (there are five networks of each size). According to a survey that we conducted among the participants, the applied inference methods span a wide range of approaches commonly used to reverse-engineer gene networks (data not shown), including statistical methods (Yip et al., 2009), Bayesian networks (Li et al., 2009), and methods based on dynamical models (Gardner et al., 2003; de la Fuente and Makhecha, 2006; Bonneau et al., 2007) (references describe methods that were applied by some of the teams that agreed to disclose their participation). The scores of the top ten teams for each sub-challenge are shown in Figure 5.3. The complete set of results, including the AUROC, AUPR, and p-values for all network predictions, is available on the DREAM3 website (`http://wiki.c2b2.columbia.edu/dream`).

The method of Yip et al. (2009), which will be further discussed in the following sections, obtained the best performance on all three network sizes. The overall p-values of their predictions are several orders of magnitude more significant than those of the next-best methods. On networks of size 100, the predictions of Yip et al.'s method were the only ones with a p-value below the numerical precision of Matlab (indicated as a score of infinity in Figure 5.3C). A representative prediction of Yip et al.'s method is shown in the example of Figure 5.2: most links are correctly recovered, but there are also some incorrect edges among the high-confidence predictions at the top of the list (the origin of these errors will become apparent in the network-motif analysis in Section 5.3.2).

As can be seen in Figure 5.3, several other methods achieved highly significant predictions. For example, on networks of size 100, the three next best teams after Yip et al. all had scores above 30 (i.e., overall p-values smaller than 10^{-30}). However, for the majority of methods the precision of the predictions is rather low (< 0.5, blue tones in Figure 5.3). In addition, a surprisingly large number of methods (11 out of the 29) produced, on average, network predictions that are not significantly better than random guessing (overall p-values > 0.01).

The performance of most methods is consistent on different network sizes.

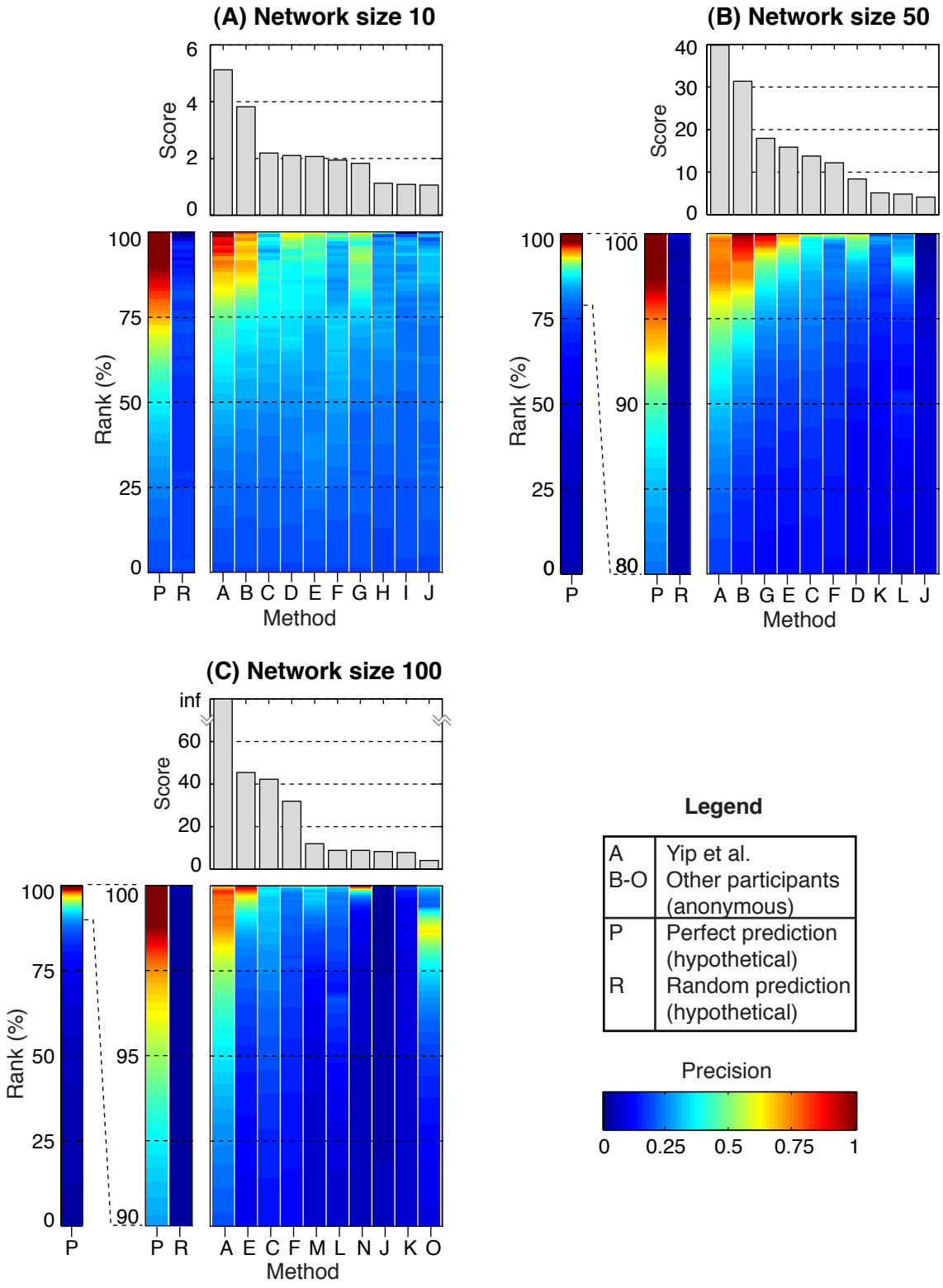


Figure 5.3: Caption on next page.

Figure 5.3: Performance of the best ten teams for each of the three sub-challenges. The bar plots on top show the overall scores, and the color bars below show the precision of the corresponding lists of predictions, as explained in Figure 5.2 (since each sub-challenge has five networks, this is the average precision of the five lists). In addition to the submitted network predictions (methods *A*-*O*), we always show the plots for a hypothetical perfect prediction *P* (all true edges at the top of the list) and a randomly generated prediction *R*, which allows to visually appreciate the quality of the submitted predictions. Remember that for networks of size 10, 50, and 100, the length of the lists is 90, 2450, and 9900 edges. Note that for networks of size 50 and size 100, we have zoomed in to the top 20% and 10% of the lists, respectively.

From the 29 methods that were applied to the networks of size 10, all but two were also applied to the networks of size 50. The two teams that participated only in the sub-challenge with networks of size 10 did not benefit from specializing on small networks. They ranked 22nd and 27th, and their predictions were not significantly better than random predictions (overall p-values of 0.24 and 0.44, respectively). After removing these two methods from the ranking, we compared the relative performance of methods on networks of size 10 and size 50 (Figure 5.4A). The methods of Yip et al. (2009) and Li et al. (2009) ranked first and second in both sub-challenges. The methods from ranks 3 to 7 are also the same for both network sizes, though not in the same order. Thus, the methods that performed best on the small networks of size 10 also performed best on the intermediate networks of size 50.

Similar observations can be made when comparing the ranking of the 22 methods that were applied both to networks of size 50 and 100 (Figure 5.4B). In this case, the correlation between the two rankings is even stronger. A notable exception is the method of Li et al. (2009), which ranked second on networks of size 10 and size 50, but ranked only 15th on networks of size 100. Even though the predictions of this method were of excellent quality for networks of size 50 (overall p-value $< 10^{-31}$), they were barely significant for networks of size 100 (overall p-value $= 0.01$). In summary, the ranking is similar in the three sub-challenges, with few exceptions. Thus, the performance of most methods was not dependent on the network size.

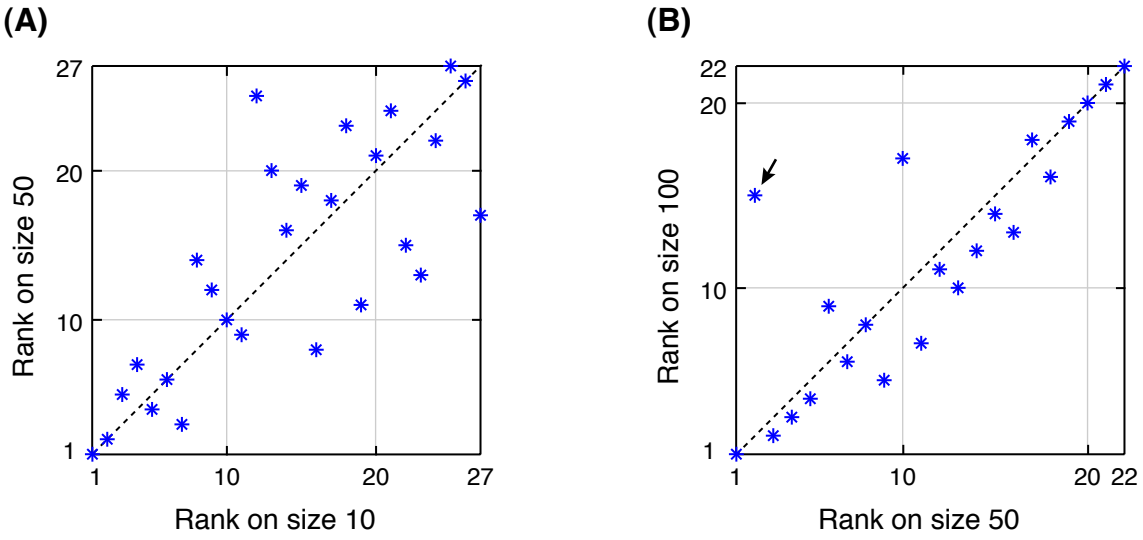


Figure 5.4: The performance of most methods is consistent on different network sizes. **(A)** The rankings of the 27 teams that participated both in the sub-challenges with networks of size 10 and 50. The methods with a good rank on networks of size 10 also performed well on networks of size 50 (points in the bottom-left corner of the plot). **(B)** The rankings of the 22 teams that participated both in the sub-challenges with networks of size 50 and 100. The ranking is very similar in the two sub-challenges. A notable exception is the method of Li et al. (2009), which had an excellent performance on networks of size 10 and 50 (2nd place) but a poor performance on networks of size 100 (15th place, indicated by the arrow).

5.3.2 Network-motif analysis reveals three types of systematic prediction errors

In the previous section, we have evaluated the overall performance of the applied inference methods. However, in order to understand the differences in performance of inference methods, we need to know what types of prediction errors they make. To this end, we have analyzed the performance of inference methods on the basic building blocks of networks, the network motifs (Milo et al., 2002). More precisely, we have analyzed how well the inference methods predict edges pertaining to different network motifs, which revealed systematic prediction errors.

We use the inference method applied by Madar and Bonneau (Bonneau et al., 2006, 2007), which ranked second on the networks of size 100, as an illustrative example to introduce the network-motif analysis. A more detailed description is given in *Methods*. The first column of Figure 5.5A shows the four types of motifs that occur in the benchmark networks of the challenge (fan-in, fan-out, cascade, and feed-forward loop), while the second column shows how well their links were predicted, on average, by the inference method of Madar and Bonneau. It can be seen that not all links of the motifs are predicted with the same median *prediction confidence* (see Section 5.5.4 for a definition)—some are predicted less reliably (at lower confidence) than others. Furthermore, some links tend to be incorrectly predicted, for example the "shortcut" (1→3) in the cascade motif.

In order to evaluate whether some edges of motifs were systematically predicted less reliably than others, we compared their median prediction confidence to the *background prediction confidence*, which is the median prediction confidence of all links of the network (independently of which motifs they belong to, see *Methods* for details). If the motifs had no effect on the prediction confidence, the edges pertaining to different motifs would all be inferred, on average, with the background prediction confidence. This is not the case, as can be seen in Figure 5.5C, which shows the divergence of the median prediction confidence of motif edges from the background prediction confidence. We evaluated the statistical significance of the divergence at a level of 0.01 using a two-sided Wilcoxon-Mann-Whitney rank-sum test and Bonferroni correction for multiple Hyothesis testing. We found three types of significant, systematic errors in the prediction of motifs (cf. Figure 5.5C):

- **The fan-out error** corresponds to a tendency to incorrectly predict edges between co-regulated nodes (2→3 and 3→2). The expression levels of co-regulated genes are often correlated. The fan-out error occurs when this correlation is wrongly interpreted as an interaction between the two genes.

- **The fan-in error** is a reduced prediction confidence for multiple inputs. In other words, fan-in links (2→1 and 3→1) are predicted less reliably than other links of the target network. This error is due to difficulties in accurately modeling and inferring *combinatorial regulation* of genes (regulation of genes by several inputs).

- **The cascade error** is a tendency for incorrectly predicted "shortcuts" in cascades. This error occurs when indirect regulation (1→2→3) is misinterpreted as direct regulation (1→3). In addition to the incorrectly predicted link (1→3), there is often a slightly reduced prediction confidence for the links (1→2) and (2→3).[2] This may be due to incorrect prediction of the shortcut (1→3) *instead of* (and not *in addition to*) the true links (1→2→3).

- The three links of **feed-forward loops** (FFLs) all have a reduced prediction confidence. This can be explained by the same types of systematic errors that occur in fan-ins and cascades. The links 1→3 and 2→3 of FFLs form a fan-in, and are thus affected by the fan-in error. The links 1→2→3 form a cascade, thus, they have a reduced prediction confidence for the same reason as the corresponding links in the cascade motif.[3]

We performed the network-motif analysis for all inference methods that were applied to the networks of size 50 and 100 (networks of size 10 are too small for

[2]In the example of Figure 5.5C, the reduced prediction confidence for the link 2→3 is not statistically significant at a level of 0.01, but for other inference methods we do observe a significantly reduced prediction confidence for this link (see Figure 5.6C).

[3]One minor effect remains to be explained: in FFLs, the prediction confidence was often slightly lower for edge 1→3 than for edge 2→3. For example, this is the case for all except Yip et al.'s method in Figure 5.6C. It seems that in addition to the fan-in error, which affects both these edges, the edge 1→3 is also affected by an additional, minor source of errors, which could be called the FFL-error. This error occurs when a method interprets the variation of gene 3 to be due solely to the indirect regulation via gene 2 (1→2→3), instead of both the indirect and the direct regulation (1→2→3 and 1→3). However, since this effect was negligible compared to the fan-in error that affects these edges, we did not consider it as a fourth type of main prediction error.

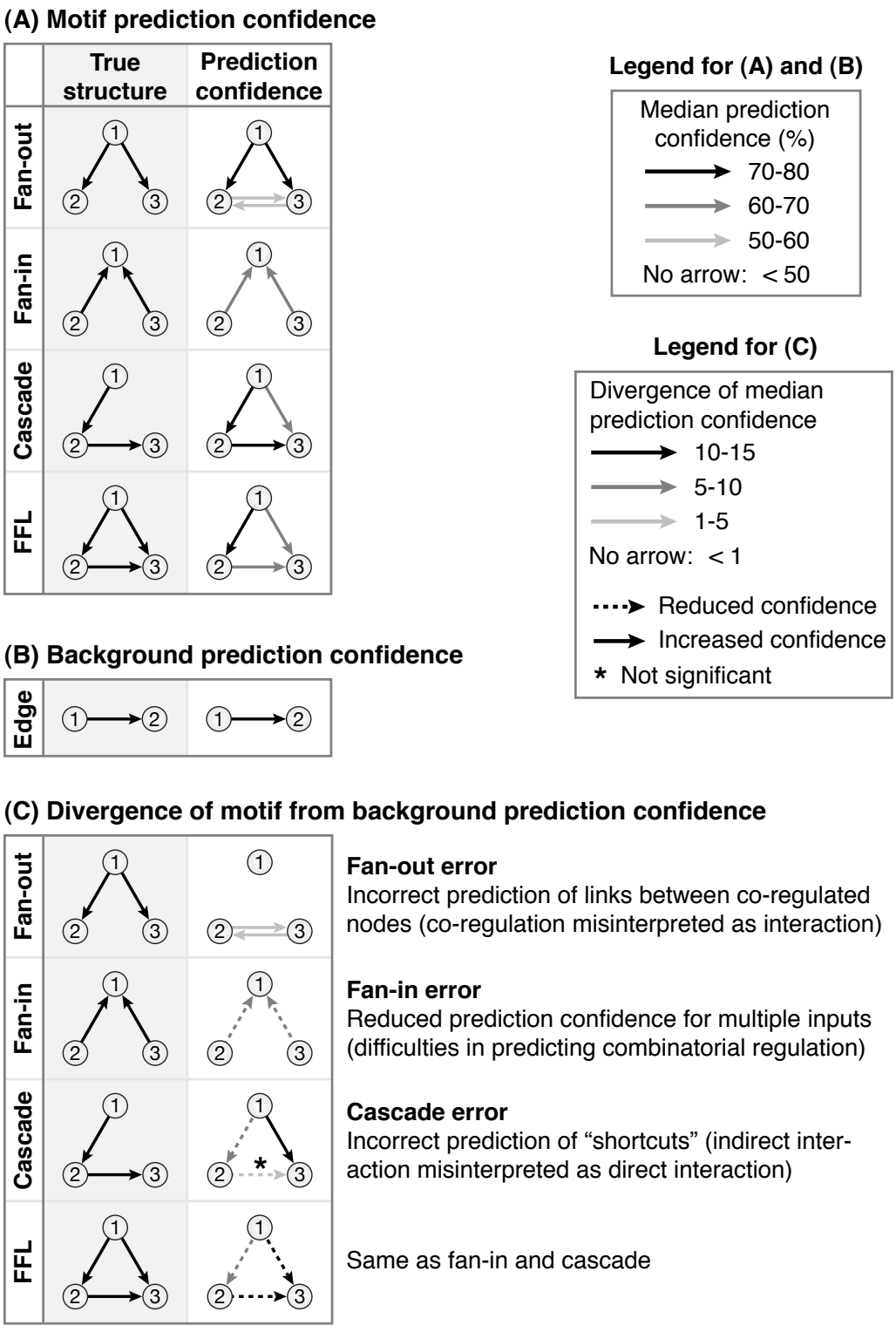
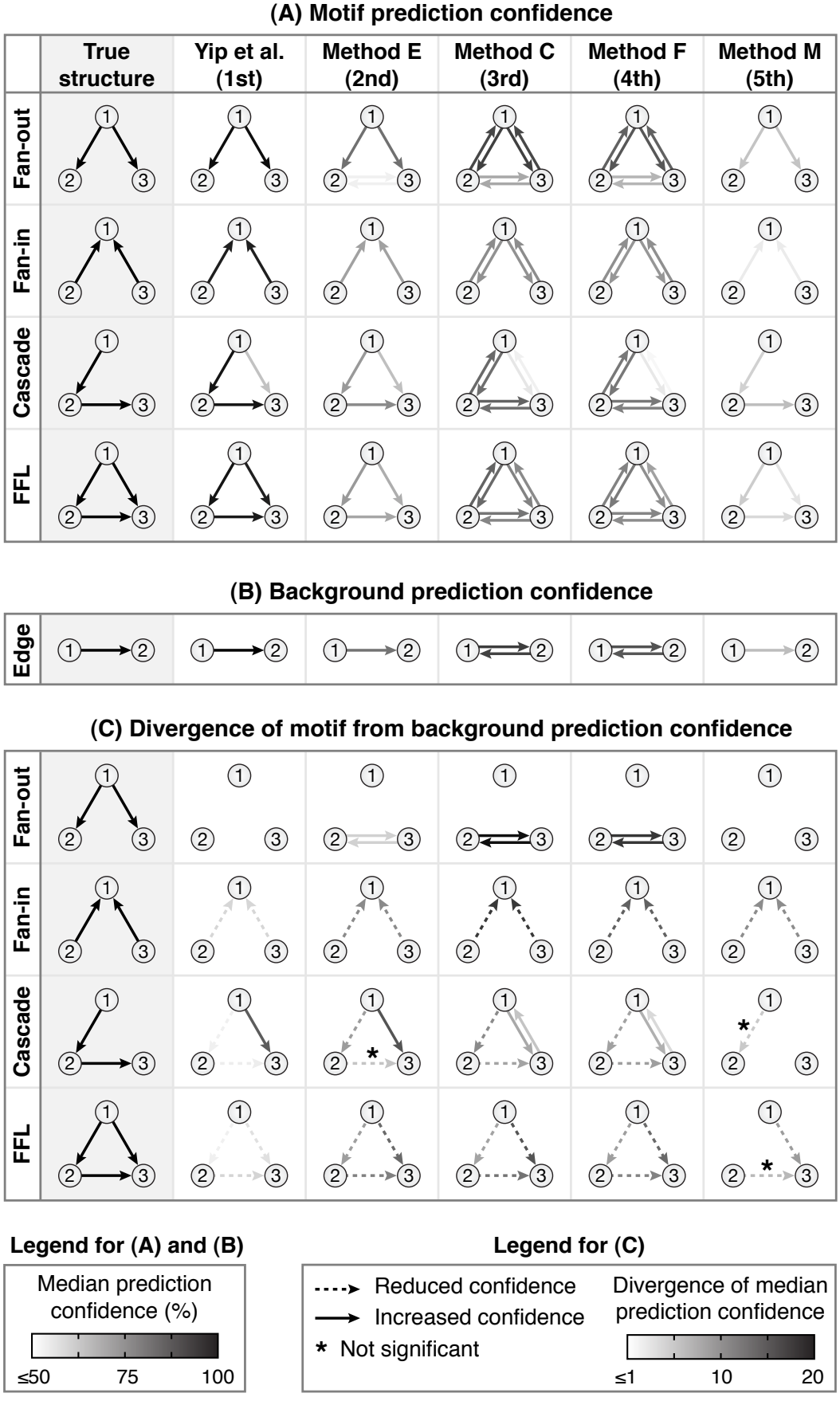


Figure 5.5: Caption on next page.

a statistically significant analysis). We did not observe other types of systematic errors than the three discussed above (except for some methods that failed to correctly infer the directionality of links). However, we found that inference methods are affected to various degrees by these errors—they have different *error profiles*. This can be clearly seen in Figure 5.6, which shows the network-motif analysis for the best five inference methods on the networks of size 100. Whereas some inference methods are more robust to certain types of error, they are more strongly affected by other types of errors. For example, only methods E, C, and F are affected by the fan-out error. All methods are affected by the fan-in error, but the method of Yip et al. to a lesser degree than the others. However, the method

Figure 5.5: Systematic errors in the prediction of motifs (figure on previous page). **(A)** The first column shows the true connectivity of the motifs. As an example, we show to the right how the motifs were predicted *on average* by the method that ranked second on the networks of size 100 (Bonneau et al., 2007). The level of gray of the links indicates their median prediction confidence. **(B)** The background prediction confidence, which is the median prediction confidence of all links of the network, independently of which motifs they belong to. **(C)** The divergence of the prediction confidence of the different motif edges from the background prediction confidence. The darkness of the links is proportional to the difference of the motif prediction confidence (A) from the background prediction confidence (B). Dashed arrows indicate a reduced, solid arrows an increased median prediction confidence. All differences are statistically significant (see main text), except for the marked edge (*). We can identify three types of systematic prediction errors: the *fan-out error* corresponds to a tendency for incorrectly predicted edges between co-regulated nodes; the *fan-in error* is a reduced prediction confidence for multiple inputs; and the *cascade error* is a tendency for incorrectly predicted "shortcuts" (1→3). On feed-forward loops (FFLs), we observe the same type of error as on fan-ins (reduced prediction confidence for multiple inputs). The reduced prediction confidence for links 1→2 in cascades and FFLs is discussed in the main text.

Figure 5.6: Inference methods are differentially affected by systematic prediction errors (figure on next page). Same as Figure 5.5, but for the best five inference methods on the networks of size 100. **(A)** Median prediction confidence of the different motif edges. Note that methods C and F apparently can't distinguish the directionality of edges. **(B)** The background prediction confidence. **(C)** Divergence of the median prediction confidence of motif edges (A) from the background prediction confidence (B). All differences are statistically significant (see main text), except for the three marked edges (*).


(A) Motif prediction confidence
True structure
Yip et al. (1st)
Method E (2nd)
Method C (3rd)
Method F (4th)
Method M (5th)
Fan-out
Fan-in
Cascade
FFL
(B) Background prediction confidence
Edge
(C) Divergence of motif from background prediction confidence
Fan-out
Fan-in
Cascade
FFL
Legend for (A) and (B)
Median prediction confidence (%)
≤50
75
100
Legend for (C)
Reduced confidence
Increased confidence
* Not significant
Divergence of median prediction confidence
≤1
10
20

of Yip et al. is more strongly affected by the cascade error than other inference methods (indeed, in the example prediction of this inference method shown in Figure 5.2B, the three incorrect predictions among the first ten edge predictions all correspond to "shortcuts" in cascades). Note that, since methods C and F can't infer the directionality of interactions, the wrongly predicted "shortcut" in cascades appears in both directions.

5.3.3 Most inference methods fail to accurately predict combinatorial regulation

The network-motif analysis of the previous section has shown that all inference methods are affected, to various degrees, by the fan-in error. The fan-in error is a reduced prediction confidence for multiple inputs (combinatorial regulation) of genes. Here, we analyze this type of error in more detail. Specifically, we compare how well, on average, inference methods predict the regulatory input(s) of genes with a single input (indegree 1), two inputs (indegree 2), three inputs (indegree 3), etc. The results of this analysis are shown in Figure 5.7A for the best five inference methods on networks of size 100. These data show that several methods predict single inputs of genes with high confidence. However, for all but the best-performing method, the prediction confidence degrades drastically as the number of inputs increases. For example, method F reliably identified links that are the only input of their targets (median prediction confidence 97%), but did no better than random guessing in predicting inputs of genes with indegree nine (median prediction confidence 46%).

As a control, we did the same analysis also for the outdegree of the genes. In contrast to multiple inputs of target genes, there is no particular reason why multiple outputs of regulators would make these edges more difficult to predict: in the network-motif analysis, we did not observe a reduced prediction confidence for edges in fan-outs (1→2 and 1→3 in Figure 5.6C). Indeed, in contrast to the indegree, the increasing outdegree of genes does not affect the prediction confidence of links (Figure 5.7B). The same observations can also be made on the networks of size 50 (Figure 5.8). Note that the results for the smaller networks of size 50 are noisier, and the confidence intervals wider, due to the smaller sample sizes.

Here, we have analyzed only the best five methods of the networks of size 50

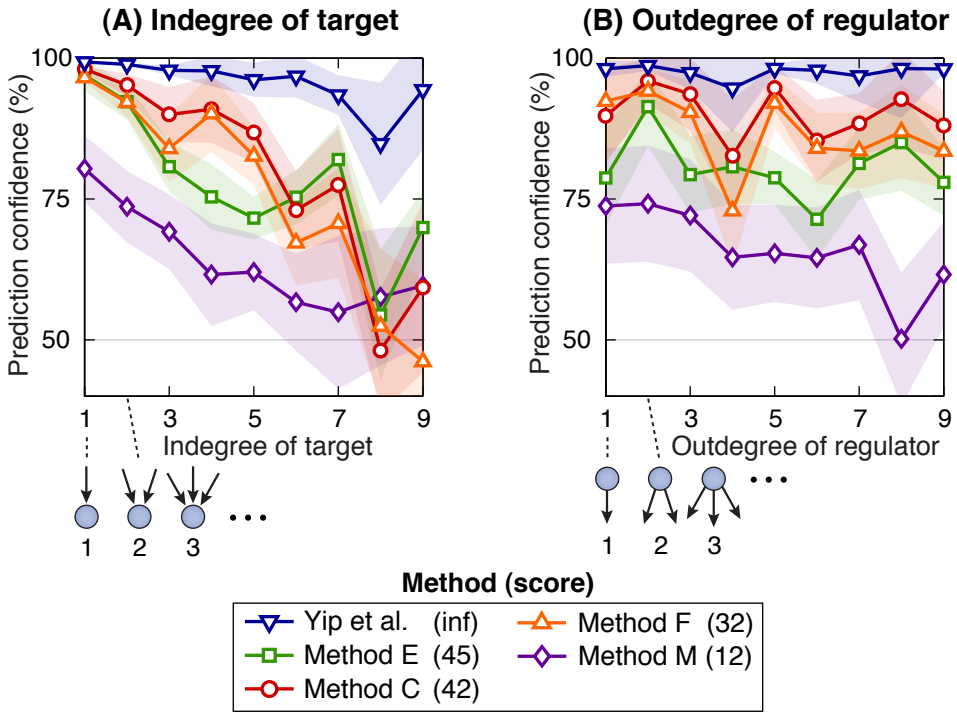


Figure 5.7: How the indegree and outdegree of genes affects the prediction confidence. The plots show the median prediction confidence for the best five methods on networks of size 100. The shaded areas indicate 95% confidence intervals for the medians. **(A)** Median prediction confidence for links that target genes of increasing indegree. Single-input links were reliably predicted with a similar, high prediction confidence by the best four methods (points in the top left corner). However, the performance of methods E, C, and F drops drastically for higher indegrees. In contrast, the best-performing method (Yip et al.) is less affected by the indegree of genes. **(B)** Median prediction confidence for outgoing links of regulators with increasing outdegree. In contrast to the indegree of target genes, the outdegree of regulators does not affect the prediction confidence of their links.

and 100. We have performed the same analysis for all inference methods of the two sub-challenges, which confirmed the observations reported here for the best five methods (results not shown). In particular, we find that Yip et al.'s method has the most robust performance of all inference methods on high indegrees.

It is not unexpected that edges that are the sole input of their target gene are easier to infer than edges towards genes with many inputs. If a gene has only one regulator, and this regulator is being perturbed, the gene would show a clear response. In contrast, if a regulator of a gene with other regulatory inputs is

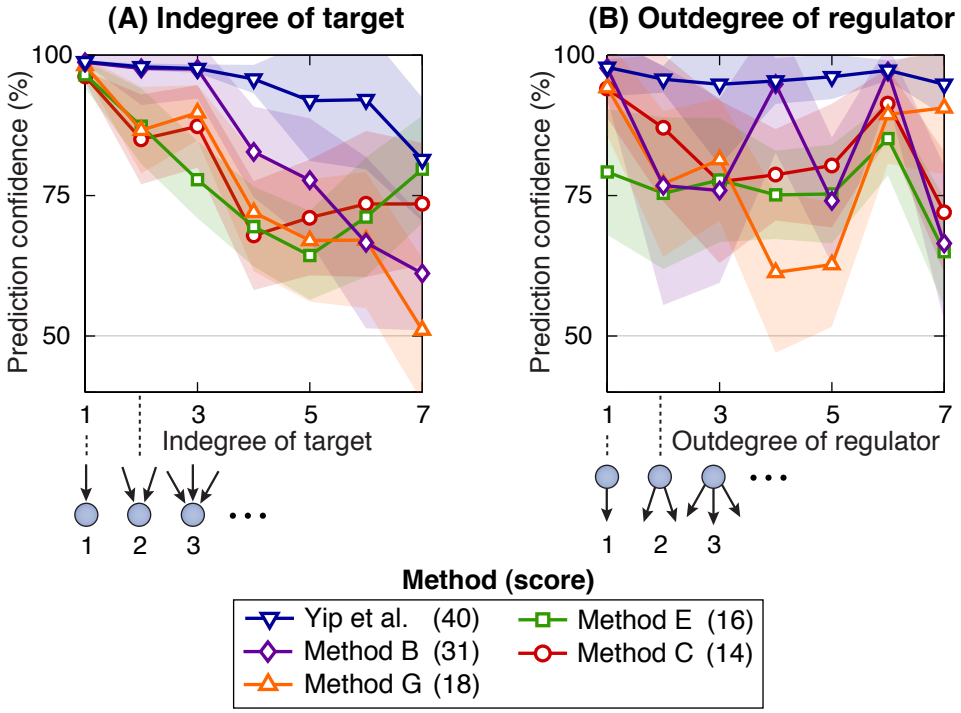


Figure 5.8: How the indegree and outdegree of genes affects the prediction confidence—network size 50. Same as Figure 5.7, but for the best five methods of the networks of size 50, instead of 100. **(A)** The top five inference methods all reliably predicted single-input links with high prediction confidence. As for networks of size 100, the method of Yip et al. has the most robust performance on high indegrees. **(B)** The outdegree does not affect the prediction confidence.

being perturbed, the effect may be partially buffered or even completely masked by the other inputs, which would make this edge more difficult to infer.[4] Thus, it is not surprising that all methods have a reduced prediction confidence for high indegrees. However, the results of this section show that the best-performing method of the challenge can cope with this difficulty much better than the other applied inference methods. In Section 5.4.2, we will discuss why this is the case.

[4]We have confirmed that edges that were missed by many teams, i.e. that were difficult to predict, targeted genes with a high indegree, and they had a weak regulatory effect on these genes (results not shown)

5.3.4 Performance of AGE on the DREAM3 benchmarks

We have also tested AGE on the benchmarks of the DREAM3 *in silico* challenge. Since AGE has been designed for inference of small modules of gene networks using nonlinear models, we have only applied it to the networks of size 10.[5] It should be noted that, unlike for the participants of the challenge, the benchmarks were obviously not blinded for us. However, we used the exact same model type (the log-sigmoid model) and setup of AGE as for our earlier experiments on the synthetic-biology benchmark network described in Chapter 3. In other words, we applied our method "as is", without adapting it to the DREAM3 challenge, thus allowing for a fair comparison with the participating teams.

In our earlier experiments, we used either steady state (Chapter 2) or time series gene expression data (Chapter 3) for reverse engineering, but not both together. AGE could be easily adapted for the joint use of steady state and time series data.[6] However, as mentioned above, we didn't want to adapt AGE specifically for the DREAM3 challenge. Thus, we exclusively used the steady state data (the wild type and the gene knockouts) for the reverse engineering. Evaluating whether the performance could be improved by combining the steady state and time series data is a topic for future work.

We performed 20 runs for each of the five networks. The confidence level of a regulatory link was defined as the fraction of times that it is present in the set of inferred networks, as described in Section 3.4.2. As shown in Figure 5.9, AGE achieved a score of 3.02, which is the third best performance of the 29 inference methods that have been applied in the challenge.

AGE clearly outperformed regression methods based on linear dynamical models (marked with an *L* in Figure 5.9). However, note that the linear regression methods have the advantage that they are scalable to larger networks. The best-performing method, which will be discussed in Section 5.4.2, relied strongly on a statistical approach to predict interactions in a first phase. In a second phase, dynamical models were used as well, but the predictions from these models were not accurate (Yip et al., 2009). The second best inference

[5] AGE is not scalable to networks of size 50 or 100. For large networks, statistical methods or regression methods based on linear models should be used, as discussed in Section 7.2.1.

[6] To use both steady state and time series data together, a measure of the overall data fit would have to be defined. This could be, for example, the average of the square error on the steady state and the time series data.

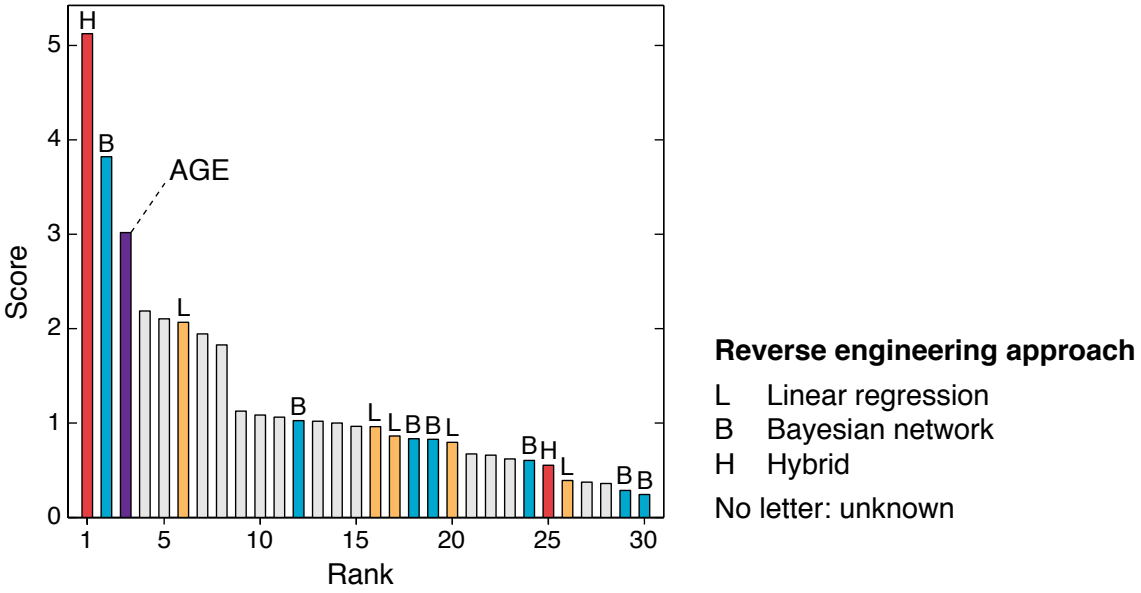


Figure 5.9: Performance of AGE on the DREAM3 networks of size 10. AGE achieves the third best performance of the 29 applied methods. For the 14 teams that have replied to our survey, the type of inference method is indicated: *L* for regression approaches based on linear dynamical models; *B* for methods based on Bayesian networks; and *H* for hybrid methods that rely both on statistical approaches and dynamical models.

method is based on Bayesian networks. We conclude that AGE obtained the best performance of all applied methods that reverse engineer dynamical models.

The networks of size 10 are too small for a statistically significant network-motif analysis. Thus, we cannot study the error profile of AGE as we did for the methods that were applied to the networks of size 50 and 100. Potentially, the network-motif analysis could be performed if AGE would be applied to a large number of different networks—we will consider this possibility in future studies.

5.4 Discussion

5.4.1 A word of caution

The *in silico* benchmarks presented here are based on networks with similar types of structural properties and regulatory dynamics as occur in biological gene networks. In particular, the network structures correspond to modules of

known gene networks of model organisms, and the kinetic model is based on a thermodynamical approach (Ackers et al., 1982), which has been shown to provide a good approximation to different types of transcriptional regulation (Setty et al., 2003; Ellis et al., 2009). However, this model is still extremely simplified compared to the real biological mechanisms. Furthermore, additional layers of control, such as post-transcriptional regulation and chromatin states, are not modeled. Thus, even though the benchmarks presented here are biologically more plausible than previously used *in silico* benchmarks in gene-network reverse engineering, they do not replace the need for careful characterization of performance *in vivo* (Cantone et al., 2009). However, they remain an important tool to systematically assess the performance of inference methods on multiple networks, because this is currently not possible *in vivo*.

A general issue of benchmarks, be it *in silico* or *in vivo*, is that the measured performance of methods is always specific to the particular networks that were being used, and does not necessarily generalize to other, unknown networks, which may have different properties. Indeed, one of the main conclusions of this chapter is that the performance of current network inference methods is strongly dependent on the properties of the network that is being inferred. For example, since methods were found to have very different network-motif error profiles, their performance depends on how many instances of each motif type are present in the network.

Thus, the overall performance (score) of the inference methods should be considered with caution, as it may vary on networks with different properties. For example, the best-performing method of the challenge relies strongly on the prior assumption that the noise in the gene expression data is Gaussian (Yip et al., 2009). This assumption was correct for the *in silico* benchmarks of the challenge, but may be inaccurate in a biological application, which could negatively affect its overall performance on real gene networks.

In contrast to the overall performance, which may vary, the systematic errors identified with the network-motif analysis are expected to be less variant on different networks. For example, a method that failed to accurately infer combinatorial regulation of genes (fan-in error), or to distinguish direct from indirect regulation (cascade error), would be expected to have similar difficulties also on biological gene networks.

5.4.2 Systematic prediction errors are induced by inaccurate prior assumptions

One of the difficulties that participants of the DREAM challenge had to face was that they did not know details of the kinetic model that was used to generate the gene expression data. This difficulty is even more pronounced in biological applications, where the mechanisms and kinetics of gene regulation underlying the expression data are more complicated, and also not known in advance. Thus, inference methods are bound to make simplifying prior assumptions, e.g., by adopting a linear gene network model, to name just one example.

However, if the prior assumptions are inaccurate, they may lead to systematic prediction errors. For example, the best-performing team mainly relied on an inference method based on a very simple principle: it predicts an interaction from gene A to gene B if, after a knockout of A, there is a significant change in the expression level of B. The significance of a change in the expression level was estimated using a Gaussian model of the noise in the gene expression data (Yip et al., 2009). This method is based on the inaccurate prior assumption that the change in the expression level of gene B is always due to a direct regulatory interaction A→B (in reality, it may also be due to an indirect interaction via other genes, e.g. A→C→B). This inaccurate prior assumption induces systematic cascade errors, as shown in Figure 5.6.

However, the best-performing inference method was also the most robust of all applied methods to the fan-in error. Interestingly, it makes a strong prior assumption on the type of noise in the gene expression data, but remains uncommitted to the type of regulatory dynamics in the networks. In contrast, according to our survey among the participants, other inference methods tend to make strong assumptions on the regulatory dynamics (e.g., by adopting simple, often linear, phenomenological functions to approximate combinatorial regulation of genes by multiple regulators). These assumptions are partly inaccurate compared to the more detailed, kinetic model of the benchmarks (they may be even less accurate compared to the complicated mechanisms and kinetics of biological gene networks). Consequently, these methods have a strongly reduced performance on genes with multiple regulatory inputs (the fan-in error), where their prior assumptions are inaccurate.

5.4.3 Conclusion

We have presented a framework for critical performance assessment of gene-network inference methods. This framework has allowed, for the first time, a large number of reverse engineering methods—applied independently by different teams—to be statistically compared on multiple benchmark networks. In addition to assessment of the overall prediction accuracy, we have evaluated the performance of the applied inference methods on individual network motifs. This network-motif analysis revealed that current inference methods are affected, to various degrees, by three types of systematic prediction errors: the fan-out error (incorrect prediction of interactions between co-regulated genes), the fan-in error (inaccurate prediction of combinatorial regulation), and the cascade error (failure to distinguish direct from indirect regulation).

Distinguishing between direct and indirect regulation is a well-known difficulty in network inference (Friedman, 2004), but was never quantitatively assessed. The network-motif analysis makes it possible to quantify how well this difficulty is resolved by different methods. Furthermore, it revealed two other types of systematic errors, which are equally important for the overall quality of predictions. The network-motif analysis, demonstrated here using basic three-node motifs, could be extended to colored motifs, higher-order motifs, or to the network dynamics by considering activity motifs (Chechik et al., 2008), for instance.

One of the main problems in gene-network reverse engineering is often considered to be the limited data, which may leave the inference problem underdetermined (Gardner and Faith, 2005). This chapter highlights another major difficulty, which has received considerably less attention: due to the complexity and our incomplete understanding of biological networks, inference methods are bound to make simplifying assumptions. One generally hopes that accurate predictions can be made despite the simplicity of currently used gene network models, i.e., that there will be a graceful degradation of performance when prior assumptions are not fully met. We have shown that this is, in general, not the case: inaccurate prior assumptions induced systematic prediction errors, which profoundly affected the performance of the applied network inference methods.

The best-performing team has demonstrated a possible paradigm for the development of reverse engineering methods that have a more robust performance despite uncertainty about the type of mechanisms (the "model") underlying the

data. In fact, they applied four distinct inference methods, each one being based on a different type of model. Subsequently, they analyzed and combined the predictions of the four inference methods to form a potentially more robust group prediction (Yip et al., 2009). We call this an *ensemble approach*, because it does not rely on a single network prediction, but on the combination of an ensemble of predictions. In the next chapter, we show that ensemble approaches indeed lead to more robust predictions in a variety of situations. In particular, we show that the predictions of several participating teams of the DREAM challenge can be combined to form "community predictions" that would have ranked first in two of the three sub-challenges.

The application of AGE to the DREAM3 challenge networks of size 10 confirmed its state-of-the-art performance for inference of dynamical models of small gene networks. AGE obtained the third best performance out of the 29 inference methods that were applied in this sub-challenge, and the best performance of all methods that reverse engineer dynamical models.

5.5 Methods

5.5.1 Gene network model

We model transcriptional regulatory networks consisting of genes, mRNA, and proteins. The state of the network is given by the vector of mRNA concentrations $\mathbf{x}$ and protein concentrations $\mathbf{y}$. We model only transcriptional regulation, where regulatory proteins (transcription factors) control the transcription rate (activation) of genes. The gene network is modeled by the system of differential equations

$$\frac{dx_i}{dt} = m_i \cdot f_i(\mathbf{y}) - \lambda_i^{\mathrm{RNA}} \cdot x_i \tag{5.1}$$

$$\frac{dy_i}{dt} = r_i \cdot x_i - \lambda_i^{\mathrm{Prot}} \cdot y_i \tag{5.2}$$

where m_i is the maximum transcription rate, r_i the translation rate, λ_i^{RNA} and $\lambda_i^{\mathrm{Prot}}$ are the mRNA and protein degradation rates, and $f_i(\cdot)$ is the so-called input function of gene i. The input function computes the *relative activation* of the gene, given the transcription factor (TF) concentrations $\mathbf{y}$. The relative activation is between 0 (the gene is shut off) and 1 (the gene is maximally activated).

Gene regulation is modeled using a standard approach based on thermodynamics (Ackers et al., 1982; Shea and Ackers, 1985). Good introductions to this type of models are given by Bower and Bolouri (2004) and Bintu et al. (2005). The basic assumption of this approach is that binding of TFs to cis-regulatory sites on the DNA is in quasi-equilibrium, since it is orders of magnitudes faster than transcription and translation. In the most simple case, a gene i is regulated by a single TF j. In this case, its promoter has only two states: either the TF is bound (state S_1) or it is not bound (state S_0). The probability $P\{S_1\}$ that the gene i is in state S_1 at an instant in time is given by the *fractional saturation*, which depends on the TF concentration y_j

$$P\{S_1\} = \frac{v_j}{1+v_j} \qquad \text{with} \quad v_j = \left(\frac{y_j}{k_{ij}}\right)^{n_{ij}} \tag{5.3}$$

where k_{ij} is the dissociation constant and n_{ij} is the Hill coefficient. At concentration $y_j = k_{ij}$ the saturation is half-maximal, i.e., the promoter is bound by the TF 50% of the time. Many TFs bind DNA as homo-dimers or higher homo-oligomers. This and other mechanisms that affect the effective cooperativity of promoter binding are approximated by the Hill coefficient n_{ij}, which determines the "steepness" of the sigmoid described by Equation (5.3).

The bound TF activates or represses the expression of the gene. In state S_0 the relative activation is α_0 and in state S_1 it is α_1. Given $P\{S_1\}$ and its complement $P\{S_0\}$, it is straightforward to derive the input function $f_i(y_j)$, which computes the mean activation of the gene as a function of the TF concentration y_j

$$f(y_j) = \alpha_0 P\{S_0\} + \alpha_1 P\{S_1\} = \frac{\alpha_0 + \alpha_1 v_j}{1+v_j} \tag{5.4}$$

This approach can be used for an arbitrary number of regulatory inputs. A gene that is controlled by N TFs has 2^N states: each of the TFs can be bound or not bound. Thus, the input function for N regulators would be

$$f(\mathbf{y}) = \sum_{m=0}^{2^N-1} \alpha_m P\{S_m\} \tag{5.5}$$

Using thermodynamics, the probability $P\{S_m\}$ can be computed for every state m. For example, the resulting expression for two regulatory inputs is

$$f(y_1, y_2) = \frac{\alpha_0 + \alpha_1 v_1 + \alpha_2 v_2 + \alpha_3 \rho v_1 v_2}{1 + v_1 + v_2 + \rho v_1 v_2} \tag{5.6}$$

where ρ is the cooperativity factor for the two TFs.

We non-dimensionalize the gene network models as described by von Dassow et al. (2000) in the supplementary information of their paper. As von Dassow et al. note: "This entails replacing every occurrence of dimensioned state variables (concentrations of molecular species and time) with scaled products yielding new state variables free of units. [...] The dimensionless model is identical to the dimensional one since it is merely an algebraic transformation." One of the advantages of non-dimensionalization is that the generation of biologically plausible instances of the dimensionless model is easier (von Dassow et al., 2000).

5.5.2 Simulation of gene expression data

Gene knockouts were simulated by setting the maximum transcription rate m_i of the deleted gene to zero, knockdowns by dividing it by two. Time series experiments were simulated by integrating the networks using different initial conditions. For the networks of size 10, 50, and 100, we provided 4, 23, and 46 different time series, respectively.[7]

For each time series, we used a different random initial condition for the mRNA and protein concentrations. The initial mRNA concentrations $x_i(0)$ were obtained by adding a random number from a Gaussian distribution with mean zero and standard deviation 0.5 to the wild-type steady-state level of every gene. We assume that the multifactorial perturbation that led to the perturbed state $x_i(0)$ (see Section 5.2) was applied for sufficient time for the protein levels to stabilize at their new steady state. Thus, the initial protein concentrations were obtained by setting $dy_i/dt = 0$ in Equation 5.2

$$y_i(0) = \frac{r_i}{\lambda_i^{\text{Prot}}} \cdot x_i(0) \tag{5.7}$$

Each time series consists of 21 time points (from t=0 until t=200). Trajectories were obtained by integrating the networks from the given initial conditions using

[7]For networks of size 50, we provided the same number of time series (23) as Pedro Mendes did for his networks of size 50 in the DREAM2 *in silico* challenges, to allow comparison of results between the two challenges. For networks of size 10 and 100, we scaled the number of provided time series with the network size (two times more for size 100, five times less for size 10).

a Runge-Kutta 4/5 solver with variable step size of the Open Source Physics library (`http://www.opensourcephysics.org`).

White noise with a standard deviation of 0.05 was added *after* the simulation to the generated gene expression data. Concentrations that became negative due to the addition of noise were set to zero.

5.5.3 Evaluation of predictions

As described in Section 5.2, the submission format of predictions was a list of predicted edges with their assigned confidence measures, constructed in decreasing order of confidence from the most reliable to the least reliable prediction. Note that the confidence measures were exclusively used to verify the order of the lists of predictions and were not used in the evaluation of the results otherwise. The quality of the predictions was measured by the area under the receiver operating characteristic curve (AUROC) and the area under the precision-recall curve (AUPR). The AUROC summarizes the tradeoff between the true positive rate and the false positive rate, and the AUPR summarizes the tradeoff between precision, which is a measure of fidelity, and recall, which is a measure of completeness. A detailed description of this approach for measuring the quality of network predictions is given by Stolovitzky et al. (2009).

In addition to the AUPR and AUROC values, we statistically evaluated predictions by computing corresponding p-values (p_{AUROC} and p_{AUPR}), which are the probability that a random list of edge predictions would obtain the same or better AUROC and AUPR than a given network prediction. Distributions for AUROC and AUPR were estimated from 100,000 instances of random lists of edge predictions. The *overall p-value* of the five networks of a sub-challenge was defined as the geometric mean of the individual p-values: $(p_1 \cdot p_2 \cdot ... \cdot p_5)^{1/5}$. The final score of a method is the log-transformed geometric mean of the overall AUROC p-value ($\bar{p}_{\text{AUROC}}$) and the overall AUPR p-value ($\bar{p}_{\text{AUPR}}$):
$\text{score} = -0.5 \cdot \log10(\bar{p}_{\text{AUROC}} \cdot \bar{p}_{\text{AUPR}})$.

5.5.4 Network-motif analysis

The goal of the network-motif analysis is to evaluate, for a given network inference method, whether some types of edges of motifs are systematically predicted less (or more) reliably than expected. This involves: (1) determining the predic-

tion confidence of edges pertaining to different motifs, (2) determining the expected prediction confidence independently of motifs (the *background prediction confidence*), and (3) evaluating whether divergences of the prediction confidence of motif edges from the expected prediction confidence are statistically significant.

Definition of prediction confidence. Given a network prediction in the format described in the previous section, we define the prediction confidence of edges as their rank in the list of edge predictions. We scale the prediction confidence such that the first edge in the list has confidence 100%, and the last edge in the list has confidence 0%.

Determining prediction confidence of motif edges. First we need to identify all three-node motif instances in the target network (the same approach could be used for higher-order motifs). We use the efficient algorithm of Wernicke (2005) for this purpose. Next, for every type of motif edge, we determine the prediction confidence of all its instances. For example, we identify all links of type 1→2 of cascades in the target network (the numbering of the nodes of the motifs is defined in Figure 5.5), and we record the prediction confidences that were assigned to these links by the given network inference method. More formally, we construct the set $C^{1\to2}_{\text{cascade}} = \{c_k\}$, where c_k is the prediction confidence of the link 1→2 in the k'th cascade of the target network. Note that we also record the prediction confidences of "absent edges" of the motifs, for example, $C^{1\to3}_{\text{cascade}}$ is the set of prediction confidences of the "shortcuts".

For every motif type m, we determine the set of prediction confidences assigned by the inference method to each of its six possible edges ($C^{1\to2}_m$, $C^{1\to3}_m$, $C^{2\to1}_m$, $C^{2\to3}_m$, $C^{3\to1}_m$, and $C^{3\to2}_m$). Note that fan-in and fan-out motifs are symmetric: nodes 2 and 3 can't be distinguished (see Figure 5.5). Thus, fan-ins and fan-outs have only three types of edges: 1→2, 2→1, and 2→3. The edges 1→3, 3→1 and 3→2 are equivalent to 1→2, 2→1, and 2→3, respectively. The prediction confidence of equivalent edges is recorded in the same set. For example, the set $C^{1\to2}_{\text{fanout}}$ contains all prediction confidences assigned by the inference method to outgoing edges of fan-outs (1→2 or 1→3).

Determining the background prediction confidence. Before we can analyze the effect of motifs on the edge prediction confidence, we first need to determine the expected edge prediction confidence independently of motifs. There are three types of predicted edges:

- Predicted edges that are *true edges* of the target network. Their background prediction confidence is given by $C_{\text{true\_edge}}$, which is the set of prediction confidences that were assigned by the inference method to the edges that are part of the target network.

- Predicted edges for which the directionality is incorrect. We call these *back edges*. Their background prediction confidence is given by $C_{\text{back\_edge}}$, which is the set of prediction confidences assigned to edges B→A that are *not* part of the target network, but for which the edge in the opposite direction A→B *is* part of the target network.

- Predicted edges between two nodes that are *not* connected by an edge in the target network. We call these *absent edges*. Their background prediction confidence is given by $C_{\text{absent\_edge}}$, which is the set of confidences of predicted edges between nodes that are not directly connected in the target network.

Note that in Figures 5.5B and 5.6B, we have only shown the median prediction confidence of *true edges* (1→2) and *back edges* (2→1)—the median prediction confidence of *absent edges* was not shown.

Evaluating the divergence of the prediction confidence of motif edges from the background prediction confidence. We use the Wilcoxon-Mann-Whitney rank-sum test to compare the motif edge prediction confidences with their corresponding background prediction confidence. The prediction confidence of true edges of motifs (e.g., $C^{1\to2}_{\text{cascade}}$ and $C^{2\to3}_{\text{cascade}}$) are compared with $C_{\text{true\_edge}}$, the prediction confidence of back edges of motifs (e.g., $C^{2\to1}_{\text{cascade}}$ and $C^{3\to2}_{\text{cascade}}$) are compared with $C_{\text{back\_edge}}$, and absent edges of motifs (e.g., $C^{1\to3}_{\text{cascade}}$ and $C^{3\to1}_{\text{cascade}}$) are compared with $C_{\text{absent\_edge}}$. We use Bonferroni correction for multiple hypothesis testing.

6 Wisdom of crowds in gene network inference

This chapter is based on the following publication:

- Marbach, D., Mattiussi, C. and Floreano, D. (2009) Combining Multiple Results of a Reverse Engineering Algorithm: Application to the DREAM Five Gene Network Challenge. *Annals of the New York Academy of Sciences*, 1158 pp. 102-113.

Synopsis

In the previous chapters, we have seen that the gene-network inference problem is often underdetermined, and one is thus often confronted with an ensemble of inferred networks that are consistent with the prior knowledge and the experimentally measured data. Such an ensemble of "plausible" networks may be the output of a single reverse engineering method, or it may have been obtained using alternative reverse engineering methods, different model types, and/or variations of the data. In this chapter, we consider the problem of combining the information contained within such a "crowd" of inferred networks in order to (1) make more accurate network predictions and (2) estimate the reliability of these predictions. We review existing methods, discuss their limitations, and point out possible research directions towards more advanced methods for this purpose. The potential of considering ensembles of networks, rather than individual inferred networks, is demonstrated by showing how ensemble approaches achieved the best performance both in the DREAM2 five-gene network challenge and in the DREAM3 in-silico network challenge.

6.1 Introduction

In the previous chapters, we have seen that there are two major difficulties in gene-network reverse engineering. The first difficulty is that the network-inference problem is in general underdetermined by the available gene expression data. The second difficulty is that current reverse engineering methods are not robust—their performance strongly depends on features of the target network that are *a priori* unknown, such as its structural properties and types of regulatory dynamics. This makes it virtually impossible to know in advance, for an unknown target network, which reverse engineering method would give the best prediction. In this chapter, we show that both difficulties can be partly overcome by using *ensemble approaches*, i.e., by combining the information contained in ensembles of inferred networks instead of focusing on individual inferred networks (Figure 6.1).

In the following, we focus on the first of the above-mentioned difficulties, i.e., the situation where a specific reverse engineering method is used, but the inference problem is underdetermined and one is thus confronted with an ensemble of plausible networks. In Section 6.5, we will come back to the second

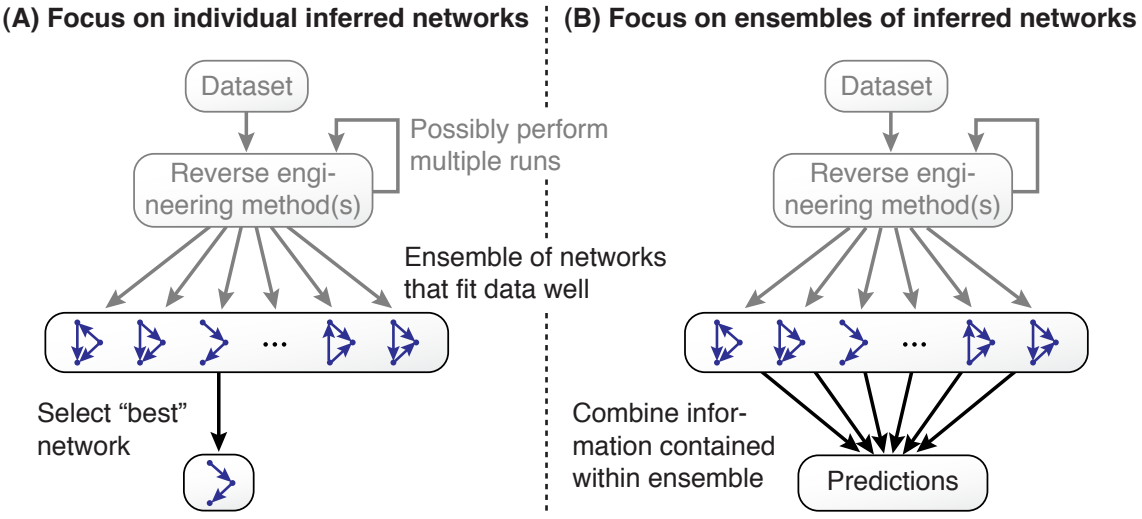


Figure 6.1: Ensemble approach in gene network inference. **(A)** The most common approach in reverse engineering aims at identifying a single "best" network, e.g., the one with the best data fit and the fewest connections. **(B)** The approach considered here aims at integrating the information from ensembles of "plausible" networks in order to make one or several network predictions and estimate the reliability of these predictions.

difficulty, and show how the results of a set of reverse engineering methods can be combined to yield a more robust network prediction.

With the typically noisy and relatively small datasets available, there are in general many different networks that are consistent with the prior knowledge and the data—the inference problem is underdetermined. Some methods attempt to identify a unique "best" network from this ensemble according to some additional criteria (Gupta et al., 2005; Gardner et al., 2003; Tegner et al., 2003), for example by posing constraints on the connectivity of the network (Figure 6.1A). Here, we advocate an alternative approach, which aims at integrating the information contained within ensembles of plausible networks that are consistent with the prior knowledge and the data (Figure 6.1B). Even though many methods have been proposed to construct such ensembles of networks, e.g., Monte Carlo techniques (Battogtokh et al., 2002), simulated annealing (Reinitz and Sharp, 1995; Jaeger et al., 2004a), or genetic algorithms (Kimura et al., 2005), the problem of how to optimally analyze the ensemble in order to estimate the "true" structure of the underlying gene network has received relatively little attention.

Below, we first motivate the use of ensemble methods in gene network reverse engineering and then proceed by formalizing the problem from a probabilistic perspective. In Section 6.2, we review existing approaches and describe a simple voting method that we use to process ensembles of inferred networks obtained with AGE. In Section 6.3, we revisit the *in silico* benchmark and the DREAM2 synthetic-biology challenge of Chapter 3. We find that in the presence of noise, predictions obtained from ensembles of inferred networks are more accurate than any of the individual networks taken alone. Finally, we consider the advantages of combining the predictions from multiple reverse engineering methods, and we show that such "community predictions" outperform individual methods in the DREAM3 *in silico* challenge.

6.1.1 Ensemble methods

The classic example of an ensemble based system in decision making is the popular game show "Who wants to be a millionaire?". When unsure about a question, the contestant has the possibility to either call a friend who he/she knows to be particularly knowledgeable (an "expert"), or to poll the studio audience, which immediately votes on the question. At first sight, one might think that

the experts would offer better help than "random crowds of people with nothing better to do on a weekday afternoon than sit in a TV studio" (Surowiecki, 2004). It turns out the opposite is true: the audience gives the correct answer with a surprisingly high accuracy of about 90%, as compared to only 65% for the experts (Surowiecki, 2004; Polikar, 2006). This is just one of many examples where ensembles of diverse individuals outperform a single expert on average.

Consider an ensemble of inferred networks obtained by a gene network reverse engineering method from a dataset of gene expression measurements. Each of these networks is a hypothesis on the true network structure, giving a prediction on the presence or absence of a regulatory link for every pair of genes. Now assume that the prediction of links is correct with probability $p > 0.5$ (better than random guessing) and that the errors in the prediction of links are uncorrelated between the different networks of the ensemble. In this case, the prediction obtained from the ensemble by voting (see next section) is on average more accurate than any of the individual networks of the ensemble (Dietterich, 2000).

In practice, the picture is more complex. First, since the different networks of the ensemble are inferred from the same dataset, the error of a given link may be correlated *between* the networks (e.g., all networks have a tendency to wrongly predict a given link). Second, there may be a correlation between the different links *within* the networks (e.g., in a given network, there is either link A or link B, but not both). The simple voting methods typically used in gene network reverse engineering ignore these correlations. Despite these limitations, we will see that in practice even simple ensemble methods are useful to process the output of reverse engineering methods and often allow to improve the accuracy compared to individual inferred networks of the ensemble.

6.1.2 A probabilistic formalization

The aim of the somewhat simplistic description above was to give an intuitive understanding of the potential advantages of ensemble methods. We now proceed with a more rigorous probabilistic formalization. Assume the reverse engineering target is a gene regulatory network of N genes (henceforth called *target network*). We assume that the reverse engineering method is based on the fitting of a model to the gene-expression dataset. The majority of data-fitting reverse engineering algorithms represents the target network by an NxN weight ma-

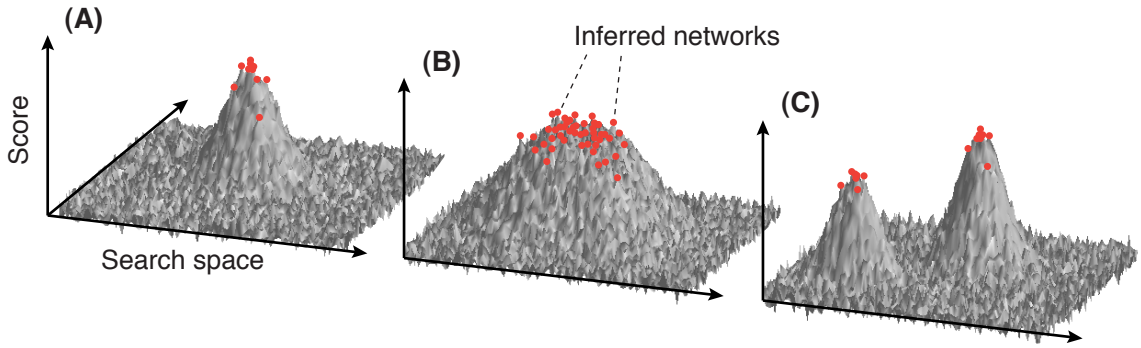


Figure 6.2: Schematic representation of possible posterior distributions in a reverse engineering problem. The horizontal plane represents the search space of all possible networks and the vertical axis corresponds to the score (e.g., the posterior probability). The dots are tentative networks inferred by a reverse engineering algorithm. **(A)** The data is sufficient to identify a unique, distinctive global optimum. **(B)** The problem is underdetermined by the available data—there are many different networks that score approximately equally well. **(C)** There are several distinctive classes of networks that fit the data well.

trix W. The entries w_{ij} of this matrix give the strength of the regulatory effect of gene j on gene i (positive for enhancers, negative for repressors, and zero for no interaction). For simplicity, let's assume that we want to determine just the weight matrix—additional parameters of the genes could be treated in an analogous way. We possess a collection D of noisy observations of the activity of the network, from which the reverse engineering algorithm infers (possibly using multiple runs) an ensemble of tentative networks. Each network has an associated score s_k that indicates how well it fits the data. The ensemble is thus a collection $E = \{(\mathrm{W}_k, s_k)\}$. The problem we consider is how to process the ensemble E to obtain an estimate of the "true" weight matrix W.

From a probabilistic perspective, the aim is to estimate the posterior probability $p(\mathrm{W}|D, I)$ for W, given the dataset D and the prior knowledge I. The ensemble is a collection of samples from this distribution (Battogtokh et al., 2002; Hartemink et al., 2002) (Figure 6.2). From this perspective, the goal of reverse engineering is not just to find the solution that maximizes the posterior probability, but rather to integrate the information contained within the complete ensemble to make predictions on the target network.

6.2 Methods for combining ensembles of inferred networks

6.2.1 Selecting the "best" network from the ensemble

If the quantity and quality of the data is sufficient to uniquely identify the target network (Figure 6.2A), it can be sufficient to simply select the network W_k with the highest score s_k as the most plausible network prediction and discard the information contained within the rest of the ensemble (Reinitz and Sharp, 1995; Moles et al., 2003; Kimura et al., 2005). For example, this is usually done when several independent runs of a stochastic search algorithm converge to this same optimal network, which is then assumed to represent the global optimum and most plausible network prediction (Reinitz and Sharp, 1995).

We used this approach in our initial experiments with noise-free data from an *in silico* network, reported in Chapter 2 (the *SOS network* test case). Remember that in this test case, we selected the network with the best data fit from 10 evolutionary runs as the final network prediction. This strategy worked well, because the noise-free dataset that we used in this test case was sufficient to uniquely identify the target network.

6.2.2 Analysis of the posterior weight distributions

Another popular approach is to analyze the posterior distribution weight by weight, i.e., without considering possible correlations between the weights. The goal is to qualitatively judge how reliable the different weights are determined by the ensemble. The more closely the collection of inferred values for a specific weight w_{ij} are clustered together, the more reliably it is assumed to be predicted. Whether the ensemble of inferred values for w_{ij} indicates a reliable prediction or not is often judged qualitatively by considering the standard deviation and plotting the distribution of inferred weights (Wahde et al., 2001; Wahde and Hertz, 2001; Jaeger et al., 2004a; Deng et al., 2005).

In general, this type of qualitative analysis is done in combination with the strategy described above: the network with the best score is chosen as the most plausible network prediction, and the posterior weight distributions are used only to indicate which of the weights are predicted reliably. Instead of taking the weight values of the best network from the ensemble, one may also consider

using the average of the predicted values (possibly weighted using the scores s_k) (Battogtokh et al., 2002).

As discussed in the introduction, using the ensemble average instead of taking simply the best network of the ensemble often leads to more robust predictions. However, averaging multiple networks only makes sense if they agree more or less on a similar network prediction. If the networks of the ensemble are very different, e.g., they fall within two categories as in Figure 6.2C, averaging leads to a meaningless "blur" of alternative structures. In this case, a more sophisticated analysis taking into account the joint probability distributions would be required.

6.2.3 Majority voting on the network structure

The quantity and quality of available data is often not sufficient to precisely infer numerical values for the weights. In this case, one may be satisfied with predicting only the network structure from the ensemble and disregard the numerical values of parameters. A straightforward approach to do so is majority voting (the same method as the audience polling in the game show mentioned above). Every network of the ensemble votes on the classification of a given link as excitatory ($w_{ij} > 0$), inhibitory ($w_{ij} < 0$), or absent ($w_{ij} = 0$ or smaller than a certain threshold). The type of the link is then defined by the majority of the votes. In addition, the votes could also be weighted by the scores of the networks.

Unsigned predictions can be treated analogously. For example, Hartemink et al. (2002) use weighted voting with Bayesian scores to estimate the probability that a given link is present in the target network. As for the averaging of the weights described in the previous section, the underlying assumption is that regulatory links are predicted independently from each other.

For signed predictions, the basic voting scheme described above may not be optimal because it treats the three possible types of a link (excitatory, inhibitory, and zero) all equal. For example, assume that two links A and B are predicted to be excitatory by 80% of all networks of the ensemble. However, link A is predicted to be zero by the remaining 20%, and link B is predicted to be inhibitory by the remaining 20%. The basic voting would predict both links to be excitatory with equal probability of 0.8. However, one may argue that in this situation 20% of inhibitory votes should be weighted stronger than 20% of zero votes because they directly oppose the excitatory predictions. The voting scheme introduced

in the next section addresses this issue.

6.2.4 Signed voting on the network structure

We have devised a simple voting scheme, which we call signed voting, that is suitable for predicting signed regulatory links from an ensemble of inferred networks. In addition, signed voting estimates a confidence level (reliability) for these predictions. In contrast to majority voting, excitatory and inhibitory votes cancel each other out, whereas votes for the absence of a link are neutral.

Assume that network structures are represented by a matrix A, where $a_{ij} = 1$ if the link is excitatory ($w_{ij} > 0$), $a_{ij} = -1$ if the link is inhibitory ($w_{ij} < 0$), and $a_{ij} = 0$ if the link is absent ($w_{ij} = 0$). Suppose we have an ensemble of K networks, and the structure of the k'th network is defined by the matrix A^k (entries a^k_{ij}). We define the signed vote v_{ij} for link a_{ij} as

$$v_{ij} = \frac{\sum_{k=1}^{K} a^k_{ij}}{K} \tag{6.1}$$

The vote v_{ij} equals 1 if the corresponding link is excitatory in all networks of the ensemble, and -1 if it is inhibitory in all networks. We now define a confidence level l that a given link a_{ij} is excitatory or inhibitory

$$l\{a_{ij} = +1\} \;:=\; v_{ij} \tag{6.2}$$

$$l\{a_{ij} = -1\} \;:=\; -v_{ij} \tag{6.3}$$

Thus, the confidence level that a link a_{ij} is excitatory is 1 if there is strong supporting evidence (all networks agree on the excitatory connection), it is 0 if there is no supporting evidence (e.g., half of the networks vote inhibitory and half excitatory, or all vote for a zero connection), and it is -1 if there is strong evidence to the contrary (all networks vote for an inhibitory connection).

Note that in contrast to majority voting, the absence of links is not explicitly predicted. Instead, one assumes that a connection is zero if there is no strong evidence for an excitatory or an inhibitory link, i.e., if the absolute value of the signed vote is smaller than some threshold $|v_{ij}| < c$. The smaller c, the more (uncertain) links are included in the final network prediction. As in any classification problem, the choice of the threshold is a tradeoff between the number of false positives (links that are predicted present, but are absent in the target network) and false negatives (links that are predicted zero, but are present in the target network).

6.3 Ensemble predictions from multiple runs of AGE

In this section, we use AGE and the same *in silico* and *in vivo* benchmark networks as in Chapter 3 (the five-gene repressilator and the synthetic-biology network of the DREAM2 challenge) for demonstrating the potential of ensemble approaches in gene-network reverse engineering. Note that the ensemble voting methods used here are not specific to AGE, they can be applied to ensembles generated by any other suitable reverse engineering method.

6.3.1 Constructing the ensembles

For generating the ensembles of tentative networks, we use the log-sigmoid model and the evolutionary reverse engineering method based on AGE as described in Chapter 3. Each run of the evolutionary algorithm evolves a population of networks that fit the data well. We found that an evolutionary run typically converges to a single network structure, i.e., in the final population the structures of most networks are identical and only the numerical values of the weights vary slightly. This is expected, because we do not use techniques that enforce diversity in the population after convergence. Thus, a single population is not well suited to construct the ensemble in our case. Instead, we construct the ensemble from multiple runs of the evolutionary algorithm. From each run, only the network with the best fitness is included in the ensemble. In the experiments reported here, we did 50 runs for every dataset.

6.3.2 Combining the ensembles and evaluating the predictions

We used the evaluation protocol of the DREAM2 challenges to assess the accuracy of network predictions. Remember that inhibitory and excitatory links were predicted separately in DREAM2. A confidence level had to be assigned to each of the N^2 possible links of the network, indicating the degree of belief that this link is excitatory/inhibitory. The network prediction is given by a list of links, ranked according to the confidence levels, and the performance is measured by the area under the precision versus recall curve (AUC) (Stolovitzky et al., 2009).

We compared three strategies to predict the network structure from the ensemble of inferred networks: (1) simply take the network with the best data fit from the ensemble and use the strength of the connections (the weights w_{ij}) as

confidence levels, (2) use signed voting, and (3) use signed voting, but allow only the M highest scoring network to vote. For the results reported here we used $M = 10$ networks (the top 20% of the ensemble).[1]

6.3.3 Ensemble voting outperforms individual networks in the presence of noise

We first tested the ensemble approach on the time-series datasets from the *in silico* five-gene repressilator of Chapter 3 at different levels of log-normal noise. Remember that the structure of the repressilator is a loop of inhibitory connections (Figure 6.3A) and the dynamics are simulated with the log-sigmoid model.

Figure 6.3B shows a dataset with log-normal noise of standard deviation 0.5, and the fit by an ensemble of networks, which were inferred with AGE. Besides from few outliers that prematurely converged to local optima, the majority of the networks fit the data reasonably well, without overfitting to noise. However, even though most networks fit the data well, they have very different numerical values for the weights (Figure 6.3C). The same is true for the gene parameters m_i, b_i, and λ_i of the model (data not shown). This indicates that for the model type used here, the reverse engineering problem is underdetermined by this relatively small and noisy dataset.

Still, the inhibitory links of the target network are correctly predicted (i.e., are assigned highest confidence of all possible links) both by the best network of the ensemble and by signed voting of the top 20%. Signed voting by the complete ensemble performs slightly worse (Figure 6.4, top three rows). Probably, the information contained in this dataset is sufficient to constrain the network structures that can fit the data to a relatively narrow peak in the fitness landscape (Figure 6.2A), and this peak seems to coincide with the true network structure. In this situation, it is not surprising that the best scoring network performs as good or better than ensemble voting.

As we add more noise to the data (standard deviation 1.0), the information

[1] We also considered taking the average of the inferred weights of the ensemble and using the standard deviations as a measure of confidence (the approach described in Section 6.2.2). However, this approach performed much worse than signed voting (results not shown), because the test cases that we consider here are underdetermined and the inferred weights therefore all have very high standard deviations.

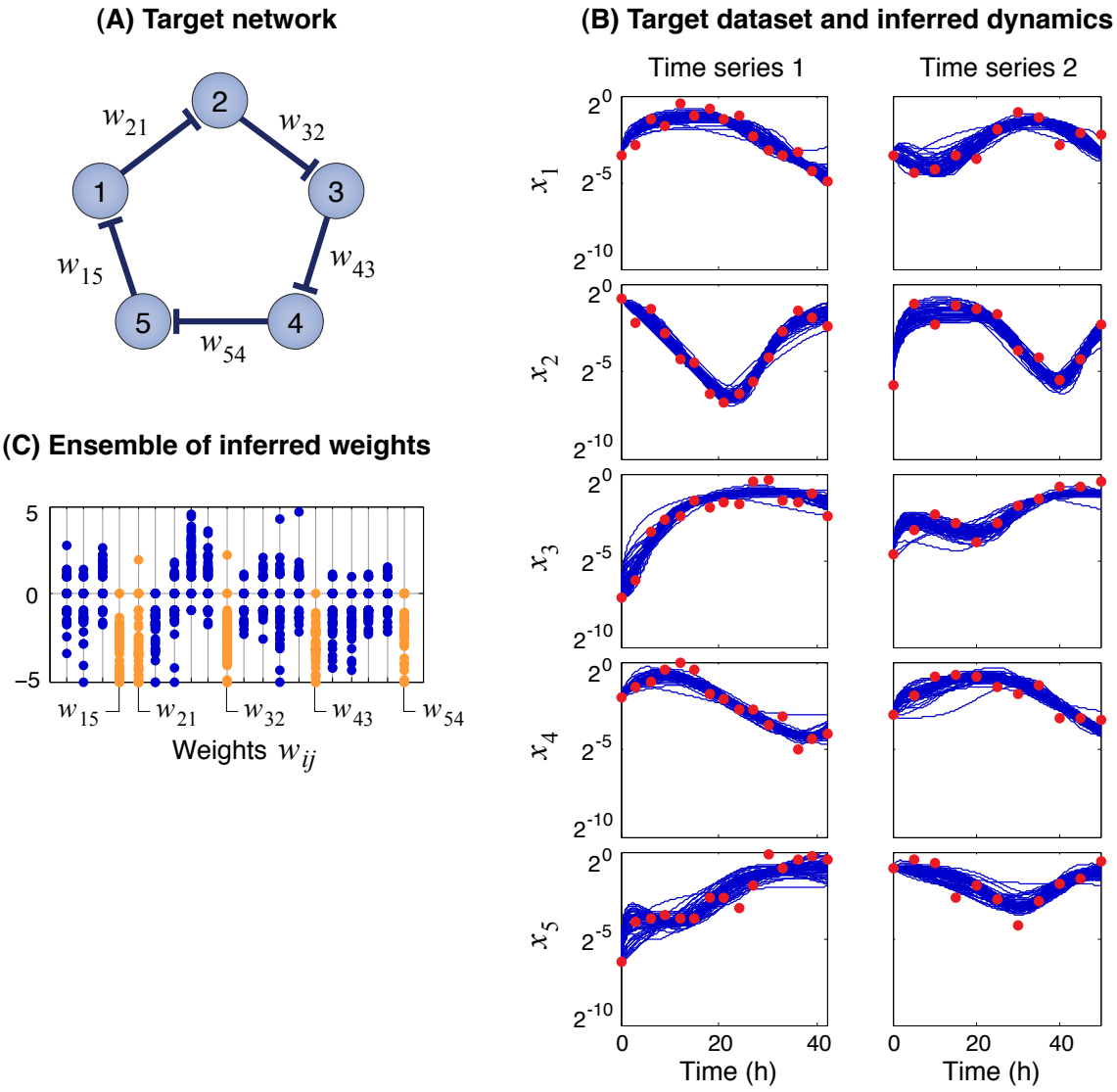
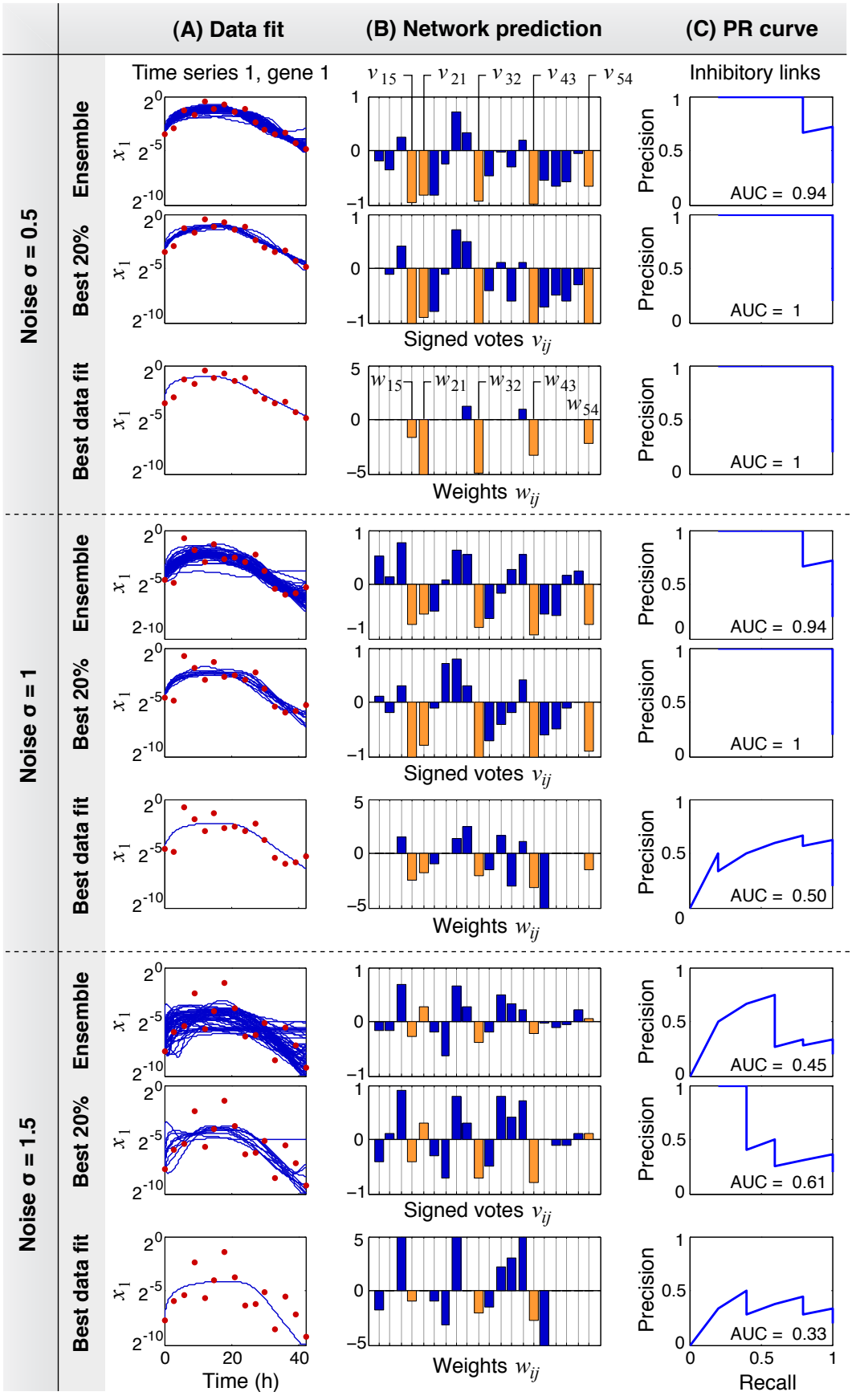


Figure 6.3: Data fit and inferred weights of 50 evolutionary runs on the *in silico* benchmark. **(A)** The target network is the five-gene repressilator. **(B)** Normalized gene expression levels—plotted on a logarithmic scale—for the two time series. The points are the input dataset with log-normal noise of standard deviation 0.5. The lines show the data fit by the 50 inferred networks of the ensemble. **(C)** The inferred weights by the networks of the ensemble (the weights that correspond to the true links of the repressilator are highlighted). Despite a good data fit by the majority of networks, the numerical values of their weights vary a lot, which indicates that the problem is underdetermined.


(A) Data fit
(B) Network prediction
(C) PR curve
Noise σ = 0.5
Noise σ = 1
Noise σ = 1.5
Ensemble
Best 20%
Best data fit
Time series 1, gene 1
Inhibitory links
Signed votes v_{ij}
Weights w_{ij}
Precision
Recall
AUC = 0.94
AUC = 1
AUC = 0.50
AUC = 0.45
AUC = 0.61
AUC = 0.33
Time (h)

content is reduced and the distribution of network structures that can fit the data broadens. Consequently, the individual networks of the ensemble are expected to be more diverse and predict the target network less accurately. Indeed, the network that fits the data best now performs poorly in predicting the network structure and has a low AUC score of 0.5. In contrast, signed voting of the top 20% still correctly predicts the network structure with a perfect AUC score of 1. Again, signed voting by the complete ensemble performs slightly worse (Figure 6.4, middle three rows).

The vastly superior performance of the ensemble as compared to the network with the best data fit can be explained as follows. The individual networks of the ensemble consistently include the five inhibitory links correctly (the signed vote is close to -1 for these links), but in addition also have many false positives. However, it seems that the false positives are sufficiently uncorrelated between the networks of the ensemble to partly "even out" and obtain a lower confidence level than the true positives.

When adding excessive noise (standard deviation 1.5), the network structure is not predicted accurately anymore. Still, the AUC score is doubled by signed voting of the top 20% compared to the network with the best data fit (Figure 6.4, last three rows).

The same quality of results was obtained on four different datasets with log-normal noise of standard deviations 0.5, 1.0, 1.5, and 2.0 (results not shown).

Figure 6.4: Predictions obtained from ensembles of inferred networks are more accurate than the "best" individual networks. For three datasets with different levels of log-normal noise (standard deviations 0.5, 1, and 1.5), we compare the predictions obtained from signed voting by the complete ensemble, signed voting by the top 20% of the ensemble, and the inferred network with the best data fit. **(A)** Data fit of the inferred networks (only the first time series of the first gene is shown). **(B)** Predictions of regulatory links (the true links are highlighted). **(C)** From the precision-versus-recall curves and the AUC scores, it can be seen that at intermediate and strong levels of noise, ensemble voting (especially by the top 20%) predicts the network structure much more accurately than the network with the best data fit.

6.3.4 Ensemble voting achieves best performance in the DREAM2 synthetic-biology network challenge

The *in vivo* synthetic-biology benchmark network and the q-PCR time-series dataset provided by Cantone et al. (2009) for the DREAM2 challenge have been described in Chapter 3. We have mentioned before that de la Fuente et al. and us, the two best performers of this challenge, both used an ensemble approach to post-process predictions and estimate confidence levels. Here, we first explain how we derived directed-signed, as well as undirected-unsigned link predictions from ensembles of inferred networks, and we analyze the performance of the ensemble approach compared to individual inferred networks with AGE. In Section 6.4, we will discuss the ensemble approach and results of de la Fuente et al.

Directed-signed link predictions

For our submission to the DREAM2 challenge, we predicted excitatory and inhibitory links using AGE coupled with *signed voting by the complete ensemble* of inferred networks. As for the *in silico* test case, the majority of the runs fit the data well (see Figure 3.4 for an example). However, the values of the inferred weights are very scattered (Figure 6.5, first row) and the reverse engineering problem seems to be largely underdetermined. In contrast to the *in silico* test case, the network structure is not accurately predicted and the AUC scores of excitatory link predictions, and in particular inhibitory link predictions, are low (Figure 6.5, second row). Note, however, that these scores compared well to other participating teams (2nd and 1st rank for excitatory and inhibitory predictions, respectively). These results, and in particular possible explanations for the low AUC scores, have been discussed in Chapter 3.

Undirected-unsigned link predictions

In order to compare our approach with reverse engineering methods that produce undirected and unsigned network predictions, we have participated also in these categories of the DREAM challenge. Confidence levels l for directed-unsigned links and undirected-unsigned links were derived from the signed

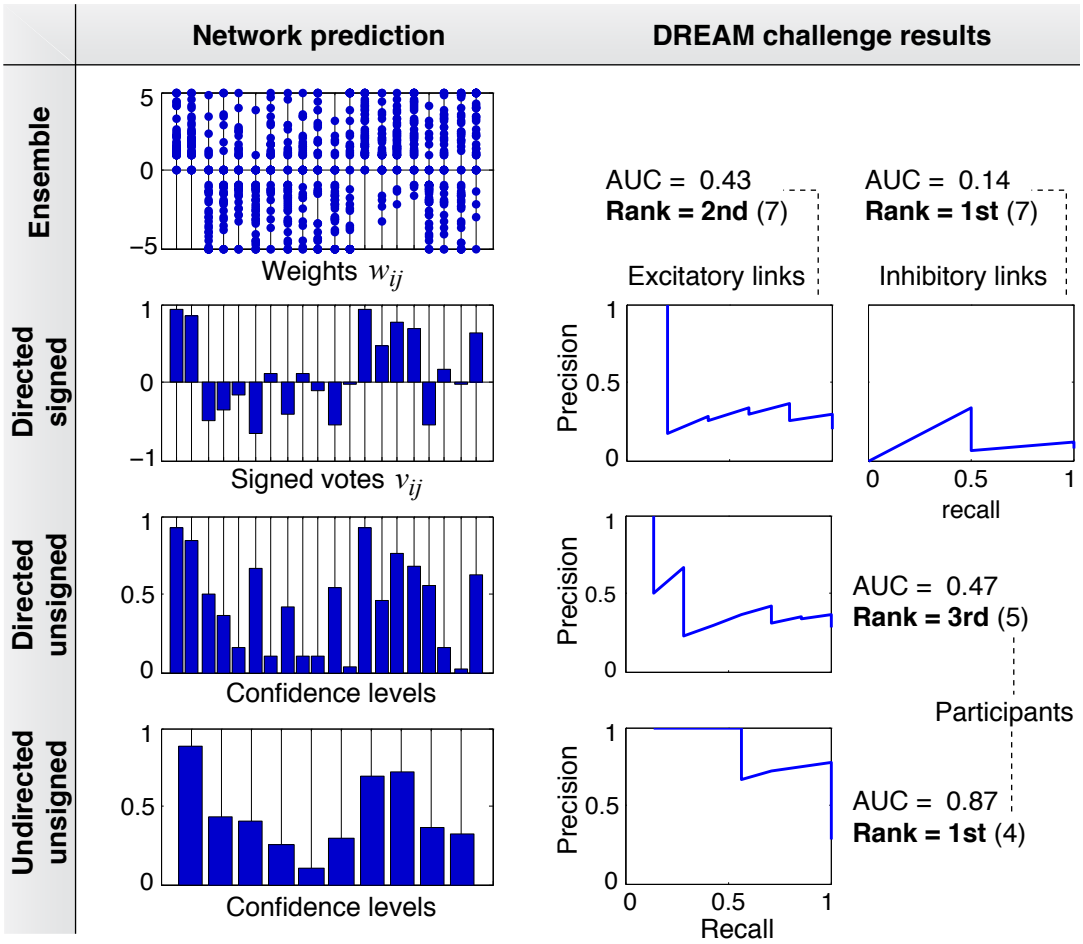


Figure 6.5: Performance of AGE, coupled with ensemble voting, on the synthetic-biology benchmark of the DREAM2 challenge. The ensemble of inferred weights is very diverse (first row). The ranking of AGE combined with ensemble voting is competitive compared to the other participating teams, but the accuracy of the predictions is not satisfactory.

votes v_{ij} of the ensemble (cf. Equation 6.1)

$$l\{|a_{ij}|=1\} := |v_{ij}| \quad \text{(directed-unsigned)}$$
$$l\{(|a_{ij}|=1) \text{ or } (|a_{ji}|=1)\} := \frac{|v_{ij}|+|v_{ji}|}{2} \quad \text{(undirected-unsigned)}$$

Note that ensemble voting is done first, and the sign is removed afterwards by taking the absolute value of the signed votes v_{ij}. Thus, if there are inconsistent signed predictions for a link (e.g, 50% excitatory, 50% inhbitory), the corresponding unsigned prediction is still zero. This would not be the case if signs were removed first, and ensemble voting done afterwards.

Our predictions derived in this way are competitive with methods that directly produce undirected and unsigned predictions (Figure 6.5, last two rows). At first sight, the AUC score seems to be very high for the undirected-unsigned predictions as compared to the directed-signed predictions that they were derived from. Is water (inaccurate directed-signed predictions) made into wine

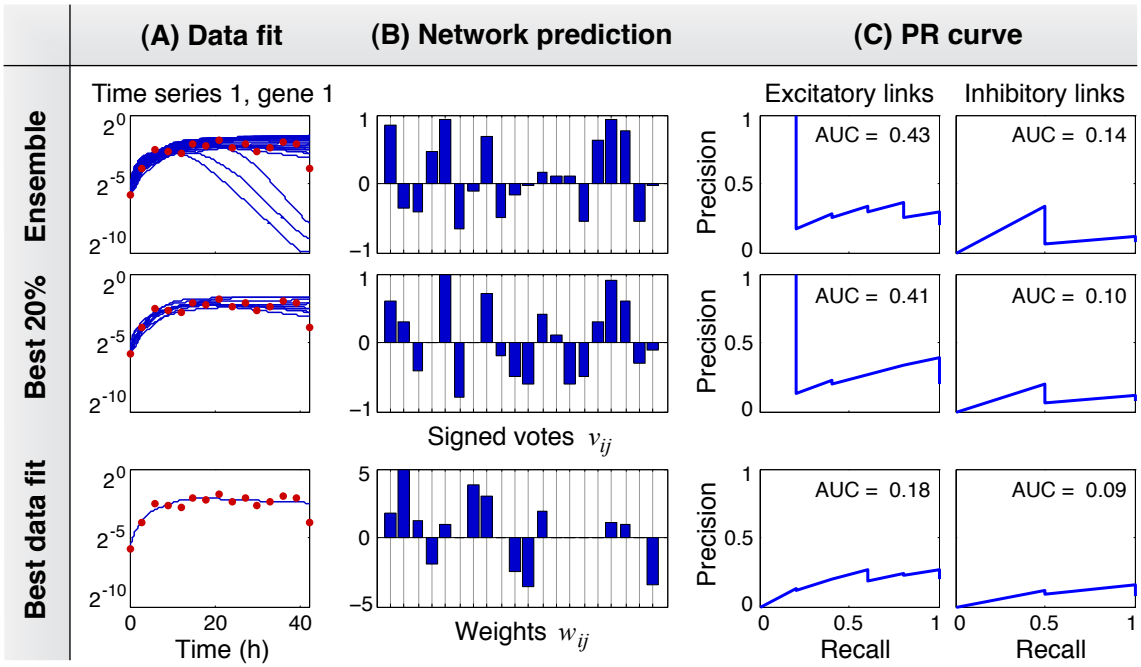


Figure 6.6: Ensemble voting outperforms the network with the best data fit in the DREAM2 challenge. Same analysis as in Figure 6.4, but for the synthetic-biology benchmark of the DREAM2 challenge. **(A)** The data fit by the inferred networks (normalized, negative q-PCR log expression ratios). **(B)** The predictions of the regulatory links. **(C)** As in the *in silico* test case, the accuracy is improved by ensemble voting. For excitatory links, the AUC is roughly doubled compared to the network with the best data fit.

(accurate undirected-unsigned predictions)? Certainly not. The undirected target network has the same number of true links, but only half as many possible links as the directed version. This makes it much easier to obtain a high AUC score.

Ensemble voting outperforms individual networks in the DREAM challenge

After the true structure of the target network was published on the DREAM website (http://wiki.c2b2.columbia.edu/dream), we analyzed the performance of ensemble voting on this benchmark. The observations on the *in silico* test case with intermediate and strong levels of noise are confirmed on the real DREAM challenge dataset. The AUC score is roughly doubled by ensemble voting compared to the network with the best data fit. Signed voting by the complete ensemble and by the top 20% perform approximately equally well (Figure 6.6).

6.4 Ensemble predictions from variations of the data

As discussed in the introduction, ensemble based systems are only interesting if the individual members of the ensemble are diverse. For example, if all inferred networks of the ensemble are affected by the same prediction errors, there is nothing to be gained in combining them. In the previous sections, we have demonstrated one approach to generate ensembles of diverse networks, namely by performing multiple runs of a stochastic reverse engineering method such as AGE. However, since each run uses the same gene-expression dataset, the errors in the inferred networks are certainly not completely uncorrelated. By using different variations of the dataset for each run, the errors between the networks could potentially be further decorrelated, and the performance of the ensemble prediction thus improved. Indeed, in the field of machine learning, the individual classifiers of an ensemble system are typically generated from different subsets of the training dataset (Dietterich, 2000; Polikar, 2006).

De la Fuente et al. used this approach in the DREAM2 synthetic-biology network challenge, where they obtained, together with AGE, the best performance (Baralla et al., 2009; Stolovitzky et al., 2009). Here, we briefly describe these results, as they illustrate a promising approach to generate ensembles of diverse networks and were not discussed in this context by de la Fuente et al.

De la Fuente et al. used different variations of the provided dataset as input for their reverse engineering method. Specifically, they considered three combinations: the first time series alone, the second time series alone, and both time series together. For each of the three combinations, two variations were obtained by using for one the original q-PCR log expression ratios, and for the other one the same data transformed to a linear scale. Thus, they considered a total of six variations of the dataset. The network was inferred from each of these variations, yielding and ensemble of six network predictions. These predictions were averaged to produce the final, submitted network prediction (Baralla et al., 2009).

After the true network structure of the challenge was released, de la Fuente et al. compared the performance of the six individual network predictions with the ensemble prediction (Figure 6.7). Indeed, for excitatory and directed-unsigned links, the ensemble prediction is more accurate than any of the individual network predictions. For undirected-unsigned links, it is about as good as the best individual network prediction. Only for inhibitory links, which were predicted with very low (not significant) AUC scores, the ensemble prediction does not perform better than individual network predictions.

The results of Fuente et al. support our conclusion of the previous sections, namely that ensemble predictions are generally more accurate than individual network predictions. Furthermore, their results indicate that using variations of the dataset to construct the ensembles, an approach that is widely used in the field of machine learning, is promising. However, a more systematic study on a number of different networks will be necessary to understand the advantages and disadvantages of this approach in gene network inference.

6.5 Ensemble predictions from a community of reverse engineering methods

In the previous chapter, we have seen that reverse engineering methods are affected, to various degrees, by systematic prediction errors. If a single reverse engineering method is used to infer an ensemble of network predictions, they are all affected by the same systematic errors. These errors are correlated and would thus not be reduced by an ensemble approach. However, if each network prediction of the ensemble is obtained using a different reverse engineer-

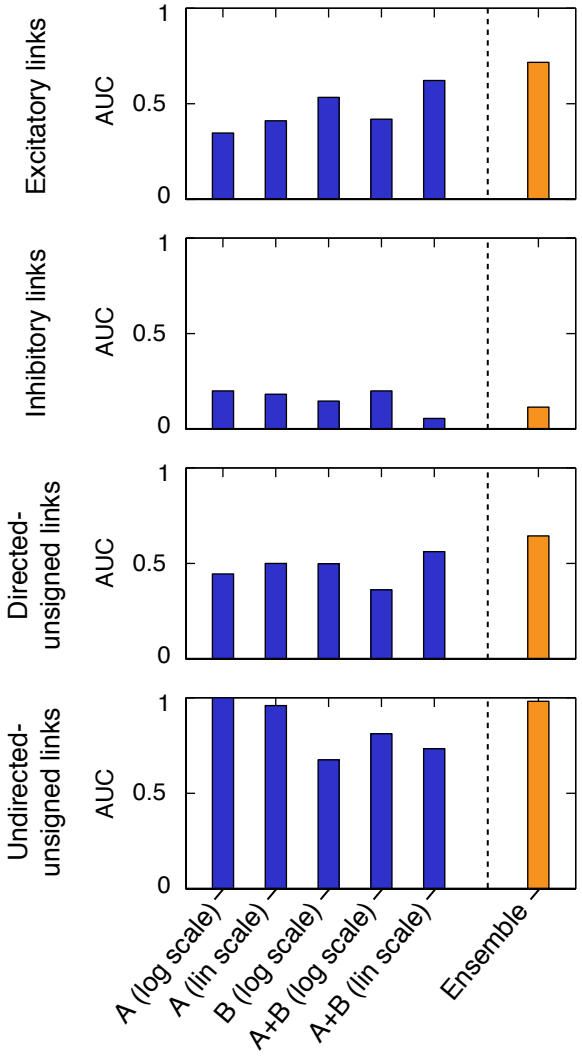


Figure 6.7: AUC scores of de la Fuente et al.'s method for predictions obtained from variations of the dataset [data from Baralla et al. (2009), Table 2]. The four plots correspond to excitatory, inhibitory, directed-unsigned, and undirected-unsigned link predictions (from top to bottom). Network predictions were obtained from six variations of the dataset (time series A, time series B, time series A+B, all both on linear and logarithmic scale). Unfortunately, the results of one variation (time series B on a linear scale) were not reported by Baralla et al. (2009) because it did not produce significant predictions. Overall, the ensemble prediction performs better than the predictions of any variation taken alone.

ing method, the errors are less correlated and could potentially be averaged out by an ensemble approach. In this section, we show that ensemble predictions obtained from multiple reverse engineering methods are often more accurate than the predictions from the individual methods taken alone.

In order to distinguish ensemble predictions obtained using a single reverse engineering method, as described in the previous sections, from those obtained using different reverse engineering methods, we call the latter *community predictions*. When analyzing the results of the DREAM2 BCL6 challenge (the goal of this challenge was to predict targets of the transcription factor BCL6), Stolovitzky et al. (2009) observed that a very accurate community prediction could be obtained by combining the predictions of the three best performing teams:

> Clearly, this shows that the "intelligence" behind the three methods can be easily aggregated to produce an integrative approach that is better than any method in isolation. This calls for a reverse-engineering-by-consensus approach that has not yet been explored by the community. (Stolovitzky et al., 2009)

The DREAM3 *in silico* network challenge gives us the opportunity to explore this approach in more detail, as for the first time, dozens of network-inference methods have been applied to the same benchmark. In the following, we first describe how we combined predictions of participating teams, and then we analyze the performance of the resulting community predictions.

6.5.1 Forming community predictions

Remember that the submission format of the DREAM3 *in silico* challenge is a ranked list of edge predictions. The question is thus how to optimally combine an ensemble of such lists, submitted by different participating teams, to form a community prediction.

We use a straightforward approach to combine the ensemble of edge-prediction lists, which consists of simply taking the average rank for each edge. Thus, the list of edge predictions of the community is ordered according to the average

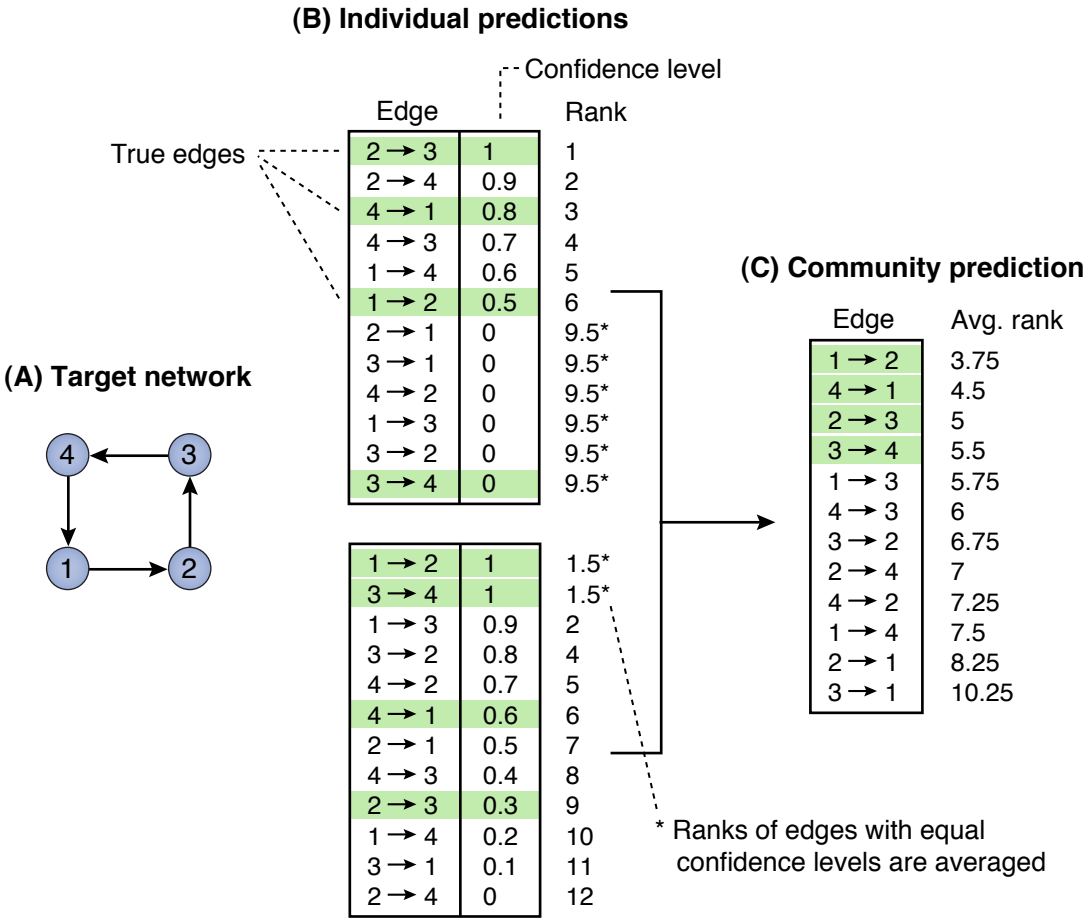


Figure 6.8: Example of a community prediction formed from two individual network predictions. **(A)** The target network of this example is a loop of four genes. **(B)** Two possible lists of edge predictions. The lists are ranked according to the confidence levels of the edges, the most confident prediction at the top of the list has rank 1. The true edges of the target network are highlighted. **(C)** The community prediction is obtained by taking for each edge the average of its rank in the two individual predictions. Here, the community prediction is perfect (all true edges are at the top of the list). This example illustrates how a community prediction can be more accurate than the individual predictions that it is composed of.

ranks of the edges in the ensemble, as illustrated in Figure 6.8. [2]

6.5.2 Community predictions are more robust than individual predictions

To gain a sense of the performance of community predictions in the DREAM3 *in silico* challenge, we systematically formed communities composed of the top-two methods, the top-three methods, the top-four methods, etc., until the last community, which contains all applied methods of a particular sub-challenge (there are three sub-challenges corresponding to networks of size 10, 50, and 100). For each of these communities, we derived community predictions for the five networks of the sub-challenge. The accuracy of the community predictions was evaluated using the same method that had also been used to evaluate the predictions of the individual teams (see Section 5.5.3).

The scores of both the community predictions and the individual teams are shown in Figure 6.9. Some of the community predictions outperform the best-performing team on networks of size 10 and size 50. For example, the community of the top-five teams would have won these two sub-challenges. As more and more teams with a bad performance are added to the community, the accuracy of the community prediction decreases. However, the performance is surprisingly robust. Even when combining all methods, the majority of which have low scores (remember from the previous chapter that about a third of the methods did not perform better than random guessing), the community prediction still ranks second on networks of size 10 and 100, and third on networks of size 50.

Note that in a real application to an unknown network, the performance of alternative methods is obviously not known in advance. Our results show that, instead of choosing a single method to trust, a more robust strategy is to apply all methods at hand and form a community prediction. When using a single method, one risks a very bad performance if the wrong choice is made.

[2] An alternative way to form a community prediction would be to take the average of the confidence levels assigned to each link by the participants. However, the confidence levels produced by the different methods have different meanings and scales, thus, averaging them makes no sense (e.g., one method may formally estimate posterior probabilities, and another method may use a more qualitative approach such as ensemble voting).

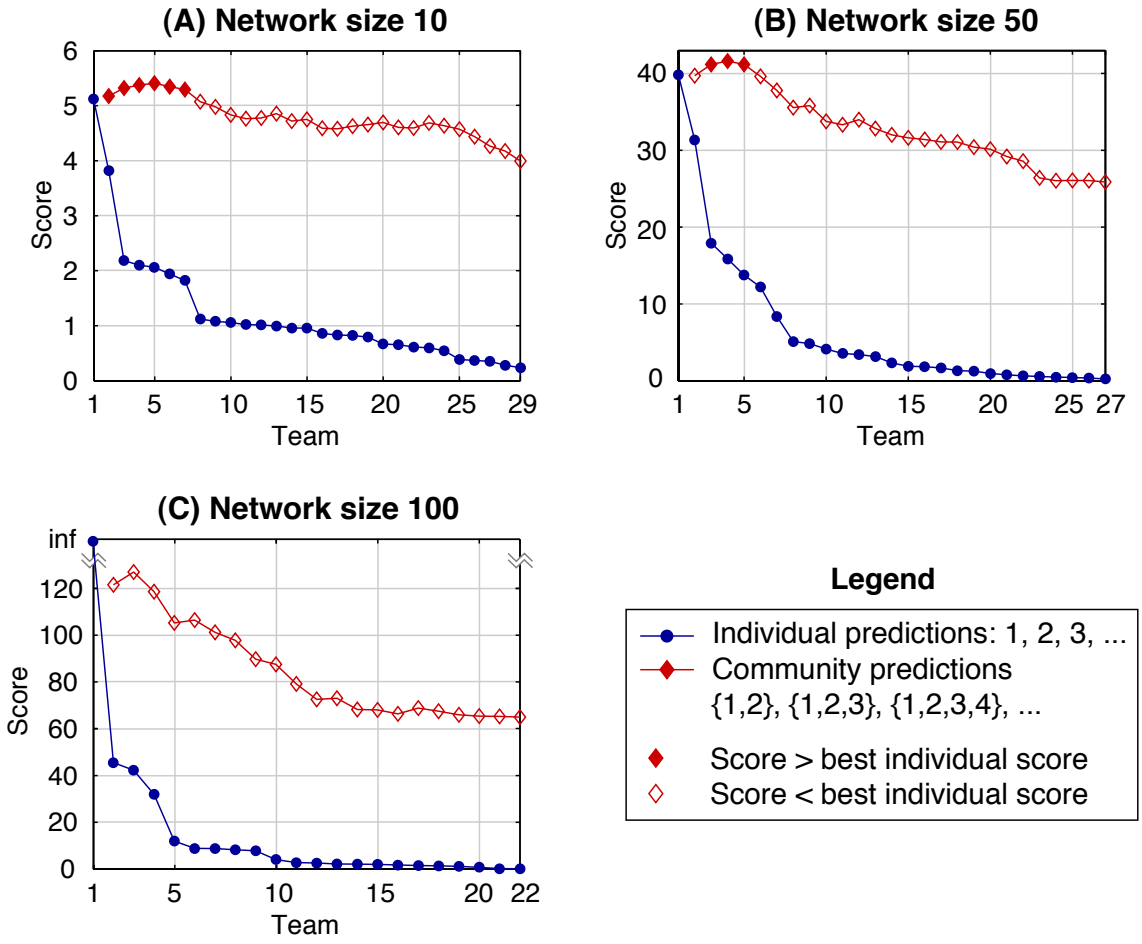


Figure 6.9: Community predictions in the DREAM3 *in silico* challenge. The filled circles are the scores of the individual teams. The diamonds correspond to the scores of the different community predictions, obtained by combining the two best teams, the three best teams, the four best teams, etc. Filled diamonds indicate that the community prediction is better than the best-performing team. The performance of the community is extremely robust. Even when including all teams (rightmost diamonds), the score of the community is better than the second-best method on networks of size 10 and 100, and better than the third-best method on networks of size 50.

However, the community prediction is consistently as good or better than the best methods of the ensemble, and this even when the majority of the methods of the ensemble have a low performance.

6.6 Conclusion

In this chapter, we have demonstrated the power of ensemble approaches in gene network inference. We have considered three strategies to construct diverse ensembles of network predictions: (1) performing multiple runs of a stochastic reverse engineering method such as AGE, (2) using variations of the gene-expression dataset, and (3) using a set of different reverse engineering methods. We have analyzed these strategies on a number of *in silico* and *in vivo* benchmarks. In all cases, the ensemble predictions were remarkably accurate and robust, having similar or better performance than the best individual network predictions of the ensemble:

- **(1) Performing multiple runs of a stochastic reverse engineering method**. As discussed in the introduction, ensemble voting boosts the performance compared to individual members of the ensemble if the prediction errors are uncorrelated. This seemed unlikely for the first of the three above-mentioned strategies, where the ensemble is inferred from the same dataset and using the same method. Yet, our results show that in practice, the prediction errors in ensembles of reverse engineered networks are sufficiently uncorrelated for ensemble voting to drastically improve the accuracy of predictions from noisy datasets. This was confirmed both on *in silico* benchmarks and on the *in vivo* synthetic-biology benchmark network of the DREAM2 challenge.

- **(2) Using variations of the gene-expression dataset**. The second strategy, that is, using variations of the dataset for inference of a diverse ensemble of networks, was applied by de la Fuente et al. in the DREAM2 challenge (Baralla et al., 2009). This approach seems promising, as it allowed de la Fuente et al. to significantly improve their predictions. However, so far it has only been tested on one network and will thus have to be studied more systematically to prove its merits.

- **(3) Using a set of different reverse engineering methods**. In the previous chapter, we have shown that gene-network inference methods have different strengths and weaknesses. Here, we have established that the predictions from different inference methods can easily be combined to form more robust and accurate "community predictions", which is the third of the above-mentioned strategies. This corresponds to a completely new paradigm in gene network inference: instead of attempting to design *the* right model and inference method, one would design a set of *complementary* models and inference methods, which would then be used to form a community prediction.

The goal of a reverse engineering method is often seen to consist in reliably finding the global optimum, i.e., the best scoring network. Here, we advocate for a different view, where the goal of reverse engineering is to construct an ensemble of diverse, good scoring networks (repeatedly recovering the global optimum is contrary to this aim). The results that we have obtained with AGE show that it is possible to make accurate predictions from such ensembles even if the problem is underdetermined and many different networks fit the noisy data equally well.

The ensemble approach, be it the first and/or the second strategy, holds the promise to improve the accuracy of any reverse engineering method that can produce sufficiently diverse network predictions. However, AGE is particularly well suited for this purpose because it reproduces, at a certain level of abstraction, the structure and evolutionary constraints of the biological genome. We hypothesize that this is an effective approach to incorporate prior knowledge and bias the search towards biologically plausible solutions (see Chapter 2). Ensembles generated by "replaying the evolutionary tape" may thus provide a better sampling of the posterior distribution than ensembles generated with other optimization methods, because the fundamental prior that biological gene networks originate from an evolutionary process is taken into account.

The straightforward methods used here (signed voting for ensemble predictions with AGE, and averaging of ranks for community predictions in the DREAM3 challenge) have an intuitive appeal, and our results show that they work well in practice. However, we believe that there is a need for more sophisticated tools for a rational, probabilistic analysis of ensembles of inferred networks, and we hope that the encouraging results presented in this chapter

will stimulate further research in this direction.

7 Discussion and outlook

Synopsis

In this chapter, we summarize the results presented in this thesis and discuss implications for future research. In particular, we discuss limitations of the current version of AGE, and how AGE could be used to infer more accurate, heterogeneous gene network models. We also consider limitations and directions for improvement of the in-silico reverse engineering benchmarks.

7.1 Accomplished work

7.1.1 Evolutionary reverse engineering of gene networks

The evolutionary reverse engineering method presented in this thesis allows to simultaneously infer both the wirings and nonlinear dynamical models of gene regulatory networks from gene expression data. The proposed method reconstructs gene networks by mimicking the natural evolutionary process that constructed them. It is based on *in silico* evolution of gene networks by means of a biomimetic artificial genome called Analog Genetic Encoding (AGE), which models both the way in which gene networks are encoded in the biological genome, and the different types of mutations and recombinations that drive their evolution. The reverse engineering method based on AGE has the following key features:

- domain-specific prior knowledge is directly built into the search method;
- it is compatible with a wide range of nonlinear gene network models;
- it can be used with different types of steady-state and time-series gene expression data.

The flexibility of AGE allows for the exploration of novel model types for gene regulatory networks, such as the log-sigmoid model introduced in Chapter 3. The log-sigmoid model provides a better approximation to different types of combinatorial regulation than commonly used models for gene network inference.

We have assessed the performance of AGE on a series of *in silico* and *in vivo* benchmarks. Given sufficient data, AGE provides a near-perfect reconstruction of *in silico* networks (Chapter 2). If the inference problem is underdetermined due to insufficient or noisy gene expression data, multiple runs of AGE can be used to construct an ensemble of plausible networks (Chapter 3). We have shown that accurate predictions can be derived from such sets of inferred networks using ensemble approaches that are well known in the field of machine learning, but haven't previously been applied in gene network inference (Chapter 6). To allow direct comparison with other network inference methods, we have applied AGE to an *in vivo* reverse engineering challenge of the DREAM2 conference, where it obtained the best performance (Chapter 3).

7.1.2 Critical performance assessment of inference methods

This thesis introduces a framework for critical performance assessment of reverse engineering methods, which consists of:

- tools for generating realistic *in silico* benchmarks;
- a community-wide network inference challenge;
- tools for evaluating the performance and analyzing the prediction errors of inference methods.

The generation of realistic *in silico* benchmark networks is based on the extraction of modules from known gene networks. We have shown that *in silico* networks generated in this way preserve functional and structural properties of the original gene network. We have applied this approach to generate benchmark suites that we released as community-wide challenges for the DREAM3 and DREAM4 conferences. With 29 participating teams, the DREAM3 challenge has become the most widely used benchmark in the field. The DREAM4 challenge, which was released in June 2009, has already been downloaded over 80 times in these past two months.

Our analysis of the 390 submitted network predictions of the DREAM3 challenge participants gives unprecedented insight into the capabilities and limitations of prevalent reverse engineering methods. Performance profiling on individual network motifs reveals that the applied inference methods are affected, to various degrees, by three types of systematic prediction errors. In particular, all but the best-performing method fail to accurately infer multiple regulatory inputs (combinatorial regulation) of genes.

The Java tool, called GeneNetWeaver (GNW), that was used to generate the DREAM *in silico* challenges and to perform the network-motif analysis is available open-source at: `http://gnw.sourceforge.net`.

7.2 Outlook

7.2.1 Current limitations of AGE

In this thesis, we have presented one possible implementation of an evolutionary, biomimetic reverse engineering approach. This implementation is based on AGE as a model of the biological encoding and the evolution of gene regulatory networks. Even though AGE successfully captures some of the fundamental evolutionary principles of gene networks, in particular the implicit encoding and the different types of possible mutations and recombinations, limitations of the current version have also become apparent.

One limitation is the absence of a mechanism to explicitly *prevent interaction* between genes. Due to the implicit encoding, the more genes (*cis*- and *trans*-acting sequences) there are, the more difficult it becomes for evolution to prevent interference (unwanted interactions) between them. This hinders evolution of large networks. In biological gene networks, there are a number of important mechanisms to prevent regulatory interactions, for example via silencing chromatin or insulators (Burgess-Beusse et al., 2002). Preliminary results show that allowing for similar mechanisms in AGE indeed significantly improves evolvability (Peter Dürr, personal communication).

As discussed in Chapter 2, the aim of the evolutionary approach is reverse engineering of small modules of gene networks using nonlinear models. With AGE, we have successfully reverse engineered networks up to size 10. The evolutionary approach is not suited for reverse engineering of large networks with hundreds or thousands of genes, for which statistical methods or regression methods based on linear models should be used. Note that limited scalability is inherent to any stochastic search method (global optimization method), not just AGE. For example, Moles et al. (2003) have compared seven state-of-the-art global optimization methods on a pathway inference problem with 36 free parameters. This is considerably less than the 130 free parameters of a gene network of size 10 (using a log-sigmoid model). Nevertheless, only one type of stochastic optimization method (Evolutionary Strategies) could solve this inference problem, which demonstrates the challenging nature of inferring nonlinear dynamical models (Moles et al., 2003). Fortunately, gene regulatory networks have a modular architecture (Hartwell et al., 1999), which lends itself to the study of small modules often comprising only a handful of genes that dynami-

cally interact to perform a specific function (von Dassow et al., 2000; Manu et al., 2009).

7.2.2 Reverse engineering heterogeneous networks

A limitation of prevalent methods for inference of dynamical models of gene networks is that each gene of the network is modeled with the same generic function $dx_i/dt = f(\mathbf{x})$ (see Equation 1.1). Only the inferred parameters of this function (e.g., the weights of the interactions) differ from gene to gene. We call this a *homogeneous network* because all the regulatory interactions and genes are modeled in the same way.

However, biological gene networks are highly *heterogeneous*. For example, a first gene may be regulated *independently* by two proteins that bind separately from each other to its *cis*-regulatory region. A second gene may be regulated *synergistically* by two proteins that first have to bind to each other, forming a *heterodimer*, and only this heterodimer has the ability to bind the *cis*-regulatory region of the gene. Consequently, the independent regulation of the first gene may be, for example, well described by $dx_1/dt = f(\mathbf{x})$, but the synergistic regulation of the second gene may require a different model $dx_2/dt = g(\mathbf{x})$.

In other words, different mechanisms of independent and synergistic combinatorial regulation of genes require different model types to be accurately described (and reverse engineered). It is thus not surprising that current inference methods, which infer homogeneous networks, make systematic errors in predicting the combinatorial regulation of genes (the fan-in error discussed in Chapter 5).

To overcome this limitation, a paradigm shift in the design of reverse engineering methods is required. Instead of relying on a single, fixed model type, which is the same for all genes and interactions, a set of alternative model types that account for different regulatory mechanisms need to be defined. The appropriate model types for the different genes and interactions of the network would then be inferred from the gene expression data, in addition to the network structure and the numerical parameter values of these models, as illustrated in Figure 7.1.

In this thesis, we have inferred homogeneous networks. However, the evolutionary approach based on AGE could also be used to reverse engineer heterogeneous networks with different types of components and interactions. In

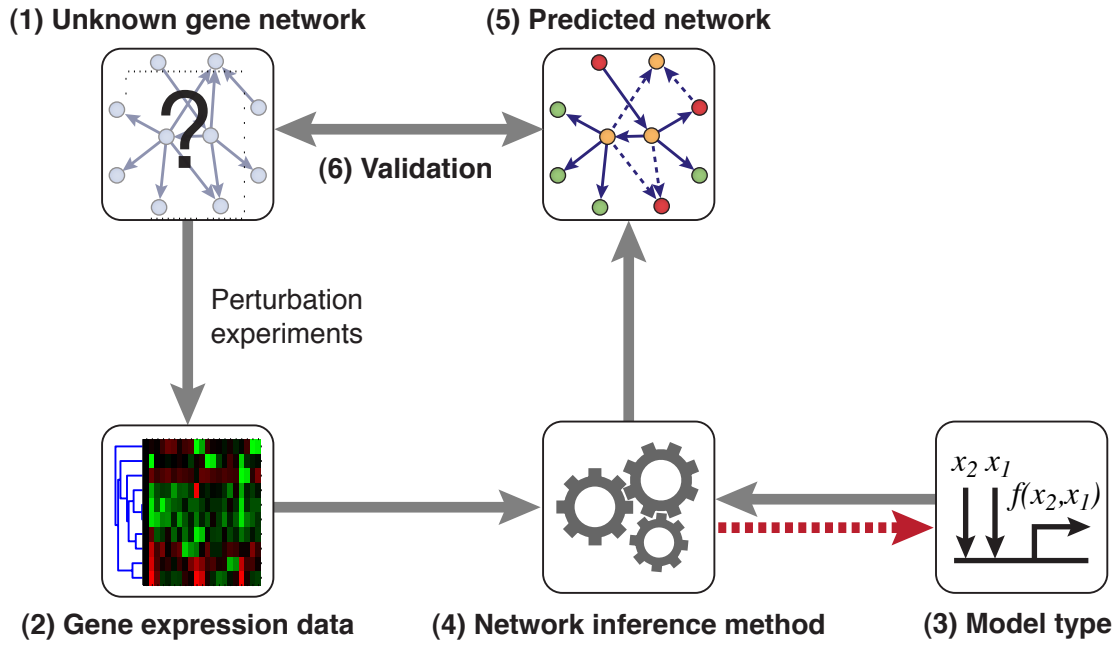


Figure 7.1: Reverse engineering heterogeneous networks. This diagram was used in Figure 1.2 to explain reverse engineering. Here, we have added the dashed arrow, representing a new approach for which the model type is not fixed beforehand. Instead, the network inference method uses the information contained in the gene expression data to infer both the network structure and the model types for the different genes and interactions. Consequently, the predicted networks may be heterogeneous (composed of different types of genes and interactions).

fact, we are already applying AGE to evolve heterogeneous networks in other domains than gene-network reverse engineering. For example, AGE has been used to evolve analog electronic circuits, which are composed of different components such as transistors and capacitors (Mattiussi and Floreano, 2007). We are also using AGE to evolve artificial neural networks (ANNs) for classification and robotic control problems. These ANNs are composed of different types of neurons, such as input and output neurons, hidden neurons, and modulatory neurons (Dürr et al., 2008, 2009).

Studying the feasibility and data requirements for reverse engineering of heterogeneous gene networks with AGE is an important field of enquiry for future research efforts.

7.2.3 Realistic *in silico* performance assessment: DREAM or reality?

Due to the complexity and our incomplete understanding of the biological mechanisms of gene regulation, reverse engineering methods are forced to make simplifying assumptions. For example, such assumptions may concern the structure of the networks (e.g., Bayesian networks are based on the assumption that there are no cycles), the noise in the gene expression data (e.g., assumption of Gaussian noise), or the dynamics of the networks (e.g., assumption that a particular phenomenological model type provides a good approximation of regulatory dynamics). For realistic *in silico* performance assessment, it is essential that the benchmarks are not based on the same simplifying assumptions as the reverse engineering methods.

In this thesis, we present tools for the generation of such benchmarks. Even though these *in silico* benchmarks are far from being realistic to the extent that they could replace *in vivo* performance assessment, they are based on more accurate and realistic models and assumptions than common reverse engineering methods. For example, the kinetic model—though still extremely simplified compared to the real biological mechanisms—does not make two of the key simplifying assumptions of most reverse engineering methods. First, mRNA and proteins are not lumped into a single state variable. Second, the regulation (input function) of genes is not phenomenologically approximated by additive or multiplicative terms, but is derived using the thermodynamical approach.

In the DREAM3 *in silico* challenge, we added Gaussian noise to the gene expression data. In recent work for the DREAM4 *in silico* challenge, which will be described elsewhere, we have adopted a more realistic approach to model the noise. First, we use stochastic simulations to model internal noise in the dynamics of the networks (Gillespie, 2000). Second, we add measurement noise to the generated gene expression datasets using a model of noise observed in microarrays (Tu et al., 2002).

In future work, we will further extend the kinetic model to include interactions between proteins and signaling molecules. As our understanding of other regulatory mechanisms progresses (e.g., the regulatory functions of non-coding RNAs and chromatin states), these mechanisms could also be modeled in the *in silico* networks to further improve the realism of the benchmarks.

7.3 Conclusion

In this thesis we demonstrate that, by taking inspiration from the mechanisms that enable the evolution of complex gene regulatory networks in nature, it is possible to design powerful reverse engineering methods for these same networks. The proposed evolutionary reverse engineering approach addresses two key problems in gene network inference. First, it permits prior biological knowledge to be embedded in the reverse engineering method. Second, it is compatible with a broad range of nonlinear models, which permits the exploration of more detailed and accurate model types than those currently used in gene network inference. For example, the evolutionary approach could provide the necessary framework for inference of heterogenous networks, where not all genes and interactions are modeled in the same way.

In an insightful article, Fisher and Henzinger (2007) distinguish between mathematical models and *executable models* of biological systems. An executable model is "an algorithm for an abstract execution engine to mimic a design or natural phenomenon" (Fisher and Henzinger, 2007). AGE is such an executable model of the evolution of gene regulatory networks. So far, we have applied AGE primarily as a *tool* for evolutionary synthesis and reverse engineering of dynamical networks. However, AGE could also be used as a *model* to study evolutionary dynamics of gene regulatory networks. Previously, conventional evolutionary algorithms—which are based on fundamentally different genetic encodings than biological evolution—were used for this purpose, for example to study emergence of robustness (Siegal and Bergman, 2002), evolvability (Bergman and Siegal, 2003; Ciliberti et al., 2007), and modularity (Kashtan and Alon, 2005). AGE would be ideally suited to study evolutionary dynamics of gene regulatory networks in a more realistic setting. The insights thus gained may in turn be used to further improve AGE for the purpose of evolutionary reverse engineering.

Nowadays, computer-aided design (CAD) is essential in many disciplines and industries. For example, cars are being designed and even crash-tested in simulation, before expensive physical prototypes are built. Early attempts in the 1980s to crash-test cars in simulation (i.e., to do *in silico* performance assessment) were unfortunately "not taken very seriously since the underlying models were very primitive and inaccurate" (Thomke, 1998). However, the accuracy of

advanced crash simulations can now exceed prototype-to-prototype variation of physical crashes (Thomke, 1998). Similar to the early attempts of simulating car crashes, current inference and simulation methods for gene regulatory networks may be considered "primitive and inaccurate". However, we are convinced that—as the young field of computational systems biology matures—simulation tools will play an equally important role in molecular biology, biotechnology, and drug design, as they do now in traditional engineering disciplines.

A Supplementary information of Chapter 2

A.1 Implementation of the implicit encoding

As described in Section 2.2.1, each gene has a sequence s_{cis} and s_{trans}, which implicitly encode the regulatory interactions and their strength (see Figure 2.2). In this Appendix, we define the interaction map I that is used to decode the strength of the regulatory interactions from the respective sequences

$$w_{ij} = I(s_{\text{trans},j}, s_{\text{cis},i}) \tag{A.1}$$

Mattiussi (2005) derived a set of requirements that the interaction map should meet in order to be evolvable. We will not include a detailed discussion here, but merely mention some of the most important requirements that were considered in the choice of the interaction map. The interaction map must

- possess a many-to-one nature in order to allow for neutral evolution;
- permit both gradual changes and major reorganizations of the network structure and interaction strengths;
- be able to determine multiple interactions that can evolve independently from each other, i.e., that are encoded by distinct subsequences of s_{cis} and s_{trans};
- be applicable to sequences of variable length;
- have reasonably low computational complexity.

Simple techniques of sequence comparison, such as Hamming distance, do not comply with these requirements. Mattiussi (2005) proposed an interaction

map based on *local alignment* of the two sequences, which does fulfill the above requirements. Below, we give a brief description of this interaction map—a detailed discussion is provided by Mattiussi (2005).

A.1.1 The interaction map

We define the interaction map I as a composed map, which is formed by a *generic interaction map* $L(s_{\text{trans}}, s_{\text{cis}})$ and a *model-specific map* $N(l)$

$$w = I(s_{\text{trans}}, s_{\text{cis}}) = N(L(s_{\text{trans}}, s_{\text{cis}})) \tag{A.2}$$

The generic interaction map $L(s_{\text{trans}}, s_{\text{cis}})$ is defined as the local alignment score of the two sequences (Gusfield, 1997), using the alignment parameters specified by Mattiussi and Floreano (2007). Figuratively speaking, the closer the match between two subsequences of s_{cis} and s_{trans}, the stronger is the interaction.

The network-specific map $N : [l_{\text{min}}, l_{\text{max}}] \mapsto [0, w_{\text{max}}]$ transforms integer local alignment scores l into floating-point weights. Alignment scores smaller than the threshold l_{min} are mapped to zero (no interaction), scores greater than l_{max} are truncated to w_{max}, and scores in between are mapped linearly or logarithmically onto the positive interval $[0, w_{\text{max}}]$. In the experiments reported in this thesis, we always used a linear mapping. For the experiments with the standard sigmoid model (Chapter 2), we set $l_{\text{min}} = 11$, $l_{\text{max}} = 31$, and $w_{\text{max}} = 2$. For the experiments with the log-sigmoid model (Chapters 3 and 5), we set $l_{\text{min}} = 10$, $l_{\text{max}} = 41$, and $w_{\text{max}} = 5$.

The alignment score given by the interaction map is always positive. In order to represent negative weights, genes actually have two sequences $s_{\text{trans}+}$ and $s_{\text{trans}-}$, which define enhancing and repressing regulatory interactions, respectively (for simplicity, only one sequence s_{trans} was described in Section 2.2.1). The weight is defined by the stronger interaction

$$w = \begin{cases} +I(s_{\text{trans}+}, s_{\text{cis}}) & \text{if} \quad I(s_{\text{trans}+}, s_{\text{cis}}) \geq I(s_{\text{trans}-}, s_{\text{cis}}) \\ -I(s_{\text{trans}-}, s_{\text{cis}}) & \text{otherwise} \end{cases} \tag{A.3}$$

Thus, there are two types of interactions in the networks (enhancing and repressing), which are defined by distinct sequences $s_{\text{trans}+}$ and $s_{\text{trans}-}$. In the same way, additional types of interactions could be encoded for reverse engineering of *heterogeneous networks* (see Section 7.2.2). For example, sequences $s_{\text{cis\_or}}$ and

$s_{\text{cis\_and}}$ could be defined for regulatory inputs that act independently (OR-type inputs) and synergistically (AND-type inputs), respectively.

B Supplementary information of Chapter 3

B.1 Elementary two-dimensional input functions

Here, we define the eight elementary types of two-dimensional input functions on which the standard sigmoid model and the log-sigmoid model were compared in Section 3.2. We define these input functions using the thermodynamical model of transcriptional regulation that is described in Section 5.5.1. Remember that the input function $f(y_1, y_2)$ computes the *relative activation* of the gene, given the transcription factor (TF) concentrations y_1 and y_2 (cf. Equation 5.2). The relative activation is between 0 (the gene is shut off) and 1 (the gene is maximally activated).

A gene that is regulated by two transcription factors (TFs) has four states (see Figure B.1): none of the TFs is bound (S_0), only the first TF is bound (S_1), only the second TF is bound (S_2), or both TFs are bound (S_3). The mean activation of the gene can be computed from the probability of each state (cf. Equation 5.5)

$$f(y_1, y_2) = \alpha_0 P\{S_0\} + \alpha_1 P\{S_1\} + \alpha_2 P\{S_2\} + \alpha_3 P\{S_3\} \tag{B.1}$$

where $\alpha_i \in [0, 1]$ is the relative activation of the gene in state S_i. Using thermodynamics, the probability $P\{S_m\}$ can be computed for every state m, given the TF concentrations y_1 and y_2. The resulting expression for the two-input function is (cf. Equation 5.6)

$$f(y_1, y_2) = \frac{\alpha_0 \,+\, \alpha_1\, v_1 \,+\, \alpha_2\, v_2 \,+\, \alpha_3\, \rho\, v_1\, v_2}{1 \,+\, v_1 \,+\, v_2 \,+\, \rho\, v_1\, v_2} \qquad \text{with} \quad v_j \,=\, \left(\frac{y_j}{k_j}\right)^{n_j} \tag{B.2}$$

where k_j are the dissociation constants, n_j the Hill coefficients, and ρ is the cooperativity factor of the two TFs. For $\rho = 1$, the binding of the two TFs to the

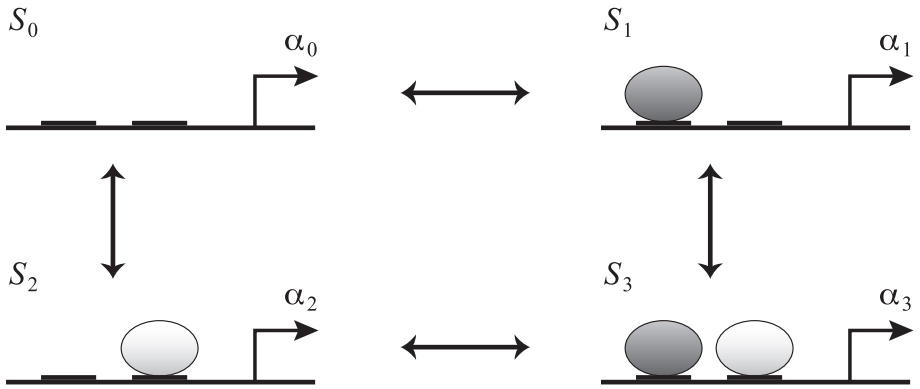


Figure B.1: A gene that is regulated by two TFs has four states. Either none, only the first, only the second, or both TFs are bound. The gene may have a different relative activation α_i in each state.

Table B.1: Parameter values defining the elementary two-input functions.

	OR	NOR	AND	NAND	IMPLIES	NIMPLIES	XOR	EQUAL
α_0	0	1	0	1	1	0	0	1
α_1	1	0	0	1	1	0	1	0
α_2	1	0	0	1	0	1	1	0
α_3	1	0	1	0	1	0	0	1

cis-regulatory region of the gene is independent; for $\rho > 1$, it is cooperative; and for $\rho < 1$, it is competitive (e.g., their binding sites are overlapping).

The eight elementary two-input functions correspond to the eight logic functions that can be performed on two inputs (AND, NAND, OR, NOR, IMPLIES, NIMPLIES, EQUAL, and XOR). They are defined by choosing different relative activations α_m for Equation (B.2), as specified in Table B.1. The Hill coefficients n_1 and n_2 only affect the steepness of the transitions and not the regulatory logic of the gene. The same is true for parameters k_1 and k_2, which only define the thresholds of the transitions. For the elementary input functions, we set all these parameters to one: $n_1 = n_2 = k_1 = k_2 = 1$. The cooperativity factor ρ is also set to one. The resulting input functions are shown in Figure B.2 (see also Figure 3.1B), they could be realized at the level of a gene as follows:

- **OR** The OR input function can be realized by two activators (enhancing

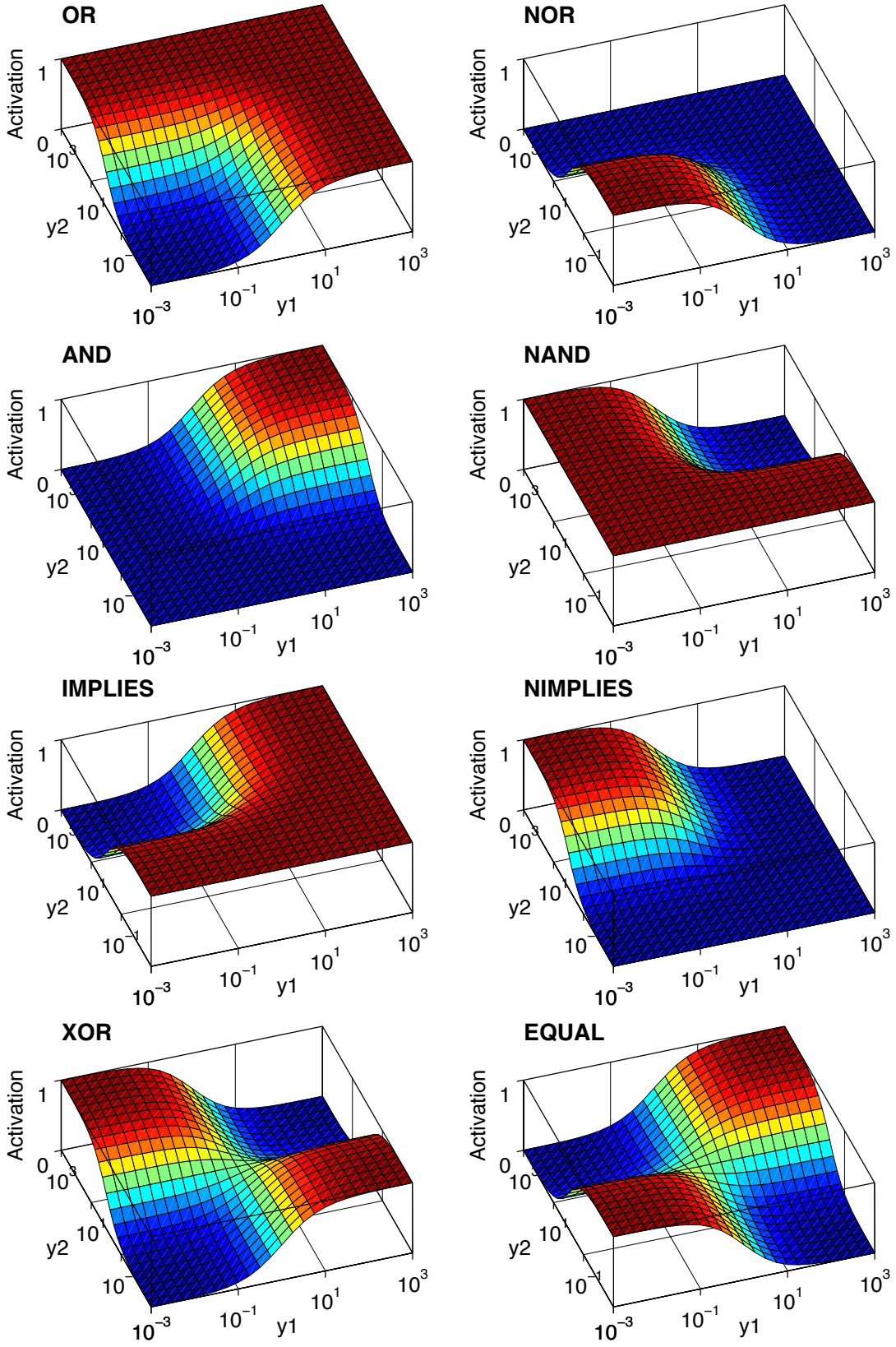


Figure B.2: The eight elementary two-input functions.

TFs) that have the capacity to fully activate gene expression independently from each other. In other words, the relative activation is zero when no activator is bound and it is maximal when the first *or* the second activator is bound.

- **NOR** The NOR input function can be realized in the same way as the OR input function, but with two repressors instead of two activators. The activation is maximal when no repressor is bound (i.e., the gene is constitutively active) and transcription is shut off when either the first *or* the second repressor is bound.

- **AND** There are two different possibilities to realize an AND input function. The first possibility is with two TFs that bind independently ($\rho = 1$), but form together a platform that initiates gene expression. Hence, there is only activation when the first *and* the second TF are bound at the promoter at the same time (state S_3). The second possibility to realize an AND input function is through strong cooperative binding of the two TFs ($\rho \gg 1$). In the extreme case, the two TFs first form a hetero-dimer and this complex binds to the promoter and activates gene expression.

- **NAND** The NAND input function may be realized by two repressors analogous to the implementation of the AND module by two activators. The first possibility is that the two repressors bind independently ($\rho = 1$), but are only effective when they are both bound simultaneously. Similar to the AND input function, the NAND input function can also be realized through strong cooperative binding of the two repressors ($\rho \gg 1$).

- **IMPLIES** The IMPLIES and NIMPLIES input functions are realized by an activator and a repressor. Without loss of generality we can assume that the first TF is the activator and the second TF is the repressor. The IMPLIES input function can be realized by a gene that is constitutively active, but transcription is shut off when the repressor is bound ($\alpha_0 = 1$, $\alpha_2 = 0$). However, the activator can disable the repressor, thereby restoring transcription ($\alpha_1 = 1$, $\alpha_3 = 1$).

- **NIMPLIES** The NIMPLIES input function is similar to the IMPLIES input function, but now the gene is only active when the activator is bound

and not the repressor. Thus, the relative activation is one in state S_1 and zero in all other states.

- **XOR** The XOR input function could be realized by two activators that, when both present, form a complex that is not activating anymore.
- **EQUAL** The EQUAL input function is the counterpart to the XOR input function, using two repressors instead of two activators. It could be realized by two repressors that, when bound together, form a complex that is not repressing anymore.

As discussed in Section 3.2, biological input functions are likely to be intermediates between these elementary input functions (i.e., the relative activations would be continuous values between zero and one).

B.2 Five-gene repressilator

In this section, we describe the complete model of the *in silico* five-gene network that we used in Chapter 3 as test case. The structure of this network is a loop of five inhibitory connections (see Figure 3.4). The dynamics are based on the log-sigmoid model, which can be written in simplified form because the network has only one input per gene (cf. Equation 3.3b)

$$\frac{dx_i}{dt} = m_i \cdot \frac{e^{b_i}\, x_j^{w_{ij}}}{1 + e^{b_i}\, x_j^{w_{ij}}} - \lambda_i x_i \tag{B.3}$$

where x_i is the expression level of gene i and x_j the expression level of its repressor. The parameter m_i is the maximum transcription rate, b_i is the bias that relates to the basal transcription rate, w_{ij} is the weight of the repressing interaction, and λ_i is the degradation rate. The values of these parameters are specified in Table B.2.

We simulated two time-series of the same duration and with the same number of time points as the data of the DREAM2 challenge (see Section 3.3). The initial conditions that we used in the two time-series for the five genes are given in Table B.3.

Table B.2: Parameter values for the five genes of the repressilator.

Parameter	Value
m_1	0.9
b_1	3.2
λ_1	0.15
w_{15}	-3.6
m_2	1.1
b_2	2.2
λ_2	0.20
w_{21}	-4.1
m_3	0.6
b_3	5.0
λ_3	0.09
w_{32}	-4.3
m_4	1.1
b_4	3.7
λ_4	0.12
w_{43}	-3.9
m_5	1.0
b_5	4.0
λ_5	0.10
w_{54}	-3.3

Table B.3: Initial conditions used for the two time-series of the repressilator.

Variable	Time-series 1	Time-series 2
$x_1(0)$	1.0	0.8
$x_2(0)$	7.6	0.2
$x_3(0)$	0.1	0.3
$x_4(0)$	4.0	1.7
$x_5(0)$	0.1	7.1

Bibliography

Ackers, G. K., Johnson, A. D., and Shea, M. A. (1982). Quantitative model for gene regulation by lambda phage repressor. *Proc Natl Acad Sci U S A*, 79(4):1129–33.

Airoldi, E. M. (2007). Getting started in probabilistic graphical models. *PLoS Computational Biology*, 3(12):e252.

Alon, U. (2007a). *An Introduction to Systems Biology: Design Principles of Biological Circuits*. Chapman & Hall/CRC.

Alon, U. (2007b). Network motifs: theory and experimental approaches. *Nat Rev Genet*, 8(6):450–461.

Ambros, V. and Chen, X. (2007). The regulation of genes and genomes by small rnas. *Development*, 134(9):1635–1641.

Anastassiou, D. (2007). Computational analysis of the synergy among multiple interacting genes. *Mol Syst Biol*, 3:83.

Bäck, T., Fogel, D. B., and Michalewicz, Z., editors (2000). *Evolutionary Computation 1: Basic Algorithms and Operators*. Institute of Physics, Bristol.

Balaji, S., Babu, M. M., Iyer, L. M., Luscombe, N. M., and Aravind, L. (2006). Comprehensive analysis of combinatorial regulation using the transcriptional regulatory network of yeast. *J Mol Biol*, 360:213–227.

Baralla, A., Mentzen, W. I., and de la Fuente, A. (2009). Inferring gene networks: dream or nightmare? *Ann N Y Acad Sci*, 1158:246–256.

Basso, K., Margolin, A. A., Stolovitzky, G., Klein, U., Dalla-Favera, R., and Califano, A. (2005). Reverse engineering of regulatory networks in human B cells. *Nat Genet*, 37:382–390.

Battogtokh, D., Asch, D. K., Case, M. E., Arnold, J., and Schuttler, H.-B. (2002). An ensemble method for identifying regulatory circuits with special reference to the qa gene cluster of neurospora crassa. *Proc Natl Acad Sci U S A*, 99:16904–16909.

Becskei, A. and Serrano, L. (2000). Engineering stability in gene networks by autoregulation. *Nature*, 405(6786):590–3.

Bergman, A. and Siegal, M. L. (2003). Evolutionary capacitance as a general feature of complex gene networks. *Nature*, 424(6948):549–552.

Bintu, L., Buchler, N., Garcia, H., Gerland, U., Hwa, T., Kondev, J., and Phillips, R. (2005). Transcriptional regulation by the numbers: models. *Curr Opin Genet Dev*, 15(2):116–124.

Bolouri, H. and Davidson, E. H. (2002). Modeling transcriptional regulatory networks. *Bioessays*, 24(12):1118–29.

Bongard, J. (2002). Evolving modular genetic regulatory networks. In *Proceedings of the 2002 Congress on Evolutionary Computation, CEC '02*, volume 2, pages 1872–1877.

Bongard, J. and Lipson, H. (2007). Automated reverse engineering of nonlinear dynamical systems. *Proc Natl Acad Sci U S A*, 104(24):9943–9948.

Bonneau, R., Facciotti, M. T., Reiss, D. J., Schmid, A. K., Pan, M., Kaur, A., Thorsson, V., Shannon, P., Johnson, M. H., Bare, J. C., Longabaugh, W., Vuthoori, M., Whitehead, K., Madar, A., Suzuki, L., Mori, T., Chang, D.-E., Diruggiero, J., Johnson, C. H., Hood, L., and Baliga, N. S. (2007). A predictive model for transcriptional control of physiology in a free living cell. *Cell*, 131(7):1354–1365.

Bonneau, R., Reiss, D. J., Shannon, P., Facciotti, M., Hood, L., Baliga, N. S., and Thorsson, V. (2006). The inferelator: an algorithm for learning parsimonious regulatory networks from systems-biology data sets de novo. *Genome Biol*, 7(5):R36.

Bower, J. M. and Bolouri, H., editors (2004). *Computational modeling of genetic and biochemical networks*. MIT Press.

Boyle, E. I., Weng, S., Gollub, J., Jin, H., Botstein, D., Cherry, J. M., and Sherlock, G. (2004). Go::termfinder–open source software for accessing gene ontology information and finding significantly enriched gene ontology terms associated with a list of genes. *Bioinformatics*, 20:3710–3715.

Bray, D. (1995). Protein molecules as computational elements in living cells. *Nature*, 376(6538):307–312.

Brazhnik, P. (2005). Inferring gene networks from steady-state response to single-gene perturbations. *J Theor Biol*, 237:427–440.

Browning, D. F. and Busby, S. J. (2004). The regulation of bacterial transcription initiation. *Nat Rev Microbiol*, 2(1):57–65.

Buchler, N. E., Gerland, U., and Hwa, T. (2003). On schemes of combinatorial transcription logic. *Proc Natl Acad Sci U S A*, 100(9):5136–5141.

Burgess-Beusse, B., Farrell, C., Gaszner, M., Litt, M., Mutskov, V., Recillas-Targa, F., Simpson, M., West, A., and Felsenfeld, G. (2002). The insulation of genes from external enhancers and silencing chromatin. *Proc Natl Acad Sci U S A*, 99 Suppl 4:16433–16437.

Butte, A. J., Tamayo, P., Slonim, D., Golub, T. R., and Kohane, I. S. (2000). Discovering functional relationships between rna expression and chemotherapeutic susceptibility using relevance networks. *Proc Natl Acad Sci U S A*, 97(22):12182–12186.

Camillo, B. D., Toffolo, G., and Cobelli, C. (2009). A gene network simulator to assess reverse engineering algorithms. *Ann N Y Acad Sci*, 1158:125–142.

Cantone, I., Marucci, L., Iorio, F., Ricci, M. A., Belcastro, V., Bansal, M., Santini, S., di Bernardo, M., di Bernardo, D., and Cosma, M. P. (2009). A yeast synthetic network for in vivo assessment of reverse-engineering and modeling approaches. *Cell*, 137(1):172–181.

Cardelli, L. (2005). Abstract machines of systems biology. *Lecture Notes in Computer Science*, 3737 LNBI:145–168.

Chechik, G., Oh, E., Rando, O., Weissman, J., Regev, A., and Koller, D. (2008). Activity motifs reveal principles of timing in transcriptional control of the yeast metabolic network. *Nat Biotechnol*, 26(11):1251–1259.

Chen, K.-C., Wang, T.-Y., Tseng, H.-H., Huang, C.-Y. F., and Kao, C.-Y. (2005). A stochastic differential equation model for quantifying transcriptional regulatory network in Saccharomyces cerevisiae. *Bioinformatics*, 21(12):2883–2890.

Ciliberti, S., Martin, O. C., and Wagner, A. (2007). Innovation and robustness in complex regulatory gene networks. *Proc Natl Acad Sci U S A*, 104(34):13591–13596.

Coleman, T. and Li, Y. (1996). An interior, trust region approach for nonlinear minimization subject to bounds. *SIAM Journal on Optimization*, 6:418–445.

Corne, D., P. C. (2004). Investigating issues in the reconstructability of genetic regulatory networks. In *Proceedings of the 2004 Congress on Evolutionary Computation, CEC2004*, pages 582–589.

Csete, M. E. and Doyle, J. C. (2002). Reverse engineering of biological complexity. *Science*, 295(5560):1664–1669.

Davis, J. and Goadrich, M. (2006). The relationship between precision-recall and roc curves. In *ICML '06: Proceedings of the 23rd international conference on Machine learning*, pages 233–240.

de la Fuente, A., Bing, N., Hoeschele, I., and Mendes, P. (2004). Discovery of meaningful associations in genomic data using partial correlation coefficients. *Bioinformatics*, 20(18):3565–3574.

de la Fuente, A., Brazhnik, P., and Mendes, P. (2002). Linking the genes: inferring quantitative gene networks from microarray data. *Trends Genet*, 18:395–398.

de la Fuente, A. and Makhecha, D. (2006). Unravelling gene networks from noisy under-determined experimental perturbation data. *IEE Proceedings - Systems Biology*, 153(4):257–262.

Dekel, E. and Alon, U. (2005). Optimality and evolutionary tuning of the expression level of a protein. *Nature*, 436(7050):588–592.

Deng, X., Geng, H., and Ali, H. (2005). EXAMINE: A computational approach to reconstructing gene regulatory networks. *Biosystems*, 81:125–136.

D'Haeseleer, P., Wen, X., Fuhrman, S., and Somogyi, R. (1999). Linear modeling of mRNA expression levels during CNS development and injury. *Pac Symp Biocomput*, pages 41–52.

di Bernardo, D., Thompson, M. J., Gardner, T. S., Chobot, S. E., Eastwood, E. L., Wojtovich, A. P., Elliott, S. J., Schaus, S. E., and Collins, J. J. (2005). Chemogenomic profiling on a genome-wide scale using reverse-engineered gene networks. *Nat Biotechnol*, 23(3):377–83.

Dietterich, T. (2000). Ensemble methods in machine learning. In *Multiple Classifier Systems*, volume 1857 of *Lecture Notes in Computer Science*, pages 1–15.

Dürr, P., Karlen, W., Guignard, J., Mattiussi, C., and Floreano, D. (2009). Evolutionary Selection of Features for Neural Sleep/Wake Discrimination. *Journal of Artificial Evolution and Applications*.

Dürr, P., Mattiussi, C., and Floreano, D. (2006). Neuroevolution with Analog Genetic Encoding. In *Parallel Problem Solving from Nature - PPSN IX*, volume 4193 of *LNCS*, pages 671–680.

Dürr, P., Mattiussi, C., Soltoggio, A., and Floreano, D. (2008). Evolvability of Neuromodulated Learning for Robots. In *The 2008 ECSIS Symposium on Learning and Adaptive Behavior in Robotic Systems*, pages 41–46, Los Alamitos, CA. IEEE Computer Society.

Ellis, T., Wang, X., and Collins, J. J. (2009). Diversity-based, model-guided construction of synthetic gene networks with predicted functions. *Nat Biotechnol*, 27(5):465–71.

Elowitz, M. B. and Leibler, S. (2000). A synthetic oscillatory network of transcriptional regulators. *Nature*, 403(6767):335–338.

Faith, J. J., Hayete, B., Thaden, J. T., Mogno, I., Wierzbowski, J., Cottarel, G., Kasif, S., Collins, J. J., and Gardner, T. S. (2007). Large-scale mapping and validation of escherichia coli transcriptional regulation from a compendium of expression profiles. *PLoS Biol*, 5(1):e8.

Fisher, J. and Henzinger, T. A. (2007). Executable cell biology. *Nat Biotechnol*, 25(11):1239–1249.

Floreano, D. and Mattiussi, C. (2008). *Bio-Inspired Artificial Intelligence: Theories, Methods, and Technologies*. Intelligent Robotics and Autonomous Agents. MIT Press, Cambridge, MA.

Fridlyander, I. (2008). From the history of aluminum alloys. *Herald of the Russian Academy of Sciences*, 78(4):416–420.

Friedman, N. (2004). Inferring cellular networks using probabilistic graphical models. *Science*, 303:799–805.

Friedman, N., Linial, M., Nachman, I., and Pe'er, D. (2000). Using Bayesian networks to analyze expression data. *J Comput Biol*, 7(3-4):601–20.

Gardner, T. S., Cantor, C. R., and Collins, J. J. (2000). Construction of a genetic toggle switch in Escherichia coli. *Nature*, 403(6767):339–42.

Gardner, T. S., di Bernardo, D., Lorenz, D., and Collins, J. J. (2003). Inferring genetic networks and identifying compound mode of action via expression profiling. *Science*, 301:102–5.

Gardner, T. S. and Faith, J. J. (2005). Reverse-engineering transcription control networks. *Physics of Life Reviews*, 2(1):65–88.

Gillespie, D. T. (2000). The chemical langevin equation. *The Journal of Chemical Physics*, 113(1):297–306.

Gupta, A., Varner, J. D., and Maranas, C. D. (2005). Large-scale inference of the transcriptional regulation of bacillus subtilis. *Comput Chem Eng*, 29:565–576.

Gusfield, D. (1997). *Algorithms on Stings, Trees, and Sequences: Computer Science and Computational Biology*, volume 28. ACM Press, New York, NY, USA.

Hartemink, A. J., Gifford, D. K., Jaakkola, T. S., and Young, R. A. (2002). Combining location and expression data for principled discovery of genetic regulatory network models. *Pac Symp Biocomput*, pages 437–49.

Hartwell, L. H., Hopfield, J. J., Leibler, S., and Murray, A. W. (1999). From molecular to modular cell biology. *Nature*, 402:C47–C52.

Haynes, B. C. and Brent, M. R. (2009). Benchmarking regulatory network reconstruction with grendel. *Bioinformatics*, 25(6):801–807.

Hintze, A. and Adami, C. (2008). Evolution of complex modular biological networks. *PLoS Comput Biol*, 4(2):e23.

Hong, E. L., Balakrishnan, R., Dong, Q., Christie, K. R., Park, J., Binkley, G., Costanzo, M. C., Dwight, S. S., Engel, S. R., and et al., D. G. F. (2008). Gene ontology annotations at SGD: new data sources and annotation methods. *Nucleic Acids Res*, 36:D577–D581.

Hoops, S., Sahle, S., Gauges, R., Lee, C., Pahle, J., Simus, N., Singhal, M., Xu, L., Mendes, P., and Kummer, U. (2006). Copasi–a complex pathway simulator. *Bioinformatics*, 22(24):3067–3074.

Huynen, M. A., Stadler, P. F., and Fontana, W. (1996). Smoothness within ruggedness: the role of neutrality in adaptation. *Proc Natl Acad Sci U S A*, 93(1):397–401.

Iba, H. and Mimura, A. (2002). Inference of a gene regulatory network by means of interactive evolutionary computing. *Information Sciences*, 145:225–236.

Jaeger, J., Blagov, M., Kosman, D., Kozlov, K. N., Manu, Myasnikova, E., Surkova, S., Vanario-Alonso, C. E., Samsonova, M., Sharp, D. H., and Reinitz, J. (2004a). Dynamical analysis of regulatory interactions in the gap gene system of drosophila melanogaster. *Genetics*, 167:1721–1737.

Jaeger, J., Surkova, S., Blagov, M., Janssens, H., Kosman, D., Kozlov, K. N., Manu, Myasnikova, E., Vanario-Alonso, C. E., Samsonova, M., Sharp, D. H., and Reinitz, J. (2004b). Dynamic control of positional information in the early drosophila embryo. *Nature*, 430:368–371.

Jaynes, E. (1984). Prior information and ambiguity in inverse problems. *SIAM - AMS Proceedings*, 14:151–166.

Kashtan, N. and Alon, U. (2005). Spontaneous evolution of modularity and network motifs. *Proc Natl Acad Sci U S A*, 102(39):13773–13778.

Kholodenko, B. N., Kiyatkin, A., Bruggeman, F. J., Sontag, E., Westerhoff, H. V., and Hoek, J. B. (2002). Untangling the wires: a strategy to trace functional interactions in signaling and gene networks. *Proc Natl Acad Sci U S A*, 99:12841–12846.

Kikuchi, S., Tominaga, D., Arita, M., Takahashi, K., and Tomita, M. (2003). Dynamic modeling of genetic networks using genetic algorithm and s-system. *Bioinformatics*, 19:643–650.

Kimura, S., Ide, K., Kashihara, A., Kano, M., Hatakeyama, M., Masui, R., Nakagawa, N., Yokoyama, S., Kuramitsu, S., and Konagaya, A. (2005). Inference of S-system models of genetic networks using a cooperative coevolutionary algorithm. *Bioinformatics*, 21:1154–63.

Kitano, H. (2002). Systems biology: a brief overview. *Science*, 295(5560):1662–1664.

Kitano, H. (2004). Biological robustness. *Nat Rev Genet*, 5(11):826–837.

Kitano, H. (2007). Towards a theory of biological robustness. *Mol Syst Biol*, 3:137.

Kremling, A., Fischer, S., Gadkar, K., Doyle, F. J., Sauter, T., Bullinger, E., Allgöwer, F., and Gilles, E. D. (2004). A benchmark for methods in reverse engineering and model discrimination: problem formulation and solutions. *Genome Res*, 14:1773–1785.

Ladd, A. N. and Cooper, T. A. (2002). Finding signals that regulate alternative splicing in the post-genomic era. *Genome Biol*, 3(11):reviews0008.

Lazebnik, Y. (2002). Can a biologist fix a radio? - or, what i learned while studying apoptosis. *Cancer Cell*, 2(3):179–182.

Li, P., Zhang, C., Perkins, E. J., Deng, Y., and Gong, P. (2009). Gene networks inference and validation of time-delayed dynamic bayesian network. *PLoS One*, to appear.

Liao, J., Boscolo, R., Yang, Y.-L., Tran, L. M., Sabatti, C., and Roychowdhury, V. (2003). Network component analysis: reconstruction of regulatory signals in biological systems. *Proc Natl Acad Sci U S A*, 100(26):15522–7.

Ljung, L. (1999). *System identification: Theory for the user*. Prentice Hall, Upper Saddle River, NJ.

Ma, H., Buer, J., and Zeng, A. (2004a). Hierarchical structure and modules in the Escherichia coli transcriptional regulatory network revealed by a new top-down approach. *BMC Bioinformatics*, 5:199.

Ma, H.-W., Kumar, B., Ditges, U., Gunzer, F., Buer, J., and Zeng, A.-P. (2004b). An extended transcriptional regulatory network of Escherichia coli and analysis of its hierarchical structure and network motifs. *Nucleic Acids Res*, 32:6643–6649.

Manu, Surkova, S., Spirov, A. V., Gursky, V. V., Janssens, H., Kim, A.-R., Radulescu, O., Vanario-Alonso, C. E., Sharp, D. H., Samsonova, M., and Reinitz, J. (2009). Canalization of gene expression in the drosophila blastoderm by gap gene cross regulation. *PLoS Biol*, 7(3):e1000049–.

Mattiussi, C. (2005). *Evolutionary synthesis of analog networks*. PhD thesis, Ecole Polytechnique Fédérale de Lausanne, Lausanne.

Mattiussi, C., Dürr, P., and Floreano, D. (2007). Center of mass encoding: A self-adaptive representation with adjustable redundancy for real-valued parameters. In *Proceedings of the Genetic and Evolutional Computation Conference, GECCO2007*, pages 1304–1311.

Mattiussi, C. and Floreano, D. (2007). Analog Genetic Encoding for the Evolution of Circuits and Networks. *IEEE Trans Evol Comput*, 11(5):596–607.

Mattiussi, C., Marbach, D., Dürr, P., and Floreano, D. (2008). The age of analog networks. *AI Mag*, 29:63–76.

Mayo, A. E., Setty, Y., Shavit, S., Zaslaver, A., and Alon, U. (2006). Plasticity of the cis-regulatory input function of a gene. *PLoS Biol*, 4(4):e45.

Mendes, P., Sha, W., and Ye, K. (2003). Artificial gene networks for objective comparison of analysis algorithms. *Bioinformatics*, 19 Suppl 2:ii122–129.

Milo, R., Itzkovitz, S., Kashtan, N., Levitt, R., Shen-Orr, S., Ayzenshtat, I., Sheffer, M., and Alon, U. (2004). Superfamilies of evolved and designed networks. *Science*, 303:1538–1542.

Milo, R., Shen-Orr, S., Itzkovitz, S., Kashtan, N., Chklovskii, D., and Alon, U. (2002). Network motifs: simple building blocks of complex networks. *Science*, 298(5594):824–827.

Mjolsness, E., Sharp, D. H., and Reinitz, J. (1991). A connectionist model of development. *J Theor Biol*, 152(4):429–453.

Moles, C. G., Mendes, P., and Banga, J. R. (2003). Parameter estimation in biochemical pathways: A comparison of global optimization methods. *Genome Research*, 13:2467–2474.

Moult, J., Fidelis, K., Kryshtafovych, A., Rost, B., Hubbard, T., and Tramontano, A. (2007). Critical assessment of methods of protein structure prediction-round vii. *Proteins*, 69 Suppl 8:3–9.

Mukherjee, S. and Speed, T. P. (2008). Network inference using informative priors. *Proc Natl Acad Sci U S A*, 105:14313–14318.

Nachman, I., Regev, A., and Friedman, N. (2004). Inferring quantitative models of regulatory networks from expression data. *Bioinformatics*, 20 Suppl 1:I248–I256.

Needham, C. J., Bradford, J. R., Bulpitt, A. J., and Westhead, D. R. (2007). A primer on learning in bayesian networks for computational biology. *PLoS Comput Biol*, 3(8):e129.

Newman, M. E. J. (2006a). Finding community structure in networks using the eigenvectors of matrices. *Phys Rev E Stat Nonlin Soft Matter Phys*, 74(3 Pt 2):036104.

Newman, M. E. J. (2006b). Modularity and community structure in networks. *Proc Natl Acad Sci U S A*, 103(23):8577–8582.

Newman, M. E. J. and Girvan, M. (2004). Finding and evaluating community structure in networks. *Phys Rev E Stat Nonlin Soft Matter Phys*, 69(2 Pt 2):026113.

Nykter, M., Aho, T., Ahdesmäki, M., Ruusuvuori, P., Lehmussola, A., and Yli-Harja, O. (2006). Simulation of microarray data with realistic characteristics. *BMC Bioinformatics*, 7:349.

Parisi, F., Koeppl, H., and Naef, F. (2009). Network inference by combining biologically motivated regulatory constraints with penalized regression. *Ann N Y Acad Sci*, 1158:114–124.

Pennisi, E. (2003). Systems biology: Tracing life's circuitry. *Science*, 302:1646–1649.

Perkins, T. J., Jaeger, J., Reinitz, J., and Glass, L. (2006). Reverse engineering the gap gene network of Drosophila melanogaster. *PLoS Comput Biol*, 2:e51.

Polikar, R. (2006). Ensemble based systems in decision making. *IEEE Circuits Syst Mag*, 6:21–46.

Ravasz, E., Somera, A. L., Mongru, D. A., Oltvai, Z. N., and Barabási, A. L. (2002). Hierarchical organization of modularity in metabolic networks. *Science*, 297:1551–1555.

Reguly, T., Breitkreutz, A., Boucher, L., Breitkreutz, B., Hon, G. C., Myers, C. L., Parsons, A., Friesen, H., Oughtred, R., and et al., A. T. (2006). Comprehensive curation and analysis of global interaction networks in saccharomyces cerevisiae. *J Biol*, 5(4):11.

Reil, T. (1999). Dynamics of gene expression in an artificial genome - implications for biological and artificial ontogeny. In *ECAL '99: Proceedings of the 5th European Conference on Advances in Artificial Life*, pages 457–466, London, UK. Springer-Verlag.

Reinitz, J. and Sharp, D. H. (1995). Mechanism of eve stripe formation. *Mech Dev*, 49:133–58.

Resendis-Antonio, O., Freyre-González, J. A., Menchaca-Méndez, R., Gutiérrez-Ríos, R. M., Martínez-Antonio, A., Avila-Sánchez, C., and Collado-Vides, J. (2005). Modular analysis of the transcriptional regulatory network of E. coli. *Trends Genet*, 21(1):16–20.

Rice, J. J., Tu, Y., and Stolovitzky, G. (2005). Reconstructing biological networks using conditional correlation analysis. *Bioinformatics*, 21(6):765–773.

Rosenfeld, N., Young, J. W., Alon, U., Swain, P. S., and Elowitz, M. B. (2005). Gene regulation at the single-cell level. *Science*, 307(5717):1962–1965.

Roy, S., Werner-Washburne, M., and Lane, T. (2008). A system for generating transcription regulatory networks with combinatorial control of transcription. *Bioinformatics*, 24(10):1318–1320.

Sauer, U., Heinemann, M., and Zamboni, N. (2007). Genetics. getting closer to the whole picture. *Science*, 316(5824):550–551.

Setty, Y., Mayo, A. E., Surette, M. G., and Alon, U. (2003). Detailed map of a cis-regulatory input function. *Proc Natl Acad Sci U S A*, 100(13):7702–7.

Shea, M. A. and Ackers, G. K. (1985). The or control system of bacteriophage lambda. a physical-chemical model for gene regulation. *J Mol Biol*, 181(2):211–230.

Shen-Orr, S. S., Milo, R., Mangan, S., and Alon, U. (2002). Network motifs in the transcriptional regulation network of Escherichia coli. *Nat Genet*, 31(1):64–8.

Siegal, M. L. and Bergman, A. (2002). Waddington's canalization revisited: developmental stability and evolution. *Proc Natl Acad Sci U S A*, 99(16):10528–10532.

Spieth, C., Streichert, F., Speer, N., and Zell, A. (2004). Optimizing topology and parameters of gene regulatory network models from time-series data. In *GECCO 2004 - Genetic and Evolutionary Computation Conference*, volume 3102 of *Lecture Notes in Computer Science*, pages 461–470.

Stolovitzky, G., Monroe, D., and Califano, A. (2007). Dialogue on reverse-engineering assessment and methods: The dream of high-throughput pathway inference. *Ann N Y Acad Sci*, 1115:1–22.

Stolovitzky, G., Prill, R. J., and Califano, A. (2009). Lessons from the dream2 challenges. *Ann N Y Acad Sci*, 1158:159–195.

Surowiecki, J. (2004). *The Wisdom of Crowds*. Random House, New York, NY.

Tegner, J., Yeung, M. K. S., Hasty, J., and Collins, J. J. (2003). Reverse engineering gene networks: Integrating genetic perturbations with dynamical modeling. *Proc Natl Acad Sci U S A*, 100:5944–5949.

Thomke, S. (1998). Simulation, learning and r & d performance: Evidence from automotive development. *Research Policy*, 27(1):55–74.

Tu, Y., Stolovitzky, G., and Klein, U. (2002). Quantitative noise analysis for gene expression microarray experiments. *Proc Natl Acad Sci U S A*, 99(22):14031–14036.

van den Bulcke, T., Leemput, K. V., Naudts, B., van Remortel, P., Ma, H., Verschoren, A., Moor, B. D., and Marchal, K. (2006). SynTReN: a generator of synthetic gene expression data for design and analysis of structure learning algorithms. *BMC Bioinformatics*, 7:43.

van Someren, E. P., Wessels, L. F. A., Reinders, M. J. T., and Backer, E. (2003). Multi-criterion optimization for genetic network modeling. *Signal Processing*, 83:763–775.

Voit, E. O. and Savageau, M. A. (1987). Accuracy of alternative representations for integrated biochemical systems. *Biochemistry*, 26(21):6869–80.

von Dassow, G., Meir, E., Munro, E. M., and Odell, G. M. (2000). The segment polarity network is a robust developmental module. *Nature*, 406(6792):188–192.

Wahde, M. and Hertz, J. (2001). Modeling genetic regulatory dynamics in neural development. *J Comput Biol*, 8:429–42.

Wahde, M., Hertz, J. A., and Andersson, M. L. (2001). Reverse engineering of sparsely connected genetic regulatory networks. In *Proceedings of the 2nd Workshop of Biochemical Pathways and Genetic Networks*.

Watkinson, J., Liang, K.-C., Wang, X., Zheng, T., and Anastassiou, D. (2009). Inference of regulatory gene interactions from expression data using three-way mutual information. *Ann N Y Acad Sci*, 1158:302–313.

Watson, J., Geard, N., and Wiles, J. (2004). Towards more biological mutation operators in gene regulation studies. *Biosystems*, 76(1-3):239–248.

Weaver, D. C. (1999). Modeling regulatory networks with weight matrices. In *Pacific Symposium on Biocomputing*, volume 1999, pages 112–23.

Wernicke, S. (2005). *Algorithms in Bioinformatics*, chapter A Faster Algorithm for Detecting Network Motifs, pages 165–177. Springer Berlin / Heidelberg.

Wildenhain, J. and Crampin, E. J. (2006). Reconstructing gene regulatory networks: from random to scale-free connectivity. *Syst Biol (Stevenage)*, 153(4):247–256.

Yeung, M. K. S., Tegnér, J., and Collins, J. J. (2002). Reverse engineering gene networks using singular value decomposition and robust regression. *Proc Natl Acad Sci U S A*, 99(9):6163–8.

Yip, K. Y., Alexander, R. P., Yan, K.-K., and Gerstein, M. (2009). Improved reconstruction of in silico gene regulatory networks by integrating knockout and perturbation data. *PLoS One*, to appear.

Yokobayashi, Y., Weiss, R., and Arnold, F. H. (2002). Directed evolution of a genetic circuit. *Proc Natl Acad Sci U S A*, 99(26):16587–91.

Yuh, C. H., Bolouri, H., and Davidson, E. H. (1998). Genomic cis-regulatory logic: experimental and computational analysis of a sea urchin gene. *Science*, 279(5358):1896–1902.

Zak, D. E., Gonye, G. E., Schwaber, J. S., and Doyle, F. J. (2003). Importance of input perturbations and stochastic gene expression in the reverse engineering of genetic regulatory networks: insights from an identifiability analysis of an in silico network. *Genome Res*, 13(11):2396–2405.

Zaslaver, A., Bren, A., Ronen, M., Itzkovitz, S., Kikoin, I., Shavit, S., Liebermeister, W., Surette, M. G., and Alon, U. (2006). A comprehensive library of fluorescent transcriptional reporters for escherichia coli. *Nat Methods*, 3(8):623–628.

Curriculum vitæ

I grew up in Bern, Switzerland, where I was born on February 14, 1980. In 1998 I did a high school exchange year in Bloomington, Indiana. I graduated from the *Gymnasium Bern-Kirchenfeld* in the year 2000, with a specialization in mathematics and natural sciences.

In the same year, I started my studies in computer science at the Swiss Federal Institute of Technology in Lausanne (EPFL). I visited the *Faculdade de Engenharia da Universidade do Porto* in Portugal during an ERASMUS exchange year in 2004. I graduated from EPFL with an M.Sc. in computer science in 2005. For my master's thesis, I developed a simulation environment and online learning algorithm for modular robot locomotion under the supervision of Prof. Auke Ijspeert.

After graduation, I started my doctoral studies in computer science at EPFL, in the Laboratory of Intelligent Systems (LIS) directed by Prof. Dario Floreano. I specialized in computational biology and developed methods for inference and modeling of gene regulatory networks. Besides from this work, which culminates in the present thesis, I have also participated in the elaboration of a project proposal that was accepted by the SystemsX.ch funding program. Within this project, our reverse engineering approach will be applied to study regulatory networks in *Drosophila* wing development. I have also contributed to teaching through the development and supervision of undergraduate research projects, master's thesis projects, and laboratory exercises for Prof. Dario Floreano's courses.

In the long term, it is my vision to contribute to a better understanding of the structure, function, and evolution of regulatory networks in the cell. After completion of my Ph.D., I will continue my work in this direction as a Postdoctoral Research Fellow in the group of Prof. Manolis Kellis at the Computer Science

and Artificial Intelligence Laboratory (CSAIL) of the Massachusetts Institute of Technology (MIT).

During my Ph.D., besides from reverse engineering biological systems, I have also created a biological system with my wife, Melusina Marbach—our son, Leonardo, was born in May 2008.