UNIVERSITA' DI PADOVA FACOLTA' DI INGEGNERIA

Dipartimento di Ingegneria dell'Informazione

Scuola di Dottorato di Ricerca in Ingegneria dell'Informazione
Indirizzo: Scienza e Tecnologia dell'Informazione e della Comunicazione

CICLO XXII

ADVANCED ALGORITHMS FOR GENOMIC DATA ANALYSIS

Direttore della Scuola: Ch.mo Prof. Matteo Bertocco

Supervisore: Ch.ma Prof. Silvana Badaloni

Dottorando: Francesco Sambo

Acknowledgements

I would like to wholeheartedly thank all the people without which this thesis would not have been possible.

Professor Silvana Badaloni, for her precious guidance of my first steps into research, Professor Gianna Maria Toffolo for her extremely useful suggestions and Professor Barbara Di Camillo for having let my entrance to Bioinformatics as smooth as possible, and for the support. A special mention also to Professor Carlo Ferrari, who first lit the sparkle, and to Doctor Marco Falda, for the countless chatters on the staircases of DEI.

Professor Marco Dorigo, for his kind hospitality at IRIDIA, Professor Thomas Stützle for his valuable suggestions and his bright sense of perspective and Doctor Marco A. Montes De Oca for having shared with me an important part of the trail. Many thanks also to Carlo Pinciroli, Manuele Brambilla, Matteo Borrotti, Jérémie Dubois-Lacoste and all the IRIDIAns, for both having been a second family in Brussels and for having lent themselves as unwitting guinea pigs for my culinary experiments.

Professor Riccardo Bellazzi, Doctor Lucia Sacchi, Doctor Fulvia Ferrazzi and all the Bioinformatics and Data Mining Group of the Pavia University, for the dataset on Human Cell Cycle and the code of the Dynamic Bayesian Networks algorithm. Our roads kept crossing during these three years, let's hope it will be the same in the future.

Giulia Doná, Erica Manesso, Enea Poletti, Diego Fiorin and all the Biomedical Data and Signal Processing Lab, for having proven that, with a little effort, a workplace can be an extremely funny environment. A special mention goes to Tiziana Sanavia, for the countless hours spent together interpreting R code.

Finally, my enriched family: my parents, for the patience and precious advices, Federico, who keeps proving me how much Brother sounds similar to Friend, Silvia, for late night and early morning vents and Enrico, for everything else.

Contents

Abstract

English

The object of this thesis is to develop new algorithmic techniques for the inference of causal relations among the genes of an organism from *DNA microarray* experiments. Cause-effect relations between genes can be inferred from microarray data (*Reverse Engineering*) and summarized in a *Gene Regulatory Network*, a graph in which nodes represent genes and edges represent causal relations among genes: this thesis presents three novel Reverse Engineering algorithms, tailored to tackle differend kinds of DNA microarray experiments and for different levels of detail in the description of the biological systems, and two studies on the difficulty of inferring Gene Regulatory Networks.

The first original contribution of the thesis is the application of the Qualitative Reasoning approach to steady state measurements of systematic gene perturbation experiments, *i.e.* experiments in which the expression of each gene is altered in turn and one sample of the expression is taken each time the system reaches a steady state.

The second proposed algorithm, CNET, is based on a heuristic scoring function designed to identify causal relations from time course experiments, *i.e.* repeated observations of the same biological system at subsequent temporal instants. The algorithm is tailored to recognize causal relations even in the presence of noise and variable regulatory delays.

We then present two original in-depth studies, the first on the relations between the performance of two network inference algorithms and the topological and structural properties of oriented Gene Regulatory Networks and the second on the fitness landscape around the optimal parameters configuration, when a class of nonlinear differential equations systems, known as Dynamic Recurrent Neural Networks, are fit to time course data. Both studies provide original and useful knowledge on the difficulty of inferring Gene Regulatory Networks from DNA microarray data.

Finally, we present a novel discrete/continuous optimization algorithm for fitting systems of nonlinear differential equations to small scale time course experiments, composed of two interacting modules: an Iterated Local Search procedure to explore the discrete space of network structures and a continuous optimization procedure to identify optimal system parameters.

The performance of the three proposed algorithms is assessed both on simulated data and, in some cases, on real DNA microarray data: the methods proved to be competitive with the state of the art of Reverse Engineering algorithms.

Italiano

Obiettivo del presente lavoro di tesi è lo sviluppo di tecniche algoritmiche innovative per l'identificazione di relazioni causali fra i geni di un organismo a partire da esperimenti di *DNA microarray*. Le relazioni causa-effetto fra i geni possono essere apprese a partire dai dati di microarray (*Reverse Engineering*) e riassunte in una *Rete di Regolazione Genica*, un grafo i cui nodi rappresentano i geni e i cui archi rappresentano le relazioni causali fra i geni: questa tesi presenta tre algoritmi innovativi di Reverse Engineering, progettati per elaborare diversi tipi di esperimenti di microarray e con diversi livelli di dettaglio nella descrizione dei sistemi biologici, e due studi sulla difficoltá nell'inferire le Reti di Regolazione Genica.

Il primo contributo originale della tesi è l'applicazione del Ragionamento Qualitativo all'elaborazione di misurazioni in stato stazionario di esperimenti di perturbazione sistematica dei geni, vale a dire esperimenti nei quali l'espressione di ogni gene a turno viene alterata e un solo campione dell'espressione genica viene misurato ogni volta che il sistema raggiunge lo stato stazionario.

Il secondo algoritmo proposto, CNET, è basato su una funzione euristica progettata per identificare relazioni causali a partire da serie temporali di espressione genica, cioè osservazioni ripetute dello stesso sistema biologico in istanti temporali consecutivi. L'algoritmo è costruito in modo tale da riconoscere le relazioni causali anche in presenza di rumore e di ritardi variabili nella regolazione.

Successivamente vengono presentati due studi approfonditi, il primo sulle relazioni fra la performance di due algoritmi di Reverse Engineering e le proprietá strutturali e topologiche della Rete di Regolazione Genica da inferire e il secondo sul panorama di fitness attorno alla configurazione ottima dei parametri di una particolare classe di sistemi dinamici non lineari, le Reti Neurali Dinamiche Ricorsive, che descriva un insieme di serie temporali di espressione genica. Entrambi gli studi hanno consentito di ottenere informazioni utili e originali sulla difficoltá nell'inferire Reti di Regolazione Genica a partire da dati di DNA microarray.

Infine, viene presentato un algoritmo innovativo di ottimizzazione mista (continua e discreta) per il fit di sistemi di equazioni differenziali non lineari a esperimenti contenenti serie temporali di espressione genica su piccola scala, composto di due moduli interagenti: una procedura di ricerca locale per esplorare lo spazio discreto delle strutture di rete e una procedura di ottimizzazione continua per l'idenficazione dei parametri ottimi del sistema.

La performance dei tre algoritmi proposti viene analizzata sia su dati simulati sia, in certi casi, su dati reali di DNA microarray: i metodi si dimostrano competitivi con lo stato dell'arte degli algoritmi di Reverse Engineering.

Introduction

The object of this thesis is to develop new algorithmic techniques for the inference of causal relations among genes from *DNA microarray* experiments, which are genome-wide observations of the expression of each gene under different biochemical conditions or at different temporal instants.

The problem is of central importance in contemporary life sciences: at present, reliable methods of *DNA sequencing* [27] allow researchers to recover the whole DNA sequence of an organism, but the task of gathering full information on the function of each gene in the DNA is still far from being accomplished. Such information would be extremely valuable for understanding the real machinery of life and would have a direct impact on medicine, allowing researchers to identify the genetic causes of diseases and the possible targets for focused medical intervention. Understanding cause-effect relations between genes is one of the key steps of this exploratory process.

A gene is *expressed* at a particular time instant if its sequence is being transcribed into messenger RNA (mRNA); mRNA is then translated into a protein and some proteins, possibly in combination with each other, have the role of activating or inhibiting the expression of other genes. This self-control mechanism of the DNA is known as *gene regulation* and induces a set of causal relations among genes.

Cause-effect relations between expressed genes can thus be inferred from DNA microarray measurements and summarized in a Gene Regulatory Network, a graph in which nodes represent genes and edges represent causal relations among genes. A detailed description of the biological problem, of the DNA microarray technique and of the topological properties of Gene Regulatory Networks is given in Chapter 1.

The problem has been widely studied in the literature (for a survey of the state of the art, please refer to Chapter 2) because some of its features make it hard to solve:

- **Dimensionality**: genomes of the various organisms contain a number of genes in the order of the thousands. DNA microarray experiments, on the other hand, do not usually consist of more than a hundred observations, because of their high cost. Thus, the whole set of causal relations is hard to be captured with a single microarray experiment.

- **Observation point**: DNA microarray experiments allow the biologist to monitor the contemporary expression of each gene, but the main biological interactions take place between proteins. The phenomenon has thus to be studied from the indirect observation of its effects.

- **Heterogenity of interactions**: interactions between proteins that have as final effect the activation of a gene are extremely heterogeneous and span different time scales. At the gene level, a consequence of this behaviour is a high variability between gene regulatory delays, *i.e.* the time lags that elapse between the expression of the regulating genes and the expression of their targets.
- **Noise**: DNA experiments are affected by a high level of noise, both inherent in the measurement technique and as an effect of unobservable environmental variables. Experiments are often replicated to cope with noise, but the number of replicas is usually limited.

For all these reasons, traditional signal processing techniques are not sufficient to infer the whole Gene Regulatory Network of an organism from a DNA microarray experiment. The problem has to be tackled with a combination of different advanced algorithmic techniques, selected from the fields of Artificial Intelligence, Data Mining, Knowledge Discovery and Optimization, and at different scales, from a genome-wide correlation analysis to a precise description of the interactions between small sets of genes. Moreover, results from multiple and heterogeneous experiments can be combined and enriched with additional knowledge from the literature. From an algorithmic point of view, the aim is thus to choose the correct technique to tackle each particular subproblem, depending on its scale and on the nature of the data to be processed.

This thesis will focus on two of these subproblems: the inference of the structure of an oriented causal network (Chapter 4 and 5) and the inference of a system of equations describing the expression of genes (Chapter 6).

In the former subproblem the objective is to infer directed causal relations between genes. In Chapter 4 two novel algorithmic techniques are described, designed to analyze two different types of expression data. The first technique is a Qualitative Reasoning algorithm (Section 4.1), developed to process steady state measurements of systematic gene perturbation experiments, *i.e.* experiments in which the expression of each gene is altered in turn and one sample of the expression is taken each time the system reaches a steady state. The second algorithm (Section 4.2) is based on a heuristic scoring function designed to identify causal relations from time course experiments, *i.e.* repeated observations of the same biological system at subsequent temporal instants. The algorithm is tailored to recognize causal relations even in the presence of noise and variable regulatory delays. It exploits the fact, known in the literature, that the network is sparse and uses it to control the number of regulators for each gene. Moreover, an in-depth study is carried out in Chapter 5 on the relations between the performance of two network inference algorithms and the topological and structural properties of oriented Gene Regulatory Networks.

For the inference of a system of equations, the second subproblem we tackle, the objective is to describe the expression of each gene with an analytical equation that relates the gene expression profile to the expression of its regulators. The scale of this subproblem is smaller than in the previous case, because of the increased level of detail in the description of gene relations that provides the ability of predicting gene expression profiles. The system of equations has to be fit to time course data and the goodness of the fit can be assessed measuring the error between real temporal profiles and profiles predicted by the model. Optimization techniques can thus be exploited to search optimal values for system parameters, minimizing an error measure. In Chapter 6, the fitness landscape around the optimal parameters configuration is first studied, for the problem of minimizing Relative Squared Error (RSE) between real and estimated gene profiles, when a class of nonlinear differential equations systems, known as Dynamic Recurrent Neural Networks, are fit to time course data (Section 6.1). For the analysis, solutions close to the optimum are sampled in three different ways and the correlation between the RSE of the solutions and their Euclidean distance from the optimum is studied. Two state-of-the-art continuous optimization algorithms, CMA-ES [28] and NEWUOA [56], are then tested on the problem and compared. The results of this analysis are exploited in the design of a mixed optimization algorithm, that searches in the discrete space of network structures with an Iterated Local Search procedure and optimizes continuous system parameter with CMA-ES (Section 6.2).

To properly assess the performance of an algorithm and to compare it with the state-of-the-art, one should rely on a set of benchmark problems and exploit them as gold standards to test average algorithm behaviour. Unfortunately, reliable knowledge on real Gene Regulatory Networks is still missing. For this reason, we concentrate mostly on simulation for the analysis of the performance of our algorithms. Chapter 3 describes the simulator we used to generate experimental data and explains in details the generated datasets. The simulator resembles some of the main features of transcriptional regulatory networks, related to topology, interaction among regulators of transcription and expression dynamics. The chapter describes also another set of simulated data, the DREAM4 In Silico Network Challenge [46, 73], that was chosen for our analysis because it is widely known and is considered a reliable benchmark for Reverse Engineering algorithms by the scientific community. Finally, to validate our approaches on real data, a dataset of DNA microarray experiments on nine human genes is presented and the (possibly incomplete) set of known biological interactions is reported.

From performance results on simulated and real microarray data the three novel algorithms presented in this thesis proved to be competitive with the state of the art in solving the different problems they address, which are all parts of the main task of Gene Regulatory Network inference from DNA microarray data. Moreover, the two analyses we carry out, on the relations between the performance of the algorithms and the topological properties of the networks and on the correlation between Relative Squared Error and distance from the optimal network, provide new valuable insights on the problem that may be exploited in the design of new algorithms.

Chapter 1

The biological problem: Inference of Gene Regulatory Networks from DNA microarray experiments.

1.1 Genetic Regulation

One of the most important discoveries of the last century in Biology is that all the information necessary for an organism to live is coded in the genes of its DNA. On the other hand, the certainty emerged that almost every biological function in all the living things is carried out by proteins. The specific relationship between genes and proteins is so important in modern biology that it is called the *central dogma* and can be enounced as: "DNA molecules contain information about how to create proteins; this information is *transcribed* into RNA molecules, which, in turn, direct chemical machinery which *translates* the nucleic acid message into a protein"[31].

DNA, RNA and proteins share the property of being a chain of chemicals known as bases or nucleotides, for DNA and RNA, and aminoacids, for proteins. In the case of DNA, nucleotides can be of four types: adenine (A), cytosine (C), guanine (G) and thymine (T); RNA has the same set of four bases, except that instead of thymine, RNA has uracil (U). On the other hand, there are 20 different naturally occurring amino acids that are assembled into proteins. Each amino acid is coded by a nucleotide sequence: since there are 20 different amino acids but only 4 nucleotides, the information required to specify an amino acid has to be contained in at least 3 nucleotides. Nucleotide triplets are called *codons*, and each amino acid is specified by one or more codon (being there $4^3 = 64$ possible codons).

Some proteins, called *transcription factors*, have the role, possibly in combination with each other, to activate or inhibit the transcription of genes and

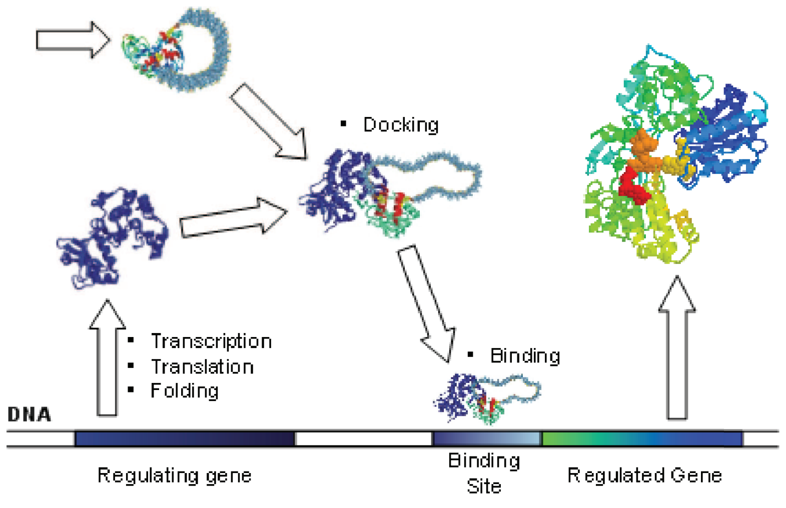


Figure 1.1: Schematic representation of gene regulation: the regulating gene (on the left) is transcribed into RNA and translated into a protein. The protein folds in a 3D structure and, potentially docking to another protein, acts as a *transcription factor* for the regulated gene (on the right), binding on the region of DNA that precedes the gene and activating (or inhibiting) its transcription process.

to control the translation of RNA into new proteins; the process by which some genes, through the proteins they code, control the *expression level* (*i.e.* the RNA transcription rate) of other genes is known as *genetic regulation*. A schematic representation of gene regulation is presented in Figure 1.1.

Object of the present research is the inference of such regulatory relations and their subsequent representation on a *Gene Regulatory Network*, a graph in which nodes correspond to genes and edges to regulatory relations between genes. Inference, or *Reverse Engineering*, of Gene Regulatory Networks is mostly based on the analysis of a particular kind of biological measurements called *DNA microarray* experiments.

1.2 DNA microarray data

In the past few years, the study of genetic regulation was drastically improved by the discovery of the new technology of DNA microarray [49], which allows researchers to monitor the expression of the whole genome under various genetic, chemical and environmental perturbations.

The DNA microarray technology is based on the fundamental property of DNA and RNA called *complementarity*: each nucleotide only binds well with its complement, A with T (or U) and G with C. Complementarity can then be exploited to detect specific sequences of bases within strands of DNA or RNA. The detection is achieved synthesizing a *probe*, i.e. a piece of DNA that is the reverse complement of a sequence one wants to detect, and having it interact with the set of genetic material that is being searched, called the *sample*. Thanks to complementarity, the probe will bind to its complement if the latter is present in the sample. The act of binding between probe and sample is called *hybridization*.

A DNA microarray (also called *gene chip*) contains a large number of different synthesized probes, carefully attached to specific positions on a flat surface. Each probe is the reverse complement of a sequence of amino acids, distinctive of a particular gene, and a chip can contain tens of thousands of different probes. The sample can then be labeled using a fluorescent tag and spread across the chip. Because the probes are attached at specific locations on the chip, if the labeled sample is detected at any position on the chip, one can determine which probe has hybridized to its complement.

The most common use of gene chips is to measure the expression level of various genes in an organism: an expression level measures the rate at which a particular gene is being transcribed, and this is used as a proxy measure for the amount of corresponding protein that is being produced within an organism's cells at a given time.

In a typical microarray study, many experiments measure the same set of genes under various circumstances or at various time points. Experiments are usually replicated to cope with noise, which is largely present in raw microarray data. However, each gene-chip experiment can cost several hundred dollars, thus in practice experiments are replicated only a small number of times, and time series experiments usually contain $O\left(10 \sim 10^2\right)$ time points.

Throughout the thesis, we will make the distinction between experiments with *external stimulation* and in *natural response*. In the former case, the biological system receives an external stimulus either right before or through the whole course of the experiment: for example, cells can be heated or irradiated with UVA, some drug can be added or the expression of one or more genes can be altered, either by making them inoperative (*gene knock-out*), by degradating their RNA product (*RNA silencing*) or by other means of external up/downregulation. This kind of experiments are usally paired with the synchronous observation of the cell in *wild type* conditions, *i.e.* without the external stimulation. In the case of natural response, on the other hand, cell evolution is observed as it is, and comparisons are usually made between cells from different tissues or in different health conditions (tumoral and normal cells, for example).

Another distinction we will make is between *time course* and *steady state* measurements: in the former case, the same cells are observed at subsequent time instants, gathering a *time series* of the expression profile. In the latter case, multiple sets of cells are observed just once, when the biological system has reached a steady state: each set can come from a different patient, or have been treated differently.

1.3 Properties of Gene Regulatory Networks

Structural and topological properties of Gene Regulatory Networks and metabolic networks have been widely studied in the past few years and a set of distinctive properties is now known:

- **Scale-free**: the vast majority of genes are not regulators, and the distribution of the out degree of the transcription factors (i.e. the distribution of the numer of nodes that each transcription factor regulates) follows a power-law of the form

$$P(k) = \alpha k^{-\gamma},$$

where k is the out degree of transcription factors, γ is in the range $2 < \gamma < 3$ and α is a normalization factor [1]. Networks that share this property are known in the literature as scale-free networks [7].

- **Sparsity**: related to the previous property, but worth mentioning, is the sparsity of regulatory networks [3]. For a network of n genes, the total number of regulatory relations is $O(n)$, much less than the total possible number of direct connections, n^2.

- **High clustering coefficient**: the clustering coefficient of a node i is defined as the ratio between the number of edges linking nodes adjacent to i and the total possible number of edges among them [77]. On random sparse network, the average clustering coefficient tends to have a low value, wich decreases with the number of nodes. On metabolic networks, on the contrary, the clustering coefficient has a higher value and it is independent of the number of nodes.

- **Small-World**: metabolic pathways between each pair of gene tend to be shorter than what expected in a random network, thus regulatory networks belong to the class of small-world networks [77].

- **Network Motifs**: some regulatory motifs, i.e. connection patterns between three or more genes, appear in regulatory networks with a significantly high frequency [2]. Each motif is supposed to have a particular biological function and is thus conserved through different kinds of organisms.

All these network properties can be exploited in the process of Reverse Engineering, either as *a priori* information for the algorithms or as a way to score multiple putative networks inferred for the same data.

Chapter 2

State of the art

The process of Reverse Engineering Gene Regulatory Networks can be decomposed in two subsequent tasks: choice of the model for describing regulatory interactions and fit of the model to data.

When choosing a model, one has always to balance between two opposite model features:

- *Accuracy* in describing real data, directly proportional to the model complexity and thus to the level of detail with which data are modeled.

- *Scalability*, in terms of the number of numerical or categorical parameters to be estimated, with respect to the number of genes present in the network to be inferred.

A large amount of models and algorithms are available in the literature. We will present in what follows a selection of the most relevant ones, pointing out the trade-off between scalability and accuracy and discussing strengths and limitations. For a summary of all the main features of each model, please refer to Table 2.1. Subsequently, we will review the most important algorithms from the literature, used for fitting each model to data. A section on performance measures will illustrate how to assess and compare performance of the various algorithms. Finally, we will outline the difficulties and limitations that each Reverse Engineering approach has to face, when dealing with real data, and present results on average algorithmic performance from five recent assessment papers.

2.1 Models

In this section a review of the most frequently adopted models in the literature is presented. In ascending order of model complexity and descending order of model scalability, we will present Oriented and Unoriented Graphs (Section 2.1.1), Boolean Networks (Section 2.1.2) Bayesian Networks and Dynamic Bayesian Networks (Section 2.1.3) and systems of equations, both linear and nonlinear, differential and additive and S-Systems (Section 2.1.4).

Oriented and Unoriented graphs • $O\left(n^2\right)$ Boolean variables • Do not allow to make predictions on the evolution of the signal • No measure for the goodness of fit	**Boolean Networks** • $O\left(n^{2^k+2}\right)$ Boolean variables, with k maximum regulators per gene • Can be used to make predictions on Boolean data • No measure for the goodness of fit
Bayesian Networks • The number of parameters depends on the type of variables (discrete-continous) and on the probability distributions relating parents and children • Can be used to make prediction on noisy data • The structure can be evaluated with posterior probability	**Systems of equations** • $O\left(n^2\right)$ continuous variables • Can be used to make predictions on noise-free data • Can be evaluated with Minum Squared Error between real and predicted gene profiles

Table 2.1: The main features of Gene Regulatory Network models.

2.1.1 Oriented and Unoriented Graphs

Probably the most straightforward way to model a Gene Regulatory Network is to view it as a graph $G = (V, E)$, which is completely determined by its sets of vertices V and edges E. Within this formalism, genes are mapped to vertices of the graph and regulatory relations are mapped to edges. An edge e is a pair (i, j), in which i and j denote vertices, *i.e.* elements of the set V. If edges are directed, *i.e.* $(i, j) \neq (j, i)$, the graph is said to be *oriented*, if the direction is not taken into account the graph is *unoriented*.

In an oriented graph the direction of the regulation can also be expressed, with the source vertex regulating the destination vertex. Moreover, directed edges can be labeled as "activating" or "inhibiting", to distinguish between the two regulatory mechanisms.

The presence or absence of a particular edge can be inferred with a set of different measures, such as Pearson Correlation and Partial Pearson Correlation [16], Mutual Information [9, 15, 47] and Conditional Mutual Information [84]. All the measures have the common goal of identifying dependence or independence between gene profiles, seen as discrete or continuous random variables, and are explained in detail in what follows.

Pearson Correlation

Pearson Correlation (PC) is probably the simplest, yet powerful measure of association between random variables. If a random variable x is associated to each gene, the (0th order) PC between the random variables x and y is

$$r_{xy} = \frac{\mathrm{Cov}\,(xy)}{\sqrt{\mathrm{Var}\,(x)\mathrm{Var}\,(y)}},$$

where Var() is the variance of the random variable and Cov() is the covariance between the two random variables.

Since correlation alone is a weak concept and cannot distinguish between direct and indirect interactions, (e.g. mediated by a common regulator gene), an algorithm for network inference can be improved by the use of partial correlations (PPC) [16]. The minimum first order partial correlation between x and y is obtained by exhaustively conditioning the pair x, y over all z and taking the minimum. If exists $z \neq x, y$ which explains all of the correlation between x and y, then the partial correlation between x and y becomes 0 and the pair x, y is conditionally independent given z. In the same way, the minimum second order partial correlation coefficient between x and y is obtained by exhaustively conditioning the pair x, y over all possible pairs z, q.

The equations for computing first and second order partial correlation coefficients are:

$$\begin{aligned} r_{xy.z} &= \frac{r_{xy} - r_{xz}r_{yz}}{\sqrt{\left(1 - r_{xz}^2\right)\left(1 - r_{yz}^2\right)}} \\ r_{xy.zq} &= \frac{r_{xy.z} - r_{xq.z}r_{yq.z}}{\sqrt{\left(1 - r_{xq.z}^2\right)\left(1 - r_{yq.z}^2\right)}} \end{aligned}$$

Since the computation is exhaustive over all n genes, the computational cost of searching for the k-th order minimum PPC is of the order of $O\left(n^{k+2}\right)$, and it becomes quickly prohibitive for $k \geq 3$, if n is of the order of the thousands.

The exhaustive conditioning over $n-2$ genes can be expressed explicitly only if the $n \times n$ matrix R of elements r_{xy} is invertible and if one can assume that the data are drawn from a multivariate normal distribution. Denoting $\Omega = R^{-1}$ the concentration matrix of elements $\Omega = (\omega_{xy})$, the partial correlation between x and y is

$$r_{xy.all} = -\frac{\omega_{xy}}{\sqrt{\omega_{xx}\omega_{yy}}}.$$

When R is not full rank, as is often the case with gene expression data, one can rely on an estimation of the inverse Ω, as described in [68].

PC and PPC are symmetric measures and thus can be used to infer parts of the unoriented regulatory network, as is done in [16] on a dataset composed by 781 genes of buddying yest (*Saccharomyces Cerevisiae*) and in [68] on 3883 genes from a study on human breast cancer tumor.

Mutual Information

In an association network, alternatively to PC and PPC, one can use the information-theoretic concept of Mutual Information (MI), together with the notion of conditional independence to discern direct from indirect interdependencies. Given a discrete random variable x, taking values in the set X, its Shannon Entropy is defined as

$$H(x) = -\sum_{\bar{x} \in X} p(\bar{x}) \log p(\bar{x}),$$

where $p(\bar{x})$ is the probability mass function $p(\bar{x}) = Pr(x = \bar{x})$, $\bar{x} \in X$. The joint entropy of a pair of variables x, y, taking values in the sets X, Y respectively, is

$$H(x,y) = -\sum_{\bar{x}\in X, \bar{y}\in Y} p(\bar{x},\bar{y}) \log p(\bar{x},\bar{y})$$

while the conditional entropy of x given y is defined as $H(x|y) = H(x,y) - H(x)$. The Mutual Information of x, y is defined as $MI(x,y) = H(x) - H(x|y)$ and can be explicitly expressed as

$$MI(x,y) = -\sum_{\bar{x}\in X, \bar{y}\in Y} p(\bar{x},\bar{y}) \log \frac{p(\bar{x},\bar{y})}{p(\bar{x})p(\bar{y})} \geq 0. \tag{2.1}$$

When the two variables are independent, the joint probability distribution factorizes and the MI vanishes:

$$p(\bar{x},\bar{y}) = p(\bar{x})p(\bar{y}) \;\Rightarrow\; MI(x,y) = 0.$$

The MI conditioned with respect to a third variable z is:

$$MI(x,y|z) = H(x,z) + H(y,z) - H(z) - H(x,y,z).$$

All pairs of nodes can be conditioned exhaustively on each of the remaining $n-2$ nodes, and the minimum of such conditional MIs

$$mMI(x,y) = \min_{z\neq x,y} MI(x,y|z) \tag{2.2}$$

can be taken as a measure of conditional independence.

Just like in the PC and PPC case, the two measures (2.1) and (2.2) can be used to construct the graph of the unoriented gene network: in [9] Mutual Information is exploited for the analysis of 2,467 genes of S. Cerevisiae, in [15] a global analysis is carried out on two datasets of 5345 genes of S. Cerevisiae and 22608 human genes and in [47] RNA products from human B lymphocytes are processed on a global scale. Finally, in [84] an unoriented network is inferred from a dataset of 527 genes from 31 human patients affected by melanoma.

In [83], empyrical evidence is shown that direct measures, such as PC and MI, are more robust in detecting coexpression situations, when two or more genes coparticipate in the same protein complex, whereas conditional measures, such as PPC and mMI, are more suited for causal regulatory interactions.

Strengths and weaknesses of the model

Oriented and unoriented graphs are among the simplest and less detailed models of gene regulation, thus exhibiting the highest level of scalability: oriented networks are completely described by n^2 boolean variables (presence or absence of an edge) and unoriented networks by $n(n-1)/2$ variables. The simplicity comes at the cost of being unable to predict the evolution of gene expression profiles across time. Moreover, the application of different measures of independence often produces different networks as output, and it is still unclear which of the measures is the more reliable. Yet, at present these are still the only tractable models applicable when inference is carried out on the global scale of a whole organism, *i.e.* when $O\left(10^3\right)$ gene profiles need to be processed.

2.1.2 Boolean networks

The Boolean Network formalism was first introduced in the pioneering work of Kauffman *et al.* [34]. Within this paradigm, genes are treated as binary random variables, which can be active (on, 1) or inactive (off, 0) and hence their products are present or absent. Moreover, interactions between genes can be represented by Boolean functions, which calculate the state of a gene from the activation of other genes. Gene interactions are supposed to be synchronous, and thus the state of each gene at the discrete temporal instant t is completely determined by the state of the system at time $t-1$.

With a more strict mathematical formalism, let the vector $\mathbf{x}$ of variables denote the state of a regulatory system of n elements. Each x_i is a binary variable, so the state space of the system consists of 2^n states. The state x_i of an element at the time instant $t+1$ is computed by means of a boolean function

$$x_i(t+1) = b_i\left[x_1(t), \ldots, x_k(t)\right], \qquad 1 \leq i \leq n$$

from the state of k of the other n elements at the previous time instant t (k can be smaller then n). For k inputs, with possibly a different k for each gene, the total number of possible Boolean functions mapping the inputs to the output is 2^{2^k}.

Strategies for the inference of Boolean Networks from time series of gene expression profiles can be found in [41, 51].

Strengths and weaknesses of the model

The Boolean Network model relies on the two strong assumptions of Boolean behaviour of genes and of synchronous regulatory interactions. While the first assumption is true at the gene level (either a gene is translated or it is not translated, in a particular time instant), DNA microarray measurements permit only the observation of the average behaviour of a population of cells, and interactions at the protein level depend more on protein concentration than on presence or absence of the protein. Microarray data is thus continuous and interactions are triggered when protein concentrations overcome some gene-specific thresholds. Identifying quantization thresholds for expression profiles is thus a hard task and can introduce additional noise in the data [19].

The second assumption is even more strict, because genes are known to exhibit variable regulatory delays [85]. Nevertheless, such an approach may be able to capture subsets of regulatory relations with time delay in the same range, ignoring relations at different time scales.

Scalability of this model strictly depends on the maximum allowable number k of regulators for each gene, because the number of possible boolean functions for each gene grows superexponentially with k. Under the two aforementioned assumptions, Boolean Networks can be used to make predictions on gene temporal profiles, and have been exploited by theoretical biologists to formulate hypothesis on the general behaviour of biological systems [34].

2.1.3 Bayesian networks

A Bayesian Network (BN) [53, 69] is fully determined by two components: a directed acyclic graph (directed tree) and a probability distribution. Nodes in the graph represent stochastic variables and edges represent direct dependencies among variables, quantified by conditional probability distributions.

Following the direction of edges, source nodes are called *parents* and destination nodes *children*. The probability distribution of each child is completely determined given its parents, and can thus be expressed in tabular form (for discrete variables) or with continuous conditional probability distributions (for continuous variables).

The *local Markov property* on Bayesian Networks states that each node is independent of its non descendants given its parents. This leads to a direct factorization of the joint probability distribution of the network variables into the product of the conditional distribution of each variable X_i given its parents $Pa(x_i)$ (in the following, capital letters denote random variables, and small letters denote their values). Therefore, the joint probability of the n network variables can be written as:

$$p(x_1, \ldots, x_n) = \prod_{i=1}^{n} p(x_i | pa(x_i)),$$

where $p(y|x)$ denotes the probability density of the variable Y, given $X = x$, and $pa(x_i)$ denotes a set of values of $Pa(x_i)$. The overall probability distribution is then broken into modules that can be interrelated, and the network summarizes concisely all the significant dependencies.

Bayesian Network have been succesfully exploited to model biological systems (see [24, 70] for two examples of application and [69] for guidelines on using BNs in genomic data analysis). Nodes in the network can represent both gene products and phenotipic information, such as occurrence of a disease or presence of a somatic trait.

When used to model genetic regulation, however, Bayesian Networks have the strong limitation of not allowing cycles in the directed graph; cycles, however, are frequent in regulatory networks. For this reason, a particular class of BN is usually exploited for microarray data analysis, Dynamic Bayesian Networks (DBNs) [50].

In DBNs each node models a variable in a particular time instant, and edges represent conditional dependence between the values of parents at time step t and the values of children at time $t + 1$. This formalism allow cycles in the network and even dependencies of a variable on its value in the previous time step. It is usually assumed that for DBNs holds the first-order Markov property, *i.e.* that connections are allowed only between subsequent time slices, and thus the state of the whole set of variables at time t is completely determined by the state at time $t - 1$.

Bayesian Networks can model both discrete and continous variables: in the former case, the dependency of each variable on its parents is represented by a set of multinomial distributions that describe the conditional distribution of the variable on each configurations of the parent variables. In the latter case, children probabilities are described by a full probability density function, given each parent.

Network structures are chosen in terms of maximum posterior probability given the data. In the special cases of networks with discrete variables and of networks with continuous Gaussian variables and linear dependencies between parents and children, the posterior probability can be computed in closed form, averaging out probability distributions and serving as a score fore just the structure.

Strengths and weaknesses of the model

Bayesian Networks, for their stochastic nature, are particularly suited to model noisy data such as gene expression profiles. Inferring a BN, however, requires both the identification of the oriented graph and the estimation of the parameters of the probability distribution for each node, with respect to its parents; as for Boolean Networks, then, scalability strictly depends on the maximum allowable number of parents for each node.

Network structures can be evaluated in terms of posterior probability given the data, and thus a coherent search in the space of networks can be carried out to identify the best structure.

2.1.4 Systems of equations

Systems of equations are the most accurate way to model a biological system: the rate of concentration of each gene, protein or reagent can be associated to a variable, and relations between the variables can be described analytically through a set of equations.

When analyzing microarray data, one usually associates variables with genes, and in some cases models other reagents as external inputs to the system of equations. Depending on the cases, equations relate the expression of regulators to either the expression of the regulated gene (*additive systems*) or the derivative of its expression (*differential systems*). The function applied to the expression of the regulators can also contain a nonlinear element, like a sigmoid function (*nonlinear sigmoidal systems*), or be intrinsecally nonlinear in its definition (*S-Systems* [67]). Each class of models is explained in details in what follows.

Additive systems

Historically among the first equation models adopted for Gene Regulatory Networks [17], additive systems have recently been used to model the SOS pathway of the organism *Escherichia Coli* [25]. With the addition of a nonlinear element, they have also been exploited to infer a network of five genes involved in the cell cycle of S. Cerevisiae [36, 58].

In its general form, an additive system of n genes is described by the set of equations

$$x_i(t) = f\left[\sum_{j=1}^{n} w_{ji} x_j(t) + u_i(t)\right] \qquad 1 \leq i \leq n,$$

where x_i denotes the rate of expression of gene i, f can be the identity function or a sigmoid function (for nonlinear additive systems), w_{ji} is the relative weight of the influence of gene j on gene i and u_i is the effect of the external input on gene i; this last term is time varying in the most general case, but can be a gene-specific constant, a global constant or even absent.

This kind of models lack, in general, the capability of fully describing gene regulatory relations, which are highly nonlinear and differential in nature, but can be exploited positively under the hypothesis that the observed system operates near a steady state [25]. The simplicity of the model makes it computationally easier to handle and more robust to noise, having only to deal with the absolute value of the expression signal and not with its derivatives. The number of parameters to estimate when inferring an additive system of n genes is n^2 (the weights w_{ji}), plus additional $O(n)$ parameters when the external input is modelled.

Differential systems

Systems of differential equations are more widely adopted, because of the aforementioned differential nature of regulatory relations.

A system of linear differential equations representing the behaviour of n genes has the form

$$\frac{dx_i}{dt} = \sum_{j=1}^{n} w_{ji} x_j + u_i \qquad 1 \leq i \leq n,$$

where x_i is the rate of expression of gene i, w_{ji} is the relative weight of the influence of gene j on the derivative of gene i and u_i is the effect of a possible external input. Linear differential equations are studied in [55] on simulated data and exploited in [52] on both a dataset of E. Coli SOS pathway (9 genes) and a dataset of 24 S. Cerevisiae cell cycle related genes.

A systems of nonlinear sigmoidal differential equations of n genes, on the other hand, has the form

$$\frac{dx_i}{dt} = \sigma \left[\sum_{j=1}^{n} w_{ji} x_j + u_i \right] - \lambda_i x_i \qquad 1 \leq i \leq n,$$

and differs from the linear model for the presence of the sigmoid function $\sigma()$ and of the coefficients λ_i, usually adopted to explicitly represent the degradation rate of each gene product. Additional parameters can control the shape of the sigmoid function and vary among genes, thus modeling gene-specific regulatory responses, at the cost of increasing the complexity of the model.

Nonlinear sigmoidal differential equations, formally identical to what is known as Dynamic Recurrent Neural Networks [39, 54] in the Machine Learning community, are exploited in [76] to match the best trascription factor to each of a set of 40 cell cycle-related genes of S. Cerevisiae, in [80, 81] to infer the E. Coli SOS pathway and in [45] on the DREAM1 *in vivo* Five-Gene-Net challenge [73].

Fitting both the linear and the nonlinear model to n gene profiles requires the estimation of n^2 continuous variables, plus additional $O(n)$ parameters for external inputs and/or degradation coefficients.

Finally, S-Systems of n genes have the form

$$\frac{dx_i}{dt} = \alpha_i \prod_{j=1}^{n} x_j^{g_{ij}}(t) - \beta_i \prod_{j=1}^{n} x_j^{h_{ij}}(t) \qquad 1 \leq i \leq n, \tag{2.3}$$

where x_i is the rate of expression of gene i, α_i and β_i are non-negative rate constants and g_{ij} and h_{ij} are kinetic orders. The equations explicitly splits the set of regulatory effects into two opposite components, the excitatory effects and the inhibitory effects. A fit of an S-System of n genes requires the estimation of $2n(n+1)$ continuous parameters.

S-Systems are used in [37] to infer a network of 24 clusters of gene profiles from a set of *Thermus thermophilus* HB8 strains and in [38] for the E. Coli SOS repair pathway; moreover, they are exploited in simulations to compare a set of Evolutionary Algorithms in [72] and to generate a set of benchmark problems for Reverse Enineering algorithms in [26].

Strengths and weaknesses of the model

Systems of equations accurately describe the behaviour of a biological system across time and can thus be used to make predictions on the evolution of the system. Moreover, when fitting systems of equations to data, measuring the accuracy of a choice of system parameteres is straightforward: it is sufficient to measure the error between real and estimated expression profiles.

This comes at the cost of a large parameter set, of size $O\left(n^2\right)$ if n is the number of genes. For this reason, systems of equations are usually adopted when modeling small sets of genes (5 to 30). Nevertheless, the size of tractable networks can be increased with the use of *a priori* information on network properties or on known links, as it is often the case in microarray studies.

When the system is differential, having to deal with derivatives poses an additional problem: if one wants to decouple the system and to fit each equation separately, derivatives of the signal have to be estimated from data. If the time sampling is sufficiently fine grained and the signal does not contain noise, derivatives can in principle be approximated with finite differences; with gene expression data, though, this is rarely the case. Numerically solving the whole system of equations, on the contrary, reduces the effect of noise, but is computationally more expensive and the system cannot be decoupled.

2.2 Algorithms

After having reviewed the main models from the literature for Gene Regulatory Networks, we will present in the following the algorithmic techniques proposed to fit these models to gene expression data. The algorithms mainly fall into two categories, depending on the presence or absence of a measure for the goodness of the fit: for oriented graphs, unoriented graphs and, in general, Boolean Networks, there is not such a measure, and thus one can only rely on pairwise measures of similarity or correlation between expression profiles and compute them extensively for each pair or triplet.

If, on the contrary, one can clearly define a measure for the goodness of fit, such as Relative Squared Error for systems of equation or maximum posterior likelihood for Bayesian Network, the process of fitting can be mapped to a search for the optimal configuration in the discrete space of network structures and in the continuous space of system parameters. In this second case, the search can be extensive, greedy, stochastic or exploit some statistical techniques.

In both cases, one can then exploit additional information on the properties of regulatory networks, such as sparsity or the scale-free distribution of node degree, and *a priori* information on some genes, like genes known to code for transcription factors, either to reduce the size of the search space or to postprocess the inferred network. Ensemble Learning strategies can also be exploited to merge the results of different algorithms on the same dataset or on multiple experiments.

The two main algorithmic approaches are described in what follows, reviewing the best solutions to the problem from the literature.

2.2.1 Pairwise Measures

Algorithms based on pairwise measures compute some indicators of causal relation between all the possible pairs of genes, creating a fully connected graph of gene relations, and then prune less probable links from the graph, with the objective of extracting the underlying unoriented graph of regulatory relations. The two similarity measures usually adopted are Pearson Correlation and Mutual Information (see Sections 2.1.1 and 2.1.1) and graphs ar pruned by conditioning the two measures over all the other genes, or by removing links whose measured score falls below some kind of threshold. While PC can be directly computed from continuous profiles, MI is originally designed for discrete random variables, and thus different algorithms propose different solutions to apply it to gene expression data.

In [9], gene profiles are quantized in 10 equally spaced bins, and then MI is computed between each pair of quantized random variables. The complete graph is then pruned, removing all the edges whose MI falls below a threshold. The threshold is computed by gathering the MI distributions of the profiles after 30 random permutations of the samples and by choosing as a threshold the highest obtained MI value. The authors define the obtained graph a *Relevance Network*.

In [15], the binning procedure of gene profiles is realized through the use of polynomial B-spline functions, allowing each measurement to be assigned to more than one bin, with weights given by the B-spline. MI is then computed among binned variables.

In [47], the ARACNe algorithm computes MI for continuous gene profiles through a Gaussian Kernel estimator. The graph is then first pruned with a threshold computed similarly to [9], and the remaining links are then pruned again exploiting the Data Processing Inequality: for each triplet of edges, the edge with the lowest value of MI is removed.

Finally, in [84], two algorithms are proposed. The first one computes MI with quantization on k equally spaced bins, deletes link below a certain threshold t_S and, for remaining links, compute the minimum Mutual Information (mMI) conditioned on all the other variables (see Section 2.1.1); if mMI drops below a second threshold, t_M, the edge is removed. Since the authors state that there is no effective automated way to set the two thresholds t_S and t_M, they propose a second algorithm, which weights each edge with the product $(MI \cdot mMI)$ and returns the network as it is, leaving the user with the task of identifying a cutoff threshold.

A similar approach is proposed in [16] with the use of Pearson Correlation: (Oth order) PC is computed for each pair, the edges whose PC falls below a threshold are pruned, then 1st order PC is computed for the remaining edges and, after a second pruning, 2nd order PC is computed on what remains.

The last example we cite for unoriented graphs is [68], in which the concentration matrix is exploited as a measure of partial correlation between genes conditioned on all the other genes. The matrix should be obtained by inverting the correlation matrix, whose rank is rarely full when dealing with microarray data. The algorithm, then, exploits a bootstrap technique for estimating the true correlation matrix and then pseudo-inverses it to find the concentration matrix. Choice on edge inclusion in the network are then made through statistical tests and p-values computation. The output of the algorithm is defined by the authors a *Graphical Gaussian Model* (GGM).

Mutual Information is exploited also for the inference of Boolean Networks by the algorithm REVEAL, as described in [41]. The authors make the implicit assumption that the update rule of the Boolean Network they are inferring has a time delay exactly equal to the sampling time in the expression measurements (see Section 2.1.2 for some considerations on this assumption); with this hypothesis, the algorithm quantizes gene profiles in 2 levels (0 and 1) and, for each gene, searches among all the possible combination of regulators the one that maximizes MI. The expression profiles of the putative regulators and of the regulated gene are aligned after a proper time shift of one step and Mutual Information is computed, progressively increasing the number k_i of possible causes for each gene i. The algorithm stops when the sets of causes that fully explain each gene are found ($MI = 1$). In a second phase, the algorithm searches the boolean rule of k_i inputs that best relates the behaviour of each gene i with the profiles of its identified regulators.

2.2.2 Search in the space of networks

Models of higher complexity, such as systems of equations or Bayesian Networks, allow the user to make predictions on the evolution of gene expression over time. The goodness of a particular choice of network structure and system parameters can thus be evaluated simply by generating expression profiles with the estimated model and by computing an error measure between real and generated gene profiles.

Moreover, for some particular classes of Bayesian Networks, namely networks with discrete variables and networks with continuous Gaussian variables and linear dependencies between parents and children, it is possible to compute in closed form the *posterior probability* of a network structure given data, which measures the goodness of a particular structure indepently of the values of network parameters [69].

When such measures of goodness of fit can be defined, a clever exploration of the spaces of network structures and of network parameters can be accomplished, exploiting either enumeration, greedy search, exact search or Stochastic Local Search algorithms.

To reduce model complexity and computational time, algorithms usually exploit the property of sparsity of Gene Regulatory Networks, designing techniques to limit the number of regulators for each gene, either by limiting the maximum allowable in-degree, by progressing the search from easier to more complex configurations of the model or by adding a complexity term to the cost function to be minimized.

Most of the approaches based on Stochastic Local Search algorithms, which can return different solutions at each run, exploit some kind of Ensemble Learning strategies to merge the results of the various independent runs.

In what follows, we present the main examples from the literature of the usage of each of the aforementioned search strategies.

In the *NIR* (Network Identification by multiple Regression) algorithm [25], a linear additive model is fit to a set of steady state expression values, obtained with multiple perturbations of a biological system. Linear additive models can be decoupled, thus the algorithm search gene by gene for the best set of regulators. All the possible combinations of k regulators are linearly regressed to each gene profile and the combination with the smallest error is chosen. For each gene, the algorithm selects the value of k that provides the best balance between coverage and false positives.

The same model and search technique are exploited in [52], with a fixed number of 4 maximum regulators for each gene. In addition, the algorithm adopts an Ensamble Learning strategy, collecting votes on each edge from the topmost combinations of regulators (with a tunable threshold) and then choosing the edges with the highest scores.

In [24], Dynamic Bayesian Networks with Gaussian variables are fit to time series of gene expression data; networks are built in a greedy fashion with the K2 algorithm [12], which evaluates models of increasing complexity as long as there is a gain in the posterior likelihood of the model structure given the data.

In [38], the problem of fitting a system of differential equations is reduced to training n classifiers, one for each gene, able to identify the sign of the derivative of the expression signal at each time instant from the expression values of all the other genes. The learning procedure is treated as a linear programming problem and solved using an interior point method [48]. Regulatory relations are then inferred from the values of the numerical parameters of each trained classifier. Derivatives, used to train the classifiers, are estimated from expression profiles, after a proper filtering with local linear regression to reduce the effects of noise.

In [36] and [58], two variants of the same approach are proposed to fit a system of nonlinear additive equations: a search in the space of structures is carried out with an Ant Colony Optimization (ACO) algorithm [21] and candidate network structures are evaluated optimizing the continuous nonzero parameters with a Particle Swarm Optimization (PSO) algorithm [35] and returning the obtained Mean Squared Error as a measure of fitness of the candidate structure. The maximum allowed number of regulators for each gene is set to 2, to limit the size of the search space. The final solution is assembled from the results of multiple runs of the same stochastic algorithm, choosing the edges that were identified in at least the 50% of the runs.

The same idea of a mixed discrete and continuous optimization approach is exploited in [81] to fit a system of nonlinear sigmoidal differential equations (Dynamic Recurrent Neural Network). The space of network structures is searched with a binary PSO; each candidate structure is then evaluated, optimizing nonzero system parameters with a continuous PSO. Sparsity is induced stochastically with an additional parameter in the particle velocity update rule of the binary PSO aglorithm. Final edges are then selected from multiple runs of the same algorithm, with a majority vote strategy. The system is differential, and derivatives of the signals are approximated with finite differences, exposing the algorithm to the problems explained in Section 2.1.4.

A hybrid algorithm, that combines Differential Evolution [75] and Particle Swarm Optimization, is used in [80] to fit a Dynamic Recurrent Neural Network to gene expression data. The algorithm alternates between the two continuous optimization approaches in each iteration. Derivatives are approximated with finite differences and final edges are selected with majority vote, as in the previous cited paper.

A biomimetic evolutionary reverse engineering method [44] is exploited in [45], in conjuction with a signed voting strategy, to infer a log-sigmoid system of differential equations. Results of multiple independent runs of the algorithm are collected; each edge then receives a score of $+1$ every time it appears with a positive sign in one of the resulting matrices, of -1 ever time it appers with negative sign and of 0 every time its absolute value falls below a certain threshold. The algorithm is thus able to compute a confidence level for each edge, cancelling out discordant results and enhancing coherent results.

Evolutionary Computation and Stochastic Local Search techniques have been applied also to S-System fitting problems: in [37], a cooperative coevolutionary algorithm is designed, with the aim of trying to decouple the fit problem without having to estimate derivatives from data. The algorithm assign each equation of the S-System (see Eq. 2.3 in Section 2.1.4) to a different evolutionary solver. At each iteration, the solvers search for the optimal values of the equation parameters and compute an estimate of the gene expression profile; in the subsequent generation, the estimate is used in the optimization process by all the other solvers. A penalty term is added to the error function to keep the indegree of each node below a fixed value.

In [42], the inference of an S-System is treated as a multi-objective optimization problem, trying to minimize the error on gene expression, the error on the derivative and the complexity of the model. Single-objective problems are solved with Hybrid Differential Evolution [10] and solutions for the multi-objective problems are searched with the ϵ-constraint method [61].

2.3 Performance Measures

Performance of Reverse Engineering algorithms are compared in terms of the ability in reconstructing the real regulatory network. Two widely used measures, borrowed from the Information Retrieval community, are *Precision* (P) and *Recall* (R) [43] and are defined as:

$$P = \frac{tp}{tp + fp}$$

$$R = \frac{tp}{tp + fn}$$

where *tp* is the number of true positives, *i.e.* the number of causal relations correctly identified by the algorithm, *fp* is the number of false positives, *i.e.* the number of relations identified by the algorithm which are not correct, and *fn* is the number of false negatives, *i.e.* the number of relations present in the real network but not identified. Both measures range in the interval $[0, 1]$ and can be used both for oriented and unoriented graphs. A graphical explanation of the two measures is given in Figure 2.1.

When possible, the comparison between two algorithms should be based on the performance on a set of Reverse Engineering tasks, rather than on a single problem. One can then compare the average behaviour of the algorithms, using statistical tools such as the exact Wilcoxon two-sample test, and consider as significant differences whose p-value is below a certain threshold [13].

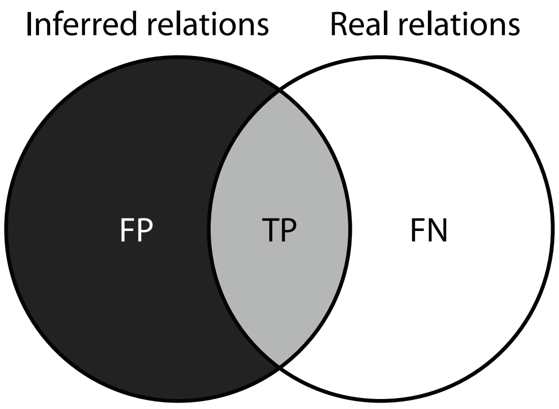


Figure 2.1: Graphical explaination of True Positives (tp), False Positives (fp) and False Negatives (fn). The left circle is the set of relations inferred by an algorithm and the right circle is the set of relations present in the real gene regulatory network. Precision is then the area of the intersection over the area of the left circle ($tp/(tp + fp)$) and Recall is the area of the intersection over the area of the right circle ($tp/(tp + fn)$).

2.4 Assessment of algorithm performance

After having reviewed the state-of-the-art of Reverse Engineering algorithms and described how to compare the performance of two algorithms, we present some results from recent assessment papers on the performance of the most widely used algorithms.

Relevance Networks (RNs), Graphical Gaussian Models (GGMs) and Bayesian Networks (BNs, sampled with an MCMC algorithm) are compared in [78], on a set of both simulated and real time course data, coming from an experiment on 11 human proteins involved in the Raf signalling network [22]. Time series are composed of 100 samples and are obtained both from experiments in natural response and from experiments with up and downregulation of some genes. GGMs and BNs outperform RNs on each dataset, and BNs show a higher performance than GGMs on the datasets with external stimulation, while no significant differences between the two are observed on the natural response datasets. BNs reach an average Precision on the oriented graphs of approximately 0.8 and 0.5 on simulated data, respectively with and without the external stimulus, and of 0.5 and 0.4 under the same conditions on real data.

In [13], a comparison between linear and nonlinear Dynamic Bayesian Networks (DBNs), Graphical Gaussian Models (GGMs) and the ARACNe algorithm is carried out on a set of simulated time series of gene expression profiles. The algorithms are compared on their ability to infer the unoriented gene networks: experiments are carried out on networks of different sizes (12, 20 and 100 genes) and with and without noise. Time series are gathered by initializing multiple times the simulated systems at random and leaving them free to evolve, sampling the evolution at subsequent time instants. From the analysis, linear DBNs show the best behaviour in data-rich situations (500 samples from 10 different experiments) both with and without noise, whereas ARACNe shows better performance when the number of samples per experiment is reduced. For networks of 100 genes, however, performance of the best method is poor even in the data-rich case (P and R below 0.4) both with and without noise; for networks of 20 genes, the best P and R are in the range of 0.5, and of 0.6 and 0.7 for 12 genes, respectively with and without noise. With fewer time samples (50 and 100) and on networks of 12 genes, both P and R of the best algorithm are in the range of 0.5.

Pairwise measures, both direct (Pearson Correlation and Mutual Information) and conditional (1st and 2nd order Partial Pearson Correlation, Graphical Gaussian Models and Conditional Mutual Information) are compared in [71] on a set of simulated networks with $n = 100$ genes, with increasing sample size m, both with systematic knock-out of each gene and in natural response experiments and both on time course and steady state experiments. Results show that steady state systematic gene knock-out experiments are the more informative; in this scenario, linear similarity measures (PC, PPC and GGM) shows the best performance. For time series data, performance is in general poorer, with the best behaviour exhibited by conditional measures (PPC, GGM and CMI) with respect to direct similarity measures. Performance tends to stabilize for steady state experiments when $m \geq n$, whereas it keeps growing for time course experiments.

In [6], ARACNe, Bayesian Networks (both static and dynamic, with the Banjo implementation [82]), NIR and a simple Hyerarchical Clustering are compared on a set of simulated networks of size 10, 100 and 1000, with 10, 100 and 1000 samples and in three conditions: steady state measurements, from perturbations of each gene and of the whole set of genes, and time course experiments obtained perturbing the 10% of the genes. In accordance with the previous assessment paper, better results are obtained in steady state experiments. On global perturbation, the highest performance is reached by Bayesian Networks; on locally perturbed data, the best algorithm is NIR, which however requires more information than the other algorithms to run. No algorithm is able to behave significantly better than random on time course experiments.

Finally, in [72] time course experiments, generated in simulation with S-Systems of 5 and 10 genes, are exploited to test a set of Evolutionary Algorithm on the network inference problem. Among Monte-Carlo search, multi-start hill climbing, binary and real-valued genetic algorithm, evolution strategies, evolution strategies with covariance matrix adaptation (CMA-ES, [28]), differential evolution and particle swarm optimization, CMA-ES is able to reach the lowest values of Relative Squared Error. No analysis, however, is carried out on Precision and Recall of the inferred networks.

2.5 Difficulties and limitations of the problem

As it is clear from the results presented in the previous section, the problem of Reverse Engineering of Gene Regulatory Networks is inherently difficult and the state-of-the-art of the algorithms is still far from reaching good results on a genomic scale ($O\left(10^3\right)$ genes). In the following, we try to resume these intrinsic difficulties and motivate them on the basis of what is known about gene regulation.

The first problem resides in the point of observation: with DNA microarray measurements, we observe the amount of RNA that is being translated in a particular time instant, but most of the biological interactions take place at the protein level; thus, we have to rely on an inderect observation of the phenomenon we are studying [49]. Moreover, we are possibly still not aware of all the components that interact in the regulatory process. For example, recent evidence emerged about micro RNA (μRNA, [59]), short molecules of non-coding RNA which interact with gene regulation, but are usually left out from normal microarray experiments because of their non-coding feature.

For all these reasons, we always observe a certain amount of *intrinsic noise* in microarray experiments, even between synchronized replicas of the same observation. Moreover, the vast amount of regulatory interactions that can take place at the protein level has the observed effect, on microarray observation, of variable regulatory delays among genes [85]. Models and algorithms that hypothesize synchronous interactions with a fixed time step usually fail to capture these kind of relations.

Another issue, known in the literature as the *curse of dimensionality* [8], is the disparity between the number of observed variables in a DNA microarray experiment ($O\left(10^3\right)$) and the usual number of observations for each variable ($O\left(10 \sim 10^2\right)$). From the theory of dynamical systems [8], if the number of observations of a system is less than the number of system variables, a whole set of different parameters settings for the system can explain the observations. Moreover, even in the case of more observations than variables, there can be the possibility that not all the dynamical states of the system are excited, and thus some relations are impossible to be inferred, simply because they were not observed in the experiment.

These are probably the reasons why state-of-the-art algorithms are not able to fully reconstruct regulatory networks even with simulated data, a large amount of samples and no noise [6, 71].

Time course experiments seem to be the less informative: there is good evidence of regulatory systems being nonlinear and differential, features that classifies them as the most difficult systems to infer from their dynamic evolution across time. Steady state measurements with systematic perturbation of genes, on the contrary, are the most informative [6, 71], but a lot of information on the dynamic evolution is lost. For these reasons, Ensamble Learning methods, that combine results of the same stochastic algorithm, of different algorithms on the same dataset or results on multiple and heterogenous experiments, are probably a good candidate to reliably infer regulatory relations on a global scale.

Chapter 3

Experimental data

We present in this chapter the experimental datasets that will be used through the thesis to test and validate the proposed approaches. Since there are a few regulatory networks which are known in the literature with sufficient confidence and no gold standards can be defined [13, 71], we mainly rely on simulation to assess the performance of the algorithms we design. In some cases, however, we test also the ability of our techniques to infer what is biologically known from real microarray data.

3.1 Simulated data

Performance assessments for each of our methods are carried out on simulated data generated with the Netsim biological network simulator [20], which is described in detail in Section 3.1.1. The performance of some algorithms is also tested on a dataset included in the DREAM4 In Silico Network Challenge [46, 73], more widely known by the Reverse Engineering community, which is presented in Section 3.1.2

3.1.1 Netsim and simulated datasets.

The biological network simulator Netsim [20] is able to mimic some important features of real regulatory networks: scale-free degree distribution, high clustering coefficient (independent of the number of nodes in the network) and low characteristic path length (small-world). Networks are built following a model hypothesized in [57] and exhibit a fractal organization: sampling from three kinds of different connection modules, which are found in regulatory networks with a frequency significantly higher than in random networks [2], nodes are connected together at different levels of network organization. Some of the links between nodes are randomly added to guarantee the desired clustering coefficient and scale-free distribution of the node degree.

Once the network is built, dynamic profiles are generated with a combination of fuzzy boolean logic and differential equations, accounting for saturation in the response to regulation and transcription activation thresholds: for each gene i, the expression of its regulators is combined with a set of fuzzy boolean rules, modeling cooperative, competitive and synergic interactions. The result is a gene-specific function $T_i(t)$, ranging from 0 to 1. A nonlinear differential model is then applied to the regulatory function, according to the two following equations:

$$S_i\left[T_i(t), \alpha_i, \beta_i\right] = \frac{1}{1 + e^{-\alpha_i \cdot (T_i(t) - \beta_i)}}$$

$$\frac{dx_i(t)}{dt} = \lambda_i \cdot \left[S_i\left(T_i(t), \alpha_i, \beta_i\right) - x_i(t)\right] \qquad 1 \leq i \leq n,$$

where α_i, β_i and λ_i are gene-specific parameters. Throughout the thesis, we will refer to this model as the *fuzzy boolean* model.

The simulator implements the possibility of observing gene dynamics by either letting the system in natural response from opportunely chosen initial conditions or by exciting it with external stimuli acting on chosen nodes.

From the same simulated networks, a second set of dynamics were generated with a different model and exploited for the experiments of Chapter 6. Since we developed a method for the optimization of a weight matrix system, we chose to model expression dynamics with a system of sigmoidal differential equations in the form

$$\frac{dx_i}{dt} = \sigma\left[\sum_{j=1}^{n} w_{ji} x_j + u_i\right] - \lambda_i x_i \qquad 1 \leq i \leq n.$$

For further details and motivations on the selected model, which is identical to what is known in literature as Dynamic Recurrent Neural Networks, please refer to Section 2.1.4. Throughout the thesis, we will refer to this model as the *recurrent neural network* model.

Simulated networks

To have a dataset which spans across different levels of complexity, we generated a set of networks with different number of genes, namely 5, 8, 10, 20, 50 and 100. To assess the average behaviour of our algorithms on different test cases, we generated 20 different networks for each size.

Simulated dynamics

A set of different simulated experiments were generated, starting from the simulated networks structures, to test the behaviour of the various algorithms on different kind of experimental data:

- **Natural response**: gene profiles of each network were initialized at random and left free to evolve until a steady state is reached. Samples were taken at equally spaced temporal instants for the boolean fuzzy model and at logarithmically spaced temporal instants for the recurrent neural network model. Increasing logarithmically the time spacing while sampling is common practice in real microarray experiments, because meaningful information usually concentrates right after the perturbation of a dynamical system.

- **External stimulation**: a second set of time series was obtained by externally stimulating each network at its *hub*, *i.e.* at the node with the highest out degree. The node was stimulated with three kinds of signals: a ramp, a step an a sinusoid[1]. Amplitude of stimulating signals lies in the range of simulated expression data and the period of the sinusoid is one third of the total observation time.

- **Knock-out**: a third dataset was obtained simulating a systematic knock-out of each gene in the network. The system was intialized at random, left free to evolve and sampled after a time lag sufficient for reaching a steady state, to simulate the so called *wild type* experiment. Expression of one gene at a time was then initialized at zero and kept constant during the evolution of the system, while all the other $n - 1$ genes were initialized with the same initial values of the wild type experiment; one sample for each of the experiments was collected at the steady state.

3.1.2 DREAM4 In Silico Network Challenge

The DREAM project (Dialogue for Reverse Engineering Assessments and Methods [46, 73]) has the main goal of catalyzing the interaction between experiment and theory in the area of cellular network inference, trying to achieve a fair comparison of the strengths and weaknesses of Reverse Engineering methods and a clear sense of the reliability of the network models they produce. With this purpose, a set of simulated Reverse Engineering challenges is published every year and researchers has the possibility to compete on their ability to infer the networks underlying the simulated data.

We exploited one of the datasets of the most recent edition, the 10 genes DREAM4 In Silico Network Challenge of 2009, to assess the performance of some of our methods. The datasets consists of a variety of different simulated experiments on a set of five networks of ten genes, whose structure is extracted from what is known in the literature of the real regulatory networks of two biological organisms, namely Saccharomyces Cerevisiae and Escherichia Coli.

For each of the five networks, various types of experiments are simulated to produce the gene expression datasets:

- **Wild type**: the dataset contains the steady-state levels of the wild-type, *i.e.* of the unperturbed network.

- **Knock-out**: the dataset contains the steady-state levels of single-gene knockouts. An independent knockout is provided for every gene of the network. A knockout is simulated by setting the transcription rate of the target gene to zero.

- **Knock-down**: the dataset contains the steady-state levels of single-gene knockdowns. A knockdown of every gene of the network is simulated by reducing the transcription rate of the corresponding gene by half.

- **Multifactorial perturbations**: the dataset contains steady-state levels of variations of the network, which are obtained by applying multifactorial perturbations to the original network. Multifactorial perturbations are

[1]Thanks to synthetic biology techniques, it is feasible to excite a system using a ”step” or a ”ramp” stimulus, whereas a sinusoidal signal is still difficult to implement in practice; however, it is interesting from a theoretical point of view to evaluate its effects.

obtained by slightly increasing or decreasing the basal activation of all genes of the network simultaneously by different random amounts.

- **Time series**: the dataset contains time courses showing how the network responds to a perturbation and how it relaxes upon removal of the perturbation. For networks of size 10, 5 different time series are provided, each with 21 time points. The initial condition always corresponds to a steady-state measurement of the wild-type. At t=0, a perturbation is applied to the network. The first half of the time series shows the response of the network to the perturbation. The perturbation is then removed and the second half of the time series shows how the gene expression levels go back from the perturbed to the wild-type state.

The simulations are based on stochastic differential equations to model internal noise in the dynamics of the networks. In addition, measurement noise is added to the generated gene expression datasets.

In our analysis, then, we first exploit the time course data to infer a model for the noise and then use the model to select the genes which are differentially expressed between the knock-out and the wild type experiments, as described in detail in Appendix B. Information on differentially expressed genes is then used by the Qualitative Reasoning algorithm presented in Section 4.1 to infer the underlying regulatory networks.

3.2 Real data

To validate the behaviour of some of our algorithms on real data, we run them on a dataset consisting of human genes involved in cell cycle [79]. From the original dataset, we extracted 9 genes whose interactions are documented in the BioGRID database[2]: CCNA1, CCNB1, CCNE1, CDC2, CDC6, CDKN3, E2F1, PCNA and RFC4. The documented relations between the 9 genes are shown in Figure 3.1.

The time series consist of 47 equally spaced samples, taken every hour; since human cell cycle is approximately 15 hours long, the time series span across approximately three complete cell cycles. The dataset has already been used in [60] to test the accuracy of a method for extracting temporal relationships between genes.

[2]www.thebiogrid.org

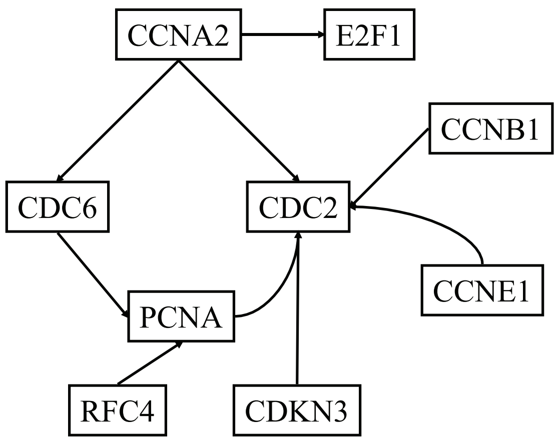


Figure 3.1: Network of known interactions for the real dataset of 9 human genes related to cell cycle.

Chapter 4

Inferring the oriented network

In this chapter, we present two novel algorithms for the inference of Oriented Networks from DNA microarray experiments. The two techniques are designed to process two different kinds of microarray data: in the first case (Section 4.1) steady state measurements of systematic gene perturbation experiments are analyzed with a Qualitative Reasoning approach. Causal relations between genes are inferred through the qualitative comparison between each perturbation experiment and the wild type, *i.e.* the experiment in which no gene is perturbed. In the second case (Section 4.2), time-course profiles of gene expression are processed with an information-theoretic approach, through the use of a heuristic scoring function for causal relations. The scoring function is able to tolerate a certain level of noise in expression profiles and of variability in gene-specific regulatory delays.

4.1 Qualitative reasoning on steady state perturbations

Steady state experiments of systematic gene perturbation are among the most informative for the purpose of reconstructing Gene Regulatory Networks [71]. The experiments consist in the systematic suppression of each gene, either by making it inoperative (gene knock-out) or by degradating its RNA product (RNA silencing), followed by a single microarray observation, sampled when the system has reached a steady state. In parallel, a *wild type* experiment is carried out, sampling the steady state of the system when no genes are perturbed.

Steady state perturbation experiments are usually processed by means of quantitative methods but, as far as we know, no systematic study on qualitative analysis of perturbation data has ever been conducted. In this section, we describe a novel qualitative reasoning approach to the inference of directed regulatory relations between genes; our approach exhibits an extremely low rate of false positives and provides meaningful insights on the amount of useful information conveyed by steady state perturbations.

The analysis is carried out on two simulated datasets of systematic gene

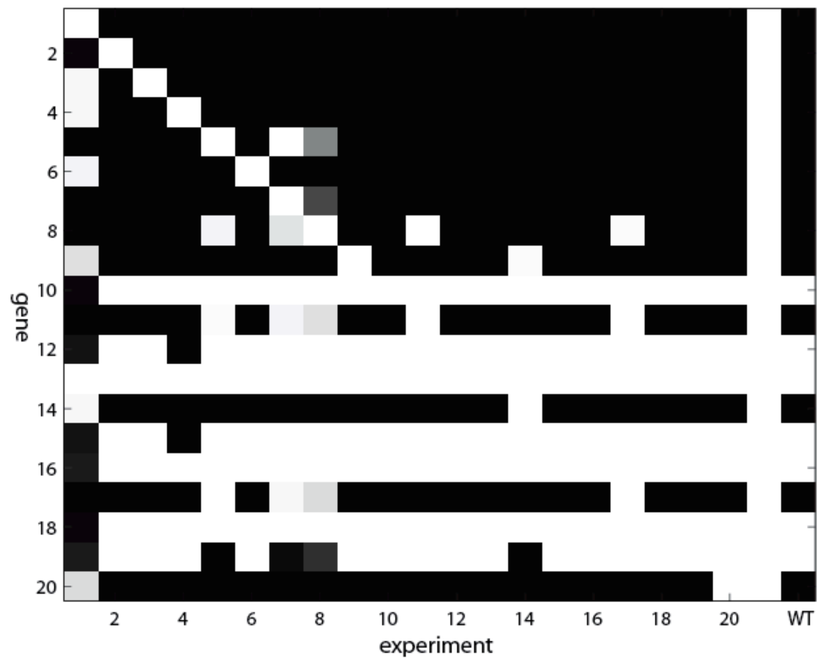


Figure 4.1: Graphical representation of a set of 20 gene knock-out experiments plus the wild type. Each row correponds to a gene and each column to an experiment. The level of gray is proportional to the relative expression value of the gene, from white (no expression) to black (maximum expression).

knock-out experiments, to assess the average performance of our algorithm on a rich set of test cases and with complete information on network topologies. The first dataset is generated with the Netsim simulator, as described in Section 3.1.1, while the second one is extracted by the DREAM4 In Silico Network Challenge, presented in Section 3.1.2.

4.1.1 Methods

Figure 4.1 shows a graphical representation of the output of a set of systematic gene knock-out experiments on a simulated network of 20 genes, flanked with the output of the wild type experiment (the rightmost column). Rows of the grid correspond to genes and columns to experiments, and the level of gray in each box is proportional to the normalized value of gene expression, from white (no expression) to black (maximum expression). One can observe that, in the majority of the experiments, the expression of most of the genes does not differ much from the wild type: just in a small set of cases a knock-out of a gene has visible effects on a large number of genes. This behaviour is possibly related to the fact that regulatory networks are scale-free, thus exhibiting few highly connected nodes and a large number of loosely connected nodes. Moreover, knocking out genes that reach a low expression value in the wild type experiment has no visible effects on the other genes.

To extract qualitative information contained in a set of knock-out experiments, one can subtract the wild type from all the other experiments and

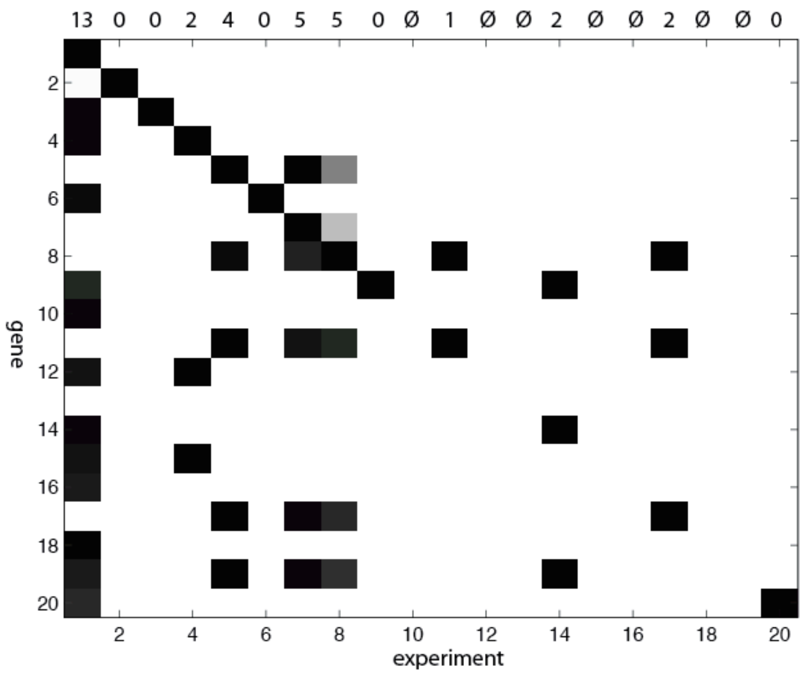


Figure 4.2: Graphical representation of the qualitative information contained in the same set of 20 knock-out experiments, obtained by subtracting the wild type from each column and taking absolute values. The number of elements whose value is larger than a fixed threshold θ is reported on top of each column.

consider all the genes whose absolute values lie above a threshold θ; for each knock-out experiment, we define these genes as the *observed effects* of the experiment. The result of such an operation on the example dataset is shown in Figure 4.2, where effects of the knock-out of each gene are clearly readable in its corresponding column. On top of each column, we reported the number of observed effects for each gene knock-out.

Moreover, we define as *not observed* the genes for which the corresponding element on the diagonal exhibits an absolute value smaller than the threshold θ. In Figure 4.2, the corresponding columns are marked with $\emptyset$. Notice that this is different from having no observed effects, as in the case of genes 2, 3, 6, 9 and 20 in the Figure.

With the aim of extracting direct regulatory relations in the form

$$y \Rightarrow x \quad ,$$

where y is one of the regulators of x and x is one of its regulated genes, we take into account for each gene the binary qualitative feature of being or not being an observed effect of a particular knock-out experiment (thus leaving aside the quantitative values of expression). All the observed effects of each gene are mapped to a string representation, as the one in Table 4.1. For convenience, we denote $\mathit{eff}(x)$ the set of observed effects of the knock-out of gene x.

The following considerations can be drawn from the string representation:

10:	not observed													
12:	not observed													
13:	not observed													
15:	not observed													
16:	not observed													
18:	not observed													
19:	not observed													
2:														
3:														
6:														
9:														
20:														
11:	8													
14:	9	19												
17:	8	11												
4:	12	13	15											
5:	8	11	17	19										
7:	5	8	11	17	19									
8:	5	7	11	17	19									
1:	2	3	4	6	9	10	12	13	14	15	16	18	19	20

Table 4.1: String representation of the observed effects for each gene.

- No inference can be carried out on the effects of not observed genes, because the information is not present in the particular set of experiments.
- For knock-out experiments with only one observed effect, a causal relation between the knocked out gene and its observed effect can always be inferred. We name these inferred relations *single effect rules.* Table 4.1, for example, reveals the causal relation $11 \Rightarrow 8$.
- If there exist x and y such that $\mathit{eff}(y) = \{x, \mathit{eff}(x)\}$, the causal relation $y \Rightarrow x$ can be inferred. The motivation for this rule is the propagation of the perturbation originating from the knock-out of y to all and only the observed effects of the knock-out of x. We name this type of inferred relations *simple inclusion rules.* Two examples from Table 4.1 are $17 \Rightarrow 11$ and $7 \Rightarrow 5$.
- If there exist x and y such that $\mathit{eff}(y) = \{x, \mathit{eff}(x), K\}$, with K an additional set of genes, to infer the causal relation $y \Rightarrow x$ one has to exclude that none of the genes in K interposes in the path between y and x (*i.e.* is not a direct or indirect cause for x). The latter condition is verified if each $k \in K$ satisfies either of the two following conditions:
  - k is observable and x is not an observable effect of k,
  - there exists a z such that k is an observable effect of z and x is not.

  We name this type of inferred relations *strict inclusion rules.* An example from Table 4.1 is the rule $5 \Rightarrow 17$: the effect list of gene 5 contains gene 17, the effect list of 17 and gene 19. However, 19 is an observed effect of 14 and 17 is not, thus 19 can not be a cause for 17 and the rule holds.

Simple inclusion and strict inclusion rules have an exception: they cannot be applied to infer the causes of a gene x if there exists at least a gene y such that $\{x, \mathit{eff}(x)\} \equiv \{ y, \mathit{eff}(y)\}$. This behaviour is in fact the evidence of the presence in the regulatory network of an oriented closed loop to which both x and y belong. For the simple inclusion and strict inclusion rules the two genes are thus indistinguishable in this qualitative framework. An example of genes for which this situation holds in Table 4.1 is the pair 7 and 8.

A Qualitative Reasoning algorithm can thus be designed to infer single effect rules, simple inclusion rules and strict inclusion rules from a set of systematic gene knock-out experiments; its pseudocode is presented in what follows.

```
QUALITATIVE($\mathbf{A}^{n\times n}, \mathbf{w}^{n\times 1}, \theta$)
 1  Subtract $\mathbf{w}$ from each column of $\mathbf{A}$, and store the absolute value in $\mathbf{D}^{n\times n}$
 2  For each element in $\mathbf{D}$, if $abs(\mathbf{D}[i,j]) < \theta$ then $\mathbf{D}[i,j] \leftarrow 0$
 3  For each $x$, extract $\mathit{eff}(x)$ as the indexes of nonzero elements of the $x$-th column of $\mathbf{D}$.
 4  $n \leftarrow ncols(\mathbf{D})$
 5  $\triangleright$ Single effect rules
 6  for $x \leftarrow 1$ to $n$
 7       do
 8           if $length(\mathit{eff}(x)) = 1$
 9              then $\mathbf{C}[\mathit{eff}(x), x] \leftarrow 1$
10  for $l \leftarrow 1$ to $\operatorname{argmax}_x length(\mathit{eff}(x)) - 1$
11       do
12           for all $x \mid length(\mathit{eff}(x)) = l$
13                do
14                    $\triangleright$ Simple inclusion rules
15                    if $\exists\, y \mid \mathit{eff}(y) = \{x, \mathit{eff}(x)\}$
16                       then $\mathbf{C}[x,y] \leftarrow 1$
17                              $\triangleright$ Strict inclusion rules
18                       else  if $\exists\, y \mid \mathit{eff}(y) = \{x, \mathit{eff}(x), K\}$
19                                then if for each $k \in K$:
20                                         $k$ observable and $x \notin \mathit{eff}(k)$
21                                         or
22                                         $\exists\, z \mid k \in \mathit{eff}(z)$ and $x \notin \mathit{eff}(z)$
23                                            then $\mathbf{C}[x,y] \leftarrow 1$
24  return $\mathbf{C}$
```

The algorithm receives as input the squared matrix of knock-out experiments $\mathbf{A}^{n\times n}$, the column vector of the wild type experiment $\mathbf{w}^{n\times 1}$ and the threshold θ; it returns as output the inferred connectivity matrix $\mathbf{C}^{n\times 1}$, in which $\mathbf{C}[x,y] = 1$ if the rule $y \Rightarrow x$ was inferred.

Analyzing the computational complexity, one can observe that the preprocessing phase (rows 1 and 3) take $O\left(n^2\right)$ operations. Searching for single effect rules takes $O\left(n\right)$ operations (rows 6−9); then, the two for loops at rows 10 and 12 scan totally $O\left(n\right)$ elements, searching for simple inclusion rules takes $O\left(n\right)$ and searching for strict inclusion rules can take $O\left(n^3\right)$ in the worst case, *i.e.* if the condition on line 22 has to be verified for every k. Thus, the algorithm has a total worst case complexity of $O\left(n^4\right)$.

The inference ability of this method is however limited: it is designed to infer at most one regulatory relation for each line of the string representation,

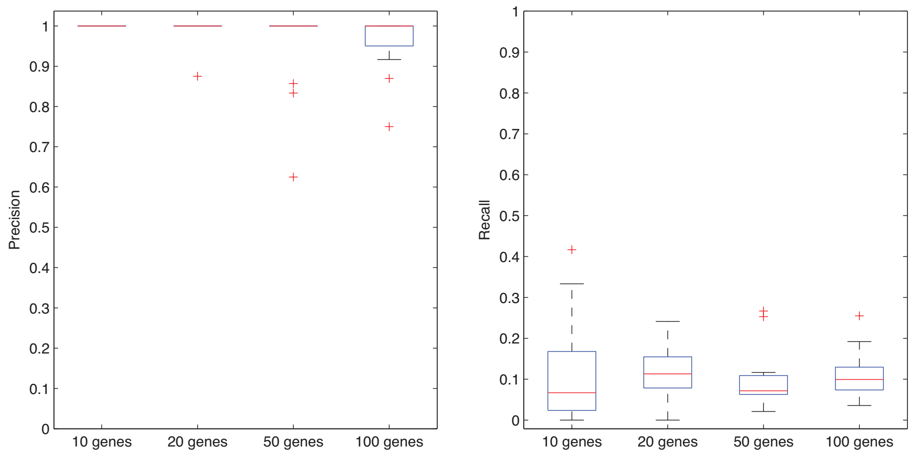


Figure 4.3: Boxplots of Precision (left) and Recall (right) of the qualitative inference algorithm, on 20 networks of sizes 10, 20, 50 and 100.

with the gene corresponding to the line as the regulator. Thus, even in the case of all observable genes, no rule can be inferred for nodes wich are leaves in the graph, *i.e.* which have no outgoing edges. The maximum possible number of inferred relations is thus $n - l$, where l is the number of leaves in the graph.

4.1.2 Results

We first run our qualitative inference algorithm on the gene knock-out simulated dataset described in Section 3.1.1, composed of 20 test network for each of the sizes 10, 20, 50 and 100. After an empyrical nonexhaustive search, we set the threshold θ to the 5% of the maximum expression value. As measures of the performance of the algorithm, we exploited Precision and Recall, described in detail in Section 2.3. Boxplots for Precision and Recall on the four sets of networks are showed in Figure 4.3.

As it is clear from the figure, Precision is 1 in the vast majority of cases, meaning that the number of false positives is extremely low. Average Recall, on the other hand, is low: on average, the method is able to infer approximately the 10% of the real regulatory relations.

The algorithm was then run on a set of simulated knock-out experiments extracted from the DREAM4 In Silico Network Challenge: the dataset consists of the sistematic knock-out of each gene, plus a wild type experiment, for five networks of ten genes. Data, in this case, contain noise both inherent in the dynamical model, a system of stochastic differential equations, and added in a second step to simulate experimental noise. To extract the genes which are differentially expressed between the wild type experiment and each of the knock-out experiments we adopted the procedure described in Appendix B, which consists in learning a model for the noise and in exploiting it to compute the confidence threshold θ.

The results on the DREAM4 data, in terms of Precision and Recall, are shown in Figure 4.4. As one can observe, the Precision of the qualitative algorithm is rather high even in the presence of noise, with an average of 0.85, and

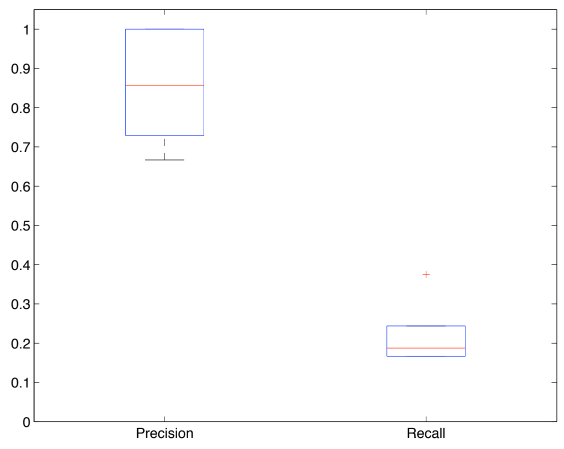


Figure 4.4: Boxplots of Precision and Recall of the qualitative inference algorithm, on the 5 networks of size 10 from the DREAM4 In Silico Network Challenge.

the average Recall, 0.20, is almost doubled with respect to the one obtained on the previous dataset.

The high level of Precision reached by our Qualitative Reasoning algorithm, together with the polynomial running time, makes it a good preprocessing tool for a general inference algorithm, able to provide valuable and reliable information on a subset of the regulatory relations and thus to reduce the size of the search space of the second algorithm.

4.1.3 Conclusions and future directions

We described in this section a novel Qualitative Reasoning algorithm for the inference of directed causal relations between genes from steady state experiments of systematic gene perturbation. The algorithm extracts from the data a qualitative description of the observable effects of each perturbation and it is both able to infer three kinds of regulatory rules and to explictly point out which parts of the network are impossible for it to infer, given the outcomes of the perturbation experiments.

The qualitative abstraction is based on a fixed numerical threshold θ, used to select the observed effects of each experiment. A possible future direction would be to make the threshold adaptive to data, relating it to either the expression of the same gene through all the experiments (gene-specific threshold) or to all the genes in each experiment (experiment-specific threshold). In the presence of noise, as we present in Appendix B, another approach is to learn the noise model from the data and to exploit statistical techniques to identify the differentially expressed genes, i.e. the genes whose expression differs from the wild type experiment.

Another possible future direction would be to extend the qualitative framework to consider also the sign of the observed effects of each perturbation, classifying them as overexpressed or underexpressed with respect to the wild type, and to study both how this affects the three rule inference procedures and if new procedures can be defined in this framework.

In the next section, we present a different approach to the inference of oriented regulatory graphs, which exploits time course rather then steady state data.

4.2 CNET: a novel information theory based algorithm for reverse engineering

A widely used approach to infer regulatory relations is the analysis of the Shannon Entropy and Mutual information between shifted time series of gene expression signals, proposed by Liang *et al.* in the REVEAL algorithm [41]. In this section, we propose an extension of the REVEAL approach, introducing a scoring function for regulatory relations and an algorithm, CNET [63, 64] that exploits the function to infer regulatory relations between genes. The function allows our approach to account for inconsistencies in gene expression time series caused by variable regulatory delays, measurement noise and quantization errors.

The output of our algorithm is an oriented regulatory graph (for a detailed description of this model, please refer to Section 2.1.1). We compare CNET with the original REVEAL algorithm and with Dynamic Bayesian Networks (DBNs) [24], an approach that was identified as promising in the three assessment papers [6], [13] and [71] and that is suitable for the inference of oriented graphs from time series data.

Comparisons are carried out on the simulated set of time course experiments described in Section 3.1.1: simulated data are useful when comparing different algorithms, because of the lack of information on real regulatory networks, which prevents one from obtaining both reliable test beds from real biological experiments and multiple cases to evaluate average performance [71]. To assess the behaviour of our algorithm on real data, we tested it also on the real dataset described in Section 3.2, consisting of the observation during cell cycle of 9 human genes.

In Section 4.2.1 we describe our inference model and the scoring function for causal relations, in Section 4.2.2 we present CNET algorithm, in Section 4.2.3 we show the results of the comparison between CNET and DBNs and in Section 4.2.4 we draw some conclusions.

4.2.1 CNET scoring function

CNET algorithm is designed to infer oriented graphs, in which nodes represent genes and edges represent causal regulatory interactions among genes. Nodes can be in three possible discrete states $\{1, 0, -1\}$, corresponding to high, medium and low (or increasing, steady and decreasing) expression level.

To adapt our algorithm to the behaviour of real gene profiles, which comprehends noise and variable regulatory delays, and to limit the effect of the error introduced by the quantization, we decided to map the REVEAL condition on Shannon Entropies to the domain of *consistent pairs*, as previously suggested by [51]: for signals with a finite set of possible values (*quantized signals*), the pair $\langle regulators, regulated\ signal \rangle$ is said to be consistent if and only if each combination of values for the regulators univocally correponds to a particular value for the regulated signal after Δ time steps. In Appendix A, we prove mathematically that the REVEAL condition is verified if and only if

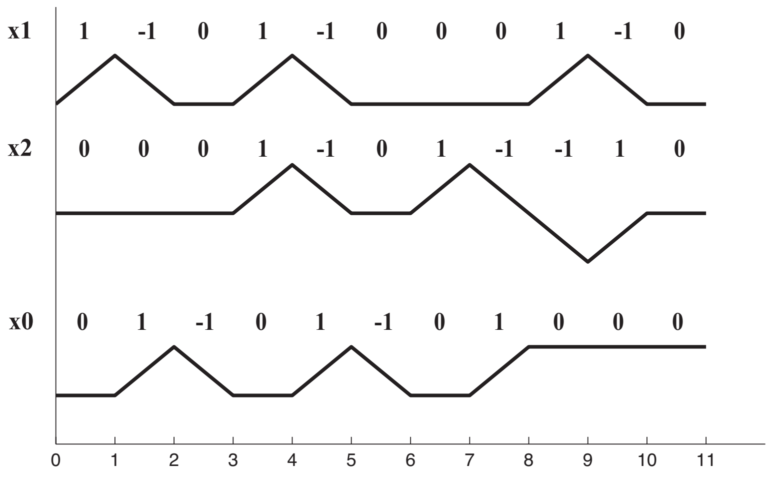


Figure 4.5: An example of a causal relation $(x_1, x_2) => x_0$ which satisfies REVEAL condition: profiles are quantized in increasing, steady and decreasing and time delay Δ is equal to 1. To each combination of values $(\overline{x}_1, \overline{x}_2)$ at time t always corresponds the same value $\overline{x}_0$ at time $t+1$: for example, to (0,0) in the regulators always corresponds a 0 in the regulated gene after one time step.

x_1	1	1	1	0	0	0	0	0	1	1	1	1	1	1
x_2	-1	-1	-1	1	1	1	1	1	-1	-1	-1	-1	-1	-1
x_0	0	0	0	0	0	-1	-1	-1	-1	-1	0	0	0	0
Block Length	BL_1= 3			BL_2= 5					BL_3= 6					
Steady State Length	SSL_1= 3			SSL_2=3					SSL_3=4					

Figure 4.6: Example of a causal relation $(x_1, x_2) \Rightarrow x_0$. Profiles are already aligned properly.

$\langle regulators, regulated\ signal\rangle$ is a consistent pair. An example which verifies this condition is shown in Figure 4.5, with two regulators and time delay $\Delta = 1$.

This switch of domains allowed us to design a novel heuristic function for regulatory pairs: each term of the function is ment to capture a particular aspect of gene expression profiles and the ensemble of terms gives an indication of how far the regulatory pair is from being consistent, thus inducing an ordering among regulatory pairs. For a given regulated signal, then, the particular combination of regulators that maximizes the scoring function can be searched.

For every gene x_0, the algorithm searches extensively for the best set of k regulators $(x_1 \ldots x_k)$ that maximizes a *scoring function* f:

$$f = w_e \frac{1}{1+e} + w_s s + w_c \frac{c}{3^k} \quad \text{with} \quad w_e + w_s + w_c = 1 \ . \tag{4.1}$$

f ranges in (0,1] and combines the contribution of an error term e, a shape term s and a completeness term c, by weighting them with parameters w_e, w_s and w_c. Each term is explained in what follows.

Error Term

To illustrate the meaning of $1/(1+e)$, we refer, without loss of generality, to the case with two regulators x_1 and x_2 for a regulated gene x_0: $(x_1, x_2) \Rightarrow x_0$. The error term e is defined as:

$$e = \sum_{i=-1}^{1} \sum_{j=-1}^{1} e_{ij} = \sum_{i=-1}^{1} \sum_{j=-1}^{1} \frac{Lp_{ij} - Mp_{ij}}{Lp_{ij}} \quad , \tag{4.2}$$

where Lp_{ij} is the number of occurrences of the input pattern $\langle i, j\rangle$ (*e.g.* $\langle 1, -1\rangle$ in Figure 4.6 occurs 9 times, then $Lp_{1,-1} = 9$), and Mp_{ij} is the value of gene x_0 that is most frequent as output in correspondence to the input $\langle i, j\rangle$ (*e.g.* in Figure 4.6, for the input pattern $\langle 1, -1\rangle$, the value $x_0 = 0$, occurring 7 times, is more frequent than -1, occuring twice, thus $Mp_{1,-1} = 7$).

If there is a univocal correspondence between input and output profiles (and then the pair $\langle (x_1, x_2); x_0\rangle$ is consistent), all the e_{ij} are equal to zero and thus $1/(1+e) = 1$, otherwise $1/(1+e) < 1$. This allows the function to relax the consistency condition and to tolerate a certain amount of noise and quantization errors in the data.

Shape term

The shape term is calculated organizing the quantized profiles into blocks of equal input combinations, such as the three blocks identified by vertical bars in Figure 4.6. The shape term s is computed as

$$s = \frac{1}{\#blocks} \sum_{i=1}^{\#blocks} s_i = \frac{1}{\#blocks} \sum_{i=1}^{\#blocks} \frac{SSL_i}{BL_i} \tag{4.3}$$

where BL_i is the length of the i-th block and SSL_i is the length of the rightmost substring of identical characters in the i-th block of x_0.

For example, in Figure 4.6, the shape terms for the three blocks are 1 (3/3), 0.6 (3/5) and 0.67 (4/6), thus leading to an average shape term of 0.76.

The shape term s is similar to e, in the sense that it relaxes the consistency condition, but it assigns lighter penalties to output inconsistencies occurring right after a change of state in regulators; it then rewards situations in which the output signal, after a change in the input, shows a short transient state followed by a longer steady state. This terms helps the algorithm to capture regulatory relations even in the presence of regulatory delays variable from gene to gene and longer than the fixed value Δ.

Completeness term

The completeness term $c/3^k$ is the normalized number of different combinations of values for regulators present in data ($1 \leq c \leq 3^k$, if k is the size of the set of regulators). For example, in Figure 4.6 two combination of values, $\langle 1, -1\rangle$ and $\langle 0, 1\rangle$, are present in input, then $c/3^k = 2/9$. This term induces the algorithm to prefer simpler solutions, *i.e.* solutions with less regulators, the other two terms being equal.

4.2.2 Algorithm

Pseudocode for CNET algorithm is as follows:

```
CNET($data, max\_causes$)
1  for $i \leftarrow 0$ to $n\_genes$
2      do $max\_fitness[i] \leftarrow 0$
3         $C[i] \leftarrow \emptyset$
4         for $k \leftarrow 1$ to $max\_causes$
5             do for $causes$ in combinations($k, n\_genes$)
6                 do $f \leftarrow$ fitness($i, causes$)
7                    if $f = max\_fitness[i]$
8                      then
9                           $C[i] \leftarrow C[i] \cup causes$
10                   if $f > max\_fitness[i]$
11                     then
12                          $C[i] \leftarrow causes$
13                          $max\_fitness[i] \leftarrow f$
14        if $C[i] > 1$
15          then weight each cause proportionally to
                 the number of times it appears in $C[i]$
16 return $C$
```

For each gene, CNET algorithm searches extensively for all the possible combinations of regulators, from one to a maximum user defined number (*max_causes*), and keeps track of the best scoring combinations. If more than one set of regulators for the same gene achieve the best score, the weight of each regulator is set proportional to the number of times it appears among the best scoring sets.

4.2.3 Experimental Results

We first test CNET on a simulated dataset, to compare its performance with the REVEAL algorithm and with Dynamic Bayesian Networks (DBNs), one of the best approaches for reverse engineering gene regulatory networks from time series data. In the implementation we chose, DBNs model continuous Gaussian variables and relations between parents and children are linear [24]; networks are inferred with the greedy search K2 algorithm, which builds the network node by node, adding regulators as long as a gain is observed in the posterior likelihood of the model structure given the data. We then validate performance on a real microarray dataset of 9 human genes. Datasets are described in detail in Sections 3.1.1 and 3.2 and are briefly reviewed in what follows.

Simulated data consist of 60 networks of 10 genes and 60 networks of 20 genes. For each network, 4 different time series of 50 samples were generated: the first time series is obtained observing the natural response of the network from random initial conditions, the other three time series are obtained stimulating the network with a sinusoid, a ramp and a step signal respectively. Networks are stimulated at their *hub*, *i.e.* the node with the highest out degree, to excite the highest number of nodes in the network.

The real dataset, on the other hand, consists of 9 genes involved in human cell cycle, for which samples were taken every hour for 47 hours (approximately three complete cell cycles). To measure the performance of CNET, we compared

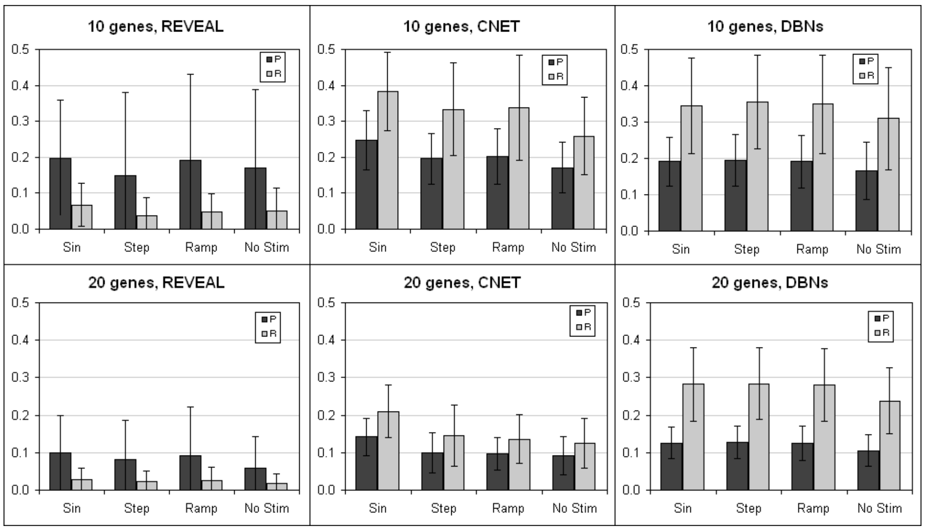


Figure 4.7: Average Precision and Recall for REVEAL, CNET and DBNs on 60 networks of 10 genes (upper row) and 60 networks of 20 genes (lower row). Values are reported for each of the three external stimuli and for the case without stimulus.

the output of the algorithm with the interactions documented in the BioGRID database (www.thebiogrid.org).

The inference algorithm for DBNs has 2 tunable parameters, the mean and the standard deviation of the prior probability distribution, μ_0 and σ_0. The tunable parameters for CNET are w_e, w_c and w_s, relative weights of the three terms in Equation 4.1. The parameters of the two algorithms were set by running DBNs and CNET on a training set consisting of 10 networks with 10 genes and 10 networks of 20 genes. Parameter values able to optimize the average performance on the training set were used for the remaining tests.

We compare the results of the three algorithms in terms of Precision and Recall, which are described in Section 2.3. Scores were compared using exact Wilcoxon two-sample tests: we considered as significant differences corresponding to a p-value < 0.05.

Results of the comparison between REVEAL, CNET and DBNs are shown in Figure 4.7. CNET significantly outperforms REVEAL both in terms of Precision an Recall, and its performance is comparable to DBNs: on networks of 10 genes, there is no significant difference between the two algorithms, whereas on 20 genes networks DBNs exhibit a significantly higher Recall. Precision and Recall are in line with the literature, being in the same range of values reported in Section 2.4 for similar experiments. CNET seems to be more sensitive than DBNs to the kind of input signal, showing significantly higher performance when the network is stimulated with the sinusoid. Both algorithms, however, show equal or higher performance when an external stimulus is present.

In microarray experiments, when a particular gene is externally stimulated, the interest usually focuses mostly on genes directly dependent from the target: for this reason, we analyzed also the performance of CNET and DBNs on the inference of regulatory relations directly outgoing from the externally stimulated gene. Results are shown in Figure 4.8. Performance on this subset of relations

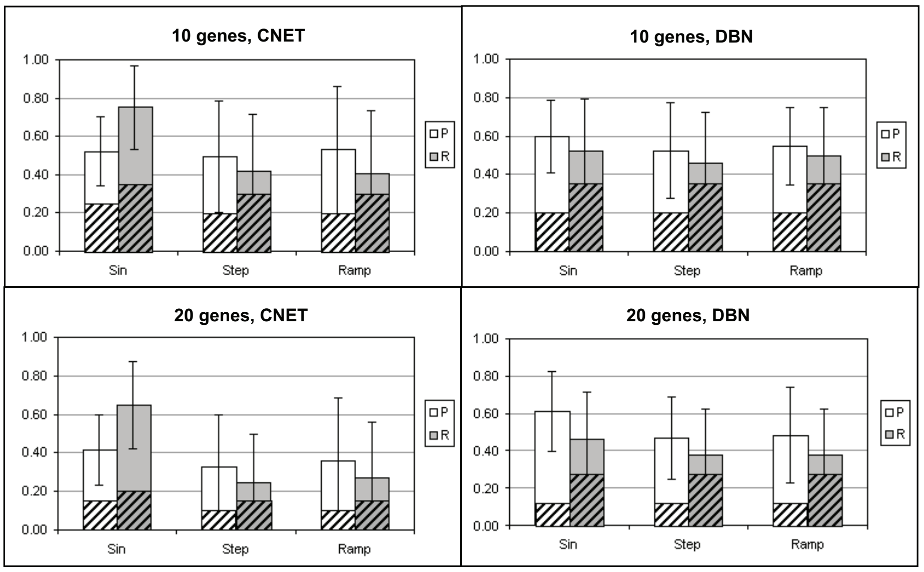


Figure 4.8: Precision and Recall for CNET and DBN, on edges directly outgoing from the stimulated gene, for networks of 10 genes (upper row) and 20 genes (lower row). The striped part represents average Precision and Recall on the whole networks.

is more than doubled, and scales better when network size increases, both for CNET and DBNs. The sensitivity of CNET to a particular external stimulus is even more evident here: in the presence of a sinusoidal signal, CNET exhibits an average Recall of 0.80 on networks of 10 genes and of 0.65 on network of 20 genes.

Results of the CNET algorithm on the real dataset are shown in Figure 4.9. On the left side of the figure known relations between genes are plotted, while on the right side the reconstructed network is represented: four of the original edges, depicted in bold, were correctly identified, and two other edges, $CCNE1 \rightarrow CDC2$ and $PCNA \rightarrow CDC2$, were identified reversed, thus leading to Precision and Recall of (0.36,0.44) on the oriented network and (0.54,0.67) on the unoriented network, which are in line with the results on simulated data. Moreover, for genes CCNA2, CCNB1, CCNE1, CDKN3 and RFC4 no regulators are known among the nine genes; if we assess the performance of CNET in inferring regulators only of the genes for whom a regulator is known, we obtain a Precision of 0.80. Precision and Recall of DBNs on the same dataset are (0.19,0.33) on the oriented network and (0.31,0.56) on the unoriented network.

4.2.4 Conclusions

In this section we presented a novel algorithm, CNET, for Reverse Engineering of Gene Regulatory Networks from time series data of DNA microarray experiments. The major point of innovation of our algorithm is the heuristic way in which it handles noise, variable regulatory delays and network complexity: a similar approach for tolerating quantization noise is presented in [51] and the completeness term, exploited to reduce the number of regulators given the predictive accuracy of a causal relation, is closely related to the Bayes Information

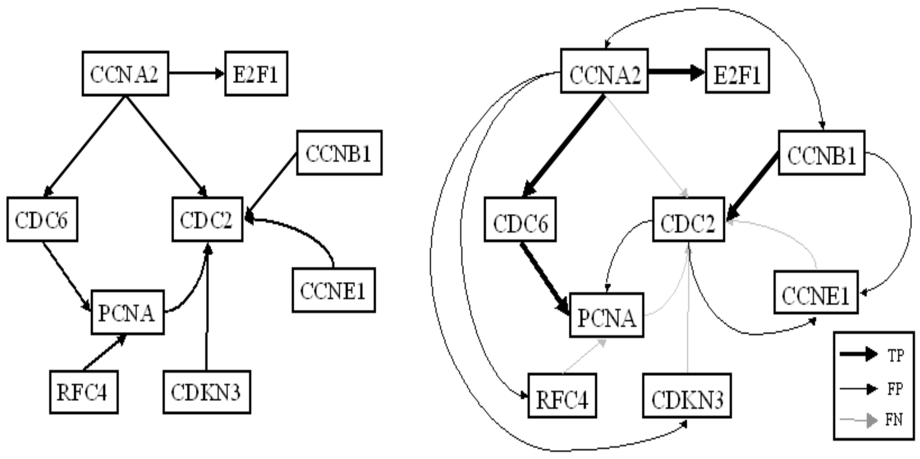


Figure 4.9: Network of known interactions (left) and reconstructed network (right) for the real dataset of 9 human genes related to cell cycle.

Criterion present in most inference algorithms for Bayesian Networks [24]. However, their contemporary usage, together with the novel shape term to capture variable regulatory delays, is the strongest point of innovation of our approach.

Tests on simulated data showed that CNET outperforms the REVEAL algorithm, of which it can be considered an improvement, and exhibits performance comparable to a state-of-art approach, Dynamic Bayesian Networks, in reconstructing both entire networks and subsets of them close to external stimulation. Similar performance was exhibited on a real dataset.

Both CNET and DBNs exhibit higher performance in the presence of an external stimulus and on the subset of edges directly outgoing from the stimulated node. This behaviour suggests that performance of the algorithms can vary across different portions of the networks. In the next chapter, we further explore CNET and DBNs behaviour on the simulated data set, to study how the performance is related to the local structure and the topological properties of the network to be inferred.

Chapter 5

Performance of the algorithms and topological properties of the networks

Recent assessments papers of the performance of different algorithms on realistically simulated data [6, 13, 71], reviewed in Section 2.4, showed that overall performance of Reverse Engineering algorithms is on average quite poor: commonly adopted performance measures, such as Precision and Recall, rarely rise beyond 0.5 in a 0 to 1 scale. Some evidence is emerging that performance is related to the design of the particular microarray experiment, for example to the presence or absence of an external stimulus, and depends on local network structure [5, 62, 65].

In this chapter, we systematically investigate these hypotheses by applying two Reverse Engineering algorithms, Dynamic Bayesian Networks (DBNs, [24]) and CNET (section 4.2), on the simulated dataset of externally stimulated data described in Section 3.1.1. The two algorithms have been chosen because both of them analyze time series data, infer oriented graphs and represent two different approaches to reverse engineering: a model based method applied to continuous expression data and an information theory based method applied to quantized data.

Networks reconstructed by the two algorithms are inspected, searching for regions of the networks correctly identified by both algorithms or, on the opposite, missed by both. It has already been shown in the previous chapter (Section 4.2) that the performance increases in the presence of an external stimulus and depends on the distance from the stimulation point in the regulatory network; in this chapter, we show empirical evidence of relations between the performance of the algorithms and some topological properties of the networks, such as the in-degree of nodes and the presence or absence of particular network motifs and of alternative paths for the regulatory signal.

5.1 Methods

For the analysis of this section, we exploit Dynamic Bayesian Networks (DBNs) and CNET, with the particular implementations described in section (4.2). We run our experiments on the simulated dataset of externally stimulated time course data presented in Section 3.1.1: we consider 60 networks of 10 genes and 60 networks of 20 genes. For each network, we take into account three time series of 50 samples, obtained stimulating the network with a sinusoid, a ramp and a step signal respectively. Networks are stimulated at the node with the highest out degree. Since we want to focus on the effects of the propagation of the stimulus through the network, we do not consider datasets in natural response for this analysis.

Overall performance of the algorithms will be measured in terms of true positives, false positives and false negatives, as defined in Section 2.3. Moreover, to search regions of the networks that are more easily detected by both Reverse Engineering methods, regardless of the stimulus adopted to excite the system, we developed a performance measure called S-score: S-score counts, for every edge in the simulated network, the number of times it was identified by one of the two algorithms in one of the different datasets, and thus ranges from 0 (never identified) to 6 (identified by both algorithms for the three data sets).

Edges are characterized in terms of:

- distance from the stimulated node,
- structure of the subnetworks containing them,
- topological properties of their source and destination nodes, such as in and out degree, clustering coefficient, cyclic coefficient and betweenness centrality,
- presence of alternative paths from their source to their destination.

For a survey and rigorous mathematical definition of topological properties of the nodes in a network, please refer to [14].

5.2 Results

Results from Section 4.2 show that the performance of the two algorithm is higher in the presence of an external stimulus, and that the sinusoid is the most informative between the three possible stimulating signals. Moreover, performance of both algorithms is almost doubled on the edges directly outgoing from the stimulated node. This seems to suggest that the behaviour of the two algorithms can be similar on particular regions of the network.

To further explore this hypothesis, we computed the six sets of true positives, false positives and false negatives of the two algorithms and the relative size of the intersection of the same set between the two algorithms. Results are shown in Figure 5.1.

One can observe that the intersection of true positives and false positives are low, *i.e.* that the two algorithms have different strategies of identifying correct and incorrect connections. The sets of false negatives, on the other

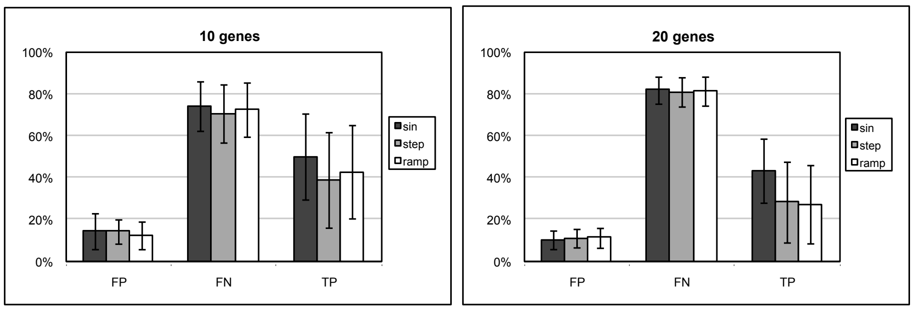


Figure 5.1: Relative size of the intersection between false positives, false negatives and true positives returned by CNET and DBNs, for the three externally stimulated data sets.

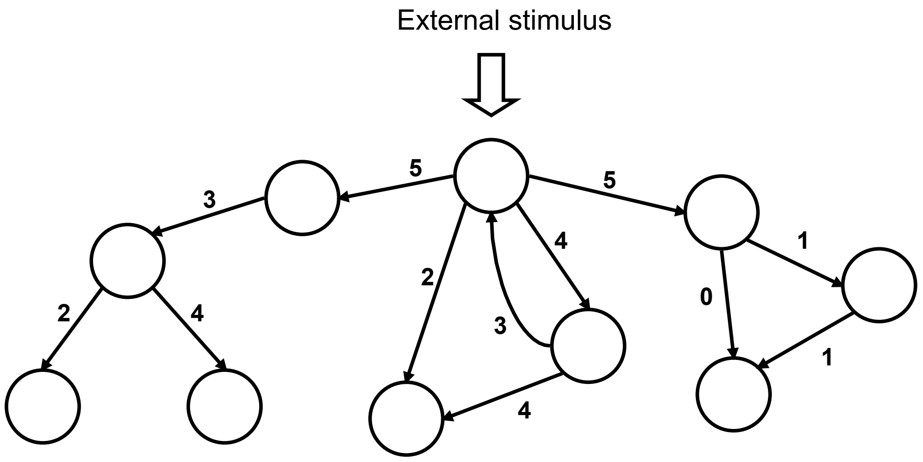


Figure 5.2: A 10 genes simulated network, with S-score reported on the edges.

hand, strongly overlap, suggesting that the portions of the network which are difficult to be identified by the two algorithms are mostly the same.

To locate these portions and to explore their properties, S-score was evaluated for every edge in the networks, as the number of times it was identified by the two algorithms for the three externally stimulated datasets. The average S-score is 2.00 for the networks of 10 genes and 1.34 for the networks of 20 genes, thus low, as expected. An example of a 10 gene network, with S-score reported on the edges, is shown in Figure 5.2.

As one can observe from the Figure, difficulty in inferring an edge is directly proportional to the distance from the stimulus, but it seems also to depend on the particular substructure in which the edge is contained: the edges on the left branch, the distance being equal, are recognized with a higher accuracy than the edges on the right one.

We indeed discovered that there are network substructures which are recognized with a significantly higher accuracy: for example, on circles of genes regulating each ohter in a chain, like the one on the left side of Figure 5.3, edges exhibit a mean S-score significantly higher than the average (3.5). The simple

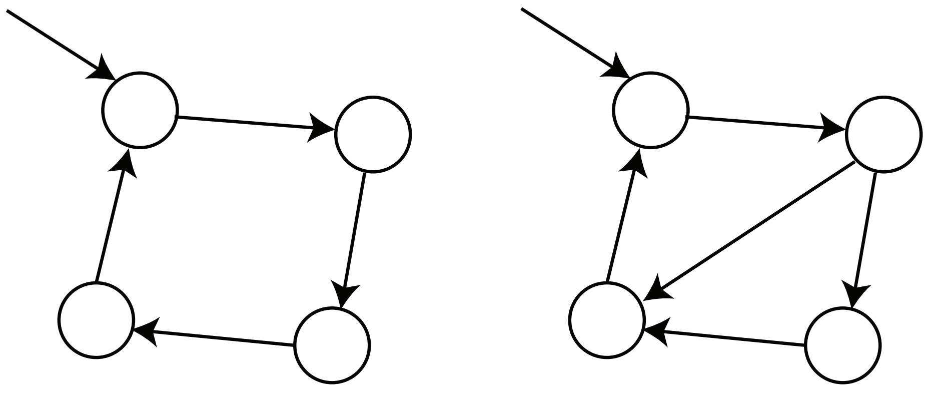


Figure 5.3: Edges in subnetworks like the one on the left exhibit an S-score significantly higher than the average. The same is not true for subnetworks like the one on the right.

addition of a diagonal edge, as in the subnetwork of Figure 5.3, is sufficent for the S-score to drop down to the average level.

Trying to motivate this behaviour, we further searched for possible correlations between S-score and the topological properties of source and destination nodes of each edge. The topological properties we considered are:

- **in and out degree**, *i.e.* the number of edges incoming to and outgoing from a node,
- **clustering coefficient**, *i.e.* the ratio between the number of edges linking the neoghbours of a node and the total possible number of edges among them, and
- **cyclic coefficient**, *i.e.* the average of the inverse of the sizes of the smallest cycles formed by a node and its neighbours.

Moreover, we considered also the **betweenness centrality** of each edge, *i.e.* the ratio between the number of shortest paths between each pair of nodes that pass through the edge and the total number of shortest path between each pair.

Among all the aforementioned measures, we found significant correlation only between S-score of an edge and the in degree of its destination node: as reported in Figure 5.4, S-score is significantly higher on edges incoming to nodes with in degree equal to one rather than edges incoming to nodes with in degree greater than one.

Results are consistent with intuition, because single one-to-one relations are likely to be identified more easily than N-to-one combinations of effects. No significant differences in S-score were observed among edges with destination nodes having in-degree greater than one; the major distinction is thus between one-to-one and N-to-one relations.

Finally, we searched for a correlation between S-score on an edge and the presence or absence of an alternative directed path between the source and the destination of the edge. As depicted in Figure 5.5, an alternatve directed path for an edge $e_{ij} = (v_i, v_j)$ can be defined as a sequence of directed edges $e_{ik_1} \dots e_{k_m j}$ $(k_1, \dots, k_m \neq i, j)$ originating from node v_i and ending on node v_j.

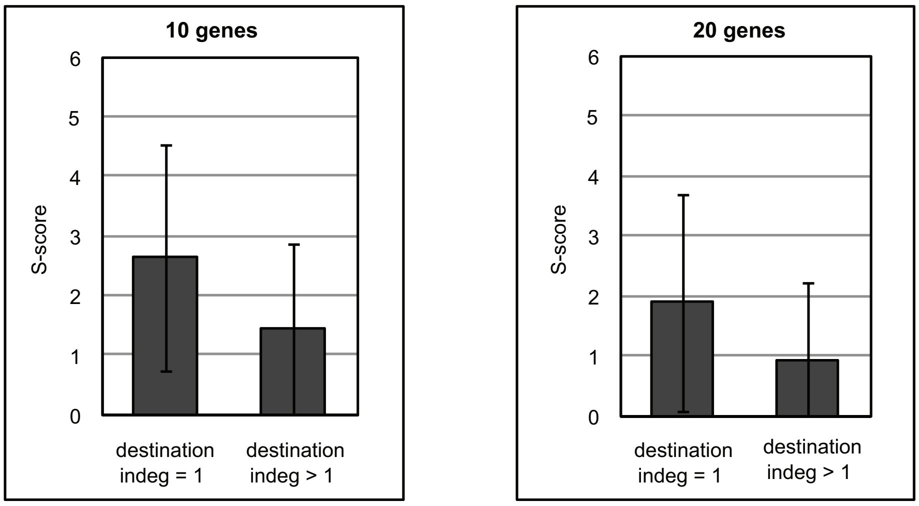


Figure 5.4: Average S-score for edges incoming on nodes with in degree equal to one and greater than one, for networks of 10 genes (left) and 20 genes (right).

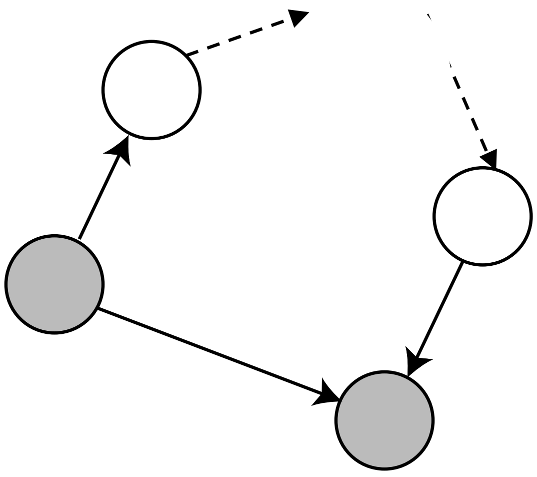


Figure 5.5: Graphical representation of an alternative directed path between the two shaded nodes.

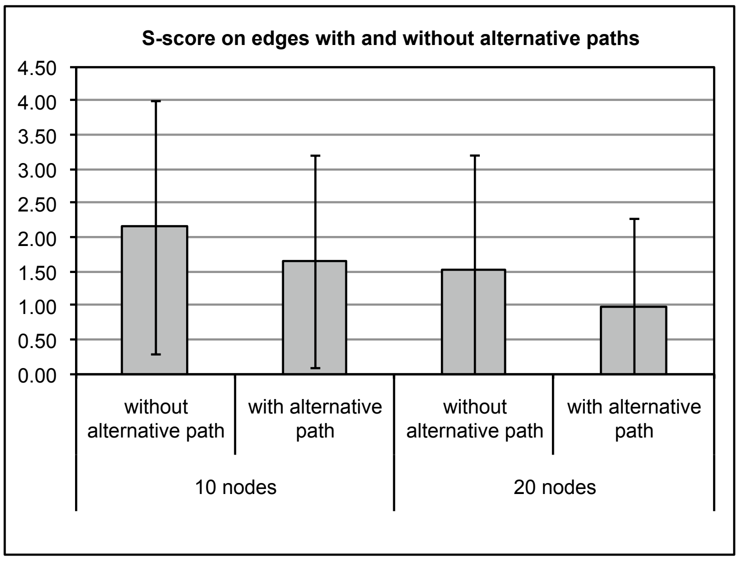


Figure 5.6: Average S-score on edges with and without an alternative path, for networks of 10 genes (left) and 20 genes (right).

We found that S-score on edges without an alternative path is indeed significantly higher than on edges with one or more alternative path, as shown in Figure 5.6. Such a result is consistent with intuition: the presence of an alternative path for the regulatory signal can create strange precedence patterns in the output of the system and confound a Reverse Engineering algorithm. No significant differences were observed between alternative paths of different lenghts.

5.3 Conclusions

Reverse Engineering methods tend to exhibit poor average performance on real or realistically stimulated data (see Section 2.4). Here, we presented an analysis of the behaviour of two Reverse Engineering algorithms, DBNs and CNET, over a dataset of 120 simulated gene networks, to understand how their performance is related to presence/absence of an external stimulus, distance from the simulated node and topological properties of the networks.

In general, the presence of an external stimulus improves the performance of both methods: significant differences were observed for systems perturbed with a sinusoidal signal in case of CNET (for both 10 and 20 genes networks) and with all the stimuli in case of DBNs (only for 20 genes networks). Moreover, the performance of both algorithms on the edges directly outgoing from the stimulated node is almost doubled, with respect to the average performance on the whole networks. This result seems to suggest the effectiveness of multi-stimulus experiment design, with consecutive stimulations of different nodes in the network.

The overlap between false positives, false negatives and true positives returned by the two algorithms shows that the algorithms make different false positives errors and similar false negatives errors, suggesting that there are parts of the networks difficult to be identified from available data, independently of the method used to reverse engineer the system.

A deeper analysis of simulated networks, through the use of an opportunely defined measure, S-score, outlined that the distance from the stimulus is not the only discriminant for performance variations: the substructure in which a node is contained has also an effect on the performance of Reverse Engineering algorithms that try to infer it. Searching for motivations of this behaviour, a further analysis revealed that edges whose destination node has in-degree equal to one and edges without an alternative path are significantly easy to be identified than the others. These results are consistent with intuition, but as far as we know this is the first systematic study that proved their empirical evidence.

Chapter 6

Inferring systems of equations

In this chapter, we present an optimization approach to the problem of reverse engineering of systems of sigmoidal differential equations from time course measurements of gene expression. If the system consists of n equations, each with n free parameters, the search space of an optimization algorithms is thus $\mathbb{R}^{n \times n}$ and the cost function to be minimized can be a measure of the error between real and estimated gene expression profiles.

The problem of inferring Gene Regulatory Networks (GRNs) from gene expression profiles using optimization techniques has proved to be difficult even when dealing with very small networks (5-10 genes) [72]. In Section 6.1, we address the issue of the difficulty of inferring GRNs by performing an analysis based on the notion of fitness-distance correlation (FDC) [32, 33, 66]. We then exploit the results obtained from this analysis to design a novel mixed discrete/continuous optimization algorithm for the inference of GRNs, described in Section 6.2. The algorithm is tested on a dataset of simulated gene profiles and its results are compared with the state-of-the-art in Section 6.2.3.

6.1 Fitness-distance correlation analysis

In this section, we study the difficulty of inferring the parameters of a system of nonlinear differential equations used to model a GRN, starting from time course observations of gene expression profiles. To model regulatory interactions, we chose a system of nonlinear sigmoidal differential equations, also known as Dynamic Recurrent Neural Networks (RNNs) [39], which has the form

$$\frac{dx_i}{dt} = \frac{k_1}{1 + \exp\left(-\sum_{j=1\ldots n} w_{ij} x_j + b\right)} - k_2 x_i\,, \qquad i = 1 \ldots n \tag{6.1}$$

where n is the number of genes in the system, x_i is the rate of expression of gene i, w_{ij} represents the relative effect of gene j on gene i ($1 \leq i, j \leq n$), b is a bias coefficient, k_1 is the maximal rate of expression and k_2 is the degradation rate. For our analysis, we set for simplicity $b = 0$, $k_1 = 1$ and $k_2 = 1$. The search space for an optimization algorithm is then formed by the matrix W of coefficients w_{ij}.

We decided not to approximate derivatives with finite differences, because such an approach implies estimating the derivatives from temporal data and thus both amplifies the effects of noise and requires a large amount of data points. On the contrary, we generate temporal profiles with numerical integration of the whole system, with a Runge-Kutta-Fehlberg method with adaptive step size control [23]. For further details on this type of model, motivations for its implementation and examples of its use in the literature, please refer to Section 2.1.4.

As a measure of the deviation of time profiles generated by an inferred network from the real profiles, we adopted the Relative Squared Error (RSE), which is defined as

$$RSE = \frac{1}{Tn} \sum_{t=1}^{T} \sum_{i=1}^{n} \frac{[\hat{x}_i(t) - x_i(t)]^2}{x_i^2(t)} ,$$

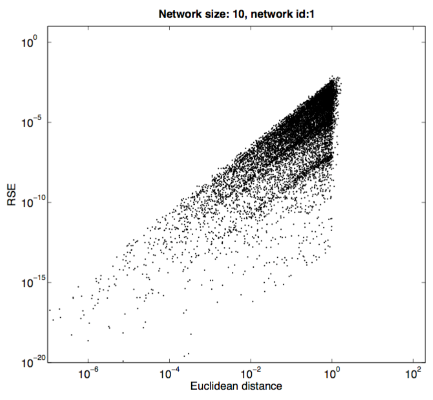
where n is the number of genes, T is the number of time samples, $x_i(t)$ is the real value for gene i at time t and $\hat{x}_i(t)$ is the estimated value for the same sample.

As a first contribution, we present an analysis of the error surface generated by the combination RNN-RSE (Section 6.1.1). The main result of this analysis is that the RNN-RSE error surface has a strong positive fitness-distance correlation; however, the data also show the existence of many local optima of extreme depth, which seems to be the main cause for the poor performance shown by optimization algorithms on this problem. A second contribution is the quantification of the effect that a priori information on the target's GRN structure has on the fitness landscape (Section 6.1.2). The final contribution is the analysis of the behavior of two state-of-the-art continuous optimization algorithms, NEWUOA [56] and CMA-ES [28], on the problem with and without *a priori* network structure information (Section 6.1.3). The results obtained from this analysis constitute strong evidence in favor of inference approaches in which the spaces of network topologies and of network parameters are decoupled.

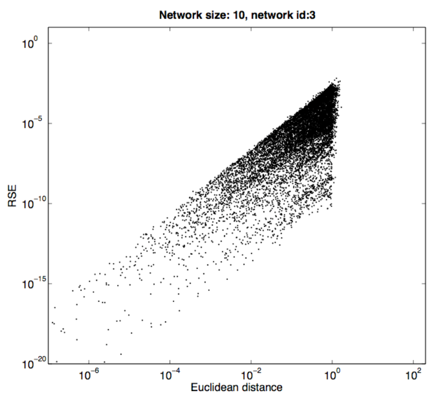
6.1.1 Fitness-distance correlation around the optimum

To investigate the structure of the fitness landscape of our optimization problem, we performed a fitness-distance correlation analysis [33, 32]: we randomly sampled interesting areas of the search space and studied, for sampled solutions, the distribution of fitness values versus distance from the optimal solution. In our case, a fitness value is considered to be better if the solution associated with it has a lower value of the fitness function, the RSE between real and estimated gene profiles. As distance measure between candidate solutions and the optimal solution, we used the Euclidean distance.

As experimental data, we exploited the natural response time course dataset described in Section 3.1.1: the expression of each gene is initialized uniformly at random and the system is let free to evolve to a steady state. 50 logarithmically spaced samples of gene expression are collected for each gene. The analyses reported in this section are carried out on gene networks of size 10, in line with



(a) Example of widely distributed samples (network 1)



(b) Example of more narrowly distributed samples (network 3)

Figure 6.1: RSE *vs* Euclidean distance of 10000 log-uniform perturbations of the elements of the optimal system matrix, for two networks of 10 genes.

the experimental results from the state-of-the-art approaches to this problem [72, 81, 80].

As a first step of our analysis, we explored relations between fitness and distance among a set of random perturbations of the optimal solution: each element of the optimal matrix was perturbed with the addition of a log-uniformly distributed random variable (*i.e.* a random variable uniformly distributed in logarithmic scale) in the interval $[10^{-a}, 10^{-0}]$, where a was tuned to account for different problem sizes, to increase the density of the samples near the optimum. Results for 10000 iterations of the perturbation procedure on two networks of 10 genes are shown in Figure 6.1.

As it can be seen from the figure, there is a strong correlation between Euclidean distance and RSE, because samples distribute along a band with positive slope, but the band is rather wide (approximately 10 orders of magnitude of RSE for Figure 6.1a and 6 orders for Figure 6.1b), thus leading to an average correlation coefficient of 0.471. We formulate the hypothesis that the difficulty in solving the particular problem instance is closely related to the width of the band in the fitness-distance plot. A large band width, in fact, suggests the presence of extremely deep valleys in the fitness landscape, in which a general purpose continuous optimization algorithm can keep decreasing the RSE without getting closer to the optimal solution. The perturbation procedure was repeated for 20 different problem instances of 10 genes. For the majority of them (17 out of 20) the band in the RSE *vs* distance plot exhibits a width close to the one of network 3 (Figure 6.1b), and for the remaining instances the width is larger, close to the one of network 1 (Figure 6.1a). Therefore, we decided to use network 1 and 3 throughout the chapter as two representative examples of problem instances, to validate empirically our hypothesis.

6.1.2 Separation between structure and parameter values

As a second step, we decided to investigate the relation between the features of the search space and the structure of the networks (*i.e.* the pattern of zero

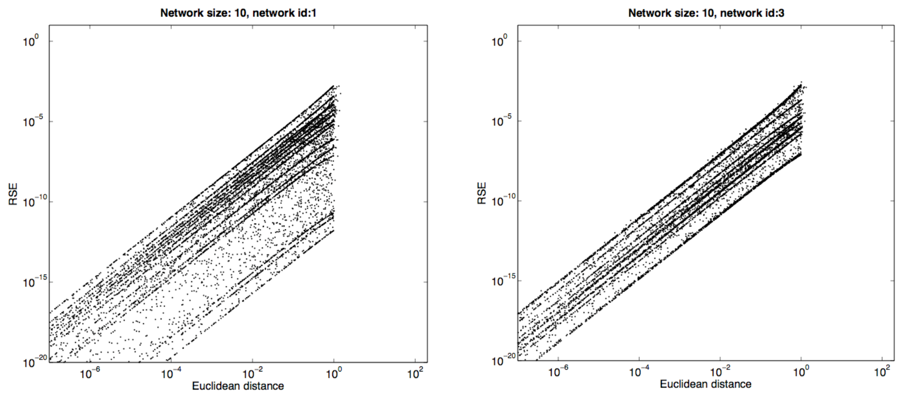


Figure 6.2: RSE *vs* Euclidean distance of 10000 log-uniform perturbations of nonzero elements of the optimal system matrix, for two networks of 10 genes. Plots are cut to keep the same scale adopted in the other figures; the diagonal lines spread with the same behavior down to 10^{-15} for Euclidean distance and 10^{-35} for RSE.

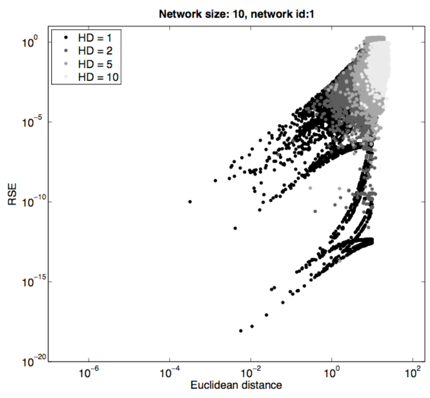
and nonzero elements in the weight matrix): gene networks are largely sparse, thus the number of parameters to be fine tuned by an optimization procedure is small with respect to the number of variables in the search space. We wanted to understand how much the search space is affected by information on the network structure.

To this end, we perturbed only nonzero elements of the optimal solution for each problem instance, fixing to zero the other elements. As before, perturbations were obtained with the addition of a log-uniformly distributed random variable. RSE vs Euclidean distance of 10000 perturbations for network 1 and 3 are shown in Figure 6.2.

Even though the average correlation coefficient is 0.424, thus slightly lower than the one form the previous step, fitness-distance plots tend to be more structured: as it is clear from the figure, most of the samples lie on straight lines parallel to the bands of the previous experiment, and the vertical span of the lines reflects the width of the bands. A deeper analysis revealed that each line corresponds to a single nonzero element of the weight matrix, thus suggesting that some variables may be optimized independently, once one knows which elements are nonzero. Such a hypothesis was not necessarily evident from the mathematical description of the system and should be explored in future works.

We then decided to further explore the shape of the fitness landscape in regions close in structure to the global optimum; for this purpose, we exploited the concept of *Hamming distance* between two connectivity matrices[1], and we randomly sampled Boolean matrices at Hamming distance 1, 2, 5 and 10 from the global optimum. We then kept original values for elements that are nonzero in both matrices, the original one and the sampled one, and set new values for the other nonzero elements, drawing them uniformly at random from the interval $[-10, 10]$. 10000 samples for each value of Hamming distance are shown in Figure 6.3, where lighter gray corresponds to higher Hamming distance, for

[1]The Hamming distance between two binary strings of the same length is the number of bits that differ between the two strings. In our case, connectivity matrices are binary matrices with 1 on element (i, j) if there is an edge from node j to node i, and a 0 otherwise; Hamming distance can thus be properly computed between two connectivity matrices.



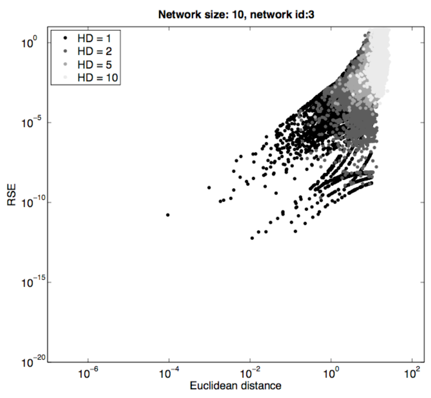


Figure 6.3: RSE *vs* Euclidean distance of 10000 log-uniform perturbations of the optimal system matrix at Hamming distance 1, 2, 5, 10, for two networks of 10 genes.

networks 1 and 3.

From the figure, it is evident that at higher Hamming distances there is no particular correlation between fitness and distance, but when the Hamming distance decreases the fitness vs. distance plot becomes more and more organized, approaching the global shape of the bars from Figure 6.1. Indeed, average correlation coefficients are 0.219, 0.196, 0.185 and 0.177 for networks at Hamming distance 1, 2, 5 and 10, respectively. At Hamming distance 1, samples tend to lie on curved lines and the structure of the plots become closer to the one from Figure 6.2.

This latter analysis outlines that portions of the search space which correspond to networks structurally close to the optimum (i.e., at a low Hamming distance) present more organization in the fitness landscape and, in general, better fitness values; these regions can thus serve as a local basin of attraction for an algorithm which searches in the discrete space of network structures. To test the quality of a particular network structure, a second algorithm can be alternated to the first, to optimize continuous nonzero values of the network; for the second algorithm, the probability of finding the optimal solution should increase as the network structure becomes closer to the optimal structure.

6.1.3 Performance of two state-of-the-art optimization algorithms

The analysis presented above gives an overall picture of the fitness-distance relationship in the search space. In addition, it is of interest to study the behaviour of some specific algorithms in the same space. The question we want to address is whether an algorithm is capable of inferring the structure of the target network using only the information provided by the RSE measure. If that is not the case, a second experiment consists in measuring the performance of the algorithm when the optimal network topology is known *a priori*. For our experiments, we use NEWUOA [56] and CMA-ES [28], two approaches that are considered to be state-of-the-art for continuous derivative-free optimization [4]. The two algorithms are described in detail in what follows.

NEWUOA

NEWUOA is a software for unconstrained continuous optimization in many dimensions that does not need information about the derivatives of the objective function $f : \mathbb{R}^n \rightarrow \mathbb{R}$ it is applied to. At each iteration, NEWUOA creates a quadratic model that interpolates k values of the objective function which is used in a trust-region procedure [11] to update the variables. The main advantage of NEWUOA is that it can be used to solve large scale optimization problems thanks to the reduced number of interpolation points it needs to build the quadratic model (usually $k = 2n + 1$, where n is the number of variables to optimize, is recommended). By definition, trust-region methods search locally, which means that they may converge to some local optimum in the case the objective function is multimodal. For this reason, we used NEWUOA with multiple restarts, so as to explore different regions of the search space in order to reduce the chances of converging to low quality local optima. In our setting, NEWUOA is restarted from a new initial solution after it has reached a maximum number of function evaluations, or when the final radius of the trust region reaches a certain threshold.

CMA-ES

The CMA-ES (Covariance Matrix Adaptation - Evolution Strategy) algorithm belongs to the class of population-based optimization algorithms called Evolution Strategies. In this kind of algorithms, a population of solutions is sampled from a multivariate normal distribution, with mean m and covariance matrix C, for a certain number of generations; at each generation, the best individuals are selected and used to adapt the sampling mechanism, in order to select potentially better solutions. In CMA-ES, the covariance matrix is dynamically adapted with three concurring strategies: (i) reproduce the best steps in the search space from the current generation, (ii) reproduce the set of moves from the previous generations and (iii) obtain uncorrelated steps with step size control. The effect of the adaptation is to skew the sampling distribution, so to obtain the highest variance along the steepest direction of the search space, remaining robust to badly scaled problems and avoiding premature convergence. Because of this last property, we do not fix a maximum number of function evaluations before stopping the search and restarting CMA-ES, relying only on its internal stopping criteria (if global standard deviation increases more than a constant factor, if the condition number of C is too large or if differences between the best values in subsequent generations are smaller than a threshold).

In Tables 6.1 and 6.2 we show the parameters used in our experiments. These parameters were chosen after an initial non-exhaustive experimentation phase.

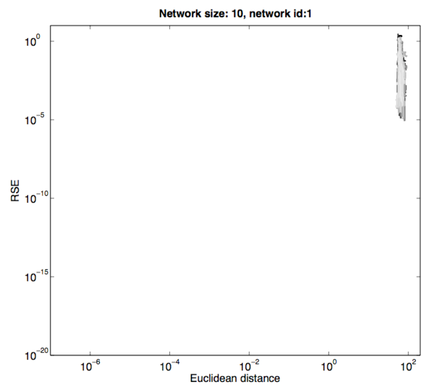
The results obtained from running both NEWUOA with multiple restarts and CMA-ES without any *a priori* information about the correct topology of the target GRN are shown in Figure 6.4. Each shade of gray represents a run of the algorithm; for each run, we generate different temporal profiles from the same network, initializing each gene uniformly at random and then letting the network evolve, so to test the robustness of the two algorithms across different simulated microarray experiments. Although the two algorithms are capable of

Table 6.1: Parameters used with NEWUOA with multiple restarts

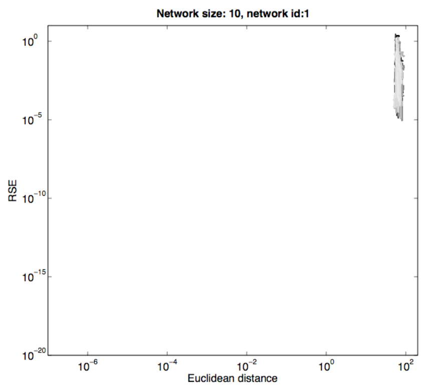
Parameter	Value
Initial trust region radius	0.2
Final trust region radius	1×10^{-10}
Number of interpolation points	$k = 2n+1$, where n is the number of variables to optimize
Maximum number of function evaluations per NEWUOA run	2×10^4
Maximum total number of function evaluations	2×10^5, with structure information
	1×10^6, without structure information
Number of independent runs	20

Table 6.2: Parameters used with CMA-ES

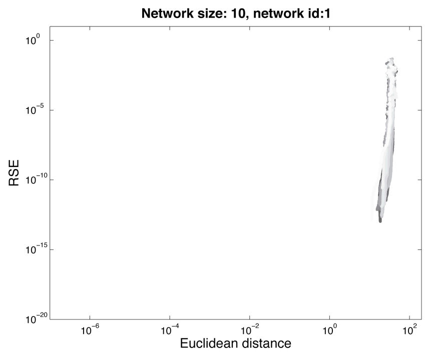
Parameter	Value
Initial search point	$\mathbf{0}^T$
Initial standard deviation	$\mathbf{3}^T$
Population size	$\lambda = 2n + 1$, where n is the number of variables to optimize
Maximum increasing factor for standard deviation	3
Maximum condition number for C	10^{14}
Minimum value for differences between best consequent solutions	10^{-17}
Maximum total number of function evaluations	2×10^5, with structure information
	1×10^6, without structure information
Number of independent runs	20



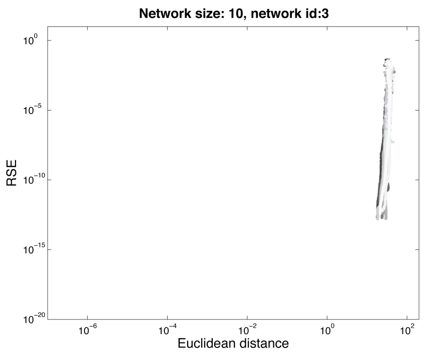
(a) NEWUOA on network 1



(b) NEWUOA on network 3



(c) CMA-ES on network 1



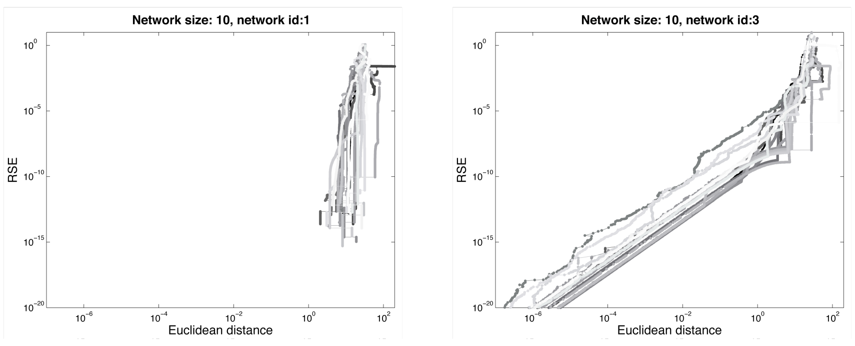
(d) CMA-ES on network 3

Figure 6.4: The progress of NEWUOA with multiple restarts and CMA-ES on two 10-gene-network inference problems. Each shade of gray represents a run of the algorithm. The plots shown correspond to the case in which no *a priori* information about the correct topology of the target GRN is provided to the algorithm.

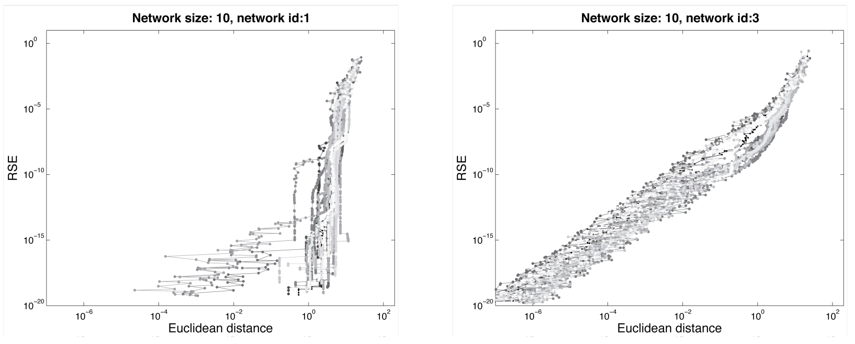
making progress in terms of the value of the objective function (they descend from a value in the order of 10 to a value in the order of 10^{-5} for NEWUOA and 10^{-13} for CMA-ES), they do not make much progress towards the actual target GRN. This can be seen by the (almost) vertical lines that appear on the upper right corner of the plots in Figure 6.4.

In Figure 6.5, we show the results obtained after running NEWUOA with multiple restarts and CMA-ES when the correct topology of the target GRN is provided to the algorithms, which is equivalent to reducing the size of the search space so that only nonzero entries are optimized. As before, each shade of gray represents a run of the algorithm. In this case, the behaviour of the two algorithms depends on the target network. With network 1, the two algorithms move towards the optimal solution while improving the value of the RSE over several orders of magnitude. However, in the vast majority of cases, the algorithms cannot find solutions that are closer than a distance of 10^{-1} to the optimal solution. In contrast, with network 3, the algorithms are capable to find the optimal solution in each run.

The results presented above, together with those of the analysis based on structure perturbations, constitute strong evidence in favor of optimization al-



(a) NEWUOA on network 1 (b) NEWUOA on network 3



(c) CMA-ES on network 1 (d) CMA-ES on network 3

Figure 6.5: The progress of NEWUOA with multiple restarts and CMA-ES on two 10-gene-network inference problems. Each shade of gray represents a run of the algorithm. The plots shown correspond to the case in which the correct topology of the target GRN is provided to the algorithm.

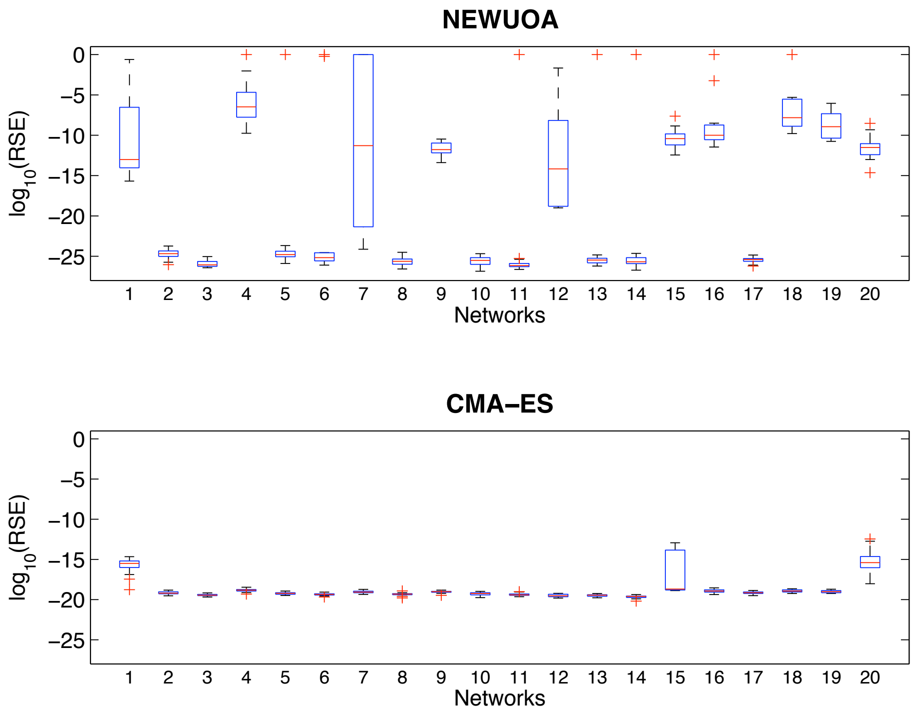


Figure 6.6: Boxplots of RSE reached in 20 runs of NEWUOA and CMA-ES on 20 network instances of 10 genes, with full information on the topology of the networks; the maximum number of function evalutations for both algorithms is set to 4×10^4.

gorithms that explicitly intertwine a network structure search phase with a network's parameters search phase. A reduction in the distance from the optimal network topology allows a continuous optimization algorithm to make more progress toward the truly optimal solution.

With this perspective, we run an experiment to test the ability of both NEWUOA and CMA-ES in returning an accurate and stable RSE value, when provided with the correct structure and when given a limited amount of function evaluations (4×10^4). The most stable between the two algorithms is the most suitable for being used as a parameter search algorithm and, implicitly, as a reliable evaluator of the goodness of a proposed structure, once we use an additional algorithm for searching the discrete space of network structures.

In Figure 6.6, we show the boxplots of RSE values achieved by 20 runs of both algorithms on 20 test networks of 10 genes. Boxplots show that CMA-ES obtains RSE values smaller than 10^{-15} on almost every run. NEWUOA, on the other hand, gets better results for some network structures, but for several others it is very poorly performing when compared to the much more stable performance of CMA-ES; the latter is thus a more reliable estimator of the goodness of the correct structure.

To further explore CMA-ES abilities, we run it on the same set of problems, with the same maximum number of function evaluations (4×10^4) but without restarting the algorithm if one of its internal stopping conditions is encountered first. Results are shown in Figure 6.7: performance does not deviate much in

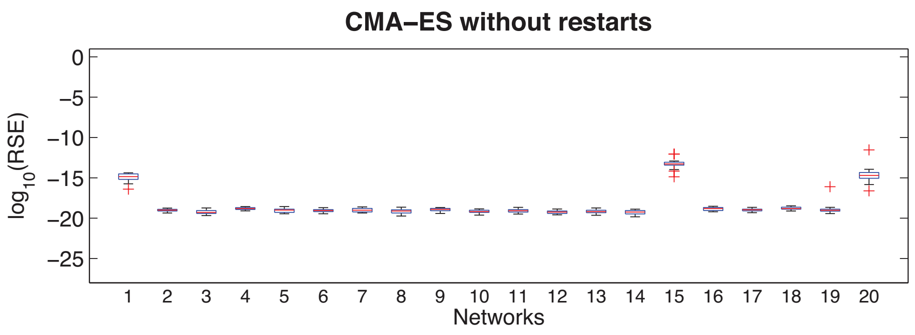


Figure 6.7: Boxplot of RSE reached in 20 runs of CMA-ES on 20 network instances of 10 genes, with full information on the topology of the networks; the maximum number of function evalutations is set to 4×10^4, but no restart is allowed if CMA-ES encounters one of its internal stopping conditions before reaching the limit of function evaluations.

terms of stability from the case of CMA-ES with restarts, but the maximum reached number of function evaluations before stopping is always lower than 4×10^4, with a median of approximately 1.3×10^4. Thus, without restarts CMA-ES exhibits roughly the same performance in terms of RSE stability, but with a time gain of a factor of 3.

6.1.4 Conclusions

In this section, we presented a study of the fitness landscape for the problem of Gene Regulatory Network inference, when Dynamic Recurrent Neural Networks are adopted as a model for gene regulation and Relative Squared Error is chosen as a fitness function. As far as we know, this is the first study on fitness-distance correlation analysis for the problem of gene regulatory networks inference.

The study consists in a fitness-distance correlation analysis of different random samplings around the problem's optimal solution, which is in the form of a weight matrix W. The optimal matrix was first perturbed globally, then only on its nonzero elements and at fixed Hamming distance. Results show that the error surface has a strong positive fitness-distance correlation, but they also reveal the presence of extremely deep valleys in the fitness landscape, which are responsible for the poor performance of optimization algorithms not designed explicitly for this problem.

The network structure perturbation analysis highlights that: (i) RSE alone is not sufficient to guide a search algorithm towards regions of the search space close to the global optimum, (ii) even if information about the optimal network structure is provided to the algorithm, convergence to the global optimum is not guaranteed because the fitness landscape presents many deep local optima, and (iii) the closer a network structure is to the one of the optimal solution, the higher the chances are that an algorithm for parameter optimization converges to the optimum. This last fact seems to be due to the higher level of organization in the fitness landscape in the proximity of the optimal structure.

Because of these observations, we conclude that a two-phase algorithm, which alternates between a search step in the discrete space of network structures and a search step in the continuous space of nonzero system parameters, has the potential of reaching high-quality solutions. Research in this direction has already been done, for example in [81, 58, 36], but no analysis of the underlying fitness landscape had been performed before.

6.2 Mixed discrete and continuous optimization algorithm

The analysis of the fitness lanscape described in the previous section provided evidence in favour of a decomposition of the problem of Gene Regulatory Network inference into two interconnected subproblems: a discrete search of the optimal network structure in the space of connectivity matrices and a continuous search of the optimal network parameters in $\mathbb{R}^m$, where m is the number of edges in the network.

In Section 6.2.1, we present a basic implementation of such an approach, exploiting an Iterative Best Improvement Local Search strategy with Multiple Restarts for the discrete search step, and CMA-ES for the continuous search step. In Section 6.2.2, we describe two communication strategies between the two optimization layers that enhance the performance of the basic algorithm. In Section 6.2.3 we report experimental results of the application of the mixed optimization algorithm to a set of simulated data and in Section 6.2.4 we present some conclusions and future directions.

6.2.1 Basic Algorithm

The Iterative Best Improvement Local Search strategy belongs to the class of Stochastic Local Search (SLS) algorithms, in which a complex or highly dimensional search space is locally explored, starting from a random position in the space and iteratively moving in the direction that locally minimizes (or maximizes) a scoring function. From the current position, SLS algorithms iteratively generate a set of *neighbours*, *i.e.* solutions that do not differ much from the current one, and replace the current solution with one of the neighbours, based on some acceptance criterion. The process is iterated until no further improvement of the current solution can be found; in this case, the current solution is called a *local optimum.* The various SLS algorithms differ in the strategies to explore the neighbourhood, to choose the next solution and to avoid local optima. In the Iterative Best Improvement Local Search strategy, the current solution is iteratively replaced with the best improving solution from its neighbourhood and the process is repeated until a local optimum is reached. Restaring multiple times the algorithm from different random solutions guarantees that the algorithm eventually converges to the *global optimum*, *i.e.* the best solution in the search space.

In our case, the discrete search space consists of a subset of all squared boolean connectivity matrices of size $n \times n$, where n is the number of genes and the element (i, j) of the connectivity matrix is equal to one if gene j regulates gene i. Feasible solutions satisfy the following additional constraints:

- each row has at least one element equal to one, thus each gene has at least a regulator (and the number of nonzero elements is greater than or equal to n),

- the number of nonzero elements is smaller than or equal to $2n$, to force sparsity of the network,
- elements on the diagonal are zero, *i.e.* self-regulation is not allowed[2].

As neighbourhood for the local search, we chose all possible flips of one bit in the connectivity matrix, excluding the flips that lead to unfeasible solutions. Nonzero parameters of each candidate structure are optimized with CMA-ES and the RSE between real and estimated profiles is used as cost function for the structure. The maximum number of function evaluation for CMA-ES is set to 4×10^4, as suggested by the results reported in the previous section, but the algorithm is not restarted if, by means of its internal stopping criteria, it terminates the computation before reaching the limit on the number of function evaluations. The neighbour solution with the lower RSE value is chosen as origin of the new neighbourhood, and the process is iterated until no improving solution can be found in the current neighbourhood. The complete algorithm is then restarted a maximum of 10 times, to increase the possibility of exploring interesting regions of the search space.

6.2.2 Enhancements of the basic algorithm

In the basic version of our mixed algorithm the two optimization components, Iterated Best Improvement Local Search and CMA-ES, communicate with each other exchanging candidate structures and RSE values. In this section, we describe two techniques for increasing the information exchange between the two components and enhancing the overall algorithm behaviour.

The first technique consists in a feedback mechanism from the continuous to the discrete optimizer: when all solutions in the neighbourhood have been evaluated, some elements of the best neighbour solution could have been estimated by CMA-ES as close to zero, *i.e.* with absolute value below a fixed threshold. If that is the case, when the center of the next neighbourhood is chosen, these elements are set to zero, thus implementing a longer move towards more promising regions of the search space.

The second communication strategy is used during the evaluation of the neighborhood, to exploit information on locality and similarities between neighboring solutions in the continuous optimization procedure. Each neighbour differs from the current solution by just one bit, *i.e.* by the presence or absence of an edge in the corresponding network. Since the network is sparse, there can be regions of the network which are not affected by the edge modification, being upstream in the signal flow through the network. Continuous values for the parameters corresponding to edges in these regions do not need to be re-optimized by CMA-ES and are thus kept fixed. Moreover, for the parameters that need to be re-optimized, initial search values for CMA-ES are chosen equal to the one obtained when evaluating the center of the neighbourhood, under the hypothesis that optimal parameters values for two networks that differ by just one edge are close to each other. The motivation for this strategy is that CMA-ES, when

[2]This last constraint is introduced for simplicity, but can be relaxed without affecting the global behaviour of the algorithm.

used to optimize a smaller set of values and initialized with a good search point, tends to reach earlier its internal stopping criteria, thus converging faster.

The pseudocode of the complete algorithm is the following:

```
DISCRETE SEARCH($\mathbf{T}^{n \times m}$)
global_best ← 1
for $i \leftarrow 1$ to 10
     do
          Sample a feasible connectivity matrix $\mathbf{C}_{curr}^{n \times n}$
          $\mathbf{rows}^{n \times 1}$ ← Boolean vector, each value set to true
          local_best ← CONTINUOUS SEARCH($\mathbf{T}, \mathbf{C}_{curr}, \mathbf{rows}$)
          neighb_best ← local_best
          while improvement
               do
                    improvement ← false
                    for $i, j \leftarrow 1$ to $n$
                         do
                              $\mathbf{C}_{neighb} \leftarrow$ FLIP($\mathbf{C}_{curr}, i, j$)
                              if ISFEASIBLE($\mathbf{C}_{neighb}$)
                                 then
                                        $\mathbf{rows} \leftarrow$ PROPAGATE($\mathbf{C}_{curr}, i, j$)
                                        $cost_{neighb} \leftarrow$ CONTINUOUS SEARCH($\mathbf{T}, \mathbf{C}_{neighb}, \mathbf{rows}$)
                                        if $cost_{neighb} < neighb\_best$
                                           then
                                                  $neighb\_best \leftarrow cost_{neighb}$
                                                  $\mathbf{C}_{best\_neighb} \leftarrow \mathbf{C}_{neighb}$
                                                  improvement ← true
                                                  Set to zero the elements of $\mathbf{C}_{best\_neighb}$
                                                  identified as close to zero by the
                                                  continuous search algorithm
                    if improvement
                       then
                              local_best ← neighb_best
                              $\mathbf{C}_{curr} \leftarrow \mathbf{C}_{best\_neighb}$
          if local_best < global_best
             then
                    global_best ← local_best
                    $\mathbf{C}_{best} \leftarrow \mathbf{C}_{curr}$
return $\mathbf{C}_{best}$
```

```
CONTINUOUS SEARCH($\mathbf{T}^{n \times m}, \mathbf{C}^{n \times n}, \mathbf{rows}^{n \times 1}$)
problem_size ← number of nonzero elements of $\mathbf{C}[\mathbf{rows}, :]$
if first call of the procedure
   then x_start ← (problem_size) random values
   else  if the current bit flip is $1 \rightarrow 0$
             then x_start ← (problem_size − 1) results of the optimization of $\mathbf{C}_{curr}$
             else  x_start ← (problem_size) results of the optimization of $\mathbf{C}_{curr}$,
                   plus an additional random value.
cost ← CMA-ES(problem_size, x_start, $\mathbf{T}$)
return cost
```

The discrete optimization procedure receives as input the matrix of time course experiments $\mathbf{T}^{n\times m}$ and returns the best connectivity matrix $\mathbf{C}^{n\times n}_{best}$. In each of the 10 iterations from line 2, an initial solution is sampled at random and evaluated (lines 4−7). At each iteration, the process of neighbourhood generation (lines 13−16), neighbour evaluation (lines 17−25) and update of the current solution (lines 26−29) is repeated until a local optimum is reached. The procedure FLIP($\mathbf{C}, i, j$) flips the bit (i, j) of the matrix $\mathbf{C}$ to its opposite value, ISFEASIBLE($\mathbf{C}$) returns *true* if $\mathbf{C}$ does not contain a row full of zeros, has less than $2n$ nonzero elements and does not contain ones on the diagonal. PROPAGATE($\mathbf{C}, i, j$) propagates the effects of the bit flip (i, j) in the matrix $\mathbf{C}$: a bit flip consists in the addition or the removal of an edge in the connection graph, thus all edges downstream of the modification are affected and PROPAGATE($\mathbf{C}, i, j$) returns *true* for each row corresponding to the destination of an affected edge.

The continuous optimization procedure, on the other hand, is exploited to evaluate candidate network structures. When called for the first time, the procedure optimizes all the nonzero values of the connectivity matrix $\mathbf{C}$, starting from a complete random vector *x_start* and scoring its progress with the RSE between real time course data $\mathbf{T}$ and time course data generated with both the connectivity matrix $\mathbf{C}$ and the current values for the set of parameters to be optimized; the optimization is carried out through the CMA-ES optimization algorithm. All the subsequent calls of CONTINUOUS SEARCH need only to optimize nonzero elements of the connectivity matrix contained in the rows indicated by **rows** (line 1) and can exploit the parameters values optimized in the previous function call as starting point for the search (lines 4−7).

6.2.3 Results

In this section, we present experimental results of our mixed discrete and continuous algorithm on a simulated dataset of time course experiments. The dataset consists of networks of size 5, 8 and 10 genes, with 20 simulated networks of each size, generated as described in Section 3.1.1. Network dynamics are obtained initializing at random a system of Dynamic Recurrent Neural Networks and sampling 50 logarithmically spaced time points. Network sizes are in line with the best results from the state-of-the-art on the inference of systems of nonlinear equations representing Gene Regulatory Networks [72, 80, 81]: a complete system of differential equations offers a detailed description of the observed phenomenon and has thus to be exploited on a small scale problem.

Boxplots of Precision and Recall of the best scoring solutions obtained in 10 restarts of our algorithm, on 20 networks of size 5, 8 and 10, are reported in Figure 6.8.

The algorithm is able to infer the correct structure (Precision and Recall equal to 1) on 19 networks of 5 genes, 8 networks of 8 genes and 7 networks of 10 genes. However, when analyzing the minimum RSE reached after 10 iterations (Figure 6.9), on 17 networks of 8 genes and 9 networks of 10 genes the algorithm is able to reach RSE values below 10^{-18}, a value extremely close to zero. This discrepancy between correct structure and low RSE is probably to be attributed to the complexity of nonlinear dynamical systems: the same set of time series, in fact, can potentially be described by more than one system of equations with great accuracy. Forcing the sparsity of the system greatly reduces the size of the

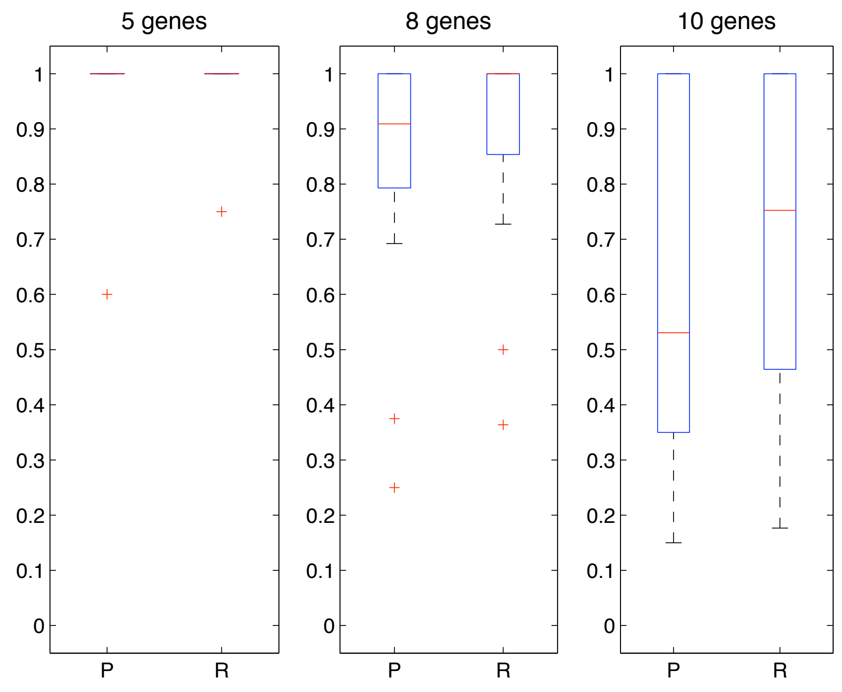


Figure 6.8: Boxplots of Precision and Recall of the best solution, *i.e.* the solution with the lowest RSE, obtained in 10 restarts of the mixed optimization algorithm on 20 networks of 5, 8 and 10 genes.

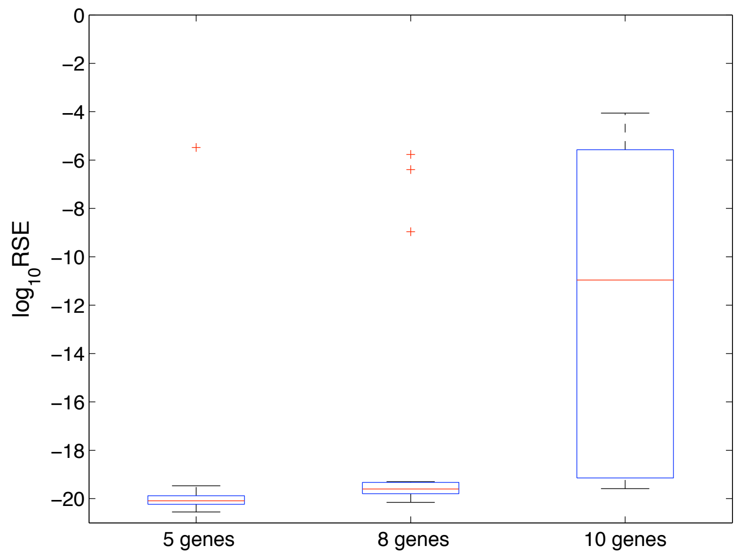


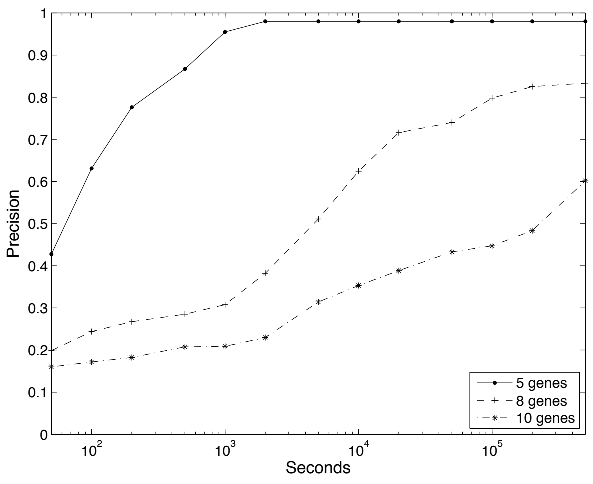
Figure 6.9: Boxplots of the logarithm of the lowest RSE values reached in 10 restarts of the mixed optimization algorithm on 20 networks of 5, 8 and 10 genes.

set of possible different systems, but apparently not enough, even with systems of 8 / 10 variables and observations of 50 time points.

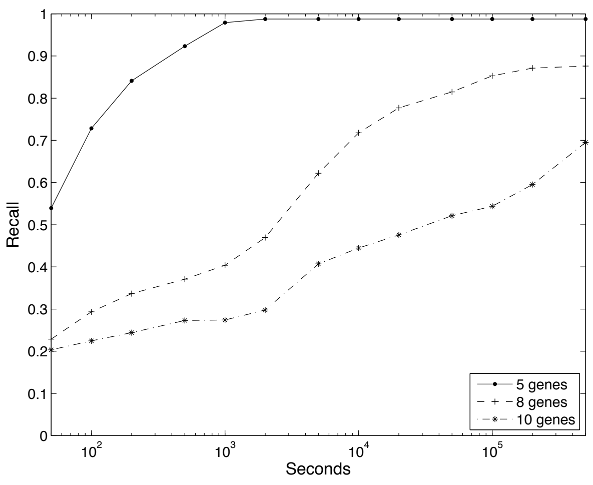
For what concerns computational time, in Figure 6.10 we plot the trends of Precision, Recall and RSE across time, on the 20 problem instances of 5, 8 and 10 genes, when restarting 10 times the algorithm and considering the solution with the lowest so far RSE. For Precision and Recall, we chose to plot the mean value over the 20 instances, whereas for the RSE we plot the median. All tests were run on a single AMD Opteron2216 HE 2,4GHz processor.

From the figure, one can observe that for the majority of networks of 5 genes a RSE lower than 10^{-18} is reached after 2×10^2 seconds, and after 5×10^4 seconds for the majority of networks of 8 genes. As said before, low RSE values do not always correspond to exact structure, but best networks are not too far, on average, from the correct structure: when the best values for RSE are reached for the majority of networks of 5 and 8 genes, Precision and Recall are both greater than 0.8. On networks of 10 genes, on the other hand, 10 restarts seem not to be enough for the algorithm to reach optimal RSE values in the majority of cases; resulting Precision and Recall are 0.6 and 0.7, respectively.

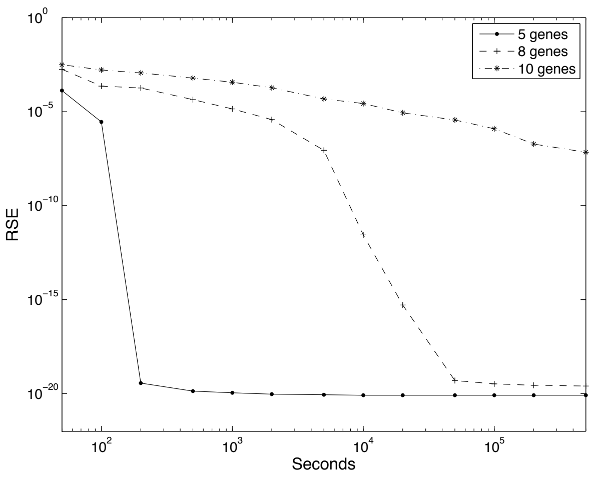
To compare our algorithm with the state-of-the-art, we chose to consider the results obtained in [80] and [81], which both present two mixed discrete and continuous optimization approaches to the problem of inferring Dynamic Recurrent Neural Networks from DNA microarray time course data. In both papers, however, experiments are carried out on a single simulated gene network (4 genes for [81] and 8 genes for [80]), thus the presented results are not really indicative of the average behaviour of the algorithms. In [81], on a network of 4 genes and with 50 time samples, reached Precision and Recall are respectively 0.44 and 0.5. In [80], on a network of 8 genes and with 3 time series of 30 samples each (thus 90 total samples), reached Precision and Recall are respectively 0.74 and 0.67. Our algorithm is thus able to reach better performance on problems



(a)



(b)



(c)

Figure 6.10: (a) average Precision, (b) average Recall and (c) median of the RSE of the solutions with the lowest so far RSE in 10 repeated restarts of the mixed optimization algorithm, on 20 networks of 5, 8 and 10 genes.

of the same size.

6.2.4 Conclusions and Future Directions

In this section, we presented a mixed discrete and continuous optimization approach to the inference of Gene Regulatory Networks from DNA microarray time course data. The algorithm is made of two interacting components, an Iterated Best Improvement Local Search algorithm, which searches in the discrete space of network structures, and a Covariance Matrix Adaptation - Evolution Strategies algorithm for the optimization of continuous system parameters. Two additional communication strategies between the two components are presented, one to exploit locality in the space of network structures and the other to provide additional feedback from the continuous optimization algorithm.

The mixed discrete and continuous optimization approach was motivated by the study of the fitness landscape presented in the previous section, and proved to be effective and competitive with the state-of-the-art on a set of simulated time course experiments. The experimental analysis, however, provided empyrical evidence of a problem related to the chosen model: time series obtained by sampling the dynamic evolution of a system of nonlinear differential equations can in general be described by more than one set of configurations of system parameters, even with long time series (50 samples), small scale systems (5 / 8 variables) and additional constraints on system sparsity. For this reason, a future research direction will be the study of Ensemble Learning strategies to merge the results of different runs of the same stochastic algorithm, with the purpose of sampling the space of all the equivalent networks able to describe the same data and providing confidence levels for each edge, proportional to how much it is conserved across the different system configurations.

Chapter 7

Conclusions

In this thesis, the problem of reverse engineering Gene Regulatory Networks from DNA microarray experiments has been studied from an algorithmic point of view. After a review of the biological problem and of the state of the art in reverse engineering algorithms, its main difficulties and limitations were pointed out. We have explained the need of decomposing the main task into a set of subproblems and of identifying the correct algorithmic approach for each subproblem, taking into account the scale and nature of the data to be processed (Chapters 1–3).

Within this perspective, three novel algorithmic techniques were presented, designed to cope with different types of microarray experiments and to address the problem at different levels of scale. Moreover, two detailed studies were carried out to further get insight into the difficulty of the problem of Gene Regulatory Network inference.

Chapter 4 focused on the problem of inferring oriented Gene Regulatory Networks, *i.e.* on identifying the presence or absence of regulatory relations between two genes, without considering the strength or the analytical shape of the relations. Two kinds of DNA microarray experiments were considered for this: steady state observations of systematic perturbations of gene expressions and time course observations of gene expression profiles. Two novel algorithms were designed to process these two types of data.

In the first case (Section 4.1), a Qualitative Reasoning algorithm is designed to process steady state perturbation experiments. The algorithm is able to infer a subset of the regulatory rules from the differences in gene expression between the perturbation experiments and the wild type experiment, *i.e.* the steady state observation of the system when none of the genes is perturbed. The algorithm can infer regulatory relations in three different scenarios, exploiting the propagation of the perturbation throughout the network. From experiments on simulated data, we observed that the Precision of the qualitative algorithm is extremely high, close to one even on large scale networks (100 genes), but that the algorithm is intrinsically unable to reconstruct the entire network. These observations, together with its polynomial running time, make the algorithm a good candidate for a preprocessing phase of some other inference technique, because of its ability to provide fast and reliable information on some regulatory relations, which can be used to reduce the size of the search space of a second Reverse Engineering algorithm. To our knowledge, this is the first systematic study on the application of Qualitative Reasoning to DNA microarray experiments of steady state perturbation.

In the second case (Section 4.2) CNET, an inference algorithm based on a heuristic scoring function for putative causal relations, is designed to process time course observations of gene expression. The scoring function is carefully tailored to identify the best set of regulators for each gene even in the presence of noise and variable regulatory delays in the expression profiles. At the same time, it rewards simpler regulatory rules. CNET proved to be competitive with one of the best algorithms for this particular subproblem, Dynamic Bayesian Networks (DBNs), both on simulated and on real time course data. Some approaches have already been proposed in the literature to tackle the three separate problems of noise, variable regulatory delays and increasing complexity of the model; the main point of innovation of CNET is its ability to face the three problems together, through the heuristic scoring function for putative causal relations.

The behaviour of CNET and DBNs was then further explored in a study to understand how the performance of Reverse Engineering algorithms is related to the structural and topological properties of Gene Regulatory Networks (Chapter 5). The analysis outlined that:

- there are regions of the regulatory networks which are intrinsically difficult to be solved by both algorithms;
- both algorithms exhibit equal or higher performance when the system is externally stimulated for the whole course of the experiment;
- when a single node i is externally stimulated, accuracy in inferring the edges directly outgoing from i is significantly higher,
- accuracy in inferring an edge depends on the network substructure containing it, being significantly higher if the in-degree of its destination node is equal to one and if alternative directed paths are absent.

All these results are consistent with intuition but, as fare as we know, this is the first systematic study that proves their empirical evidence.

Chapter 6 focused on the inference of systems of nonlinear differential equations from time course data. The scale of this second problem is smaller, because the objective is to fit to time course data a system able to analytically describe dependencies between genes and capable of making predictions on the evolution of the gene expression across time. For our analysis, we chose a class of systems of nonlinear sigmoidal differential equations known as Dynamic Recurrent Neural Networks [54], already exploited in the literature to analyze gene expression profiles [80, 81, 76] and provided with a set of interesting features: regulatory relations, in fact, are known to exhibit a differential and strongly nonlinear nature and Dynamic Recurrent Neural Networks are among the simplest systems able to reproduce a similar behaviour [39].

The ability to predict the evolution of the system across time provides one with a direct measure of the goodness of a particular configuration of the system, in terms of the error between real profiles and profiles predicted by the current system configuration. The inference problem can thus be mapped to an optimization problem, in which model parameters form the search space and the error is the fitness function to be minimized. In Section 6.1, we present a detailed analysis on the problem, when Dynamic Recurrent Neural Networks have to be inferred and the Relative Squared Error (RSE) has to be minimized. The analysis is carried out by sampling the search space around the optimal solution and by studying the correlation between the RSE of the samples and their Euclidean distance from the optimum. The study outlined that:

- the correlation between RSE and distance from the optimum is strong, but the search landscape exhibits deep valleys, in which a general purpose continuous optimization algorithm can be trapped; it can in fact keep reducing RSE without a real decrease in the distance from the optimum;
- information on the structure of the networks, in terms of the pattern of zero and nonzero elements in the system, is valuable and provides the search space with a higher level of organization, useful for a continuous optimization algorithm;
- organization in the fitness landscape is inversely proportional to the accuracy of the provided additional information on the structure, in terms of Hamming distance from the original connectivity matrix.

This is the first study in the literature on fitness-distance correlation analysis for the problem of inferring Gene Regulatory Networks and its results can be exploited for the design of novel Reverse Engineering algorithms.

The aforementioned properties of the search space motivated the design of a novel mixed optimization algorithm, described in Section 6.2, which searches at a higher level in the discrete space of connectivity matrices and exploits a continuous optimization algorithm to tune the nonzero parameters of candidate matrices, which are evaluated through the obtained minimum RSE. The algorithm exploits Iterative Best Improvement Local Search with Multiple Restarts [30] for the discrete search and Covariance Matrix Adaptation - Evolution Strategies (CMA-ES [28]) for optimizing continuous nonzero parameters. The main point of innovation of our algorithm are two communication strategies between the discrete and the continuous optimizer, which allow them to exploit locality in the search space and to progress faster towards the global optimum. Performance results on simulated data showed that the algorithm is able to infer correct GRNs and is competitive with the state-of-the-art on the same subproblem, but pointed out a limitation of the chosen model: sampled nonlinear differential dynamics can potentially be described by a set of different configurations of the system of equations with extremely low RSE, even with a large set of samples and a small number of system variables.

Most of our performance results are obtained in simulation, because of the need for large set of benchmark problems to test the average behaviour of the algorithm. Reliable information on real regulatory systems is still missing, thus a set of gold standards is hard to find in the literature [6, 71]. However, we tested the behaviour of one of our algorithms, CNET, on a real gene expression dataset, analyzing its ability to infer what is known of real regulatory relations; moreover, we assessed the performance of our Qualitative Reasoning algorithm on a widely known simulated benchmark for Reverse Engineering algorithms, the DREAM4 In Silico Network Challenge. As a future direction, we intend to test the Qualitative Reasoning and the mixed optimization algorithms on a real problem, although with the proper care in considering the results of the algorithms and in comparing them with information from the literature on the biological system under analysis.

Since the inference of a Gene Regulatory Network from a single experiment proved to be unfeasible, as further shown by our presented studies on the difficulties of the inference process, we will exploit both problem decomposition and complementary information provided by multiple heterogenous experiments to try to fully understand the complex mechanisms beyond regulatory systems. Ensamble Learning strategies will thus be studied, with the aim of combining the results of our three novel algorithms, when applied to different datasets originating from the same biological system. For example, from a dataset of both steady state perturbation experiments and time course experiments, the Qualitative Reasoning algorithm can be exploited to gather reliable information on a subset of regulatory relations, which can then be provided as a priori information to CNET and to the mixed optimization algorithm, reducing the size of their search space. Moreover, given the stochastic nature of the mixed optimization algorithm, we will study the design of one or more voting strategies to merge the results of multiple runs of the algorithm, thus providing confidence levels for the inferred edges.

Finally, the mixed discrete/continuous optimization technique developed for the inference of Gene Regulatory Networks can also be adapted to other network inference problems, such as learning of Bayesian Networks or Artificial Neural Networks: traditional approaches to these problems usually adopt greedy algorithms [12] or heuristic rules of thumb [29] to identify good network structures and then exploit optimization techniques to find the best configuration of network parameters. Our mixed approach can be applied to these problems, maintaining the paradigm of discrete search in the space of structures and continuous search in the space of network parameters, selecting from time to time the best optimization algorithms for the two components and developing novel techniques for the communication between the two search layers.

Appendices

Appendix A

Equivalence between pair consistency and causal relation

In this appendix we prove mathematically the equivalence between the REVEAL condition for causal relations, based on Shannon Entropy, and the consistency of the pair $\langle regulators; regulated_gene\rangle$, *i.e.* the condition on which the scoring function of CNET algorithm is based (see Section 4.2).

Definition 1 *The pair* $\langle (X_1 \ldots X_K); X_0\rangle$ *is* consistent *if, every time a given combination of values* $(\overline{x}_1 \ldots \overline{x}_K)$ *appears for* $(X_1 \ldots X_K)$*, the value of* X_0 *after* Δ *time steps is always the same.*

Definition 2 *The* Shannon Entropy *for a sequence* X *of symbols* x_i *from an alphabet of size* b *is*

$$H(X) = -\sum_{i=1}^{b} p(x_i)\log_b p(x_i) \tag{A.1}$$

where $p(x_i)$ *is the probability of observing the particular symbol* x_i.

Definition 3 *The* Joint Shannon Entropy *for the sequences* $(X_1 \ldots X_K)$ *of symbols from an alphabet of size* b*, is*

$$H(X_1, ..., X_K) = -\underbrace{\sum_{i=1}^{b} \ldots \sum_{i=1}^{b}}_{K\ times} p(x_{1i}, \ldots, x_{Ki})\log_b p(x_{1i}, \ldots, x_{Ki}) \tag{A.2}$$

where $p(x_{1i}, \ldots, x_{Ki})$ *is the probability of observing simultaneously the particular combination of symbols* $x_{1i}, \ldots, x_{Ki}$ *in sequences* $X_1, \ldots, X_K$.

Proposition 1 *The causal relation*

$$X_1 \ldots X_K \Rightarrow X_0 \tag{A.3}$$

holds if and only if

$$H(X_1 \ldots X_K) = H(X_0, X_1 \ldots X_K) \tag{A.4}$$

after a proper shift of sequence X_0, to account for the fixed delay Δ in the causal relation.

Proven in [41].

Theorem 1 *The causal relation A.3 holds if and only if the pair $\langle (X_1 \dots X_K); X_0 \rangle$ is consistent.*

Proof From Proposition 1, equation (A.3) holds if and only if

$$H(X_1 \dots X_K) = H(X_0, X_1 \dots X_K) \tag{A.5}$$

but then, from definition 3

$$H(X_1, ..., X_K) = -\underbrace{\sum_{i=1}^{b} \dots \sum_{i=1}^{b}}_{K\ times} p(x_{1i}, \dots, x_{Ki}) \log_b p(x_{1i}, \dots, x_{Ki}) \tag{A.6}$$

and, supposing w. l. o. g. that $X_i \in \{i, s, d\}$[1],

$$H(X_0, X_1 \dots X_K) = -\underbrace{\sum_{i=1}^{b} \dots \sum_{i=1}^{b}}_{K+1\ times} p(x_{0i}, x_{1i}, \dots, x_{Ki}) \log_b p(x_{0i}, x_{1i}, \dots, x_{Ki}) =$$

$$-\underbrace{\sum_{i=1}^{b} \dots \sum_{i=1}^{b}}_{K\ times} p((x_0 = i) \wedge (x_{1i} \dots x_{Ki})) \log p((x_0 = i) \wedge (x_{1i} \dots x_{Ki}))$$

$$-\underbrace{\sum_{i=1}^{b} \dots \sum_{i=1}^{b}}_{K\ times} p((x_0 = s) \wedge (x_{1i} \dots x_{Ki})) \log p((x_0 = s) \wedge (x_{1i} \dots x_{Ki}))$$

$$-\underbrace{\sum_{i=1}^{b} \dots \sum_{i=1}^{b}}_{K\ times} p((x_0 = d) \wedge (x_{1i} \dots x_{Ki})) \log p((x_0 = d) \wedge (x_{1i} \dots x_{Ki})) \tag{A.7}$$

If the pair $\langle (X_1 \dots X_K); X_0 \rangle$ is consistent, every time a given combination $(x_{1i} \dots x_{Ki})$ appears for $(X_1 \dots X_K)$, x_0 after Δ time steps has always the same value $\overline{x}$, being it *increasing*, *steady* or *decreasing*. Then,

$$p(x_{0i}, x_{1i}, \dots, x_{Ki}) = \begin{cases} p(x_{1i}, \dots, x_{Ki}) & \text{if } x_{0i} = \overline{x}; \\ 0 & \text{otherwise.} \end{cases}$$

And then, in equation (A.7), for every combination of values $(x_{1i} \dots x_{Ki})$ there is exactly one term in one of the three summations which is different from zero. And then equation (A.5) holds.

[1]Which stand for *increasing*, *steady* and *decreasing*.

Appendix B

Selection of differentially expressed genes from noisy data

In DNA microarray experiments, for the case in which an external perturbation of the system is present in some experiments and absent in some others (*wild type* experiments), genes are said to be *differentially expressed* in the perturbation experiment if their level of expression differs significantly from their level in the wild type experiment.

In this appendix, we present the methodology we adopted for the selection of differentially expressed genes from noisy knock-out experiments. As a case study, throughout the appendix we will show the application of this methodology to a dataset extracted from the DREAM4 In Silico Network Challenge [46, 73], namely the Size 10 Subchallenge, which is described in detail in Section 3.1.2 and is used as a benchmark for our Qualitative Reasoning algorithm in Section 4.1.

The procedure can be divided in three subsequent steps:

1. Identification of flat profiles from time course experiments,
2. Inference of a noise model from flat profiles,
3. Selection of differentially expressed genes from knock-out experiments with the inferred noise model.

Each step is described in detail in what follows.

B.1 Selection of flat profiles from time course data

The core idea behind our methodology is to infer an accurate description of the noise present in the data. With this aim, we want to identify a set of profiles which can be considered representative of the noise at different levels of gene expression. If both steady state and time course experiments are present in the dataset under analysis, as it is the case in the DREAM4 In Silico Network

Challenge, one can exploit the dependencies among the expression samples at subsequent temporal intervals to reliably identify *flat* profiles, *i.e.* profiles whose expression is not changing significantly over time. All the variations present in flat profiles are caused only by noise, thus one can exploit a set of flat profiles at different levels of expression to gain a precise model for the noise and then use this model to select differentially expressed genes from the steady state experiments.

To this purpose, we exploited the procedure described in [74] and implemented in the open-source EDGE software [40]: for each gene expression time series in the dataset, the method fits a model under the null hypothesis that there is differential expression, and then under the alternative hypothesis that there is no differential expression. The null model is the curve that minimizes the sum of squares among the general class of natural cubic splines. The alternative model is the flat line that minimizes the sum of squares among all possible flat lines. A statistic is calculated that compares the goodness of fit of the two model: the lower is the p-value of the statistic, the higher the chances are that the gene expression profile is flat.

For each of the five networks of ten genes from the DREAM4 In Silico Network Challenge, we processed altogether the five time course experiments with the EDGE software and selected, among the ten profiles with the lower p-values, the five profiles with the better span across the whole normalized gene expression range. The five selected profiles for each network are shown in Figure B.1.

B.2 Inference of a noise model

The second step of our analysis consists in the inference of a model for the noise from the previously identified flat profiles. With this purpose, we follow the procedure described in [18]: given two measurements (x_i, x_j) of the expression of the gene x and assuming additive noise we can write

$$\left\{ \begin{array}{lcl} x_i & = & \mu_i + \epsilon_i \\ x_j & = & \mu_j + \epsilon_j \end{array} \right. \tag{B.1}$$

where μ_i and μ_j represent the real (unknown) gene expression values and ϵ_i, ϵ_j are two realizations of the noise variable ϵ.

Since our purpose is to identify genes whose espression is significantly different between the knock-out experiment and the wild type experiment, we want to infer a model for the variable

$$\delta = x_i - x_j = (\mu_i - \mu_j) + (\epsilon_i - \epsilon_j) \tag{B.2}$$

at different levels of the average expression intensity

$$\overline{x_i x_j} = \frac{x_i + x_j}{2} \tag{B.3}$$

If the Null Hypothesis H_0 is defined as the situation in which the generic gene x is not differentially expressed in two different arrays, and thus $\mu_i = \mu_j$, δ can be seen as a distribution in Null Hypothesis conditions. Therefore, from δ distribution knowledge it is possible to derive a confidence interval for the selection of differentially expressed genes. In our case, the δ distribution can be

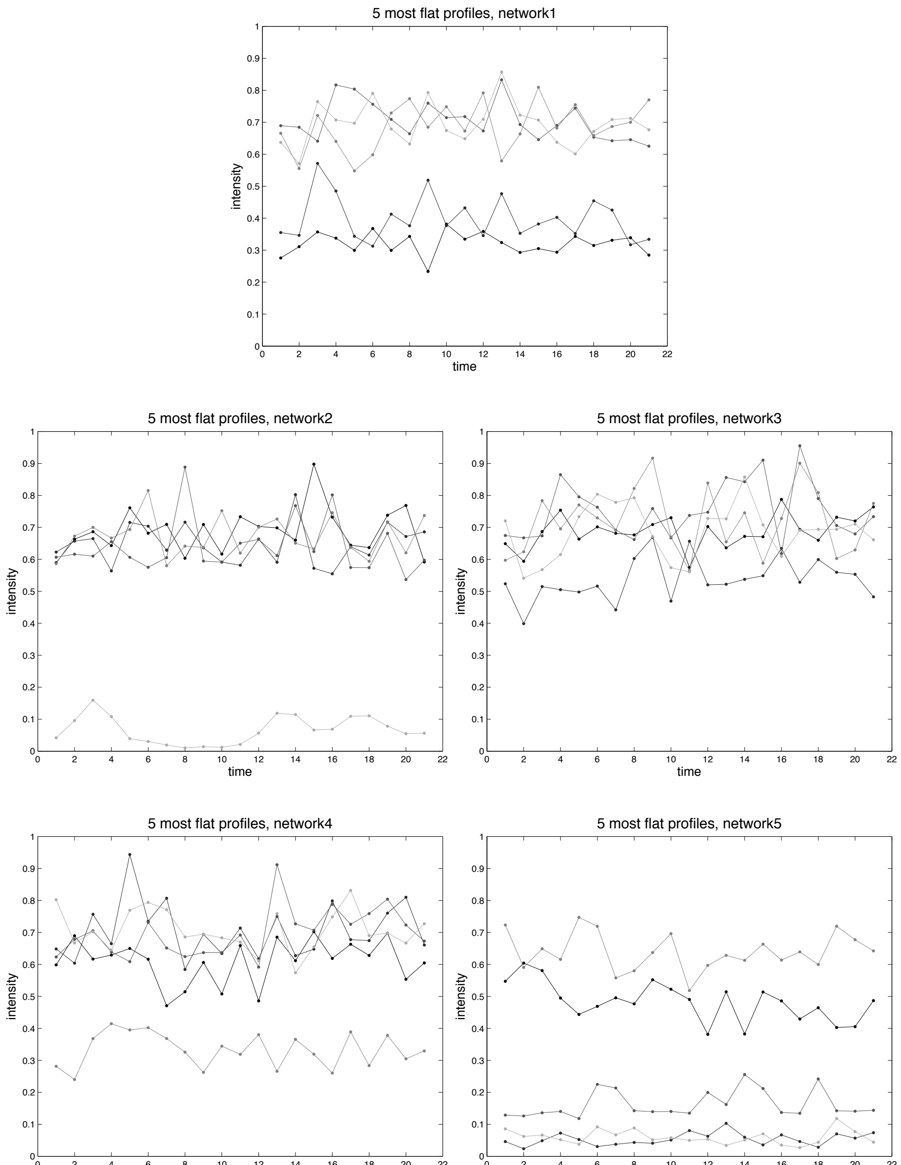


Figure B.1: Five most flat profiles for each of the five networks in the DREAM4 In Silico Network Challenge, selected with the EDGE software for differentially expressed time course experiments analysis.

inferred from flat time course profiles and then exploited to select differentially expressed genes in the knock-out and wild type experiments.

The inference procedure is composed by the following steps:

- A bootstrap operation is carried out on the set of flat profiles, extracting (with replacement) 100 pairs of measurements (x_i, x_j) for each flat profile.

- Average expression intensity and δ are computed for each pair of measurements with equations B.3 and B.2. δ *vs* average intensity scatterplots of the obtained samples for the DREAM4 data, one for each network, are represented in Figure B.2. As it is clear from the figure, in some cases the samples do not cover the whole span of the average expression intensity, because the dadaset does not contain flat profiles for all the possible average intensities.

- The range of average intensity values is discretized in 20 intervals of variable sizes, grouping sets of 25 samples with consecutive intensity values.

- For each of these intervals, the standard deviation of δ is calculated using

$$SD_\delta = \sqrt{\frac{1}{n-1} \cdot \sum_{i=1}^{n} (\delta_i - m_\delta)^2} \tag{B.4}$$

  where n is the number of samples and m_δ is the average of the n observed δ_i in the considered interval.

- In order to explicit the relation between the standard deviation of δ and the average intensity of the gene expression, the standard deviation is fitted using Weighted Least Squares method. The model we chose for the fit, after having inspected the sampled distributions of standard deviations, is of constant coefficient of variation: the samples were thus fitted to straight lines. The samples of the standard deviations, together with the fitted models, for the DREAM4 data are represented in Figure B.3. Some cases of deviation from the fitted linear model can be imputed to the aforementioned lack of flat profiles to cover the whole range of average expression intensity.

The inferred noise model, *i.e.* the description of the standard deviation of δ as a function $SD_\delta(\overline{x_i x_j})$ of the average expression intensity, is then used in the next step to identify the genes which are differentially expressed between knock-out and wild type experiments.

B.3 Selection of differentially expressed genes from knock-out experiments

To select the genes whose expression intesity differs significantly between the knock-out and the wild type experiments we exploit the noise model through the following procedure:

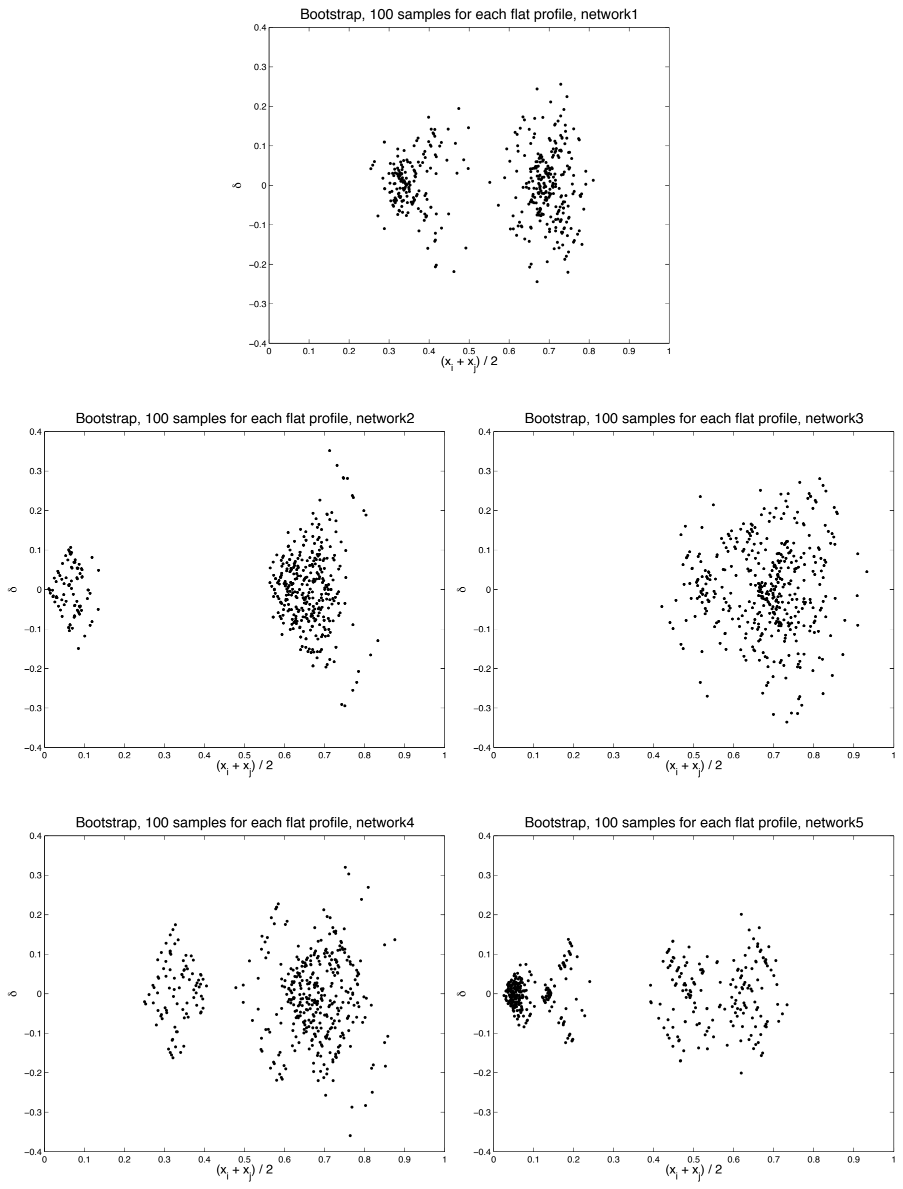


Figure B.2: δ *vs* average intensity scaterplots of the bootstrap sampling for each of the five networks in the DREAM4 In Silico Network Challenge. 100 samples were gathered for each of the 5 flat profiles.

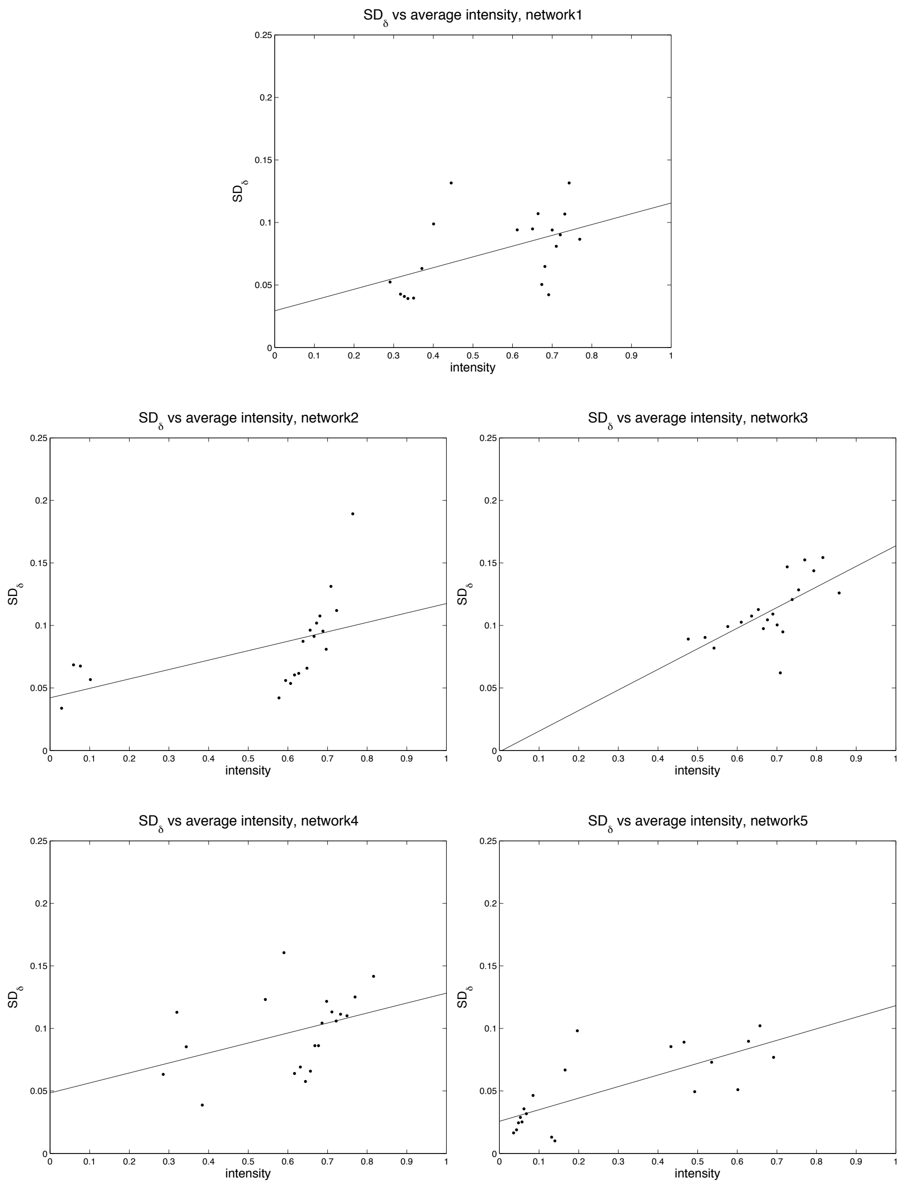


Figure B.3: SD_δ *vs* average intensity plots for each of the five networks in the DREAM4 In Silico Network Challenge, with the results of Weighted Least Squares fits of models with constant coefficient of variation.

- For each gene and each knock-out experiment, we compute the two measures
$$\delta_{ij} = x_{ij} - wt_i, \qquad \overline{x_{ij}} = \frac{x_{ij} + wt_i}{2}, \tag{B.5}$$
where x_{ij} is the intensity of the expression of gene x_i measured during the knock-out of gene x_j and wt_i is the intensity of the expression of the same gene measured in the wild type experiment.

- For each pair $(\delta_{ij}, \overline{x_{ij}})$ we compute
$$\delta'_{ij} = \frac{\delta_{ij}}{SD_\delta(\overline{x_{ij}})}, \tag{B.6}$$
normalizing each δ_{ij} by its corresponding standard deviation, computed from the noise model inferred in the previous step.

- The distribution of the variable δ' reproduces the null hypothesis distribution and, assuming Gaussian noise, it can be considered equivalent to the standard normal distribution. Fixing a significance level α, where α represents the probability of rejecting the null hypothesis when the null hypothesis is true, one can thus compute a confidence threshold θ directly from the inverse of the normal cumulative distribution function. θ can then be used to identify differentially expressed genes through the following inequalities:
$$\begin{cases} \delta'_{ij} > \theta & \Rightarrow \text{ Overexpressed gene} \\ \delta'_{ij} < -\theta & \Rightarrow \text{ Underexpressed gene} \\ -\theta \le \delta'_{ij} \le \theta & \Rightarrow \text{ Gene not differentially expressed} \end{cases} \tag{B.7}$$

For the case of the DREAM4 In Silico Network Challenge, we set a conservative significance level $\alpha = 0.002$, thus obtaining a threshold $\theta = 3.09$. Information on differentially expressed genes was then used for the Qualitative Reasoning algorithm described in Section 4.1 to infer direct causal relations between genes from gene knock-out experiments.

Bibliography

[1] R. Albert. Scale-free networks in cell biology. *Journal of Cell Science*, 118:4947–4957, 2005.

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

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

[4] A. Auger, N. Hansen, J. M. Perez Zerpa, R. Ros, and M. Schoenauer. Empirical comparisons of several derivative free optimization algorithms. In *Acte du 9ime colloque national en calcul des structures*, May 2009.

[5] S. Badaloni, M. Falda, and F. Sambo. Scale-free structure and topological properties in reverse engineering of gene regulatory networks. In *Workshop Italiano di Vita Artificiale e Computazione Evolutiva WIVACE2008.*, Venezia, Italy., Sept. 8-10 2008.

[6] M. Bansal, V. Belcastro, A. Ambesi-Impiombato, and D. di Bernardo. How to infer gene networks from expression profiles. *Mol. Syst. Biol.*, 3(78), February 2007.

[7] A.-L. Barabasi and R. Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, Oct. 1999.

[8] R. E. Bellman. *Adaptive control processes - A guided tour.* Princeton University Press, Princeton, New Jersey, U.S.A., 1961.

[9] A. J. Butte and I. S. Kohane. Mutual information relevance networks: functional genomic clustering using pairwise entropy measurements. In *Pacific Symposium on Biocomputing*, pages 418–429, 2000.

[10] J.-P. Chiou and F.-S. Wang. Hybrid method of evolutionary algorithms for static and dynamic optimization problems with application to a fed-batch fermentation process. *Computers and Chemical Engineering*, 23(9):1277 – 1291, 1999.

[11] A. R. Conn, N. I. M. Gould, and P. L. Toint. *Trust-Region Methods.* MPS-SIAM Series in Optimization. SIAM, Philadelphia, PA, 2000.

[12] G. F. Cooper and E. Herskovits. A bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 09(4):309–347, October 1992.

[13] A. Corradin, B. Di Camillo, G. Toffolo, and C. Cobelli. In silico assessment of four reverse engineering algorithms: role of network complexity and multi-experiment design in network reconstruction and hub detection. In *ENFIN-DREAM Conference Assessment of Computational Methods in Systems Biology, Madrid, April 28-29*, 2008.

[14] L. d. Costa, F. A. Rodrigues, G. Travieso, and P. R. V. Boas. Characterization of complex networks: A survey of measurements. *Advances in Physics*, 56(1):167–242, January 2005.

[15] C. O. Daub, R. Steuer, J. Selbig, and S. Kloska. Estimating mutual information using b-spline functions - an improved similarity measure for analysing gene expression data. *BMC Bioinformatics*, 5:118, 2004.

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

[17] P. D'Haeseleer, X. Wen, S. Fuhrman, and R. Somogyi. Linear modeling of mRNA expression levels during CNS development and injury. In *Pacific Symposium on Biocomputing*, pages 41–52, 1999.

[18] B. Di Camillo. *Modelling Dynamic Gene Expression Profiles: Insulin Regulation in Skeletal Muscle.* PhD Thesis in Bioengineering, Department of Information Engineering, University of Padova, Italy, 2003.

[19] B. Di Camillo, F. Sanchez-Cabo, G. Toffolo, S. K. Nair, Z. Trajanoski, and C. Cobelli. A quantization method based on threshold optimization for microarray short time series. *BMC Bioinformatics*, 6:S11, 2005.

[20] B. Di Camillo, G. Toffolo, and C. Cobelli. A gene network simulator to assess reverse engineering algorithms. *Annals of the New York Academy of Sciences*, 1158(1):125–142, 2009.

[21] M. Dorigo and T. Stützle. *Ant Colony Optimization (Bradford Books).* The MIT Press, July 2004.

[22] M. K. Dougherty, J. Mller, D. A. Ritt, M. Zhou, X. Z. Zhou, T. D. Copeland, T. P. Conrads, T. D. Veenstra, K. P. Lu, and D. K. Morrison. Regulation of raf-1 by direct feedback phosphorylation. *Molecular Cell*, 17(2):215 – 224, 2005.

[23] E. Fehlberg. Low-order classical runge-kutta formulas with step size control and their application to some heat transfer problems. Technical Report 315, NASA, 1969.

[24] F. Ferrazzi, P. Sebastiani, M. F. Ramoni, and R. Bellazzi. Bayesian approaches to reverse engineer cellular systems: a simulation study on nonlinear gaussian networks. *BMC Bioinformatics*, 8 Suppl 5, 2007.

[25] T. S. Gardner, D. di Bernardo, D. Lorenz, and J. J. Collins. Inferring genetic networks and identifying compound mode of action via expression profiling. *Science*, 301(5629):102–105, July 2003.

[26] P. Gennemark and D. Wedelin. Benchmarks for identification of ordinary differential equations from time series data. *Bioinformatics*, 25(6):780–786, 2009.

[27] N. Hall. Advanced sequencing technologies and their wider impact in microbiology. *J Exp Biol*, 210(9):1518–1525, May 2007.

[28] N. Hansen. The CMA evolution strategy: a comparing review. In *Towards a new evolutionary computation. Advances on estimation of distribution algorithms*, pages 75–102. Springer, 2006.

[29] S. Haykin. *Neural networks : a comprehensive foundation.* 1999.

[30] H. H. Hoos and T. Stützle. *Stochastic Local Search : Foundations & Applications (The Morgan Kaufmann Series in Artificial Intelligence).* Morgan Kaufmann, September 2004.

[31] L. Hunter. Life and its molecules: A brief introduction. *AI Magazine - Special issue on AI and Bioinformatics*, 25(1):9–22, 2004.

[32] T. Jones. Evolutionary algorithms, fitness landscapes and search. Working Papers 95-05-048, Santa Fe Institute, May 1995.

[33] T. Jones and S. Forrest. Fitness distance correlation as a measure of problem difficulty for genetic algorithms. In *Proceedings of the 6th International Conference on Genetic Algorithms*, pages 184–192, San Francisco, CA, 1995. Morgan Kaufmann.

[34] S. A. Kauffman. Metabolic stability and epigenesis in randomly constructed genetic nets. *Journal of Theoretical Biology*, 22(3):437–467, March 1969.

[35] J. Kennedy, R. Eberhart, and Y. Shi. *Swarm Intelligence.* Morgan Kaufmann, San Francisco, CA, 2001.

[36] K. Kentzoglanakis, M. Poole, and C. Adams. Incorporating heuristics in a swarm intelligence framework for inferring gene regulatory networks from gene expression time series. In *ANTS '08: Proceedings of the 6th international conference on Ant Colony Optimization and Swarm Intelligence*, pages 323–330, Berlin, Heidelberg, 2008. Springer-Verlag.

[37] S. Kimura, K. Ide, A. Kashihara, M. Kano, M. Hatakeyama, R. Masui, N. Nakagawa, S. Yokoyama, S. Kuramitsu, and A. Konagaya. Inference of S-system models of genetic networks using a cooperative coevolutionary algorithm. *Bioinformatics*, 21(7):1154–1163, 2005.

[38] S. Kimura, S. Nakayama, and M. Hatakeyama. Genetic network inference as a series of discrimination tasks. *Bioinformatics*, 25(7):918–925, 2009.

[39] J. F. Kolen and S. C. Kremer. *A Field Guide to Dynamical Recurrent Networks.* Wiley-IEEE Press, 2001.

[40] J. T. Leek, E. Monsen, and J. D. Dabney, A. R. ans Storey. Edge: extraction and analysis of dierential gene expression. *Bioinformatics*, pages 507–508, 2006.

[41] S. Liang, S. Fuhrman, and R. Somogyi. Reveal: a general reverse engineering algorithm for inference of genetic network architectures. In *Pacific Symposium on Biocomputing*, pages 18–29, 1998.

[42] P.-K. Liu and F.-S. Wang. Inference of biochemical network models in s-system using multiobjective optimization approach. *Bioinformatics*, 24(8):1085–1092, 2008.

[43] J. Makhoul, F. Kubala, R. Schwartz, and R. Weischedel. Performance measures for information extraction. In *Proceedings of DARPA Broadcast News Workshop*, pages 249–252, 1999.

[44] D. Marbach, C. Mattiussi, and D. Floreano. Bio-mimetic Evolutionary Reverse Engineering of Genetic Regulatory Networks. In E. Marchiori, J. H. Moore, and J. C. Rajapakse, editors, *5th European Conference on Evolutionary Computation, Machine Learning and Data Mining in Bioinformatics (EvoBIO 2007)*, volume 4447 of *Lecture Notes in Computer Science*, pages 155–165. Springer-Verlag Berlin, 2007.

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

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

[47] A. A. Margolin, I. Nemenman, K. Basso, C. Wiggins, G. Stolovitzky, R. Dalla Favera, and A. Califano. Aracne: an algorithm for the reconstruction of gene regulatory networks in a mammalian cellular context. *BMC Bioinformatics*, 7 Suppl 1, 2006.

[48] S. Mehrotra. On the implementation of a primal-dual interior point method. *SIAM Journal on Optimization*, 2(4):575–601, 1992.

[49] M. Molla, M. Waddell, D. Page, and J. Shavlik. Using machine learning to design and interpret gene-expression microarrays. *AI Magazine - Special issue on AI and Bioinformatics*, 25(1):23–44, 2004.

[50] K. Murphy and S. Mian. Modelling gene expression data using dynamic bayesian networks. Technical report, Computer Science Division, University of California, Berkeley, CA., 1999.

[51] D. Nam, S. Seo, and S. Kim. An efficient top-down search algorithm for learning boolean networks of gene expression. *Machine Learning*, 65:229–245, 2006.

[52] D. Nam, S. H. Yoon, and J. F. Kim. Ensemble learning of genetic networks from time-series expression data. *Bioinformatics*, 23(23):3225–3231, 2007.

[53] J. Pearl. *Probabilistic Reasoning in Intelligent Systems : Networks of Plausible Inference*. Morgan Kaufmann, September 1988.

[54] B. A. Pearlmutter. Dynamic recurrent neural networks. Technical Report CMU-CS-90-196, Carnegie Mellon University, Pittsburgh, PA, 1990.

[55] R. Peeters and R. Westra. On the identification of sparse gene regulatory networks. In *Proc 16th Intern Symp on Mathematical Theory of Networks*, 2004.

[56] M. J. D. Powell. *Large-Scale Nonlinear Optimization*, volume 83 of *Nonconvex Optimization and Its Applications*, chapter The NEWUOA software for unconstrained optimization, pages 255–297. Springer-Verlag, Berlin, Germany, 2006.

[57] E. Ravasz, A. L. Somera, D. A. Mongru, Z. N. Oltvai, and A. L. Barabasi. Hierarchical organization of modularity in metabolic networks. *Science*, 297(5586):1551–1555, August 2002.

[58] H. W. Ressom, Y. Zhang, J. Xuan, Y. Wang, and R. Clarke. Inference of gene regulatory networks from time course gene expression data using neural networks and swarm intelligence. In *IEEE Symposium on Computational Intelligence and Bioinformatics and Computational Biology*, pages 1–8. IEEE, 2006.

[59] G. Ruvkun. Molecular biology. glimpses of a tiny rna world. *Science*, 294(5543):797–799, 2001.

[60] L. Sacchi, C. Larizza, P. Magni, and R. Bellazzi. Precedence temporal networks to represent temporal relationships in gene expression data. *Journal of Biomedical Informatics*, 40(6):761–774, 2007.

[61] M. Sakawa. *Fuzzy Sets and Interactive Multiobjective Optimization*. Plenum Press, New York, 1993.

[62] F. Sambo, B. Di Camillo, M. Falda, G. Toffolo, and S. Badaloni. Evaluation of local reliability of gene networks inferred from time series expression data. In *RECOMB Satellite on Regulatory Genomics and Systems Biology - Abstract Book.*, page 121, Boston, MA., oct 29 – nov 2 2008.

[63] F. Sambo, B. Di Camillo, M. Falda, G. Toffolo, and S. Badaloni. CNET: an algorithm for the inference of gene regulatory interactions from gene expression time series. In *Proceedings of the 14th Workshop on Intelligent Data Analysis in Medicine and Pharmacology IDAMAP09*, pages 23–28, Verona, Italy, July 19 2009.

[64] F. Sambo, B. Di Camillo, and G. Toffolo. CNET: an algorithm for reverse engineering of causal gene networks. In *Bioinformatics Methods for Biomedical Complex Systems Applications. 8th Workshop on Network Tools and Applications in Biology NETTAB2008.*, pages 134–136, Varenna, Italy, May 19-21 2008.

[65] F. Sambo, B. Di Camillo, and G. Toffolo. Role of network structure and experimental design on the performance of two reverse engineering methods. In *7th European Conference on Computational Biology ECCB2008.*, Cagliari, Italy, Sept. 22-26 2008.

[66] F. Sambo, M. A. Montes de Oca, B. Di Camillo, and T. Stützle. On the difficulty of inferring gene regulatory networks: A study of the fitness landscape generated by relative squared error. In *Proceedings of the 9th Conference on Artificial Evolution*, Lecture Notes in Computer Science, Strasbourg, France, Oct. 26–28 2009.

[67] M. A. Savageau. *Biochemical Systems Analysis: a Study of Function and Design in Molecular Biology*. Addison-Wesley, Reading, MA, 1976.

[68] J. Schäfer and K. Strimmer. An empirical bayes approach to inferring large-scale gene association networks. *Bioinformatics*, 21(6):754–764, March 2005.

[69] P. Sebastiani, M. Abad, and M. F. Ramoni. Bayesian networks for genomic data analysis. In E. R. Dougherty, I. Shmulevich, J. Chen, and Z. J. Wang, editors, *EURASIP book series on signal processing and communications: genomic signal processing and statistics*, pages 281–320. Hindawi, New York, NY., 2005.

[70] P. Sebastiani, M. F. Ramoni, V. Nolan, C. T. Baldwin, and M. H. Steinberg. Genetic dissection and prognostic modeling of overt stroke in sickle cell anemia. *Nature Genetics*, 37(4):435–440, March 2005.

[71] N. Soranzo, G. Bianconi, and C. Altafini. Comparing association network algorithms for reverse engineering of large-scale gene regulatory networks: synthetic versus real data. *Bioinformatics*, 23(13):1640–1647, July 2007.

[72] C. Spieth, R. Worzischek, F. Streichert, J. Supper, N. Speer, and A. Zell. Comparing evolutionary algorithms on the problem of network inference. In M. Cattolico, editor, *Genetic and Evolutionary Computation Conference, GECCO 2006, Proceedings, Seattle, Washington, USA, July 8-12, 2006*, pages 305–306. ACM, 2006.

[73] G. Stolovitzky, R. J. Prill, and A. Califano. Lessons from the DREAM2 challenges: A community effort to assess biological network inference. *Annals of the New York Academy of Sciences*, 1158(1):159–195, 2009.

[74] J. D. Storey, W. Xiao, J. T. Leek, R. G. Tompkins, and R. W. Davis. Significance analysis of time course microarray experiments. *Proceedings of the National Academy of Sciences of the United States of America*, 102(36):12837–12842, September 2005.

[75] R. Storn and K. Price. Differential evolution: A simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11(4):341–359, 1997.

[76] Vu, T. Thi, Vohradsky, and Jiri. Nonlinear differential equation model for quantification of transcriptional regulation applied to microarray data of saccharomyces cerevisiae. *Nucleic Acids Research*, 35(1):279–287, January 2007.

[77] D. J. Watts and S. H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393(6684):440–442, June 1998.

[78] A. V. Werhli, M. Grzegorczyk, and D. Husmeier. Comparative evaluation of reverse engineering gene regulatory networks with relevance networks, graphical gaussian models and bayesian networks. *Bioinformatics*, 22(20):2523–2531, 2006.

[79] M. L. Whitfield, G. Sherlock, A. J. Saldanha, J. I. Murray, C. A. Ball, K. E. Alexander, J. C. Matese, C. M. Perou, M. M. Hurt, P. O. Brown, and D. Botstein. Identification of genes periodically expressed in the human cell cycle and their expression in tumors. *Molecular Biology of the Cell*, 2002.

[80] R. Xu, G. K. Venayagamoorthy, and D. C. Wunsch, II. Modeling of gene regulatory networks with hybrid differential evolution and particle swarm optimization. *Neural Networks*, 20(8):917–927, 2007.

[81] R. Xu, D. Wunsch II, and R. Frank. Inference of genetic regulatory networks with recurrent neural network models using particle swarm optimization. *IEEE/ACM Trans. Comput. Biol. Bioinformatics*, 4(4):681–692, 2007.

[82] J. Yu, V. A. Smith, P. P. Wang, A. J. Hartemink, and E. D. Jarvis. Advances to bayesian network inference for generating causal networks from observational biological data. *Bioinformatics*, 20(18):3594–3603, December 2004.

[83] M. Zampieri, N. Soranzo, and C. Altafini. Discerning static and causal interactions in genome-wide reverse engineering problems. *Bioinformatics (Oxford, England)*, May 2008.

[84] W. Zhao, E. Serpedin, and E. R. Dougherty. Inferring connectivity of genetic regulatory networks using information-theoretic criteria. *IEEE/ACM Trans. Comput. Biol. Bioinformatics*, 5(2):262–274, 2008.

[85] R. Zhu, A. S. Ribeiro, D. Salahub, and S. A. Kauffman. Studying genetic regulatory networks at the molecular level: Delayed reaction stochastic models. *Journal of Theoretical Biology*, 246(4):725 – 745, 2007.