Ph.D. Thesis

Reverse Engineering Gene Regulatory Networks for elucidating transcriptome organisation, gene function and gene regulation in mammalian systems

Director of studies: Dr. Diego di Bernardo

Candidate: Vincenzo Belcastro

External supervisor: Degree in Computer Science

Prof. Mario di Bernardo ARC: Telethon Institute of Genetics and Medicine

Year 2007/2010
Contents

LIST OF TABLES vii

LIST OF FIGURES viii

1 Introduction 3

2 Introduction to reverse-engineering 7
    2.1 Microarray technology and microarray data repositories . . . . 12
    2.2 Reverse-engineering . . . . . . . . . . . . . . . . . . . . . . . . 14
       2.2.1 Bayesian networks . . . . . . . . . . . . . . . . . . . . . . 14
       2.2.2 Association Networks . . . . . . . . . . . . . . . . . . . . . 19
       2.2.3 Ordinary differential equations (ODEs) . . . . . . . . 25
       2.2.4 Other Approaches . . . . . . . . . . . . . . . . . . . . . . . 27

3 Comparison of Reverse-engineering algorithms 29
    3.1 Gene network inference algorithms . . . . . . . . . . . . . . . . . . . 30
       3.1.1 BANJO . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 31
       3.1.2 ARACNe . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 34
       3.1.3 NIR . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 37
3.2 In-silico and experimental data .......................... 44
  3.2.1 Generation of ‘In silico’ data .......................... 44
  3.2.2 Experimental data ................................. 50
  3.2.3 Assessing the performance of algorithms .............. 53

3.3 Results: ‘in silico’ evaluation .............................. 57
  3.3.1 Application of parallel NIR ............................ 60

3.4 Results: Experimental evaluation ........................... 61
  3.4.1 Reverse Engineering the IRMA network .................. 62

3.5 Discussion and Conclusion ................................. 67

4 Reverse-engineering gene networks from massive and heterogeneous gene expression profiles 69
  4.1 Introduction .............................................. 70
  4.2 A new algorithm for reverse-engineering ................. 71
    4.2.1 Normalisation of Gene Expression Profiles .......... 71
    4.2.2 Mutual Information ............................... 72
  4.3 Application to simulated data ............................ 75
    4.3.1 Simulated Dataset .................................. 75
    4.3.2 Performance on simulated data and comparison with
          state-of-the-art reverse-engineering algorithms .......... 76
  4.4 Discussions and conclusions ............................. 79

5 Reverse-engineering of human and mouse regulatory networks 80
  5.1 Introduction ............................................. 81
CONTENTS

5.2 Human and Mouse gene expression profiles .................. 82
5.3 Application of the reverse-engineering algorithm ............. 83
  5.3.1 Human network ........................................ 85
  5.3.2 Mouse network ......................................... 93
5.4 The modular structure of the human and mouse networks ... 97
  5.4.1 Hierarchical clustering procedure ....................... 103
  5.4.2 Communities of communities: “Rich-clubs” ............... 105
5.5 Connections tend to be conserved across species .............. 109
5.6 Discussions and conclusions .................................. 113

6 Transcriptome organisation and gene function ................. 114
  6.1 Biological validation of Protein-Protein interactions ..... 115
    6.1.1 Yeast-two-Hybrid assays ............................. 117
  6.2 Chromatin structure and gene expression .................... 119
  6.3 Prediction of gene function and protein localisation ...... 125
    6.3.1 Computational method ............................... 126
    6.3.2 Biological validation of gene expression localisation .. 127
  6.4 Elucidating the function of the granulin precursor ‘disease-gene’130
    6.4.1 Experimental material ............................... 134
    6.4.2 Identification of Binding sites in Granulin promoter re-
           gion ................................................... 136
  6.5 Gene signature analysis .................................... 136
    6.5.1 A case study ......................................... 137
  6.6 Discussions and conclusions ................................ 139
List of Tables

3.1 Features of the network inference algorithms reviewed . . . . . . 32
3.2 Experimental datasets used as examples . . . . . . . . . . 51
3.3 Results of the application of network inference algorithms on
the simulated Global perturbation . . . . . . . . . . . . . . . . 58
3.4 Results of the application of network inference algorithms on
the simulated Local perturbation . . . . . . . . . . . . . . . . 58
3.5 Results of the application of network inference algorithms on
the simulated Time Series data . . . . . . . . . . . . . . . . 59
3.6 Total execution times in seconds . . . . . . . . . . . . . . . . 61
3.7 Results of the application of network inference algorithms on
the experimental datasets . . . . . . . . . . . . . . . . . . . . . 62
4.1 Performance comparison on in-silico expression profiles . . . . 78
5.1 Details of the Golden Standard Interactome . . . . . . . . . . 87
5.2 Human hub genes . . . . . . . . . . . . . . . . . . . . . . . . . 90
5.3 Mouse hub genes . . . . . . . . . . . . . . . . . . . . . . . . . 94
5.4 List of conserved modules between human and mouse . . . . . 112
LIST OF TABLES

6.1 Biological validation of Protein-Protein interactions . . . . . . 118
6.2 GOEA: guilty-by-associations study . . . . . . . . . . . . . . . 126
6.3 Genes tested for mitochondrial localisation . . . . . . . . . . . 130
6.4 Top 20 genes significantly associated to Lysosome . . . . . . . 131
6.5 Top 20 genes significantly associated to Lysosome Organization 132
6.6 Mesenchymal gene expression signature analysis . . . . . . . . 138
# List of Figures

1.1 Different levels of networks in Biology. .......................... 5

2.1 Example of influence network. ................................. 11

2.2 Systematic overview of the theory underlying different models for inferring gene regulatory networks .......................... 16

3.1 System identification process ........................................ 31

3.2 Flow chart to choose the most suitable network inference algorithms according to the problem to be addressed .................. 32

3.3 Core BANJO Objects .................................................. 33

3.4 Structure of the synthetic network .................................. 54

3.5 Application of Banjo on IRMA data. .............................. 64

3.6 Application of ARACNe on IRMA data. .......................... 65

3.7 Application of Nir on IRMA data. ................................. 66

4.1 Parallel processor organization ....................................... 74

4.2 Gene network inference performances on a set of simulated expression profiles ......................................................... 77
5.1 Comparison of the results changing the number of bins . . . . 84
5.2 Fitting of a Gamma distribution over the human MIs . . . . 86
5.3 In-silico validation of the Human network . . . . . . . . . . 88
5.4 Relation between human gene connection degrees and expres-
sion levels . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 91
5.5 Relation between human gene connection degree and protein
dosage sensitivity . . . . . . . . . . . . . . . . . . . . . . . . . 92
5.6 Fitting of a Gamma distribution over the mouse MIs . . . . 93
5.7 Relation between mouse gene connection degrees and gene ex-
pression levels . . . . . . . . . . . . . . . . . . . . . . . . . . . . 95
5.8 Relation between mouse gene connection degree and protein
dosage sensitivity . . . . . . . . . . . . . . . . . . . . . . . . . 96
5.9 Modular structure of the human network . . . . . . . . . . . 98
5.10 Modular structure of the mouse network . . . . . . . . . . . 99
5.11 Communities validation through gene ontology analysis . . . 101
5.12 Distance matrix update example with $p = 4$ processes . . . 106
5.13 Parallel hierarchical clustering algorithm . . . . . . . . . . . 107
5.14 Community-wise human network . . . . . . . . . . . . . . . 108
5.15 Community-wise mouse network . . . . . . . . . . . . . . . 110
5.16 Percentage of mouse conserved human connections . . . . . 111

6.1 Subnetworks obtained by collecting the top 1000 connections
with the highest MI within the network . . . . . . . . . . . . . 116
6.2 Connection tendency and genomic distance . . . . . . . . . . 120
6.3 Intra-community physical contact . . . . . . . . . . . . . . . . 122
6.4 Genes that are co-expressed tend to be physically close at the 3D chromatin level ........................................... 123
6.5 Biological validation of predicted mitochondrion protein localizations ........................................... 128
6.6 GRN is involved in lysosomal biogenesis and function ........................................... 133
B.1 Parallel mutual information algorithm ........................................... 151
Chapter 1

Introduction

The study of individual gene and protein function has been largely the main focus of traditional biology known as “reductionist” approach. With the advent of new technologies, particularly high-throughput technologies, have considerable changed the traditional approach. Nowadays, the aim is to develop computational tools able to integrate different sources of information and to put side-by-side biology and mathematics.

The common believe that one gene was responsible of one disease is definitively death. Life is so complicated that it is impossible for one gene to be solely responsible for one function and therefore one disease. Almost all genes identified have multiple domains (function units) with different “potential” function. So, mess up one gene would certainly have more than one consequences. This has been confirmed in many species, from bakers yeast to human. So, it is almost impossible to predict the occurrence of one disease just by analyzing the function/structure of one gene or its product. It has to
be a combination of various information. With that being said, is it helpful to analyze one gene if it is know that this particular gene (or its mutant formats) always associates with certain disease.

Modern biology often follows the “Holism”\(^1\) idea, which states that all the properties of a given system cannot be determined by its components parts alone. The general principle of holism was concisely summarized by Aristotle in the *Metaphysics*: “The whole is more than the sum of its parts”. Under this paradigm the “systems biology” area of research was born. Systems biology aims to study a biological system as a whole more than study its components alone, whose main challenge is the identification of gene regulatory networks transforming high-throughput datasets into biological insights about the underlying mechanism. The identification or “reverse-engineering” of the system allows to study one of its component, for example a gene or a protein, by considering it together with the other components of the system.

In a topological sense, a network is a set of nodes and a set of directed or undirected edges between the nodes. Biological networks exist at various levels as shown in Figure 1.1

- Transcription regulatory networks: Genes are the nodes and edges are directed. A gene serves as the source of a direct regulatory edge to a target gene by producing a RNA or protein molecule that function as a transcriptional activator or inhibitor of the target gene. If the gene is an activator, then it is the source of a positive regulatory connection; if an inhibitor, then it is the source of a negative regulatory connection.

\(^1\)Greek word meaning all, whole, entire, total
Chromatin Immunoprecipitation (ChIP) experiment is a way to detect the physical binding of transcription factor on the promoter of a gene. ChIP-chip experiment is a way of doing ChIP at large-scale level that detects binding at genome scale.

- Protein networks: Proteins are the nodes and edges are the interactions between two proteins. These networks are bidirectional (or undirected). Experimental way to infer protein-protein interaction network is by doing Yeast 2 hybrid experiment or Mouse 2 hybrid experiment. Proteins are connected if it has been experimentally demonstrated that they bind together.

- Metabolite networks: Metabolites are the nodes and an edge represent the reaction catalyzed by an enzyme between the two metabolites. This kind of network is directional.

The aim of this thesis is to develop a reverse-engineering algorithm able to infer gene-gene regulatory networks by analysing gene expression profiles.
from high-throughput technologies. The gene regulatory networks so obtained will elucidate transcriptome organisation, gene function and gene regulation in mammalian systems.

The thesis is organized as follows: In chapter 2, we gave the background of gene regulatory network and described various mathematical models used for inference of gene regulatory network. In chapter 3, we did a comparative study among different approaches to reverse-engineer using ready-to-use softwares from each class of models described in chapter 2. We tested these approaches on experimental data sets as well as on an *in-silico* data. In chapter 4, we described a novel approach to infer gene regulatory network using massive datasets made of gene expression profiles. In Chapter 5, we *in-silico* validated the inferences by comparing the results with known protein-protein and other types of interactions collected from literature. Moreover, we shown how to make use of the topology of the gene network to predict gene function. In chapter 6, we *experimentally* validated some of the predictions of the reverse-engineering algorithm in terms of protein-protein interactions and gene function prediction. In chapter 7, we concluded the thesis highlighting the limitation of the proposed approach and introduce a statistical model able to generalise the proposed algorithm.
Chapter 2

Introduction to reverse-engineering

Reverse-engineering techniques have principally focused on decoding the mechanisms of transcription control, the first step in gene expression. This is because DNA microarray technology has enabled researchers to efficiently measure the concentration of all RNA transcripts in a cell, making such data abundant. Measuring peptide, protein and metabolite regulators of gene expression is generally more difficult, and such data are not often available. But with improved technologies for protein and metabolite measurement, reverse-engineering techniques may be extended to the second and third steps of gene expression.

Reverse-engineering techniques can be divided in two main classes: “physical” and “influence” approaches. A physical approach seeks to identify the protein factors that regulate transcription, and the DNA motifs to which the
factors bind. In other words, it seeks to identify true physical interactions between regulatory proteins and their promoters. An advantage of this strategy is that it can reduce the dimensionality of the reverse-engineering problem by restricting possible regulators to TFs. The second strategy, which we call the “influence” approach, seeks to identify regulatory influences between RNA transcripts. In other words, it looks for transcripts that act as inputs whose concentration changes can explain the changes in “output” transcripts. Each transcript may act as both an input and an output. The input transcripts can be considered the regulators of transcription. By construction, such a model does not generally describe physical interactions between molecules since transcription is rarely controlled directly by RNA (and never by messenger RNA, which is the type of RNA predominantly measured by DNA microarrays). Thus, in general, the regulator transcripts may exert their effect indirectly through the action of proteins, metabolites and effects on the cell environment (Fig. 2.1). Nevertheless, in some cases, the regulator transcripts may encode the TFs that directly regulate transcription. In such cases, the influence model may accurately reflect a physical interaction. An advantage of the influence strategy is that the model can implicitly capture regulatory mechanisms at the protein and metabolite level that are not physically measured. That is, it is not restricted to describing only transcription factor/DNA interactions.

The identification of gene regulatory networks is crucial for understanding pathways and functions that take place within a cell. Gene regulatory networks can be inferred by analysing the transcriptional response of a pop-
ulation of cells in multiple experimental conditions. High-throughput technologies such as microarray and more recently next generation sequencing allow to measure quantitatively the expression levels of the genes under specific experimental conditions. The capability to “obtain” the fingerprint of a cell at a specific time and condition, together with the large number of expression data now available allow to use methods from engineering mathematics and statistics to explore and analyse gene expression data. In the following chapter we present different approaches to infer or “reverse-engineer” gene regulatory networks from gene expression profiles measured by microarray technology.

Assume that one is interested in analysing the response of a cell under certain conditions, for example when the expression of some of its genes is modified (or perturbed) by an external agent (i.e. drug, heat shock). The genes that are directly or indirectly regulated by the perturbed genes will then likely change their expression too. From an engineering point of view, knowledge of how genes’ expression change following the perturbation experiment allow to identify the network of regulatory interactions occurring among them. This identification process takes different names depending on the field of application, such as: system identification and reverse-engineering.

Here we focus our attention on the reverse-engineering of gene regulatory networks from expression profiles. Depending on the class of inference algorithm, the identification process allows to infer gene-gene relationships that can be both physical or functional [35]. Those based on the physical interaction aim to infer transcription factor - target genes interactions or
protein-protein interactions. Whereas functional, or influence interactions, aim to identify relationships between the expression profiles of the genes.

A functional interaction between two genes in a gene network does not necessarily imply a physical interaction, but can also refer to an indirect regulation via proteins, metabolites and ncRNA that have not been measured directly (Figure 2.1). We must mention that the concept of influence interaction is not well defined and strongly depend on the mathematical formalism the gene network is approximated with. Therefore in what follows we will refer to functional interactions by the term “connection”, whereas the term “interaction” will be used only when a physical interaction between the DNA, RNA or protein products of the genes is occurring.

We refer to a **gene network** as the collections of gene-gene influences captured from expression data. A gene network is not just a static list of edges (connections between genes) but it contains information about the topological organisation of its nodes (genes). For example, a community in a network of genes identify a group on genes that are highly connected among them and poorly connected with genes outside the group. Communities of genes can be used to detect the modularity of the cell, where groups of genes cooperate to accomplish to a specific function.

In what follows we will describe algorithms to infer gene-gene influence interactions (connections). A description of other methods based on the physical approach and more details on computational aspects can be found in [50, 5, 38, 89, 35, 13, 106].
Figure 2.1: Biological networks are regulated at many levels. (A) Shows an example network where transcription factors (blue and green shapes) influence the expression of different transcripts (brown lines). One protein is a membrane-bound metabolite transporter. The metabolite it imports (brown triangle) binds one of the transcription factors enabling it to bind DNA and initiate transcription. (B) A gene network model of the real network in (A). Because the model is inferred from measurements of transcripts only, it describes transcripts as directly influencing the level of each other, even though they do not physically interact.
2.1 Microarray technology and microarray data repositories

A DNA microarray (DNA chip) is a collection of small DNA oligomers laying on a solid surface of approximately 1 cm$^2$ (chip). DNA oligomers on the chip are organized in approximately 250,000 “spots” (depending on the chip model), and each spot contains millions of copy of the same DNA sequence, called probe. Microarrays are used to simultaneously measure the expression of thousands of genes starting from total RNA extracted from a population of cells.

Total RNA is converted into cDNA through reverse transcriptase and marked with a fluorescent marker. cDNA is then placed on the microarray chip and the complementarity between two fragments allows the hybridization of a cDNA sequence on the corresponding DNA “spot”. The number of hybridized probes in a spot is directly related to the expression level of the gene represented by the spot. Gene expression levels are quantified through fluorescence analysis, the higher the number of hybridized copy of a probe the higher the fluorescence level measured that is associated to that spot.

There exists many types of microarray all based on the same principles: the two color microarrays are used to measure both the gene expression levels of treated cells and gene expression levels of the control cells on the same chip; other types of microarray can be used to measure single nucleotide polymorphism, fusion genes, alternative splicing, and so on. Here, we concentrate on DNA single color microarray. Hereon we use the term hybridization and gene
expression profile (GEP) to refer to the set of gene expression levels collected on a single microarray chip. Moreover, we refer to a set of hybridizations with the term experiment.

The fluorescence levels that are collected from a microarray hybridization are called raw data. The hybridizations in the same experiment are usually “normalized”. The normalization process is a data processing step where the measurement noise is removed, the background fluorescence is subtracted and the average fluorescence level among the spots associated to the same probe is computed. The process yields a set of comparable gene expression profiles. We refer to a set of normalized GEPs (or experiment) with the term processed GEPs.

There exists two major GEPs repositories: Gene Expression Omibus (GEO [32]) and Array Express. Array Express [86] is a public repository of gene expression profiles described in literature. GEPs are logically organized into experiments. An experiment is a collection of GEPs (usually performed in a single laboratory) together with their METADATA to trace information such as the applied experimental protocol, type of samples (cell types or tissues), and all the other information MIAME standard [18] requires. Recently, data from GEO have been curated (METADATA curation) and imported in Array Express. Most of the experiments in Array Express come with the processed GEPs. Here, we downloaded and analyzed processed GEPs from Array Express repository.
2.2 Reverse-engineering

2.2.1 Bayesian networks

Definition 2.1 A Bayesian Network (BN) is a directed acyclic graph (DAG) \( G = (V, A) \) together with a set of local probability distributions \( P \). The vertices \( V \) correspond to variables, and the arcs or edges \( A \) represent probabilistic dependency between the variables. An arc from variable \( X \) to variable \( Y \) states a probabilistic dependence between the two variables, i.e. the state of \( Y \) depends on the state of \( X \). In this case, \( X \) is called a parent of \( Y \). A node with no parents is unconditional. \( P \) contains the local probability distributions of each node \( X \) conditioned on its parents.

A Bayesian Network provides a graphical representation of probabilistic relationships among a set of random variables \( X_i \), with \( i = 1 \ldots n \). An example of Bayesian network is provided in Fig. 2.2. A variable in a Bayesian network can be either discrete or continuous. Bayesian networks can handle incomplete data sets and the statistical nature of the formalism provides a framework to account for noisy data typical of biological experiments.

Owing to its advantages, researchers have devoted considerable attention in recent years to the use of Bayesian network approaches for reverse-engineering gene networks [40, 79, 96, 31, 118, 97, 46, 119, 99, 101].

A Bayesian network describes the relationship between variables at both qualitative and quantitative level. At a qualitative level, the relationships between variables \( X_i \) are simply dependence and conditional independence. These relations are encoded in the structure of a directed graph, \( G \), to achieve
a compact and interpretable representation. Vertices of the graph correspond to variables and directed edges between vertices represent dependencies between variables.

At a quantitative level, relations between variables are described by a family of joint probability distributions $P(X_1, \ldots, X_n)$ that are consistent with the independence assertions embedded in the graph $G$ and has the form:

$$P(X_1, \ldots, X_N) = \prod_{i=1}^{N} P(X_i = x_i \mid X_j = x_j, \ldots, X_{j+p} = x_{j+p})$$  (2.1)

where the $p + 1$ genes on which the probability is conditioned are called the *parents* of gene $i$ and represent its regulators, and the joint probability density is expressed as a product of conditional probabilities by applying the chain rule of probabilities and independence. This rule is based on the Bayes theorem:

$$P(A, B) = P(B \mid A)P(A) = P(A \mid B)P(B)$$  (2.2)

We observed that the JPD (joint probability distribution) can be decomposed as the product of conditional probabilities as in Eq. 2.1 only if the *Markov assumption* holds, i.e. each variable $X_i$ is independent of its non-descendants, given its parents in the directed acyclic graph $G$. A schematic overview of the theory underlying Bayesian networks is given in Figure 2.2.

The joint probability distribution, $P(A, \ldots, H)$, for the Bayesian network
2.2.1 Bayesian networks

Figure 2.2: Systematic overview of the theory underlying different models for inferring gene regulatory networks. Bayesian networks: A is conditionally independent from D and E given B and C; Information-Theoretic networks: Mutual information is 0 for statistically independent variables, and Data Processing Inequality helps pruning the network; Ordinary Differential Equations: Deterministic approach where the rate of transcription of gene A is a function (f) of the level of its direct causal regulators.

In order to reverse engineer a Bayesian network model of a gene network, we must find two sets of parameters: the conditional probability functions relating the state of the regulators to the state of the transcripts and the directed acyclic graph $G$ (i.e. the regulators of each transcript) that “best” describes the gene expression data $D$, where $D$ is assumed to be a steady-state data set.

The model-learning algorithm usually assumes a specific form of the conditional probability function. Any function can be used, including Boolean and linear functions. But there will be a tradeoff between model realism and
model simplicity. More realistic models will have more parameters, which will require more experimental data and greater computational effort to solve.

The network structure is usually determined using a heuristic search such as greedy-hill climbing approach, Markov Chain Monte Carlo method or simulated annealing (which do not guarantee convergence to the global optimal solution). Heuristics approaches are used because trying out all possible combinations of interactions among genes is a NP-hard problem. For each network structure $G$ visited in the search, the algorithm learns the maximum likelihood parameters for the conditional probability functions. It then computes a score that evaluates each graph $G$ (i.e., a possible network topology) with respect to the gene expression data $D$. The score can be defined using Bayes rule:

$$P(G | D) = \frac{P(D | G)P(G)}{P(D)}$$

(2.4)

where $P(G)$ can either contain some a priori knowledge on network structure, if available, or can be a constant non-informative prior, and $P(D | G)$ is a function, to be chosen by the algorithm, that evaluates the probability that the data $D$ has been generated by the graph $G$. The most popular scores are the Bayesian Information Criteria (BIC) and the Bayesian Dirichlet equivalence (BDe). Both scores incorporate a penalty for complexity to guard against over fitting of data. The BDe score is based on the full Bayesian posterior probability $P(G | D)$ and has an inherent penalty for complexity since it computes the marginal likelihood $P(D | G)$ by integrating the probability of the data over all possible parameters assigned to $G$. The BIC score is an asymptotic approximation to the BDe score that used an explicitly penalized
estimate to the likelihood. One then selects the highest-scoring network as the correct network.

In Bayesian networks, the learning problem is usually underdetermined and several high scoring networks are found. To address this problem, one can use model averaging or bootstrapping to select the most probable regulatory connection and to obtain confidence estimates for the connection. For example, if a particular interaction between two transcripts repeatedly occurs in high-scoring models, one gains confidence that this edge is a true dependency. Alternatively, one can augment an incomplete data set with prior information to help select the most likely model structure.

The main limitation of Bayesian networks is that they assume that the network structure is acyclic (i.e. no feedback loops). Dynamic Bayesian networks [118, 31, 79] overcome this limitation and can be used to infer cyclic phenomena such as feedback loops that are prevalent in biological systems. Dynamic Bayesian networks, an extension of Bayesian networks, are able to infer interactions from a dataset $D$ consisting of time-series data rather than steady-state data.

A word of caution: Bayesian networks model probabilistic dependencies among variables and not causality, i.e. the parents of a node are not necessarily also the direct causes of its behaviour. However, we can interpret the edges as a causal links if we assume that the Causal Markov Condition holds. This can be stated simply as: a variable $X$ is independent of every other variables (except the targets of $X$) conditional on all its direct causes. It is not known whether this assumption is a good approximation of what
happens in real biological networks.

For more information and a detailed study of Bayesian networks for gene network inference we refer the reader to [40].

2.2.2 Association Networks

Association networks connect transcripts that exhibit high statistical similarity (i.e. statistical dependence) in their responses in all experiments in the dataset. To measure similarity, algorithms often use Pearson correlation, which assumes linear dependence between variables, or mutual information, which makes no assumptions about the form of the dependence. If no assumptions are made, association networks are undirected. Algorithms in this class of model add a connection between all transcripts pairs with expression profiles that exceed a given threshold of similarity. This step, however, does not distinguish between direct and indirect relations. To address this problem, a pruning process can be undertaken to remove connections that are better explained by a more direct path through the graph.

Correlation

Association networks based on correlations often uses Pearson correlation coefficient between pair of transcript $X$ and $Y$ to compute similarity. This is computed as:

$$r_{XY} = \frac{\sum_{i=1}^{M} (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{M} (x_i - \bar{x})^2 \sum_{k=1}^{M} (y_k - \bar{y})^2}}$$

(2.5)
2.2.2 Association Networks

de la Fuente et al [27] used Pearson correlation or Spearman rank correlation to connect all similar transcripts. To prune the network i.e. to remove redundant connections, they used partial correlation coefficients. Partial correlation measures the correlation between two variables after subtracting the correlation between each variable and one or more additional variables.

**Mutual Information**

Consider a variable $X$ with $C$ possible states, $x_1, \ldots, x_C$ each with its corresponding probability $p(x_i)$. The average amount of information gained from a measurement that specifies one particular value $x_i$ is given by entropy $H(X)$ and is computed as:

$$H(X) = - \sum_{i=1}^{C} p(x_i) \log(p(x_i))$$

(2.6)

The entropy $H(X)$ has the following properties:

- If an outcome of the measurement is completely determined by $x_i$ i.e. the probability $p(x_i)$ is one and all other probabilities $p(x_j)$ with $i \neq j$ are zero, then $H(X) = 0$.

- For equiprobable events the entropy $H(X)$ is maximum and is given by:

$$p(x_i) = \frac{1}{C} \rightarrow H(X) = \log(C)$$

- Entropy remains unchanged when impossible events are added.
If the logarithm to base $C$ is used, the entropy is normalized i.e.

$$0 \leq H(X) \leq 1$$

The joint entropy $H(X,Y)$ of two discrete variables $X$ and $Y$, with $Y$ assuming values in the set \{y_1, \ldots, y_C\}, is given by:

$$H(X,Y) = - \sum_{i=1}^{C} \sum_{j=1}^{C} p(x_i, y_j) \log p(x_i, y_j)$$  \hspace{1cm} (2.7)

$p(x_i, y_j)$ denotes the joint probability that $X$ is in state $x_i$ and $Y$ in state $y_j$. If the systems $X$ and $Y$ are statistically independent the joint probability factorize and the joint entropy $H(X,Y)$ becomes:

$$H(X,Y) = H(X) + H(Y)$$  \hspace{1cm} (2.8)

The mutual information $MI(X,Y)$ or $MI_{XY}$ between variables $X$ and $Y$ is defined as:

$$MI(X,Y) = H(X) + H(Y) - H(X,Y) \geq 0$$  \hspace{1cm} (2.9)

Two of the main strategies to estimate $MI$ are:

1. Histogram technique: Consider a collection of $N$ simultaneous measurements of two continuous variables $X$ and $Y$. First, the range of values of each variable is calculated and then that range is divided into $M$ sub-ranges. The data are thus partitioned into $M$ discrete bins $a_i$
and \( k_i \) denotes the number of measurements of \( X \) that lie within bin \( a_i \). Probabilities \( p(a_i) \) are approximated by the corresponding relative frequencies of occurrence:

\[
p(a_i) \rightarrow \frac{k_i}{N}
\]

and the \( MI(X,Y) \) between two datasets \( X \) and \( Y \) may be expressed as:

\[
MI(X,Y) = \log N + \frac{1}{N} \sum_{ij} k_{ij} \log \frac{k_{ij}}{k_i k_j}
\]  \hspace{1cm} (2.10)

\( k_{ij} \) denotes the number of measurements where \( X \) lies in \( a_i \) and \( Y \) in \( b_j \).

2. Kernel Density Estimation: With a generalized kernel function \( K(X) \), the kernel density estimator \( \hat{f}(X) \) is given by:

\[
\hat{f}(X) = \frac{1}{Nh} \sum_{i=1}^{N} K \left( \frac{X - x_i}{h} \right)
\]  \hspace{1cm} (2.11)

The parameter \( h \) is called smoothing parameter or window width and the kernel function \( K(X) \) is required to be a normalized probability density. For Gaussian kernel, density estimate then reads:

\[
\hat{f}(X) = \frac{1}{N h \sqrt{2\pi}} \sum_{i=1}^{N} \exp \left( - \frac{(X - x_i)^2}{2h^2} \right)
\]  \hspace{1cm} (2.12)
For two-dimensional Gaussian kernel estimate is written as:

\[
\hat{f}(X,Y) = \frac{1}{2\pi Nh^2} \sum_{i=1}^{N} exp \left( -\frac{(X - x_i)^2 + (Y - y_i)^2}{2h^2} \right)
\]  

(2.13)

Appropriate value of \( h \) depends on the unknown density being estimate. Monte Carlo simulations can be used to obtain optimal value of \( h \) [12].

We refer reader to [104] for more details on MI and its estimation.

There are many algorithms which have successfully applied the association network based on MI [19, 12, 36] and shown its application in biological systems.

**Relevance Network**: Butte et al [19] showed the application of relevance network on 2467 genes in *Saccharomyces cerevisiae*. They estimated the MI pair-wise from 79 measurements using the histogram technique. Each transcript in network was thus completely connected to every other transcript with calculated mutual information. To prune the network, they chose a threshold mutual information (TMI), which was based on estimating average MI after permuting data n number of times and selecting the maximum MI. They selected an edge if its MI was higher than TMI.

**ARACNe**: Basso et al [12] showed the application of ARACNe on 7907 genes in Human B cells. The MI between each pair of genes was estimated using kernel density estimation. To select the smoothing parameter they used Monte Carlo simulation. The algorithm created an initial graph by connecting all transcript pairs with a mutual information value above a p-value computed again using Monte Carlo simulation. Final pruning of the network was done
using data processing inequality (DPI) principle that asserts that if both 
(x, y) and (y, z) are directly interacting and (x, z) is indirectly interacting 
through y, then $MI(x, z) \leq MI(x, y)$ and $MI(x, z) \leq MI(y, z)$. ARACNe 
is more detailed in Chapter 3 in section 3.1.2.

**CLR:** Faith et al [36] inferred the network of 4345 genes from 445 expression 
data in *Escherichia coli* using CLR which is an extension of relevance network. For pruning the network, CLR calculated the statistical likelihood 
of each MI value within its network context by computing the distribution 
of MI scores for all possible regulators of gene $i$ and distribution of MI scores 
for all possible targets of gene $i$.

The definition of MI requires each data point, i.e. each experiment, to be 
statistically independent from the others. Thus information-theoretic ap-
proaches, as described here, can deal with steady-state gene expression data 
set, or with time-series data as long as the sampling time is long enough to 
assume that each point is independent.

Edges in networks derived by information-theoretic approaches represent 
statistical dependences among gene expression profiles. As in the case of 
Bayesian network, the edge does not represent a direct causal interaction 
between two genes, but only a statistical dependency.

Theoretically, the main difference between MI and Pearson correlation co-
efficient is that MI can quantify also nonlinear dependencies between vari-
ables. Moreover, Pearson correlation cannot imply that two variables are 
statistically independent. In practical application, however, MI and Pearson
correlation may yield almost identical results [104].

In addition MI can be extended to more than two variables, whereas Pearson Correlation is limited to two variables.

### 2.2.3 Ordinary differential equations (ODEs)

Reverse-engineering algorithms based on ordinary differential equations (ODEs) [15, 41, 111, 117, 110, 107] relates changes in gene transcript concentration to each other and to an external perturbation. By external perturbation we mean an experimental treatment that can alter the transcription rate of some of the genes in the cell. An example of perturbation is the treatment with a chemical compound (i.e. a drug), or a genetic perturbation involving over-expression or down-regulation of one or more genes.

This is a deterministic approach not based on estimation of conditional probabilities, unlike Bayesian networks and association network approaches. A set of ordinary differential equations, one for each gene, describes the gene regulation as a function of other genes:

\[
\dot{x}_i(t) = f_i(x_1, \ldots, x_N, u, \theta_i) \tag{2.14}
\]

where \(\theta_i\) is a set of parameters describing interactions among genes (the edges of the graph), \(i = 1 \ldots N\), \(x_i(t)\) is the concentration of transcript \(i\) measured at time \(t\), \(\dot{x}_i(t)\) is the rate of transcription of transcript \(i\), \(N\) is the number of genes, and \(u\) is an external perturbation to the system. Since ODEs are deterministic, the interactions among genes (\(\theta_i\)) represent causal
interactions, and not statistical dependencies as the other methods.

An algorithm usually presupposes the form of the influence functions $f_i$ and nonlinear functions can lead to exponential rise in the unknown parameters to be estimated. Researchers have studied various functions, including sigmoidal functions [113], linear [29, 41, 23] and non-linear [44] functions.

Reverse-engineer a network using ODEs means to choose a functional form for $f_i$ and then to estimate the unknown parameters $\theta_i$ for each $i$ from the gene expression data $D$ using some optimization technique.

The easiest form that this function can assume is the linear form where Equation 2.14 becomes:

$$\dot{x}_i(t) = \sum_j \omega_{ij}x_j + p_i$$

(2.15)

where $\omega_{ij}$ represents the influence of transcript $j$ on transcript $i$ and $p_i$ is an externally applied perturbation to the level of transcript $i$. Linear functions has proven to be the most versatile in the analysis of experimental data sets [29, 41]. In part, this is due to the simplifying power of linear functions; they dramatically reduce the number of parameters needed to describe the influence function and avoid problems with overfitting. Thus, the amount of data required to solve a linear model is much less than that required by more complex nonlinear models. This advantage is crucial in light of the high cost of experimental data and the high dimensionality of the systems. On the other hand, linear functions do not show a rich variety of dynamic behaviour. They only have one isolated stationary state in which the temporal change
of transcript vanishes, once reaching this state the concentrations of the network components remain constant. Furthermore, linear model places strong constraint on the nature of regulatory interactions in the cell. Therefore, oscillations or multistationarity, which are both important properties of true biological networks, and are nonlinear phenomena, cannot be captured with linear models. Also, higher noise in the microarray data limits their application to make only qualitative statement and not quantitative statement about the underlying network.

ODE-based approaches can be applied to both steady state and time-series expression profiles. Advantage of ODE approaches is that once the parameters, $\theta_i$ for all $i$ are known, Equations 2.14 and 2.15 can be used to make predictions on the behaviour of the network to different conditions (i.e. gene knock-out, treatment with an external agent, etc.) [30].

2.2.4 Other Approaches

Clustering

Clustering, although not properly a network inference algorithm, is the current method of choice to visualize and analyze gene expression data. Clustering is based on the idea of grouping genes with similar expression profiles into clusters [33]. Clusters typically contain genes that function within related pathways or biological processes. It is therefore possible to assign functions to previously uncharacterized genes based on the functions of other genes in the same cluster.
**Boolean Networks**

Boolean networks [1, 2, 64, 49, 54] offer a binary, discrete-time description of a system. This model assumes that each gene is in two possible states, expressed or not expressed. Interactions among genes are modeled through Boolean logic functions. These models are quite simple and can be easily applied but the underlying assumption seems very unrealistic, in particular, modeling genes as discrete entities assuming one of only two states. Also the amount of data needed to infer the network increases as $2^n$, where $n$ is the number of genes.
Chapter 3

Comparison of

Reverse-engineering algorithms

In this chapter we test and compare different reverse-engineering algorithms. Each algorithm is based on a different mathematical or statistical model. The input data to the algorithms are the gene expression profiles and, in some cases, detail on the performed perturbation experiments. The output is a gene regulatory network where each node is a gene and each edge express the influence of one gene on the other. Part of the work described in this chapter has been published in [9].

Generally speaking the identification of a system involves three main entities [69],

1. A collection of experimental data.

These are usually expression data, which are generated through microarray hybridisation experiments. The data contain the information
that allow the identification, or reverse-engineering, of the system that generated them. Gene expression profiles obtained by microarrays are often referred to as observations.

2. A set of candidate models.

The set of candidate models is obtained by specifying the collection of models that can possibly describe the observed data. In this step, it is possible to account for the prior knowledge on the characteristics of the biological system. Each model comes with a set of parameters that can be learned from the data according to how well they explain the observations.

3. A set of rules that allow to accept or reject a candidate model.

The set of rules are applied over the set of candidate models. The rules are usually expressed in terms of an evaluation function (or objective function) used to check the accuracy of the models in fitting the experimental observations.

Once the model is selected, a model validation step is performed. This step allows to check for the “goodness of fit” of the model to the experimental observations. Figure 3.1 reports the classical system identification process.

3.1 Gene network inference algorithms

Hereon we indicate with $x_i$ the set of gene expression measurements of gene $i$; with $D$, the set of all the expressions of all the genes, and with $a_{ij}$ a connec-
Figure 3.1: System identification process.

The choice of the algorithm to reverse-engineer a gene network depends of the type of data available (time-series or steady state). The choice of the algorithm also determines the type of network that is possible to infer (directed or undirected, weighted or not, cyclic or acyclic). An overview of current softwares along with their range of applicability is reported in Figure 3.2 and in Table 3.1, respectively.

3.1.1 BANJO

Revers-engineering a Bayesian network means to search for the directed acyclic graph, $G$, that better describes the influence relationships hidden into the data, $D$, as detailed in Section 2.2.1. Figure 3.3 reports the steps
Figure 3.2: Flow chart to choose the most suitable network inference algorithms according to the problem to be addressed. (*): check for independence of time points (see text for details); (BN): Bayesian Networks; (DBN): Dynamic Bayesian Networks.

<table>
<thead>
<tr>
<th>Software</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>BANJO</td>
<td>S/D</td>
</tr>
<tr>
<td><a href="http://www.cs.duke.edu/~amink/software/banjo">www.cs.duke.edu/~amink/software/banjo</a></td>
<td></td>
</tr>
<tr>
<td>ARACNe</td>
<td>S/D</td>
</tr>
<tr>
<td>NIR</td>
<td>S</td>
</tr>
<tr>
<td><a href="http://dibernardo.tigem.it">http://dibernardo.tigem.it</a></td>
<td></td>
</tr>
<tr>
<td>Hierarchical Clustering</td>
<td>S/D</td>
</tr>
<tr>
<td><a href="http://bonsai.ims.u-tokyo.ac.jp/">http://bonsai.ims.u-tokyo.ac.jp/</a> mdehoon/software/cluster/</td>
<td></td>
</tr>
</tbody>
</table>

Table 3.1: Features of the network inference algorithms reviewed. S: steady-state; D: dynamic time-series.
followed by the Banjo [118] inference process.

The “searcher” is the core of the Banjo algorithm. The first step, **Proposer**, consists in selecting a graph structure, $G_{\text{rough}}$, to be evaluated according to the data. The strategies currently implemented in Banjo are the *simulated annealing* and the *hill climbing*.

The graph $G_{\text{rough}}$ is then scanned for cycles, **Cycle Checker**, and the acyclic graph $G$ generated. The two possible strategies to visit the graph and remove the cycles are the *Breadth First Search* and *Depth First Search*.

The acyclic graph $G$ is then evaluated, **Evaluator**, according to a scoring function. Banjo implements a Bayesian score, defined as the logarithm of the
probability of the proposed graph model given the data,

\[
BayesianScore(G) = \arg \max_G \{ \log p(G \mid D) \} = \arg \max_G \{ \log p(G) + \log p(D \mid G) - \log p(D) \} \approx \arg \max_G \{ \log p(D \mid G) \}
\]

where Eq. 3.2 is a good approximation of Eq. 3.1 under the hypothesis of equiprobability both over the graph models and over the different data configurations.

The Decider decides, possibly stochastically, whether to accept the proposed network (as the new current network) and best scored networks are then reported.

### 3.1.2 ARACNe

ARACNe (Algorithm for the Reconstruction of Accurate Cellular Networks) [12, 72] belongs to the family of association networks for identifying transcriptional interactions between gene products. Relevance networks algorithms, first proposed in [19] (see also [36]), are described in Section 2.2.2.

The computational core of ARACNe consists in the evaluation of pairwise Mutual Informations in Equation 2.9 between each pair of genes of interest. The computation of Equation 2.9 requires the knowledge of the marginal and joint probability distributions of the two genes X and Y that have to be estimated from the expression profile. In [14] various methods used to estimate probability distributions from observations are discussed.
ARACNe implements a Gaussian Kernel estimator first reported in [103] that defines a function $\hat{f}$ in terms of a Gaussian kernel function known as Kernel Estimator,

$$\hat{f}(x) = \frac{1}{M} \frac{1}{h\sqrt{2\pi}} \sum_{i=1}^{M} \exp\left(-\frac{(x - x_i)^2}{2h^2}\right), \quad (3.3)$$

where $M$ is the number of microarray hybridisations. This approach has a straightforward graphic explanation; it consists in drawing $M$ “bumps”, one for each observation, with mean $x$ and a priori fixed height. A classical gaussian shaped plot is than obtained by summing up the bumps. After the joint probability distributions are estimated ARACNe computes the MI between $X$ and $Y$ as

$$\hat{MI}(X; Y) = \frac{1}{M} \sum_{i=1}^{M} \log \frac{\hat{f}(x_i, y_i)}{\hat{f}(x_i)\hat{f}(y_i)}, \quad (3.4)$$

and uses all the pairwise MIs to build up the gene network. We remind that $MI(X, Y) = 0$ if and only if $X$ and $Y$ are statistically independent. In experimental setting, the estimated MI never equals zero. Under this scenario the recovered gene network would be full connected (each gene is connected to all the other genes of the network). To remove redundant hence false predicted connections among genes ARACNe implements a bootstrap strategy that allows to compute a random MI given the number of observations. This approach allows to set a threshold that discriminates between statistically dependent and independent pairs of genes, given the data.

The threshold over the MIs allows to remove most of the false positives
predicted interactions. Biological networks are very sparse and usually the pruning strategy just described is not sufficient. There are cases where, for example, a common regulator of two genes forces the two regulated genes to have an high MI. Consider the network scheme in Figure 2.2, gene $B$ and $C$ will have an high MI score even though they are not directly connected. The information “flows” between $B$ and $C$ through $A$, hence there is a common shared information (i.e. MI) between $B$ and $C$, but the interaction is not direct.

The false predicted connections should be kept only when the interest is on the functional relationship between the genes. In this case the recovered network will preserve the connections between the genes whose expression profiles coherently change across a set of hybridisations. ARACNe implements a pruning strategy based on the concept of “Data Processing Inequality” (DPI), in order to remove indirect interactions.

**Definition 3.1** The data processing inequality in information theory states that given three random variables $X$, $Y$ and $Z$ which form a Markov chain in the order $X \rightarrow Y \rightarrow Z$, then the mutual information between $X$ and $Y$ is greater than or equal to the mutual information between $X$ and $Z$. That is $MI(X;Y) \geq MI(X;Z)$.

The ARACNe implementation of the DPI scans all the full connected triplets of genes in the network and removes the recovered connection with lowest MI (according to a certain tolerance).

The main limitation of the ARACNe algorithm is that it requires a set of comparable gene expression profiles, i.e. all the expressions it uses to
reverse-engineer a gene network must be normalised together. This constitutes a problem in presence of an heterogeneous set of data (different tissues, different laboratories and so on). In Chapter 4 we propose a new reverse-engineering algorithm that overcomes this limitation and thus can be applied to massive and heterogeneous dataset.

3.1.3 NIR

ODE-based algorithms have been developed (Network Identification by multiple Regression, NIR) that use a series of steady state gene expression profiles to reconstruct gene regulatory networks [41].

The network is described as a system of linear ordinary differential equations [25] representing the rate of synthesis of a transcript as a function of the concentrations of every other transcript in a cell and the external perturbation:

\[ \dot{x}_i(t_k) = \sum_{j=1}^{N} a_{ij} x_j(t_k) + b_i u(t_k), \]  

(3.5)

where \(i = 1 \ldots N; k = 1 \ldots M\), \(N\) is the number of genes, \(M\) is the number of time-points, \(x_i(t_k)\) is the concentration of transcript \(i\) measured at time \(t_k\); \(\dot{x}_i(t_k)\) is the rate of change of concentration of gene \(i\) at time \(t_k\), i.e. the first derivative of the mRNA concentration of gene \(i\) measured at time \(t_k\); \(a_{ij}\) represents the influence of gene \(j\) on gene \(i\); \(b_i\) represents the effect of the external perturbation on \(x_i\) and \(u(t_k)\) represents the external perturbation at time \(t_k\).

In the case of steady-state data, \(\dot{x}_i(t_k) = 0\) and Eq. 3.5 for gene \(i\) becomes
3.1.3 NIR

independent of time and can be simplified and rewritten in the form of a linear regression:

\[ \sum_{j=1}^{N} a_{ij} x_j = -b_i u \]  

(3.6)

The NIR algorithm [41] computes the edges \( a_{ij} \) from steady-state gene expression data using Eq. (3.6). NIR needs as input: the gene expression profiles following each perturbation experiment \( (x_j) \), knowledge of which genes have been directly perturbed in each perturbation experiment \( (b_i u) \) and optionally, the standard deviation of replicate measurements. NIR is based on a network sparsity assumption, i.e. a maximum number of ingoing edges per gene (maximum number of regulators per gene), which can be chosen by the user. The output is in matrix format where each element is the edge \( a_{ij} \). The inference algorithm reduces to solving Eq. (3.6) for the unknown parameters \( a_{ij} \), i.e. a classic linear regression problem.

Other algorithms based on ODEs have been proposed in literature [15, 110, 29, 107].

The NIR parallel version and its implementation

In proximity of the steady-state the concentrations of the \( N \) transcripts do not change in time \( (\dot{x}_i(t) = 0 \ \forall \ i) \) so that the (3.6) can be rewritten as:

\[ a_i^T x = -b_i u_i, \ i = 1, \ldots, N. \]  

(3.7)

Let us suppose that we have conducted \( M \) experiments such that we know the genes directly perturbed \( (u_i(k), \ k = 1, \ldots, M) \) as well as the expression
profiles following the experiments (transcript concentration levels from microarray data $x_j(k), k = 1, \ldots, M$). We can then solve the equation (3.7) for the unknown parameters $a_{ij}$, and thus obtain the ingoing edges per gene.

NIR applies the multiple linear regression method to estimate the unknown model parameters. It relies on the assumptions that the data $x$ are realizations of a normally distributed random variable with known variances and the perturbations, $u$, are random variables, also normally distributed with known variances. Generally, the response $y$ may be related to $k$ regressors and the model

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_k x_k,$$

(3.8)

is termed a multiple linear regression with $k$ regressors.

Having $M$ experiments (response observation points) at our disposal, the model for each gene of the network becomes:

$$y_i = \beta_0 + \sum_{j=1}^{k} \beta_j x_{ij}, \quad i = 1 \ldots M.$$

(3.9)

The response ($y$) is given by the experimental perturbation values $u_i \in \mathbb{R}^{1 \times M}$, the regression variable values ($X$) are given by the concentrations of the gene transcripts and the regression variable parameters are given by the components of the vector $a_i$, so that, in matrix form, the model becomes:

$$-u_i^T = u_i^T X, \quad i = 1 \ldots N,$$

(3.10)
with $X \in \mathbb{R}^{N \times M}$. The regression analysis aims to best-fit the data by estimating the parameters of the model.

NIR estimates the parameters of the regression variables for each gene, using the least squares method. These are the values for which the first derivative of the residual sum square function is zero:

$$\hat{a}_i = -u_i X^T (X X^T)^{-1},$$  \hspace{1cm} (3.11)

under the assumption that the regressors are linearly independent.

Biological networks are sparse [82], thus NIR adopts the sparsity assumption that imposes an upper bound on the number of ingoing edges per gene (i.e. maximum number of regulators per gene), $\text{restk} < N$, which can be chosen by the user.

For each gene the $\text{restk}$ parameters that result in the smallest mean square deviation identify the $\text{restk}$ ingoing edges for that gene. The weight of the identified edges is given by the value of the estimated parameters. The choice of $\text{restk}$ affects either the sensitivity to measurement errors or the execution time. A low value of $\text{restk}$ induces an increase in the solution sensitivity to measurement errors. A high value prohibitively increases the computational time needed to identify the regulatory network due to the high number of the regressor combinations to be included in the model. This number is equal to the number of combinations without repetitions of $N$
objects taken $restk$ at a time:

$$D_{N,restk} = \frac{N!}{restk!(N-restk)!},$$  \hspace{1cm} (3.12)$$

This is polynomial of degree $restk$ in the number of genes. The exhaustive approach which evaluates the regression for each combination is not feasible for gene networks larger than 100 genes (with 100 genes and $restk = 10$ the number of combinations is of the order of $10^{13}$), thus NIR uses the following heuristic approach.

For each gene $i$:

- At the first step NIR computes (3.11) $N$ times by considering the regression variables one at time; the $topd$ variables for which the sum of the squared deviations is minimized are selected as possible ingoing edges for the gene.

- At the second step NIR computes (3.11) by considering the remaining $N - 1$ variables jointly with each of the first $topd$ selected ones, that is $\frac{topd(2N-topd-1)}{2}$ (Gauss formula) pairs of variables; the $topd$ pairs of variables for which the sum of the squared deviations is minimized are selected as possible pairs of ingoing edges for the gene.

- At step $k+1$ NIR computes (3.11) by considering the remaining $N - k$ variables jointly with each of the $topd$ sets of $k$ variables selected at the previous step, that is $\frac{topd(2N-2k-topd+3)}{2}$ sets of $k$ variables are considered; the $topd$ sets of $k + 1$ variables for which the sum of the
squared deviations is minimized are selected as possible sets of $k + 1$
ingoing edges for the gene.

- The process ends when the number of regression variables selected
reaches $restk$; the set of $restk$ variables for which the sum of the squared
deviations is minimized identifies the set of parameters $a_i$ corresponding
to the input regulations affecting expression profile of gene $i$.

The final output is an adjacency matrix, where each element is the edge $a_{ij}$,
that encodes the directed graph. The number of times (3.11) is calculated
for each gene is $O(restk \cdot topd \cdot N)$, so the overall number of times (3.11) is
calculated is $O(restk \cdot topd \cdot N^2)$. The computational complexity of (3.11) at
step $k + 1$, for the submatrix of $X$ whose rows correspond to a set of $k$ vari-
ables ($0 \leq k \leq restk - 1$), is $O(k^2N)$. The overall computational complexity
is therefore $O(topd(restk \cdot N)^3)$.

The NIR algorithm can be easily parallelized to handle large problems in
a computationally efficient manner by distributing the overall computational
burden among different processors to reduce the total execution time. In or-
der to address the high computational cost issue of the NIR algorithm we
have applied some specific implementation optimizations along with parallel
programming techniques.

The computational core of the NIR algorithm is the equation (3.11) where
$X$ is a submatrix of the gene expression matrix composed only of $k$ rows (with
$k = 1, 2, ..., restk$). From the matrix-matrix product definition applied to the
submatrix $X(v, v)$, with $v = [i_1, \ldots, i_k]$ vector of $k$ indices, it follows that:

$$X(v, v)(X(v, v))^T = (XX^T)(v, v).$$  \hspace{1cm} (3.13)

Therefore for each step of the algorithm, we don’t compute any matrix-matrix product operation $X(v, 1:N)(X(v, 1:N))^T$. On the contrary the product $XX^T$ is computed once and for all at the beginning of the program. At each step $k$, our implementation just selects the symmetric submatrix of $XX^T$ whose row and column indices correspond to the $k$ possible ingoing edges for the gene. Let $S$ be this submatrix of dimension $k$ stored in packed format.

In each experiment only one gene is perturbed. This implies that for each gene $i$ the perturbation vector $u_i$ is equal to $(0, \ldots, 0, 1_i, 0, \ldots, 0)$ and then the product $u_iX^T$ reduces to the $i$-th row of $X^T$. Denote this row by $r$.

$S$ is positive definite so we can apply the Cholesky factorization to the matrix $S$ in order to compute $\hat{a}_i$ as solution of the system of equations $S\hat{a}_i = -r$, thus avoiding the matrix inversion. We rely on the LAPACK routine DPPSV to solve this system of equations with a computational complexity of $O(k^3)$.

By avoiding the matrix product in (3.11), the parallel algorithm complexity is decreased by one order of magnitude: at the generic step the computational complexity is $O(k^3)$, the overall computational complexity is therefore $O(topd \cdot restk^4 \cdot N^2)$

The parallelization is implemented by assigning different genes to different computing processes: each process takes care of $N/p$ genes where $p$ is the
number of processes available. The computing steps described in the previous section can be performed independently for each gene so each process can compute the results for its genes independently without communication. The parallel algorithm has been implemented in C using the MPI standard. This work is partly described in [43].

3.2 In-silico and experimental data

Due to the lack of knowledge of all the real interactions between genes in a real biological network, it is very challenging to test gene regulatory network inference algorithms. Moreover, the availability and the heterogeneity of biological experiments do not always allow to reverse-engineer a network of interest. For these reasons we adopted a simulative environment to run the algorithms, test and compare their performances.

3.2.1 Generation of ‘In silico’ data

An in-silico gene regulatory network consists of a set of nodes that represent the genes and a set of edges that represent the interactions between the genes. There exist many implementations of network simulators such as [100, 26, 109] able to generate networks whose topology reflects the topology of a biological network (i.e. scale-free and small-world effect, [3, 11, 82]) with different mathematical models for simulating gene expression data.

To simulate gene expression data and gene regulations in the form of a network, we used linear ODEs relating the changes in gene transcript concen-
3.2.1 Generation of ‘In silico’ data

In-silico networks

The comparative analysis was performed by generating three different sets of 20 networks each, with different network dimensions (10, 100 and 1000) and different average in-degree (2 and 10). To generate each random network, we first created a random network using MATLAB© command `rss` that need as input the order of the model, $N$, the number of input and output of the system, $u$ and $y$ respectively. The `rss` command generates stable continuous-time state-space models,

\[
\dot{x} = Ax + Bu \\
y = Cx + Du
\]

where $x$ represents the state vector; $y$ is the output vector; $u$ is the input vector; $A$ is the state transition matrix which defines the dynamics of the
system; $B$ is the input coefficient matrix; $C$ is the output coefficient matrix; $D$ is the feedthrough (or feedforward) matrix. The $rss$ command generates $A$ (N x N), $B$ (N x 1), $C$ (N x N), $D$ (N x 1) which altogether define the state-space models [68].

The algorithm codified behind the $rss$ command, generates a network of size $N$, first it randomly chooses the number of complex and real eigen-values for matrix $A$ (complex eigen-values always exist in pairs, eigen-value and its complex conjugate). For real part of eigen-values, negative random numbers generated from normal distribution are taken and for imaginary part positive random numbers (again from normal distribution). These eigen-values are placed on the diagonal of a zero matrix, $Z$, of order $N \times N$. If the eigen-value pair is complex, then the real and imaginary part of the eigen-values and its complex conjugate are placed on block diagonal of $Z$. Another matrix, $T$, of size $N \times N$ is generated randomly from normal distribution and is then orthogonalised. Finally $A$ is computed as $T^{-1}ZT$. $B$ is generated randomly using normal distribution.

Once we generated the state-space model, we set $C$ equal to identity matrix and $D$ equal to a zero matrix so that the output of the system equals its states. Here $A$ represents the network which is a full rank matrix with eigen-values whose real part are less than 0 to ensure the stability of the dynamical system [68] (i.e. all the gene mRNAs reach an equilibrium between their transcription rate and degradation rate after a given time period). For each row of $A$, we then randomly selected $K$ elements (including the diagonal), and set the other elements to zero (to make the network sparse). We
3.2.1 Generation of ‘In silico’ data

then checked the stability of this new network by computing its eigen-values, and if $A$ was not stable we selected a different set of $K$ elements in each row unless we got a stable network $A$.

We used the above-described method to generate the network of size 10 and 100. But to generate network of 1000 genes, we used another approach. We first generated the network of 100 genes in the same way as described. We then expanded the network of 100 genes to obtain a network of 1000 genes. To do this, we replicated each row in $A$, 10 times, adding to each element noise of mean 0 and standard deviation equal to 10% of the absolute value of the corresponding element of the original row in $A$. While replicating, we removed the diagonal element from the original row and moved it to the diagonal position of the row where it is replicated. It means that each gene has self feedback loop. All the remaining elements in the expanded 1000x1000 matrix are filled with zeros. This is done to simulate clusters of genes that are co-regulated. In the final network of 1000 genes, each gene will be controlled only by the original 100 genes in $A$, and by self feedback, and not from the other genes in the same cluster. We then checked the stability of $A$ by computing its eigen-values that should be negative [68]. If matrix $A$ were not stable, we then repeated the above steps with different random noise until we got a stable network $A$. In this way, we generated a sparse network in which 100 genes are the main regulators controlling most of the remaining genes in the network, or in network theory words, they act as hubs. Remaining genes are on the network periphery and they do not control any genes but themselves.
3.2.1 Generation of ‘In silico’ data

**Perturbations**

For each network, we simulated steady state and time-series expression data.

- **Steady State** in which the measurement is taken once the system reaches the steady state after a perturbation.

- **Time Series** in which measurement is taken at $M$ different time points following a perturbation.

Matrix $B$ contains the information of which gene(s) is (are) perturbed in the network. $B$ has all its elements equal to 0 except for the gene(s) that is (are) perturbed. For all datasets, $M$ was chosen equal to 10, 100 and 1000 experiments.

We simulated microarray resulting from two kinds of perturbations, $P$

1. **Global** perturbation: all the genes are perturbed simultaneously in each perturbation experiment. We randomly generated the perturbation matrix $P$ of size $N \times M$ with values generated from Gaussian distribution with mean $\mu = 0$ and standard-deviation $\sigma = 1$. Global perturbations simulate environmental perturbations to the cell state, such as increase of temperature, drug treatment and so on.

2. **Local** perturbation: in each experiment a single different gene is perturbed. We selected $P$ equal to the identity matrix representing single gene perturbation in each experiment. In *local* perturbation, a maximum of $M = N$ experiments can be obtained. Local perturbations simulate, for example, single gene over-expression or knockdown.
3.2.1 Generation of \textit{`In silico’} data

\textbf{Steady State}

To simulate the data, we used Eq. 3.6. In compact form it is written as:

\[ AX = -BU \]  \hspace{1cm} (3.14)

or

\[ AX = -P \]  \hspace{1cm} (3.15)

where \( P \) is equal to \( BU \). Data matrix \( X \) is obtained by taking the inverse of \( A \) and left multiplying it with the perturbation matrix:

\[ X = -A^{-1}P \]  \hspace{1cm} (3.16)

\textbf{Time-series data}

Time-series simulated microarray data are generated by perturbing 10\% of the genes simultaneously in the network. The genes to be perturbed are chosen randomly and this information is stored in \( B \). \( U \) (1 x M) contains the information about what kind of perturbation is applied. In our simulations we applied a constant step perturbation of amplitude equal to 1. Once matrix \( A \) (N x N) and \( B \) (N x 1) were generated, we simulated the gene expression profile dataset \( X = [X(t_1), \ldots, X(t_M)] \) of equally sampled time points which was obtained using \textit{lsim} command in MATLAB\textsuperscript{©} by solving:

\[ \dot{X} = AX + BU \]  \hspace{1cm} (3.17)
lsim command simulates the time response of the state space model to the given input signal \( U \) and gives an output at the sampling time specified. Initial condition for all genes is set to 0. Time \( t_M \) was chosen equal to 4 times the inverse of the real part of the smallest eigen-value of \( A \) [68]. This ensures that at time \( t_M \) all the genes are close to their steady state value.

**Noise in the data**

Biological data are noisy. In a real scenario, the inference of a regulatory network is performed on noisy expression profiles. To conform the simulated data to real expressions we add noise to any generated gene expression. The noise follows a Gaussian distribution with zero mean and standard deviation proportional to the level of the expression simulated [41],

\[
X_n = X + \sigma_n \ | \ X \ | \odot Y
\]

(3.18)

where \( X_n \) is the noisy data, \( | X | \) represents the absolute values of the elements of \( X \), \( \odot \) refers to element wise product of two matrices, \( Y \) is a matrix of same size as \( X \) and is obtained from Gaussian distribution with mean \( \mu = 0 \) and standard-deviation \( \sigma = 1 \) (in the simulations \( \sigma = 0.1, \ 10\% \) of noise).

### 3.2.2 Experimental data

We selected three different organisms collecting six dataset of gene expression data. Table 3.2 lists the organisms selected along with the details of the
3.2.2 Experimental data

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Cell/Organism</th>
<th>eX</th>
<th>Samples</th>
<th>Genes</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Human B cells</td>
<td>S</td>
<td>254</td>
<td>7907</td>
<td>26 MYC targets [12]</td>
</tr>
<tr>
<td>B</td>
<td>S. cerevisiae</td>
<td>S</td>
<td>300</td>
<td>6312</td>
<td>844 TF-gene interactions [63]</td>
</tr>
<tr>
<td>C</td>
<td>Human B cells</td>
<td>S</td>
<td>254</td>
<td>23</td>
<td>11 MYC targets+11 non-targets [12]</td>
</tr>
<tr>
<td>D</td>
<td>S. cerevisiae</td>
<td>S</td>
<td>300</td>
<td>90</td>
<td>subset of TF-gene interactions [63]</td>
</tr>
<tr>
<td>E</td>
<td>E. coli</td>
<td>S</td>
<td>9</td>
<td>9</td>
<td>9 gene network [41]</td>
</tr>
<tr>
<td>F</td>
<td>E. coli</td>
<td>T</td>
<td>6</td>
<td>9</td>
<td>9 gene network [41]</td>
</tr>
</tbody>
</table>

Table 3.2: Experimental datasets used as examples. S: steady-state; T: time-series.

dataset.

The datasets differ in the number of genes that ranges from tenths to thousands. This scenario helps in testing the algorithms in different real experimental settings. The largest datasets, in terms of number of genes, are the B-cell datasets (A and C) and S. cerevisiae’ one (B). As the reference “true network” we used 23 genes, 11 known targets of MYC, 11 that are not targets of MYC, and one is the MYC itself [12], to test the predictive performance of different algorithms.

*Saccharomyces cerevisiae* (dataset B and D) is a well studied organisms and many biological confirmed interactions are already catalogued. As a reference of true network we used 844 TF-target (transcription factor - target) interactions obtained from Chromatin immunoprecipitatatin (ChIP) experiments performed on microarrays containing a genome-wide set of yeast promoter regions. These experiments were performed by Lee et al [63]. As in the case of human B-cells, we selected a subset of genes to test algorithms on a system of smaller dimension.

*Escherichia coli* dataset contains the expressions of the genes that are
involved in the SOS DNA repair pathway of this organism [41]. The steady state expressions, were obtained by over-expressing different genes with an arabinose-controlled episomal expression plasmid. Cells were grown under constant physiological conditions to their steady state (~5.5 hours after the addition of arabinose) and change in expression relative to unperturbed cells was measured. In time-series expressions cells were induced with Norfloxacin drug and total RNA was extracted at the following time points: 0, 12, 24, 36, 48 and 60 minutes from the drug treatment (triplicate).

IRMA: In vivo assesment of Reverse-engineering and Modeling Approaches.

We also tested these reverse-engineering algorithms on a synthetic network (Fig. 3.4) built in *Saccharomyces cerevisiae* [20]. Five well-known genes were chosen in order to obtain a network containing different kinds of interactions, so that it can model both transcriptional and protein-protein interactions, positive and negative feedbacks and it can be turned on and off by changing growing conditions. In order to isolate the synthetic network from the cellular environment non-essential and non-redundant genes are selected and endogenous ones are deleted. The coding sequence of each gene was assembled with a non-self specific promoter in order to create the desired transcriptional interactions between the network’s members, and each cassette was integrated by homologous recombination within the locus of another gene obtaining its simultaneous deletion. Thus, the system was built in such a way that the cho-

\[1\] http://gardnerlab.bu.edu
sen promoters are regulated only by the transcription factors of the network and not by external gene products. The following genes have been chosen for the synthetic network:

- As activators and repressors encoding genes: CBF1, GAL4, SWI5, GAL80 and ASH1.
- As promoter genes: HO, MET16, GAL1-10 and ASH1.

These activators and repressors regulate each other with the desired network topology, shown in figure 3.4.

Gal4 activates GAL10 promoter only in the presence of galactose that releases the inhibitor Gal80 from Gal4; in contrast when cells are cultured in glucose Gal80 binds to Gal4 activation domain and thus inhibits GAL10 transcription.

The resulting synthetic network is a ‘switchable’ isolated system which can be easily turned on and off by changing the medium (galactose or glucose). Its behavior can be altered only by perturbing the expression of the network genes. This makes such a network a good model to test the performance of different modeling and inference strategy.

### 3.2.3 Assessing the performance of algorithms

Suppose to have two graphs with the same set of nodes, how can we define a distance between them? Such a distance could then be used to compare the algorithm performances by setting one of the graph as the true gene network while the other would be the predicted network. We have chosen two
Figure 3.4: Structure of the synthetic network. The genes are all non-essential. Each red box shows the promoter (pr), each blue one the coding sequence of the gene and each green ones the Tag. Solid green lines represent regulation (arrow: upregulation, minus sign: inhibition); dashed black lines represent translation. Ellipses represent protein products. The HO open reading frame was deleted by integrating the CBF1-GFP cassette downstream to HO endogenous promoter. SHE1, CBF1 and SWI5 loci were deleted by integration of MET16pr-GAL4, GAL1-10prSWI5-MYC9 and ASH1prGAL80-3xFlag, respectively. The deletion of SHE2, CBF1 and SWI5 loci has been done from 200-500 bp upstream of the start site to the Stop of each gene. The triple HA tag has been integrated before the Stop codon of ASH1 gene.
parameters to test and compare the predictions: one is the Positive predicted values (PPV), that measures the percentage of correct predicted interactions; the other is the Sensitivity that measures how much of the real network is covered by the predictions. The two parameters are computed from the True Positives, False Positives and False Negatives predictions as $PPV = TP/(TP + FP)$ and $Sensitivity = TP/(TP + FN)$.

We also consider as random the performance of an algorithm that randomly connect genes assuming a uniform probability distribution over the set of possible edges.

Some algorithms infer the network just as an undirected graph, and others as a directed and/or signed graph, thus, in order to facilitate comparison among algorithms, we computed PPV and Sensitivity by first transforming the real network (signed directed graph) and the inferred network (when directed and/or signed) in an undirected graph (labeled $u$ in the table).

If the algorithm infers a directed graph and/or a signed directed graph, we also compared PPV and Sensitivity in this case (labeled $d$ and $s$ in the table, respectively). While computing PPV and Sensitivity we did not include self-feedback loops (diagonal elements of the adjacency matrix) since all the genes in the simulated networks have self-feedback loops and this could be an advantage for some algorithms as NIR that always recover a network with self-feedbacks.

To transform the signed network to directed network in matrix form, we took the absolute value of the numbers in the weighted adjacency matrix. To transform directed network into undirected network, we symmetrize the
matrix. While computing PPV and Sensitivity for undirected network, we considered only the upper triangular matrix, i.e. only the upper half of the matrix, $A$, to avoid counting the connections twice.

To check if the algorithm has performed significantly better than random, we computed the $p$-value using a Binomial distribution to estimate the probability of getting the correct number of edges using a probability of success equal to the random probability.

**Dialogue for Reverse Engineering Assessment and Methods: testing performances**

The Dialogue for Reverse Engineering Assessment and Methods (DREAM\textsuperscript{1}) [90] is an annual meeting to compare and discuss new reverse-engineering methodologies. It is possible to run an algorithm on biological or simulated dataset, which are provided from the organisers. There are different challenges and one of these consists in reverse-engineering gene regulatory networks from gene expression profiles.

The algorithm performances are compared by applying two scoring metrics: the Area Under the ROC curve (AUROC), which summarizes the trade-off between the true positive prediction ratio and the false positive prediction ratio; and the Area Under the Precision Recall curve (AUPR) that summarizes the PPV-Sensitivity tradeoff. The predicted edges, sorted according to their significance, may only contain a subset of the true edges. The remaining (null predicted) true edges are then ranked randomly, and this procedure

\textsuperscript{1}http://compbio.mit.edu/recombsat/
3.3 Results: ‘in silico’ evaluation

All of the algorithms were run on all the datasets using default parameters (Table A in appendix). BANJO was not run on the 1000 genes dataset since it was crashing due to memory limitations, whereas NIR needed an excessively long computation time. The parallel version of NIR [43] overcome this limitation, as described in Section 3.1.3. Results of the comparison are described in Table 3.3.

Algorithm performances are higher when applied on local perturbations data (one gene perturbed at time), with respect to global perturbation data (multiple genes perturbed). Multiple perturbation data should carry more information with respect to single perturbation data due to the higher number of genes perturbed. However, results show that algorithms are more able to recover a signal from single perturbation data. Also the number of expression profiles are is an important parameter. When we try to recover a network by using very few hybridisation with respect to the number of genes, due to the weakness of the signal, performances are very low independently from the algorithm applied (only NIR is slightly better than random).

Despite the dataset configuration BANJO’s sensitivity (or network cover-
### 3.3 Results: ‘in silico’ evaluation

Table 3.3: Results of the application of network inference algorithms on the simulated Global perturbation. PPV: Positive Predicted Value; Se: Sensitivity. In bold the algorithms that perform significantly better than random, using as a random model a Binomial distribution.

<table>
<thead>
<tr>
<th>Datasets</th>
<th>ARACNe PPV/Se</th>
<th>BANJO PPV/Se</th>
<th>NIR PPV/Se</th>
<th>Clustering PPV/Se</th>
<th>Random PPV</th>
</tr>
</thead>
<tbody>
<tr>
<td>10x10</td>
<td>0.43(^u)/0.43(^u)</td>
<td>0.41(^u)/0.52(^u)</td>
<td>0.60(^u)/0.53(^u)</td>
<td>0.36(^u)/0.34(^u)</td>
<td>0.36(^u)</td>
</tr>
<tr>
<td>100x10</td>
<td>0.63(^u)/0.65(^u)</td>
<td>0.97(^u)/0.09(^d)</td>
<td>0.72(^u)/0.88(^u)</td>
<td>0.40(^u)/0.38(^u)</td>
<td>0.36(^u)</td>
</tr>
<tr>
<td>1000x10</td>
<td>0.21(^u)/0.12(^u)</td>
<td>0.20(^u)/0.04(^u)</td>
<td>0.26(^u)/0.04(^u)</td>
<td>0.20(^u)/0.12(^u)</td>
<td>0.19(^u)</td>
</tr>
<tr>
<td>10x100</td>
<td>0.29(^u)/0.24(^u)</td>
<td>0.71(^u)/0.00(^u)</td>
<td>0.63(^u)/0.17(^u)</td>
<td>0.00(^u)/0.00(^u)</td>
<td>0.19(^u)</td>
</tr>
<tr>
<td>100x100</td>
<td>0.57(^u)/0.44(^u)</td>
<td>0.99(^u)/0.05(^u)</td>
<td>0.93(^u)/0.84(^u)</td>
<td>0.27(^u)/0.17(^u)</td>
<td>0.27(^u)</td>
</tr>
<tr>
<td>1000x1000</td>
<td>0.04(^u)/0.23(^u)</td>
<td>0.07(^u)/0.01(^u)</td>
<td>0.17(^u)/0.02(^d)</td>
<td>0.10(^d)</td>
<td>0.05(^u)</td>
</tr>
</tbody>
</table>

Table 3.4: Results of the application of network inference algorithms on the simulated Local perturbation. PPV: Positive Predicted Value; Se: Sensitivity. In bold the algorithms that perform significantly better than random, using as a random model a Binomial distribution. (*) Results obtained running the parallel version of NIR [43]
### 3.3 Results: ‘in silico’ evaluation

Table 3.5: Results of the application of network inference algorithms on the simulated Time Series data. PPV: Positive Predicted Value; Se: Sensitivity. In bold the algorithms that perform significantly better than random, using as a random model a Binomial distribution.

<table>
<thead>
<tr>
<th>Datasets</th>
<th>ARACNe PPV/Se</th>
<th>BANJO PPV/Se</th>
<th>NIR PPV/Se</th>
<th>Clustering PPV/Se</th>
<th>Random PPV</th>
</tr>
</thead>
<tbody>
<tr>
<td>10x10</td>
<td>0.35(^u)/0.84(^u)</td>
<td>0.36(^u)/0.34(^u)</td>
<td>-</td>
<td>0.34(^u)/0.35(^u)</td>
<td>0.36(^u)</td>
</tr>
<tr>
<td></td>
<td>0.24(^d)/0.22(^d)</td>
<td>0.39(^s)/0.01(^s)</td>
<td></td>
<td></td>
<td>0.20(^d)</td>
</tr>
<tr>
<td>10x100</td>
<td>0.36(^u)/0.96(^u)</td>
<td>0.38(^u)/0.45(^u)</td>
<td>-</td>
<td>0.33(^u)/0.34(^u)</td>
<td>0.36(^u)</td>
</tr>
<tr>
<td></td>
<td>0.23(^d)/0.29(^d)</td>
<td>0.20(^s)/0.13(^s)</td>
<td></td>
<td></td>
<td>0.20(^d)</td>
</tr>
<tr>
<td>100x10</td>
<td>0.19(^u)/0.63(^u)</td>
<td>0.18(^u)/0.05(^u)</td>
<td>-</td>
<td>0.19(^u)/0.27(^u)</td>
<td>0.19(^u)</td>
</tr>
<tr>
<td></td>
<td>0.10(^d)/0.03(^d)</td>
<td>0.03(^s)/0.00(^s)</td>
<td></td>
<td></td>
<td>0.10(^d)</td>
</tr>
<tr>
<td>100x100</td>
<td>0.19(^u)/0.84(^u)</td>
<td>0.11(^u)/0.03(^u)</td>
<td>-</td>
<td>0.19(^u)/0.31(^u)</td>
<td>0.19(^u)</td>
</tr>
<tr>
<td></td>
<td>0.11(^d)/0.03(^d)</td>
<td>0.06(^s)/0.01(^s)</td>
<td></td>
<td></td>
<td>0.10(^d)</td>
</tr>
<tr>
<td>100x1000</td>
<td>0.19(^u)/0.87(^u)</td>
<td>0.20(^u)/0.03(^u)</td>
<td>-</td>
<td>0.19(^u)/0.32(^u)</td>
<td>0.19(^u)</td>
</tr>
<tr>
<td></td>
<td>0.11(^d)/0.02(^d)</td>
<td>0.06(^s)/0.01(^s)</td>
<td></td>
<td></td>
<td>0.10(^d)</td>
</tr>
<tr>
<td>1000x1000</td>
<td>0.02(^u)/0.96(^u)</td>
<td>-</td>
<td>-</td>
<td>0.02(^u)/0.37(^u)</td>
<td>0.02(^u)</td>
</tr>
</tbody>
</table>

Performance is again random for the time-series ‘dynamic’ dataset (Table 3.5), except for BANJO albeit with a very low Sensitivity. In this case, we run ARACNe as well, although the time-points cannot be assumed independent from each other. BANJO has been shown to work on dynamic data but needs a very high number of experiments (time-points) as compared to the number of genes [118].
3.3.1 Application of parallel NIR

We carried out two kind of tests: to measure the result accuracy of the algorithm and to measure the efficiency in terms of speed-up. In order to measure the result accuracy we ran the program, by using the ‘in-silico’ data discussed in Section 3.2.1, on 20 different networks counting 1000 genes, with 10 as average in degree per gene. ‘In-silico’ data are gene expression data generated by a computer model of the gene regulation that enable one to check the performance of algorithms against a perfectly known truth. For each network we generated 1000 experiments perturbing a different single gene at time (local steady-state data).

The program has been executed on 100 processors of an HP XC6000 Cluster with Itanium 2 biprocessors nodes and a Quadrics ELAN 4 network. On average it took 984 seconds to generate the results for each gene network. The program recovered most of the true interactions as shown in the table 3.4.

Parameters were set as, \(restk = 11\), \(topd = 50\). As shown in table 3.4, NIR performs better than ARACNe even in the case of 1000 gene networks.

Moreover we ran the program on a 2500 gene network, even though we didn’t have any other software predictions to compare the results with. We set the same values for the parameters \(restk\) and \(topd\) as before. It took around 12,450 seconds to generates the results and we obtained the following values for PPV and Sensitivity: 0.26\(^d\), 0.27\(^u\) and 0.10\(^d\), 0.11\(^u\) (random PPV: 0.008\(^u\) and 0.004\(^s\)), respectively. In order to measure the parallel efficiency we run the program for a 1000 gene network on different number of processors.
3.4 Results: Experimental evaluation

<table>
<thead>
<tr>
<th>number of procs</th>
<th>time (secs)</th>
<th>speed-up</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>98896</td>
<td>-</td>
</tr>
<tr>
<td>10</td>
<td>9725</td>
<td>10.1</td>
</tr>
<tr>
<td>20</td>
<td>4798</td>
<td>20.6</td>
</tr>
<tr>
<td>40</td>
<td>2406</td>
<td>41.1</td>
</tr>
<tr>
<td>60</td>
<td>1643</td>
<td>60.2</td>
</tr>
<tr>
<td>80</td>
<td>1259</td>
<td>78.6</td>
</tr>
<tr>
<td>100</td>
<td>969</td>
<td>102.0</td>
</tr>
</tbody>
</table>

Table 3.6: Total execution times in seconds

The execution times and speed-up are shown in table 3.6. To evaluate the parallel speed-up, we developed a serial version of the algorithm with the implementation optimizations discussed in the previous paragraph.

3.4 Results: Experimental evaluation

The algorithms have been tested on real expression data collected from literature. The details of the data are reported in Table 3.2. The results of the reverse-engineering is reported in Table 3.7. We were unable to run all the algorithms on all the dataset because of the limitations outlined in Section 3.3. However, we should point out that these results could not be suitable for algorithm comparison purposes due to the limited dataset dimensions and to the limited knowledge of the real biological networks.

Due to the dimensions of the dataset, ARACNe was the only software able to infer the network from dataset A and B. Dataset E and F are very small and come from dense biological networks forcing the random performance to be very high. NIR can only be applied on dataset E, because all the other datasets were missing the information regarding the target of the
3.4.1 Reverse Engineering the IRMA network

In order to reverse-engineer the IRMA network, we used gene expressions collected after having grown IRMA on two different media, one containing glucose (switch-off) and one containing galactose (switch-on). To generate steady-state data we perturbed this network of 5 genes by over-expressing each gene in a separate experiment (a gene at time). To over-express the genes, we expressed them under the control of a strong constitutive CPP promoter.

**Time-series** data were collected by growing IRMA cells both in glucose

<table>
<thead>
<tr>
<th>Datasets</th>
<th>ARACNe PPV/Se</th>
<th>BANJO PPV/Se</th>
<th>NIR PPV/Se</th>
<th>Clustering PPV/Se</th>
<th>Random PPV</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>0.00u/0.01u</td>
<td>-</td>
<td>-</td>
<td>0.00u/0.21u</td>
<td>0.00u</td>
</tr>
<tr>
<td>C</td>
<td>0.78u/0.64u</td>
<td>0.60u/0.27u</td>
<td>-</td>
<td>0.45u/0.91u</td>
<td>0.48u</td>
</tr>
<tr>
<td>D</td>
<td>0.07u/0.17u</td>
<td>0.17u/0.02u</td>
<td>-</td>
<td>0.11u/0.44u</td>
<td>0.02u</td>
</tr>
<tr>
<td>E</td>
<td>0.69u/0.34u</td>
<td>0.78u/0.44u</td>
<td>0.80u/0.88u</td>
<td>0.8u/0.63u</td>
<td>0.71u</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.67u/0.24d</td>
<td>0.74u/0.67d</td>
<td></td>
<td>0.63d</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.50u/0.02u</td>
<td>0.59u/0.53u</td>
<td></td>
<td>0.32s</td>
</tr>
<tr>
<td>F</td>
<td>0.75u/0.37u</td>
<td>0.73u/0.69u</td>
<td>-</td>
<td>0.90u/0.59u</td>
<td>0.71u</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.61u/0.39d</td>
<td></td>
<td></td>
<td>0.63d</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.00u/0.00u</td>
<td></td>
<td></td>
<td>0.32s</td>
</tr>
</tbody>
</table>

Table 3.7: Results of the application of network inference algorithms on the experimental datasets. PPV: Positive Predicted Value; Se: Sensitivity. In bold the algorithms that perform significantly better than random, using as a random model a Binomial distribution. Details concerning the datasets analysed are reported in Table 3.2.
3.4.1 Reverse Engineering the IRMA network

(switch-off) and galactose (switch-on). Measurements were taken every 20 minutes by using real time PCR over the transcript of the 5 genes.

**Application of the Banjo algorithm**

In order to reverse-engineer IRMA we applied Banjo to the “switch on” and “switch off” time-series data. Banjo recovers the Dynamic Bayesian Network that better describes the observed data. In order to estimate the joint probability distribution (see Section 2.2.1) of all the variables in the network, Banjo first discretizes the data using a *quantile discretization procedure*. The *Proposer/Sercher* strategies were set to *random local move* and *simulated annealing*, respectively. The amount of time Banjo uses to explore the Bayesian Network space was set to *one minute*. All the other parameters such as *reannealingTemperature*, *coolingFactor*, and so on, were left with their default values. Of course the parameter values were not arbitrary chosen; those values were selected as the best values (in terms of network inference accuracy) in [9]. As well as for time-series observations, Banjo is able to infer the Directed Acyclic Bayesian Network on steady state observations. We applied Banjo on both “Glucose steady-state”, “Galactose steady-state” dataset and to the switch on and switch off time-series. Results are reported in Figure 3.5.

**Application of the ARACNE algorithm**

In order to reverse-engineer IRMA we applied ARACNE on both “Glucose steady-state”, “Galactose steady-state” datasets, and concatenated them to
3.4.1 Reverse Engineering the IRMA network

Figure 3.5: Reverse-engineering IRMA from experimental data using the Bayesian Network approach. (A-B) Inferred network using Banjo for the switch on and switch off time-series. Gray lines: inferred interactions that are not present in the true network. PPV (Positive Predictive Value = \( \frac{TP}{TP + FP} \)) and Se (Sensitivity = \( \frac{TP}{TP + FN} \)) values for an unsigned directed graph. The random PPV is equal to 0.40. (C-D) Inferred network using the Banjo for the steady-state experimental data from network gene overexpression in cells grown in galactose or glucose medium, respectively.
3.4.1 Reverse Engineering the IRMA network

Figure 3.6: Reverse-engineering IRMA from experimental data using the Information-Theoretic approach. (A-B) Inferred network using ARACNE for the steady-state experimental data from network gene overexpression in cells grown in galactose or glucose medium, respectively. Grey lines: inferred interactions that are not present in the true network. PPV (Positive Predictive Value = $\frac{TP}{TP + FP}$) and Se (Sensitivity = $\frac{TP}{TP + FN}$) for an unsigned undirected graph. The random PPV for the unsigned undirected graph is equal to 0.70 (7/10).

obtain a larger one. Whereas, the lack of the statistical independence assumption for time-series data did not allow to run ARACNE on dataset containing time-series experimental data. All the parameter were left with their default values. For instance, Kernel width and Number of bins are automatically detected by the software; no threshold and p-value among both MI values and MI P-value were used, respectively; DPI tolerance to remove false positive “mirrored” connections was left to its default value, 0.15.

Application of the NIR algorithm

NIR [41] solves equation (3.6) to obtain the network matrix $A$ from gene expression data. We considered a fixed number of regressors for each gene
3.4.1 Reverse Engineering the IRMA network

ODE NETWORK INFERENCE (NIR)

Figure 3.7: Panel A (B) report the inferred interactions by reverse-engineering IRMA network from steady-state data when Galactose (Glucose) is provided to the cells.

\( k = 2 \), i.e. we assume that each gene can be regulated by a maximum of 2 other genes. The regressor set was chosen according the residual sum of square error (RSS) minimization criterion. The small size of the system (5 genes) allowed to exhaustively search the best regressors in the space of all the possible couples of genes. Perturbation values were set equal to 1. Results are reported in Figure 3.7.

**Computation of the random performance**

In order to compute the random PPV, that is the one of an algorithm that connects genes in the network randomly from a uniform distribution, we considered the expected value of an hypergeometrically distributed random
variable whose distribution function and expected value are, respectively:

\[ P_x = \binom{M}{x} \binom{N - M}{n - x}, \quad E[x] = M \binom{N - 1}{n - 1} = M \frac{n}{N}. \]

In our case, \( N \) is the number of possible edges in the network; \( M \) is the number of true edges, \( n \) is the number of predicted edges. Then we computed as random PPV:

\[ PPV_{rand} = \frac{TP_{rand}}{TP + FP} = \frac{E[x]}{n} = \frac{M}{N}. \]

Reverse-engineering algorithms can infer interactions with direction of regulation (A regulates B and not vice-versa, directed graph), or just an undirected interaction (A regulates B, or, B regulates A, undirected graph). For unsigned directed networks the value of the random PPV is equal to \( 8/20 = 0.4 \), for unsigned directed networks, it is equal to \( 7/10 = 0.70 \).

### 3.5 Discussion and Conclusion

We analysed and compared the performances of different algorithms on both in-silico and on experimental expression data. In particular concerning the simulated expression data we used two class of perturbations that we defined as “local” and “global”. As outlined in the result section, for the local
perturbation (Table 3.3) and the global perturbation (Table 3.4), there is a difference in the amount of information that is possible to store and recover by following the two types of perturbations. Algorithms perform better when only a small subset of genes (possibly only one gene) are perturbed at time. The less genes are perturbed, the clearer is the stored information, and the easier it is to recover. The noise should not depend on the type of perturbation. Due to the normal distribution we used, the noise from different measurements will sum up and cancels out.

Results on in-silico data are in line with respect to the results on experimental data. Since BANJO requires the estimation of a probability density function it can only be applied when many data points are available. Moreover, BANJO is very precise but the real network coverage of the inferences is very low. ARACNe performs generally better than BANJO, and even though it is not generally theoretically correct, can also be applied on time-series dataset. ARACNe computes pair-wise “distances”, and this allows to run it on larger dataset with respect to the other algorithms that relate a given gene to many others. NIR performs well also when the amount of data is small, it can be applied on steady-state data and performs well both on locally than on globally perturbed expressions.
Chapter 4

Reverse-engineering gene networks from massive and heterogeneous gene expression profiles
4.1 Introduction

Tens of thousands of protein-protein, protein-DNA and protein-RNA interactions have been experimentally identified in mammalian organisms [56, 114]. However, they constitute only a small part of the complex network of regulatory interactions occurring in a cell. Due to the time required to experimentally identify specific interactions, efforts have been made to infer gene regulatory networks directly from gene expression profiles, using a variety of “reverse-engineering” algorithms [21, 9, 10, 12, 28]. Among the plethora of different approaches to reverse-engineering, the most successful, and generally applicable, are those based on information-theoretic approaches [9]. The network among genes is reconstructed by considering pairs of genes and checking whether the two genes in each pair are co-expressed across the experimental dataset. Co-expression can be measured either by correlation, or by a more robust measure, called mutual information (MI). As described in Chapter 2.2.4 gene-gene “connection”, thus inferred, is not necessarily a physical interaction between the protein (or RNA) products of the two genes, but can also imply a functional, but indirect, regulation.

Reverse-engineering becomes much more powerful as the number of gene expression profiles used to infer the network increases as shown in Chapter 3.5 [9, 71]. However, the requirement of using homogeneous gene expression profiles (i.e. from a specific cell type, tissue or condition) typically limits their number to the order of hundreds. Indeed, it is common belief that reverse-engineering a gene network by combining gene expression profiles from heterogeneous samples will generate too much noise, thus hampering
the detection of biologically relevant signals. Our starting hypothesis was that, despite the extreme heterogeneity of gene expression profiles coming from different cell types, tissues, and conditions, it is indeed possible to infer a meaningful “consensus” gene regulatory network.

In this chapter we describe a new reverse-engineering algorithm able to use heterogeneous and massive sets of gene expression profiles to infer a gene network.

4.2 A new algorithm for reverse-engineering

4.2.1 Normalisation of Gene Expression Profiles

Gene expression profiles (GEPs) are usually organized in experiments. GEPs within experiments are normalized together using standard techniques described in 2.2.4. This procedure yields GEPs that are normalized within each experiment, but that are not normalized across experiments. This means that the expression level of a gene in two different experiments, although with the same absolute value, can refer to two different levels of expression. Normalised GEPs cannot therefore be merged across experiments. One way to obtain a single set of comparable GEPs from different experiments is to consider them as part of the same experiment, hence considering raw expression data for each GEP, and then normalise them all at once. Since standard normalisation techniques are both difficult to apply to large datasets and could yield incorrect results due the assumptions underlying these techniques, we decided to proceed differently.
In order to derive numerically comparable measures of gene expression for different experiments, we discretized normalized GEPs within each experiment via quantile discretization [67]. This method is based on equal frequency binning. Here, the expression values of all arrays within an experiment are discretized into a predetermined number of bins (three). The three bins, with equal number of values, are determined using the three quantiles of the normalized expression values as cut points. Each expression value is then replaced by an integer value corresponding to the bin it falls into.

The choice of the number of bins may affect the subsequent results of the reverse-engineering analysis, however as shown in Section BINNING SECTION this is not the case in our settings.

4.2.2 Mutual Information

As described in Chapters 2.2.4 and 3.5, mutual information (MI) is a pseudo distance between probability distributions; it measures the amount of information two random variables share. We decided to use a MI reverse-engineering approach due to the good advantages of this approach over the others reverse-engineering approaches as detailed in Chapter 3.5.

Genes can be seen as random variables and their profiles as a random process. Once gene expression profiles are properly discretized into bins, it is possible to compute MI for each couple of genes by merging all the discretized data together. Specifically, for each pair of probes, we considered two discrete random variables $I \in \{1, ..., 3\}$ and $J \in \{1, ..., 3\}$ respectively describing the
4.2.2 Mutual Information

discretised state of the two genes. In this context MI can be defined as:

\[ MI_{ij} = \sum_{i=1}^{3} \sum_{j=1}^{3} \pi_{ij} \log \frac{\pi_{ij}}{\pi_{i+} \pi_{+j}}, \]  
(4.1)

where \( \pi_{ij} \) represents the joint probability \( P(I = i, J = j) \) and \( \pi_{i+} = \sum_j \pi_{ij} \) and \( \pi_{+j} = \sum_i \pi_{ij} \) are respectively their marginal probabilities \( P(I = i) \) and \( P(J = j) \).

Let \( n_{ij} \) be the counts of the outcomes of pair \( (I = i, J = j) \) in the discretized sample set, then the frequency \( \hat{\pi}_{ij} := \frac{n_{ij}}{n} \) gives rise to a probability matrix \( \hat{\Pi} = (\hat{\pi}_{ij}) \) of dimension 3x3, where \( n \) is the number of experiments where both \( I \) and \( J \) appear. These values can be used as an estimate of the unknown true joint probabilities matrix \( \Pi = (\pi_{ij}) \). This leads to a point estimate of MI equal to

\[ \hat{MI}_{ij} = \sum_{ij} \frac{n_{ij}}{n} \log \frac{n_{ij}n}{n_{i+}n_{+j}}, \]  
(4.2)

Parallel implementation of the MI

The computational complexity of our algorithm is \( o(N^2 \cdot K) \) where \( N \) is the number of genes (or probes in the case of microarrays) and \( K \) is the total number of expression profiles. Typically, both the number of genes measured and the number of available GEPs in public repositories are in the order of \( 10^4 \). Due to the high computational cost, we decided to implement a parallel version of the algorithm to reduce computational time. The parallel algorithm distributes the gene expression profiles among several computing processes. Each process gets \( N/p \) probes where \( p \) is the number of processes available
and $N$ is the total number of probes. The processors are named $P_0$ to $P_{p-1}$ and logically organized in a topological ring where $i$ follows $j$ if $i > j$ and $P_0$ follows $P_{p-1}$.

Figure 4.1: Processors are organized in a topological ring. Data is passed among processors according to their topological organization.

At the beginning of the computation each process calculates the discretization of its own gene expression profiles and computes the mutual information for each pair of its $N/p$ probes. At the first communication step each process sends its probes to the following process and computes the MI for all the pairs of probes where the first is the local probe while the second is the received probe. At the $i^{th}$ communication step process $P_j$ receives probes from process $P_{(j-i)\mod(p)}$. The algorithm completes in $\lceil \frac{p}{2} \rceil$ communication steps. The parallel algorithm has been implemented in C using the MPI standard. The code is available in Appendix B. The algorithm has been executed on 105 processors of an HP XC6000 Cluster with Itanium 2 biprocessors nodes.
4.3 Application to simulated data

4.3.1 Simulated Dataset

We applied our reverse engineering algorithm on simulated in-silico gene expression data described in Chapter 3.5 [9]. Specifically, the set of in-silico datasets we used consists of 20 networks of 10, 100 and 1000 genes. For each network, we used a set of 100 expression profiles generated by perturbing one gene at the time (local perturbation) and a the set of 1000 gene expression profiles obtained via global perturbation, as reported in Section 3.2.1.

Moreover, in order to make this in-silico dataset comparable to real gene expression profiles coming from different experiments, and therefore with values not comparable across experiments, we proceeded as follows. For each gene network, we subdivided each set of the 1000 in-silico expression profiles into 10 different subsets of 100 expression profiles each. We then added a constant value to each of the subsets to simulate heterogeneity in the data. This dataset was mainly used to test the efficacy of the quantile discretization (see 4.2.1 for details). We refer to this simulated dataset as Dataset A.

A second set of simulated expression profiles (using a continuous differential equation model) were used to test and compare the inference with other available algorithms. This set of expression profiles we used to run the algorithms are part of the forth challenge of the third edition of the Dialogue for
Reverse Engineering Assessment and Methods (DREAM) [90]. In this set, data are generated from 5 networks of genes (sub-networks of biologically validated gene networks, 2 for E. Coli and 3 for S. Cerevisiae). Regulation dynamics were assumed to follow a thermodynamic model. Gaussian noise was added to the generated expression profiles.

Each network consists of 100 genes. Two sets of 101 expression profiles were generated for each network. The $j^{th}$ expression profile is simulated by deleting a copy of the $j^{th}$ from the gene network (heterozygous mutant); the $101^{th}$ expression profile corresponds to the response of the wild type (original) network. In a second version of this dataset, expression profiles are simulated by removing both the copies of a gene (homozygote null mutant). We refer to these datasets as Dataset B and C.

4.3.2 Performance on simulated data and comparison with state-of-the-art reverse-engineering algorithms

As described in Chapters 2.2.4 and 3.5, information theoretic approaches to reverse-engineering gene regulatory network have been first applied more than a decade ago [19]. The very first practical application came with ARACNe [73], successfully applied to infer the regulatory network in human B-Cells [12]. Our approach differs from ARACNe in the way of computing the MI: ARACNe uses the continuos version of the formula described in Section 2.2.2 and described in Section 3.1.2, where the distribution function is obtained by fitting a gaussian distribution over the set of expression values; our method applies the discrete version of the Equation 4.1.
4.3.2 Performance on simulated data and comparison with state-of-the-art reverse-engineering algorithms

We applied ARACNe [73] on Dataset A. Since ARACNe needs a single normalised dataset to run properly, and since each network in Dataset A consists of 10 different expression subsets, we first applied ARACNe on each subset and then considered the union of the gene networks thus inferred in further analysis. On Dataset A we also applied the new MI reverse-engineering approach described previously. We observed that the average precision of our approach across the 20 networks (True Positives/ (True Positive plus False Positive)) is 47%, whereas ARACNe reaches a precision of 21%. For comparison, when ARACNE is run on all the 1000 in-silico expression profiles together, its precision is 66% [9]. This means that ARACNE is very good at inferring gene networks but only if gene expression profiles are coming from homogeneous samples, where standard normalization procedures can be applied.
Both the approaches were also applied on Datasets B and C. The set of algorithm predictions are compared by applying two scoring metrics: the Area Under the ROC curve (AUROC) that summarizes the tradeoff between the true positive prediction ratio and the false positive prediction ratio; whereas, the Area Under the Precision Recall curve (AUPR) that summarizes the Precision-Recall tradeoff. The edges predicted, sorted according to their weights, may only contain a subset of the true edges. The remaining (null predicted) true edges are then ranked randomly, and this procedure gives rise to a p-value associated to both the AUROC ($P_{AUROC}$) and the AUPR ($P_{AUPR}$) scores. The 5 p-values (one for each network), for each metric, are then evaluated together by taking their geometric means. The final score is computed as $-\frac{1}{2} \log_{10} (P_{AUROC} \times P_{AUPR})$ [90].

Table 4.1: The performances of the two theoretical-information based algorithms were compared on a set of simulated expression profiles (Dataset B and C) [90]. We report the averaged performances obtained by analysing the heterozygous mutants and homozygote null mutant generated expression profiles. The “Pos” column contain the position of the two algorithms when compared with the performances of the other teams that participated at the DREAM3 (challenge 4).

<table>
<thead>
<tr>
<th>Score</th>
<th>Overall $P_{AUROC}$</th>
<th>Overall $P_{AUPR}$</th>
<th>Pos</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARACNe</td>
<td>7.38</td>
<td>0.26</td>
<td>$8.93 \times 10^{-11}$</td>
</tr>
<tr>
<td>our method</td>
<td>5.49</td>
<td>0.40</td>
<td>$8.93 \times 10^{-11}$</td>
</tr>
</tbody>
</table>

Results are reported in Table 4.1. Given that the real synthetic network is directed and the inferred network has no directionality, and due to the symmetric nature of the mutual information, we forced the real network to be directed before checking the algorithm performances.
It can be seen that even when expression profiles are optimal, that is data are not heterogeneous as in the case of simulated Datasets B and C, the two algorithm perform similarly and would rank among the top 10 in the DREAM competition.

4.4 Discussions and conclusions

Gene expression profiles are widely used as a collection of information to infer gene-gene and other types of regulations. The amount of biological data produced during the last decade was followed by a proliferation of reverse-engineering algorithm that make use of these data.

In Chapter 3 algorithms, under different assumption of network models, were tested and compared. Those algorithm can only be applied when the GEPs are comparable, that is part of the same experiment, thus strongly limiting the number of expression profiles that can be analysed. Here, we presented a novel algorithm able to use expression profiles, even when them are not comparable.
Chapter 5

Reverse-engineering of human and mouse regulatory networks
5.1 Introduction

In this chapter we show the application of the new reverse-engineering approach presented in Chapter 4.4. The algorithm was run on two very large dataset of GEPs. We show the reliability of the results by comparing the recovered gene networks with literature validated sets of gene-gene interactions. After the validation of the method we study few properties concerning the topology of the recovered network and we show how these results are in line with previous findings. Moreover, we apply a communities finding algorithm on the gene networks to discover groups of genes that are highly connected to each other and are part of the same biological function. This “component-wise” topological organisation of the genes in the gene in the networks reflects the modularity that is typical of the genes in the cells.

Towards this goal, we collected a massive and heterogeneous dataset of more than 20,000 gene expression profiles from a variety of human samples. We then developed and applied a mutual information (MI) “reverse-engineering” approach capable to infer a network from such a large dataset. We reconstructed a comprehensive gene network by inferring connections among more than 22,000 transcripts. We show that the resulting network can be used to discover protein-protein interactions, biological function and subcellular localisation of a protein, and to elucidate the function of a disease-gene.
5.2 Human and Mouse gene expression profiles

We collected 20,255 GEPs from 591 different experiments performed on a variety of human tissues, cell types, and conditions from a public microarray repository Array Express. A total of 22,283 different transcripts were measured, corresponding to the number of probes of the Affymetrix HG-U133A chip. From the same repository, we collected 8,895 GEPs from 614 experiment performed on a variety of mouse tissues, cell types, and conditions. Our aim was to exploit this massive datasets to yield consensus gene regulatory networks among the 22,283 human transcripts, and 45,101 mouse transcripts, using the reverse-engineering approach described in Chapter 4.4.

We applied MI that measures how coherently two genes change in their expression levels across a set of different conditions; MI equals zero when there is no coordinated expression between the two genes across the dataset as described in Chapter 2.2.4, due to the complexity of this massive dataset, we could not apply ‘off-the-shelf’ approaches to compute MI [12, 37], since gene expression profiles are not comparable to each other across experiments in different tissues, cell types or conditions.
5.3 Application of the reverse-engineering algorithm

The algorithm presented in Section 4.2 was applied to the set of GEPs presented in the previous section. The algorithm was applied to the set of GEPs from human and mouse separately. The algorithm produced a MI value for each pair of probes in each of the two species. MI is always greater or equal to zero. MI equals zero only when the expressions of two probes are statistical independent. It is rare to find a pair of probes whose MI is exactly zero. Probes almost always share a signal, even though imperceptible, due to pre-processing and to finite sample size, which prevent to compute the exact MI value.

One the algorithm parameter is the number of discretisation states (or bins) the expression data have to be discretised. It is possible to chose it arbitrarily according to data availability.

However the change in the results is not crucial as shown in Figure 5.1. Since the results are not changing significantly, and the computational complexity of the algorithm inference process depends on the number of bins used, and the computational cost increases quadratically on the number of bins we decided to use three as a good balance between these factors.

The number of probes in HG-U133A is 22,283. This means that the number of massorted pairs of probes is 248,254,903. The number of probes in Mouse430A.2 is 45,101, reveling in a number of massorted pairs of probes equal to 1,017,027,550. The reverse-engineering algorithm produced two com-
Figure 5.1: Comparison of the results changing the number of bins. Simulated network of size 100 generated in [9] were used for the comparison. Data used in panel A are related to the inference from locally perturbed GEPs (100 data points). Whereas, in panel B were used simulated GEPs related to the global perturbation (1000 data points). The plots report the ROC curve of the performances. The percentage of the positive predictions and the percentage of the coverage (sensitivity) over the interactions of the simulated networks are reported.
pletely connected undirected networks. The assumption of biological independence among most pairs of genes imply the statistical independence of their expression profiles. Under this assumption, we adopted a null hypothesis that allowed us to select a threshold over the MIs. This threshold was used to discriminate statistically significant gene-gene connections. Under the null hypothesis of independence, the distribution of the MI follows approximately a Gamma distribution [42, 52].

We therefore fitted a Gamma distribution to the values of the MI across all the probes’ pairs in human or mouse, using Maximum Likelihood estimation [45] are shown in Figures 5.2 and 5.6. We thus could assign a p-value to the MI of each gene pair and retained only those MI with a p-value < 0.01. This correspond to a MI of 0.04 for human and 0.025 for mouse.

We could have further pruned the networks by using one of the recently proposed schemes, such as the CLEAR method [37] and the Data Processing Inequality (DPI) method introduced by [73] and described in Section 3.1.2, but we decided against it, since we wanted to keep as many interactions as possible to have a broader overview of gene function and regulation. We were not interested in identifying ‘direct’ interactions as done by [12], but we focused on the identification of “co-expressed” genes, i.e. both direct and indirect interactions.

5.3.1 Human network

The number of significant connections is 4,817,629 connections (among 22,283 probes). To our knowledge, this is the largest and most comprehensive dataset
Figure 5.2: Fitting a Gamma distribution the histogram of the MIs recovered from human expression profiles along with the gamma pdf.

ever used to reverse-engineer the human gene regulatory network.

To validate at least a subset of predictions, we generated a reference human interactome, Golden Standard (GoS), consisting of 105,588 experimentally verified interactions including protein-protein [114], transcription factor-target gene, and metabolic interactions [56] (Table 5.1).

In order to validate the biological relevance of the predicted interactions of our MI approach, we built a golden standard (GoS) interaction network from the following interaction databases:

Reactome [56] is a curated knowledge-base of biological pathways; It represents a resource of core pathways and reactions in human biology. It collects a total of 32,821 metabolite-metabolite interactions\(^1\) reported in literature.

Cipher [114] is a tool to predict disease-genes on the bases of a computa-

\(^{1}\)number of interactions obtained by mapping the specific dataset identifiers over the identifiers complained within the HG-U133A microarray model
Table 5.1: Details of the Golden Standard Interactome. Number of interactions in the interaction databases used to build the Golden Standard, along with the cardinality of their intersection.

<table>
<thead>
<tr>
<th>ID</th>
<th>Source</th>
<th>Probewise</th>
<th>Probes</th>
<th>Genewise</th>
<th>UnigeneIDs</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Cipher+TissueSpecific</td>
<td>270,937</td>
<td>13,525</td>
<td>80,828</td>
<td>7,766</td>
</tr>
<tr>
<td>2</td>
<td>Reactome</td>
<td>113,867</td>
<td>3,303</td>
<td>32,821</td>
<td>1,759</td>
</tr>
<tr>
<td>1&amp;2</td>
<td>1&amp;2</td>
<td>350,116</td>
<td>13,911</td>
<td>102,112</td>
<td>7,980</td>
</tr>
</tbody>
</table>

5.3.1 Human network

Table 5.1: Details of the Golden Standard Interactome. Number of interactions in the interaction databases used to build the Golden Standard, along with the cardinality of their intersection.

Authors provide a protein interaction network assembled from the HPDR, BIND, MINT and OPHID protein-protein interaction databases. It accounts for 40,649 interactions.

**Tissue Specific Protein Interactions** [16] is a global human interaction network obtained by integrating data from 21 different sources to define a network of a total of 67,200 physical interactions.

Genes in the GoS network are identified with human UNIGENE IDs. On the other hand, microarray model HG-U133A identifies genes with probe IDs. In order to perform the comparison of the human interactome against the GoS network, we first mapped HG-U133A probes to 14,340 UNIGENE IDs using the annotation file provided by Affymetrix. Each gene-gene connection may correspond to multiple probe-wise connections because each gene can be associated to multiple probes. We therefore assigned to each gene-gene connection a MI value equal to the maximum MI among the corresponding probes-wise connections. The probe-oriented network was thus reduced to a gene-oriented one and made compatible with the GoS network.

Figure 5.3 shows the percentage of human predicted connections that
5.3.1 Human network

Figure 5.3: In-silico validation of the Human network compared with an experimental Golden Standard (GoS) interactome collected from different databases. The human network (blue-line) significantly outperforms correlation-based approaches (CoexpresDB dashed line). The Venn diagram reports the details of the interactions part of the GoS interactome.

were confirmed by the GoS interactome. Connections are ordered according to their MI, from the highest to the lowest. The network reaches a maximum of 90% of correct predictions, with an average precision of 32%. The percentage of correct connections, had these been randomly guessed, would have been equal to 0.0028%. The GoS interactome includes only a subset of the interactions occurring in a cell. Moreover, a high MI does not necessarily imply physical interactions, as reported in the GoS, but more often functional relationships. Despite these limitations, the GoS interactome provides evidence of the biological significance of the inferred connections.

Figure 5.3 also shows the percentage of correctly predicted connections by
CoexpresDB [83], a database of human co-expressed genes measured using a classic Pearson Correlation Coefficient (PCC). CoexpresDB performance is better than random, but much worse than our mutual information based approach.

The structure of the human network

The structure of the human network is typical of biological networks [81, 93]: it has an exponential degree distribution consisting of a large number of genes with very few connections, and very few genes with a large number of connections, termed hubs reported in Table 5.2.

We observed that, as the number of connections of a gene increases, so does its average expression level as reported in Figure 5.4.

In order to relate gene degree to the average expression level of a gene, we proceeded as follow: since GEPs are normalized within experiments, and not comparable across experiments, we could not use the whole dataset to estimate the average expression of each gene. To this end, we used 618 human gene expression profiles from Array Express [87] measuring expression across a variety normal tissues. GEPs were normalised together by applying the Affymetrix MAS5 algorithm. Average gene expression levels were then compared with gene connection degrees, i.e. the number of connections of a gene. Genes were first divided into equally sampled quantiles (500 each in human) then the average expression levels and the average gene connection degrees were computed. Due to the high degree variance in some of the quan-
5.3.1 Human network

<table>
<thead>
<tr>
<th>Probeset ID</th>
<th>Unigene ID</th>
<th>Gene Symbol</th>
<th>Connections</th>
</tr>
</thead>
<tbody>
<tr>
<td>220531_at</td>
<td>Hs.710464</td>
<td>FLJ14126</td>
<td>5733</td>
</tr>
<tr>
<td>215366_at</td>
<td>Hs.585343</td>
<td>SNX13</td>
<td>5449</td>
</tr>
<tr>
<td>207969_x_at</td>
<td>Hs.169222</td>
<td>ACRV1</td>
<td>5364</td>
</tr>
<tr>
<td>216292_at</td>
<td>Hs.677419</td>
<td>—</td>
<td>5326</td>
</tr>
<tr>
<td>216900_s_at</td>
<td>Hs.10734</td>
<td>CHRNA4</td>
<td>5267</td>
</tr>
<tr>
<td>207140_at</td>
<td>Hs.37009</td>
<td>ALPI</td>
<td>5262</td>
</tr>
<tr>
<td>207685_at</td>
<td>Hs.533022</td>
<td>CRYBB3</td>
<td>5156</td>
</tr>
<tr>
<td>210923_at</td>
<td>Hs.104637</td>
<td>SLC1A7</td>
<td>5105</td>
</tr>
<tr>
<td>216739_at</td>
<td>Hs.589088</td>
<td>—</td>
<td>5010</td>
</tr>
<tr>
<td>1-194_f_at</td>
<td>Hs.439056</td>
<td>CYP2A6</td>
<td>5002</td>
</tr>
<tr>
<td>214558_at</td>
<td>Hs.123034</td>
<td>GPR12</td>
<td>4989</td>
</tr>
<tr>
<td>215511_at</td>
<td>Hs.475018</td>
<td>TCF20</td>
<td>4979</td>
</tr>
<tr>
<td>215557_at</td>
<td>Hs.658129</td>
<td>—</td>
<td>4973</td>
</tr>
<tr>
<td>214923_at</td>
<td>Hs.712567</td>
<td>ATP6V1D</td>
<td>4957</td>
</tr>
<tr>
<td>220222_at</td>
<td>—</td>
<td>C8orf39</td>
<td>4919</td>
</tr>
<tr>
<td>32540_at</td>
<td>Hs.655661</td>
<td>PPP3CC</td>
<td>4918</td>
</tr>
<tr>
<td>211788_s_at</td>
<td>Hs.644635</td>
<td>TREX2</td>
<td>4870</td>
</tr>
<tr>
<td>206878_at</td>
<td>Hs.113227</td>
<td>DAO</td>
<td>4858</td>
</tr>
<tr>
<td>216440_at</td>
<td>Hs.658200</td>
<td>ERC1</td>
<td>4840</td>
</tr>
<tr>
<td>216159_s_at</td>
<td>Hs.654267</td>
<td>—</td>
<td>4798</td>
</tr>
<tr>
<td>208486_at</td>
<td>Hs.380681</td>
<td>DRD5</td>
<td>4778</td>
</tr>
<tr>
<td>34846_at</td>
<td>Hs.351887</td>
<td>CAMK2B</td>
<td>4720</td>
</tr>
<tr>
<td>215479_at</td>
<td>Hs.663736</td>
<td>—</td>
<td>4720</td>
</tr>
<tr>
<td>221466_at</td>
<td>Hs.673854</td>
<td>P2RY4</td>
<td>4715</td>
</tr>
<tr>
<td>207991_x_at</td>
<td>Hs.169222</td>
<td>ACRV1</td>
<td>4698</td>
</tr>
<tr>
<td>220671_at</td>
<td>Hs.639842</td>
<td>CCRN4L</td>
<td>4697</td>
</tr>
<tr>
<td>220826_at</td>
<td>Hs.677183</td>
<td>C2orf77</td>
<td>4670</td>
</tr>
<tr>
<td>202485_s_at</td>
<td>Hs.25674</td>
<td>MB2</td>
<td>4653</td>
</tr>
<tr>
<td>216437_at</td>
<td>Hs.677301</td>
<td>—</td>
<td>4638</td>
</tr>
<tr>
<td>221199_at</td>
<td>Hs.302025</td>
<td>GFRA4</td>
<td>4636</td>
</tr>
<tr>
<td>207960_at</td>
<td>—</td>
<td>—</td>
<td>4592</td>
</tr>
<tr>
<td>215246_at</td>
<td>Hs.642978</td>
<td>LARP7</td>
<td>4590</td>
</tr>
<tr>
<td>210271_at</td>
<td>Hs.322431</td>
<td>NEUROD2</td>
<td>4578</td>
</tr>
<tr>
<td>216116_at</td>
<td>Hs.655006</td>
<td>NCKIPS_D</td>
<td>4575</td>
</tr>
<tr>
<td>221378_at</td>
<td>Hs.248204</td>
<td>CER1</td>
<td>4570</td>
</tr>
<tr>
<td>206256_at</td>
<td>Hs.2246</td>
<td>CPN1</td>
<td>4566</td>
</tr>
<tr>
<td>211314_at</td>
<td>Hs.591169</td>
<td>CACNA1G</td>
<td>4563</td>
</tr>
<tr>
<td>221460_at</td>
<td>Hs.258574</td>
<td>OR2C1</td>
<td>4559</td>
</tr>
<tr>
<td>217293_at</td>
<td>—</td>
<td>—</td>
<td>4542</td>
</tr>
<tr>
<td>220082_at</td>
<td>Hs.192927</td>
<td>PPP1R14D</td>
<td>4529</td>
</tr>
</tbody>
</table>

Table 5.2: Human hub genes. Top 40 most connected human genes.
Figure 5.4: Relation between human gene connection degree (x-axis) versus gene expression level (y-axis). Genes were grouped in equally sampled (500 genes each) quantiles and the average gene connection degrees computed (x-axis). The log average expression level of the genes falling into each quantile is reported in the y-axis.

\[ y = 0.19x + 6.1 \]

Correlation = 0.94
\[ P = 6.4 \times 10^{-16} \]
tiles, we decided to keep only those human quantiles where the connection degree standard deviation was less 30.

On the contrary, the intrinsic protein disorder [66] of the protein product of a human gene significantly decreases ($P = 0.009$) as its connection degree increases, as shown in Figure 5.5. Protein disorder, defined as the length of the unstructured part of a protein, is an important determinant of gene dosage sensitivity [112].

In order to relate gene degree to the average protein disorder of a gene, we proceeded as before, but this time we used GlobPlot [66] to compute the protein disorder for each gene. We thus obtained a protein disorder score for 12,494 human genes (19,860 probes).
5.3.2 Mouse network

We repeated the same procedure described in Section 5.3.1 to infer the mouse network. The number of probes in Mouse430A.2 is 45,101. This means that the number of indirected pairs of probes are 1,017,027,550. However, the number of significant connection are 14,461,095 (Fig. 5.6).

The structure of the mouse network is very similar to the human network: it has an exponential degree distribution consisting of a large number of genes with very few connections, and very few genes with a large number of connections (hubs) reported in Table 5.3.

Figure 5.6: Fitting of a Gamma distribution over the MIs. The figure shows the histogram of the MIs recovered from mouse expression profiles along with the Gamma pdf.
Table 5.3: Mouse hub genes. Top 40 most connected mouse genes.
Figure 5.7: Relation between gene connection degree (x-axis) versus gene expression levels (y-axis) in mouse. Genes were grouped in equally sampled (1000 genes each in mouse) quantiles and the average gene connection degrees computed (x-axis). The log average expression level of the genes falling into each quantile is reported in the y-axis.

**Degree of connection and gene expression**

Also for the mouse network, the number of connections of a gene increases, so does its average expression level as reported in Figure 5.7.

In order to relate gene degree to the average expression level of a gene, we proceeded as follows: since GEPs are normalized within experiments, and not comparable across experiments, we could not use the whole dataset to estimate the average expression of each gene. To this end, we used 247 mouse expression profiles from Array Express [87] measuring expression of genes across normal tissues. GEPs were normalized together by applying MAS5 algorithm. Genes average expression levels were then compared with gene
5.3.2 Mouse network

connection degrees, i.e. the number of connections of a gene. Genes were first divided into equally sampled quantiles (1000 each in mouse) then the average expression levels and the average gene connection degrees were computed. Due to the high degree variance in some of the quantiles, we decided to keep only those mouse quantiles where the connection degree standard deviation was less 50.

As for the human network, we observed that the intrinsic protein disorder [66] of the protein product of a mouse gene significantly decreases ($P = 0.009$) as its connection degree increases, as shown in Figure 5.8. Protein disorder, defined as the length of the unstructured part of a protein, is an important determinant of gene dosage sensitivity [112].
5.4 The modular structure of the human and mouse networks

In order to relate gene degree to the average protein disorder of a gene, we used GlobPlot [66] to compute the protein disorder for each gene. We thus obtained a protein disorder score for 17,498 mouse genes (33,289 probes).

5.4 The modular structure of the human and mouse networks

A cell is able to regulate its complex behaviour thanks to groups of genes which perform different, but coordinated, functions to carry out a specific task. We asked whether we could find such functional modules within the inferred networks, which could reveal how the cell transcriptome is organized.

We searched the network for modules, which are defined as “communities” and “rich-clubs” in network theory. A community is a group of genes highly inter-connected to each other, but with few connections to genes outside the group. A “rich-club” can be defined as a “community of communities”, i.e. a group of closely inter-connected communities.

In order to identify communities, we represented the human network as a matrix, as shown in Figure 5.9A where the human adjacency matrix is reported. Each entry $m_{ij}$ of the matrix was set to 1, or 0, depending whether, or not, the MI between gene $i$ (in the $i^{th}$ row) and gene $j$ (in the $j^{th}$ column) was significant. We observed that genes lying within the same chromosome (squares in Fig. 5.9A) tend to be connected to each other more often than what would be expected by chance.

In order to identify communities from the adjacency matrix, we applied
5.4 The modular structure of the human and mouse networks

Figure 5.9: Modular structure of the human network. Adjacency matrix of the network before (A) and after (B) the hierarchical clustering procedure used to identify communities. Each dot represents a connection among two genes, that is a matrix entry, whose MI is greater than the significance threshold. (A) Genes are sorted according to their chromosomal location. Numbers on the x and y axes indicate chromosomes. (B) Genes are sorted according the community they belong to. Square dimensions are proportional to the number of genes in each community. The inset shows an enlargement of an area of the adjacency matrix where single dots are visible.
Figure 5.10: Modular structure of the mouse network. Adjacency matrix of the network before (A) and after (B) the hierarchical clustering procedure used to identify communities. Each dot represents a connection among two genes, that is a matrix entry, whose MI is greater than the significance threshold. (A) Genes are sorted according to their chromosomal location. Numbers on the $x$ and $y$ axes indicate chromosomes. (B) Genes are sorted according the community they belong to. Square dimensions are proportional to the number of genes in each community.
a classic hierarchical clustering algorithm with the average linkage distance on the matrix describing the network using as a distance the “Jaccard” metric. This is defined as the ratio between the number of common interactions between two genes (i.e. two rows or two columns being the matrix symmetric) divided by the total number of interactions. The clustering procedure is outlined in Section 5.4.1. The dendrogram, produced by the average linkage algorithm, was cut in order to maximize the number of clusters with more than 4 nodes 5.11A,B. At the end of this procedure, we identified 393 communities in the human network and 865 communities in the mouse network. Figure 5.9 and Figure 5.10 show the matrixes representing the human and mouse networks before (A) and after (B) the hierarchical clustering procedure (see Supplementary Table 2 for the list of communities). Communities appear as dark squares, with genes belonging to the same community grouped together, giving a striking check-board pattern.

We then checked whether communities were enriched for genes sharing a common biological function. To this end we applied Gene Ontology Enrichment Analysis on the list of genes in each community. GOEA is a commonly used technique that allows the identification of statistically over-represented Gene Ontology terms in a set of genes. Suppose to have a set of $N$ genes, $m$ of which are annotated as associated to a Gene Ontology term of interest. Suppose to draw a subset of $n$ genes from the complete list of $N$ genes, then the probability of obtaining $k$ genes all sharing the same Gene Ontology term
5.4 The modular structure of the human and mouse networks

Figure 5.11: Communities validation through gene ontology analysis. Linkage step versus number of human (A) and mouse (B) communities with at least 4 nodes. Human (C) and Mouse (D) Communities were sorted according to their cardinalities. Each blue dot represents a community whose genes are enriched for a GO term whereas red dots express communities were no enrichment have been found.
of interest follows an hypergeometric probability distribution:

\[ Pr(X = k) = \frac{{m \choose k} \cdot \left( \frac{N-m}{n-k} \right)}{{N \choose n}} \]  \hspace{1cm} (5.1)

from this, we can compute the cumulative distribution and hence the significance, or p-value, of the draw; i.e. the probability of having at least k genes sharing the same GO term of interest. Figure 5.11C,D show the result of the analysis where the percentage of positive predictions (y-axis) are related to the number of communities analysed (x-axis). Communities are ranked in descending order according to their cardinalities.

We assessed that 36% of the human communities are enriched for a specific biological function by Gene Ontology analysis (Supplementary Table 3). This percentage increases up to 47%, when considering only communities composed by more than 10 genes. We also found that 6 out of 393 communities of the network are significantly enriched for disease-genes (\( P < 0.05 \), Gene Set Enrichment Analysis): community number 1, 11, 22, 40, 54 and 96. The most significant community, number 40, is composed by genes whose protein products localize to the lysosome, and is highly enriched for disease-genes involved in lysosomal storage disorders. Other examples include community 11, whose genes are related to cell adhesion and extracellular matrix organization, and include disease-genes causing developmental or cardiovascular defects; community 22, related to the immune system and including genes causing related disorders; community 54, composed by genes involved in oxygen transport and enriched for genes involved in hematological disor-
5.4.1 Hierarchical clustering procedure

Hierarchical clustering methods [62] aim to group data items into a hierarchical set of clusters, organized in a tree structure, according to a specific distance function.

In order to apply one of these methods, it is necessary that all the distances between the items to be clustered are known, therefore a distance matrix must be calculated before the clustering procedure can be applied.

The hierarchical clustering algorithm starts with each item considered as a separate cluster. At each step, according to the distance matrix, the pair of closest clusters is merged into a new single cluster. The distance matrix is then updated by computing the distances between the new cluster and the others. This process iterates until all the items belong to a single cluster. It requires \( N - 1 \) steps where \( N \) is the number of the data items.

There are many possible distance functions that can be used in hierarchical clustering algorithms. We have adopted the unweighted average distance (UPMGA) [102]: the distance between any two clusters \( C_i \) and \( C_j \) is taken...
to be the average of all distances between pairs of items \( p \) in \( C_i \) and \( q \) in \( C_j \):

\[
\frac{1}{|C_i| \cdot |C_j|} \sum_{p \in C_i, q \in C_j} d(p, q).
\]

To constructs the distance matrix starting from the network adjacency matrix we have used the Jaccard similarity coefficient. For two two rows of the matrix, it is defined as the cardinality of their intersection divided by the cardinality of their union:

\[
d(p, q) = \frac{|p \cap q|}{|p \cup q|}.
\]

**Parallel Implementation of the hierarchical clustering algorithm**

The parallel algorithm distributes the gene adjacency matrix among the computing processes. Each process gets \( N/p \) rows of the adjacency matrix where \( p \) is the number of processes available and \( N \) is the total number of probes. Therefore each process stores a submatrix of the distance matrix with size \( N/p \times N \).

As the distance matrix is symmetric, only the elements corresponding to the upper triangular half are calculated applying the Jaccard similarity coefficient to the rows of the adjacency matrix taken pairwise.

During the clustering process, the cluster index constantly changes: the singleton clusters are numbered from 1 to \( N \), while each newly-formed cluster is assigned the index \( N + I \), where \( I \) is the index of the iterative step \( (1 \leq I \leq N - 1) \).

At each iterative step \( I \), each process finds the minimum of the distances locally. The global minimum distance of the closest clusters is found through
an all-reduce operation and its indices are broadcasted to all processes. The computational time complexity is then reduced to $O(nClusters_I^2/p)$, instead of $O(nClusters_I^2)$, where $nClusters_I$, varying from $N$ to 1, is the number of the clusters at step $I$. Therefore, the overall computational time complexity for this part of the algorithm is $O(\frac{N^3}{2p})$.

Each process updates the distances of the newly-formed cluster from the clusters that it stores. If the new cluster is obtained by merging the clusters $i$ and $j$, then the new distances to be calculated are stored in the column and row $i$ while the row and column $j$, from now on, will be excluded from the search for the minimum distance. Thus, each process just updates the portion of the column $i$ ($N/p$ elements) that it owns. The row $i$ is instead updated through a gatherv operation in which all processes send its portion of the column $i$ ($N/p$ elements) to the root process that owns the row $i$ (Figure 5.12). Therefore, the overall computational time complexity for this part of the algorithm is $O(\frac{N^3}{p})$. The parallel algorithm is summarized in Figure 5.13 and was implemented in C using the MPI standard.

5.4.2 Communities of communities: “Rich-clubs”

In order to better elucidate community function and interactions among them, we defined the Interaction Strength (IS) between two communities as the number of connections occurring among genes belonging to the two different communities, divided by the expected number of connections. The IS is equal to 0 if no connections exist among genes belonging to the two different communities.
### 5.4.2 Communities of communities: “Rich-clubs”

Formally, considering two communities \( A \) and \( B \), we define \( n_{AB} \) as the number of edges in the human or mouse network across the \( K_A \) genes in community \( A \) and \( K_B \) genes in community \( B \), then the Interaction Strength (\( IS_{AB} \)) is

\[
IS_{AB} = \log\left(\frac{n_{AB}}{K_A \times K_B \times f}\right)
\]  

(5.2)

where \( f \) is the average frequency of edges across all the genes in the human or mouse network.

We then constructed a community-wise network by creating an adjacency matrix whose element in row \( i \) and column \( j \) is the \( IS_{ij} \) between community \( i \) and community \( j \), if the \( IS_{ij} > 0 \).

We the used this community-wise adjacency matrix to identify rich-clubs, i.e. communities of communities. We computed the IS between all the pairs
of 393 communities for a total of 77,028. Only 5,074 pairs of communities had an IS greater than 0. Similarly to the gene-wise network, also the community-wise network can be represented as a matrix. We can therefore apply a clustering procedure to group the communities into sets of highly interconnected communities (“rich-clubs”) [39]. To this end, we applied a novel message-passing clustering algorithm [39] which is able to return the number of clusters without any user-specified parameter, using as inter-community distances the ISs. We thus obtained 58 human and 227 mouse clusters of communities, i.e. rich-clubs (as defined in network theory). The process was iterated by considering the ISs among the new formed clusters (rich-clubs).
5.4.2 Communities of communities: “Rich-clubs”

Figure 5.14: Community-wise human network. Each node is a community. A color and a number identify each rich-club (i.e. a group of highly interconnected communities). The width of each edge reflects the IS between communities. “Exemplars” are indicated by triangles. Examples of rich-clubs: (A) communities of genes involved in house-keeping functions: *gene expression* (rich-club 197), *translation* (rich-club 246), *RNA processing* (rich-club 2) and *cell cycle* (rich-club 7); (B) communities involved in the *extracellular matrix maintenance*, and *cell mobility*; (C) communities involved in *immune response*; (D) communities involved in *intracellular trafficking*.
5.5 Connections tend to be conserved across species

To understand whether, and to what extent, connections among genes are conserved across human and mouse species, we first removed from the human network those genes without an ortholog in mouse, resulting in a “reduced” network of 11,318 genes. We then found that 218,700 connections (12%) were conserved in mouse as shown in Figure 5.16.

This percentage is in line with previous studies. In yeast, it has been reported that between 10% [94] and 30% [85] of protein-protein interactions occurring during the cell cycle of *Saccharomyces pombe* (fission yeast) and *Saccharomyces cerevisiae* (budding yeast) are conserved; another cross-species (fly and yeast) protein interaction study [8] resulted in a ratio of conservation ranging from 6% to 15%; in [92], the authors report a database of protein-protein interactions occurring among transcription factors in human and mouse, where the percentage of effective conserved interactions is about 16% (the estimated range is 34%-64% when taking into account the False Positive Rate of the experimental technique). A recent genome-wide analysis [4] integrating heterogeneous sets of experimental data (including 338 expression profiles in human and 1048 in mouse) showed a conservation of
Figure 5.15: Community-wise mouse network. Each node is a community. A color and a number identify each rich-club (i.e. a group of highly interconnected communities). The width of each edge reflects the IS between communities and rich-clubs. “Exemplars” are indicated by triangles. The node of the network are identified with an integer number (see Supplementary Table 2).
Figure 5.16: Percentage of mouse conserved human connections. Connections are sorted according to their MI value. The pick of the curve is at 28%, and the genes involved in those conserved interactions are highly enriched for cell cycle $P = 1.0 \times 10^{-15}$.

15% of interactions between the two mammalian species.

We investigated whether communities found in the human network were conserved as communities in the mouse network. A human community was deemed to be conserved in mouse, if there was at least one mouse community composed by a significant fraction of genes orthologous to genes in the human community. Ninety-two out of 389 human communities (24%) were enriched in one, or more, mouse communities (Supplementary Table 5). We only considered those human communities with at least one mouse orthologous gene. It appears that the conservation at the community level (24%) is higher than conservation at the gene-gene level (12%) suggesting that many circuits embedded within the networks were conserved during evolution. Our results are in agreement with previous studies on the conservation of protein
5.5 Connections tend to be conserved across species

Table 5.4: List of conserved modules along with the human and mouse communities IDs, number of orthologous genes and the p-value associated to the Enrichment analysis.

<table>
<thead>
<tr>
<th>Gene ontology</th>
<th>Human community</th>
<th>Mouse community</th>
<th>Orthologous genes</th>
<th>p-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>cell cycle</td>
<td>2</td>
<td>7</td>
<td>142</td>
<td>$1.47 \times 10^{-24}$</td>
</tr>
<tr>
<td>proteasome</td>
<td>4</td>
<td>2</td>
<td>483</td>
<td>$8.96 \times 10^{-216}$</td>
</tr>
<tr>
<td>oxidative phosphorylation</td>
<td>4</td>
<td>2</td>
<td>483</td>
<td>$8.96 \times 10^{-216}$</td>
</tr>
<tr>
<td>ribosome</td>
<td>9</td>
<td>2</td>
<td>113</td>
<td>$1.79 \times 10^{-43}$</td>
</tr>
<tr>
<td>immune system</td>
<td>8</td>
<td>8</td>
<td>114</td>
<td>$2.02 \times 10^{-129}$</td>
</tr>
<tr>
<td>signal transduction</td>
<td>19</td>
<td>33</td>
<td>33</td>
<td>$2.85 \times 10^{-54}$</td>
</tr>
<tr>
<td>lipid metabolism</td>
<td>120</td>
<td>51</td>
<td>7</td>
<td>$1.61 \times 10^{-13}$</td>
</tr>
</tbody>
</table>

complexes across species [98]. Example of conserved communities are listed in Table 5.4: as expected many of the basic cellular processes are conserved [105].

We observed that human community 4 and the conserved mouse community 2 are significantly enriched for genes involved in proteasome (human $p = 7.25 \times 10^{-22}$, mouse $p = 7.83 \times 10^{-08}$) but also for genes involved in oxidative phosphorylation (human $p = 1.36 \times 10^{-55}$, mouse $p = 9.16 \times 10^{-12}$). Therefore, genes belonging to these different pathways are co-regulated, and this co-regulation is conserved across species. Proteasome-dependent protein degradation represents one of the most expensive processes in term of energy; oxidative phosphorylation is a metabolic pathway that uses energy released by the oxidation of nutrients to produce ATP. Therefore it would make sense for these two pathways to be coupled.
5.6 Discussions and conclusions

In this chapter we have shown that massive and heterogeneous gene expression datasets can indeed be exploited to yield biologically relevant information on gene regulation at transcriptional level. We have identified groups of genes that tend to be highly “co-expressed” (communities) and that share a common biological function.

We observed that genes that are connected, i.e. co-expressed, in human tend to be also co-expressed in mouse, more than what is expected by chance; however this is not the general trend, since only 12% of connections are conserved. Albeit this estimate may be lower than the real one, due to false negatives and orthologous gene mis-assignments, our results are confirmed by other large-scale experimental studies in human and mouse[92, 4]. Our observation adds weight to the hypothesis that regulation of gene expression may be very different between species, even if they share a similar proteome.

The gene network structure is typical of complex network with the presence of hub genes with a large number of connections. Our finding suggests that hub genes are highly expressed and may have been selected to be less dosage sensitive, i.e. not pathological when their expression is increased, as confirmed by their tendency to have a lower protein disorder.

We identified biologically relevant functional modules within the network, thus providing a modular view of the wiring diagram of a cell, going from genes to communities, and from communities to rich-clubs.
Chapter 6

Transcriptome organisation and gene function

In this chapter we described the application of a new Mutual Information reverse-engineering approach to infer a human-specific and mouse-specific gene regulatory network. A network is not just a static list of connections, it has a topology as reported in 5.3, which is by itself a source of information for scientific discoveries as we show in the current chapter. The final goal of an inference process applied to biological dataset is to gain new insights on the biological mechanisms of the cells. These insights include identification of physical interactions among proteins (P-P interactions), prediction of the biological function of a genes, and so on. In this chapter we experimentally validated a set of physical interactions predicted by connections in the human network, and we propose a new approach called “guilty-by-association”, able to predict the biological processes of genes, or its cellular localization of the
6.1 Biological validation of Protein-Protein interactions

Genome-wide inference validation was a difficult process to accomplish due to the lack of a golden standard network of interactions among genes. As outlined in 5.3.1 we built a golden standard (GoS) interaction network by merging different sources of interaction databases. Results are reported in Figure 5.3 and show that the inference performance reaches a maximum of 90% of correct predictions. The ROC curve highlight how the higher the MIs the more we can trust our predictions.

We investigated the identity of the top one thousand connections with the highest MI in the network (Fig. 6.1A and Supplementary Table 4). Forty percent of these connections, involving a total of 302 genes, were confirmed by the GoS interactome. An additional 13% of the connections were predicted among genes in the same gene-family, which, therefore, may well be functionally related, although not physically interacting.

In order to test the predictive ability of the network, we focused on the subnetwork (B) in Figure 6.1 consisting of 13 genes most of which (CENPF, NUSAP1, KIF2C, BUB1B, ASPM, ZWINT, PRC1, and CCNB2) involved in mitotic spindle checkpoint, chromosome motility, and mitotic progression.
6.1 Biological validation of Protein-Protein interactions

Figure 6.1: Subnetworks obtained by collecting the top 1000 connections with the highest MI within the network. Subnetwork (A) contains genes that codify for the ribosomal protein complex. Subnetwork (B) is enriched for genes involved in the spindle checkpoint, for clarity only a subset of interactions are shown. Subnetworks (C) is enriched for metallothionein genes, a family of low molecular weight, heavy metal binding proteins. Interestingly these genes are all present as a cluster on chr 16q13. Subnetwork (D) contains major histocompatibility complex proteins. Subnetwork (E) is enriched for immunoglobulin genes.
6.1.1 Yeast-two-Hybrid assays

According to the GoS interactome, protein-protein interactions are known to occur among some of the genes in subnetwork (B) (red lines in Fig. 6.1), therefore we decided to verify whether the predicted connections, could be yet undiscovered protein-protein interactions. We selected only the subset of 8 genes (NUSAP1, KIF2C, BUB1B, ASPM, ZWINT, PRC1, KIAA0101, and RRM2), which according to our network formed a tight cluster of genes all connected to each other. We performed a series of yeast two hybrid (Y2H) assays to test a total of 28 connections (i.e. all the possible interactions among 8 proteins); 15 of these (54%) were confirmed by Y2H corresponding to the green lines in Figure 6.1 (see Table 6.1 for the results of all the assays).

The statistical tests were accomplished with the hypergeometric distribution as reported in Equation 5.1, where, for each pair of proteins, and for each test (strong, weak, negative signal), \( m \) is the number of replicated Y2H assays (outcomes), \( k \) is the number of positive outcomes (strong, weak, negative), \( N \) is the number of total tests for all the pairs of proteins, and \( n \) is the number of controls.

6.1.1 Yeast-two-Hybrid assays

The Yeast two Hybrid (Y2H) kit “ProQuest Two-Hybrid System” (Invitrogen) included the S. cerevisiae MAV 203 strain (MAT\( \alpha \), leu2-3,112, trp1-901, his3\( \Delta \)200, ade2-101, gal4\( \Delta \), gal80\( \Delta \), SPAL10::URA3, GAL1::lacZ, HIS3UAS GAL1::HIS3@LYS2, can1R, cyh2R), the bait vector pDEST32 and the prey vector pDEST22. The “Ultimate ORF” (Invitrogen) of the genes of interest
### 6.1.1 Yeast-two-Hybrid assays

<table>
<thead>
<tr>
<th>Protein</th>
<th>Protein</th>
<th>strong signal</th>
<th>weak signal</th>
<th>negative</th>
<th>total</th>
<th>p-value strong</th>
<th>p-value weak</th>
<th>p-value negative</th>
</tr>
</thead>
<tbody>
<tr>
<td>NUSAP1</td>
<td>RRM2</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>6</td>
<td>0.43</td>
<td>0.00</td>
<td>0.05</td>
</tr>
<tr>
<td>NUSAP1</td>
<td>ZWINT</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>6</td>
<td>0.43</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>NUSAP1</td>
<td>ASPM</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>6</td>
<td>0.02</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>NUSAP1</td>
<td>KIF2C</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>6</td>
<td>0.12</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>NUSAP1</td>
<td>BUB1B</td>
<td>2</td>
<td>4</td>
<td>0</td>
<td>6</td>
<td>0.12</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>NUSAP1</td>
<td>PRC1</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>6</td>
<td>0.02</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>NUSAP1</td>
<td>KIAA0101</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>3</td>
<td>0.56</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>ZWINT</td>
<td>RRM2</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>6</td>
<td>0.12</td>
<td>0.03</td>
<td>0.05</td>
</tr>
<tr>
<td>ZWINT</td>
<td>ASPM</td>
<td>3</td>
<td>0</td>
<td>3</td>
<td>6</td>
<td>0.02</td>
<td>0.37</td>
<td>0.05</td>
</tr>
<tr>
<td>ZWINT</td>
<td>BUB1B</td>
<td>2</td>
<td>0</td>
<td>4</td>
<td>6</td>
<td>0.12</td>
<td>0.37</td>
<td>0.24</td>
</tr>
<tr>
<td>ZWINT</td>
<td>PRC1</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>0.56</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>ZWINT</td>
<td>KIF2C</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>3</td>
<td>0.13</td>
<td>0.19</td>
<td>0.21</td>
</tr>
<tr>
<td>ZWINT</td>
<td>KIAA0101</td>
<td>1</td>
<td>0</td>
<td>5</td>
<td>6</td>
<td>0.43</td>
<td>0.37</td>
<td>0.60</td>
</tr>
<tr>
<td>RRM2</td>
<td>PRC1</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td>0.56</td>
<td>0.19</td>
<td>0.67</td>
</tr>
<tr>
<td>RRM2</td>
<td>KIF2C</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>6</td>
<td>0.43</td>
<td>0.03</td>
<td>0.24</td>
</tr>
<tr>
<td>RRM2</td>
<td>KIAA0101</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>3</td>
<td>0.13</td>
<td>0.19</td>
<td>0.21</td>
</tr>
<tr>
<td>RRM2</td>
<td>ASPM</td>
<td>2</td>
<td>0</td>
<td>4</td>
<td>6</td>
<td>0.12</td>
<td>0.37</td>
<td>0.24</td>
</tr>
<tr>
<td>RRM2</td>
<td>BUB1B</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>3</td>
<td>0.13</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>PRC1</td>
<td>ASPM</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>6</td>
<td>0.12</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>PRC1</td>
<td>KIF2C</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>0.01</td>
<td>0.19</td>
<td>0.02</td>
</tr>
<tr>
<td>PRC1</td>
<td>BUB1B</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>3</td>
<td>0.13</td>
<td>0.19</td>
<td>0.21</td>
</tr>
<tr>
<td>PRC1</td>
<td>KIAA0101</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>3</td>
<td>0.13</td>
<td>0.19</td>
<td>0.21</td>
</tr>
<tr>
<td>KIF2C</td>
<td>ASPM</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>6</td>
<td>0.12</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>KIF2C</td>
<td>BUB1B</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>6</td>
<td>0.70</td>
<td>0.00</td>
<td>0.05</td>
</tr>
<tr>
<td>KIF2C</td>
<td>KIAA0101</td>
<td>2</td>
<td>0</td>
<td>4</td>
<td>6</td>
<td>0.12</td>
<td>0.37</td>
<td>0.24</td>
</tr>
<tr>
<td>KIAA0101</td>
<td>BUB1B</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>3</td>
<td>0.13</td>
<td>0.19</td>
<td>0.21</td>
</tr>
<tr>
<td>ASPM</td>
<td>KIAA</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td>0.56</td>
<td>0.19</td>
<td>0.67</td>
</tr>
<tr>
<td>ASPM</td>
<td>BUB1B</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>3</td>
<td>0.13</td>
<td>0.19</td>
<td>0.21</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>strong signal</th>
<th>weak signal</th>
<th>negative</th>
<th>total</th>
<th>p-value strong</th>
<th>p-value weak</th>
<th>p-value negative</th>
</tr>
</thead>
</table>

| negative control | 7             | 2           | 21       | 30    |                |              |                   |
| positive control | 15            | 0           | 0        | 15    |                |              |                   |
| positive weak control | 6 | 9 | 0 | 15 | | | |

Table 6.1: Biological validation of Protein-Protein interactions. Protein-protein interactions that have been tested via Yeast two Hybrid assays. In bold we report the significant p-values associated to the biological assay.
were used to generate prey and bait plasmids using the GateWay technology and protein-protein interaction assays were performed according to manufacturer instructions.

6.2 Chromatin structure and gene expression

The three-dimensional structure of chromosomes in the nucleus brings into close spatial proximity loci that are far apart in terms of genomic distance [65]. However, little is known about the relationship between chromatin spatial structure and gene expression.

We investigated whether genes that are connected to each other according to our network, were also physically close to each other at the chromatin level, in the cell nucleus. We used a recent comprehensive mapping of chromosomal loci physical interactions using an innovative “Hi-C” chromatin capture technology [65].

The physical interaction map was measured in 2 human cell lines (GM06690 and K562) [65]. Authors provide an intra-chromosomal [65] contact probability matrix at 100 Kilobases resolution, and a genome wide contact probability matrix at 1 Megabases resolution.

This map of physical interactions can be represented as a matrix \( (M_p) \), where each entry \( m_{ij} \) reports the probability of the \( i^{th} \) Mb of the genome to be in physical contact with the \( j^{th} \) Mb, according to “Hi-C” experimental results. Figure 6.4B is a graphical representation of this map for chromosome 19. In [65] authors observe that genes in the same chromosome that are close
6.2 Chromatin structure and gene expression

Figure 6.2: Connection tendency and genomic distance. “Tendency” (y-axis) of the human genes within a certain genomic distance to be connected according to our network. We computed a connection tendency as the number of connections among genes below a given genomic distance, divided by the expected number of connections for the same number of genes, independently of their distance. We color coded each chromosome, from red (shorter chromosomes) to blue (longer chromosomes). We see that the closer two genes are in terms of genomic distance, (x-axis, log-scale), the higher their tendency to be connected.

Each to other in terms of genomic distance have an high probability to be physically in contact. By using the information of the connections among the genes that is stored in the human network, we computed a “connection tendency” and compared it with the linear genomic distance.

Figure 6.2 shows the result of the analysis. Plots are associated to chromosome and are color coded according with the length of the chromosomes. Specifically, for each normalised chromosomal distance (y-axis), the number of connections among the genes that are within that distance is computed. This value is then divided by the expected number of connections (given the network) and the log2 is computed and reported in the x-axis. This quan-
tity is referred to as the “connection tendency”. The analysis shows that genes that are close each to other in terms of linear genomic distance, have an higher tendency of being connected in the network with respect to genes that are far apart. The analysis reported in Figure 6.2 is similar to the one performed in [65] where in place of connection tendency among genes they consider physical contact probability.

Communities in our network can be considered as functional modules consisting of genes whose expression is coordinated, and that carry out specific biological functions as observed in Section 5.4. We asked whether these gene modules could have a ‘physical counterpart’ in the cell. We therefore further investigated the correspondences between physical contact probability and connection tendency.

In [65], authors defined a correlation matrix $C_p$ in which element $(i, j)$ is the Pearson correlation between the $i^{th}$ row and $j^{th}$ column of $M_p$. This $C_p$ matrix exhibits a strong ‘plaid-pattern’ as shown in Figure 6.4B. $C_p$ is the correlation matrix that illustrates the correlation (range from 1 (blue) to +1 (red) in Fig. 6.4) between the intrachromosomal interaction profiles of every pair of 1 Mb loci along a chromosome (chromosome 19 in Fig. 6.4). The plaid pattern indicates the presence of two compartments within the chromosome.

The computation of the correlation between each pair of chromosome loci is justified by the fact that if two loci are nearby in space, they reasonably will share neighbours and have correlated interaction profiles.

The analysis reported in Figure 6.3 was performed computing the contact correlation mean (from matrix $C_p$) among the chromosomal regions associ-
6.2 Chromatin structure and gene expression

Figure 6.3: Intra-community physical contact (y-axis) expressed in terms of correlation mean is plotted for each of the 393 communities (x-axis) identified in Section 5.4.

ated to each of the community found with the analysis presented in Section 5.4. To associate chromosomal regions to communities, we simple take all the chromosomal regions that contain at least a gene of the community. Then, for each community we averaged the correlations among the chromosomal regions so associated (blue spots in Fig. 6.3). For each community we also check for the random correlation by shuffling the labels of the community, hence the genes, while keeping the size of the community (red spots indicate random mean, while red bars indicate standard deviation over 100 attempts).

In order to compare the physical contact probabilities of the chromosomal regions \(C_p\) to the human network, we first derived a “connection tendency matrix” \(M_c\) at 1 Mb resolution from the human network adjacency matrix. In the \(M_c\) matrix the element in position \((i, j)\) reports the connection tendency between genes in the \(i^{th}\) Mb and genes in the \(j^{th}\) Mb. To generate
Figure 6.4: Genes that are co-expressed tend to be physically close at the 3D chromatin level. (A) Connection tendency matrix of chromosome 19. Grey stripes highlight regions with no probes designed for the microarray model HG-U133A. A red color indicates two different 1 Mb loci whose genes are strongly connected to each other. (B) Physical contact matrix of chromosome 19. Grey stripes highlights chromosomal regions where centromeres are located, plus unalignable regions. A red color indicates two different 1 Mb loci that are physically close to each at the chromatin level. Physically close regions may also contain genes that are not co-expressed and vice-versa: region (I) in (A) has an opposite tendency with respect to the corresponding region (I) in (B). This means that regions that are not in physical contact may contain genes that are co-expressed. The opposite can also be true, for example region (II) shows that loci physically interacting with each other do not necessarily contain genes that are co-expressed.
the $M_c$ matrix, we first subdivided the adjacency matrix by grouping together probes referring to genes which where within 1 Mb distance between each other, we then computed the IS (Eq. 5.2) genome-wide between the 1 Mb regions, with $K_A$ equal to the number of genes in the A region (of 1 Mb), $K_B$ the number of genes in the B region (of 1 Mb), $n_{AB}$ the number of interactions between genes in region A and genes in region B, and $f$ the average frequency of interactions across all the genes in the human network. We then derived a correlation matrix $C_c$ as shown in Figure 6.4A, where the element in position $(i,j)$ is the Pearson correlation between the $i^{th}$ row and $j^{th}$ column of the $M_c$ matrix.

Chromosome-wise analysis and genome-wide analysis was accomplished by computing the 2-dimensional Pearson correlation coefficient (PCC) between the $C_p$ matrix, describing the physical contact probability, and the $C_c$ matrix, describing the interaction probabilities among the genes in the human network. We computed the p-values following classic statistical theory for PCC. The p-values were then corrected following a Bonferroni False Discovery Rate (FDR) procedure to control the percent of false predictions in the set of predictions. We deemed as significant those PCC with an $FDR < 0.05$.

In both matrices (Fig. 6.4A,B), there is a clear “plaid pattern” highlighting chromosomal regions whose genes are strongly connected to each other (red in Fig. 6.4A) and regions which are physically close to each other at the chromatin level (red in Fig. 6.4B). These regions have a striking overlap (correlation = 0.4, $P = 7.3 \times 10^{-123}$) especially the p-arm of chromosome 19 (upper left square in both matrices), revealing that genes that are physically
close to each other at the chromatin level tend to be “co-expressed” (i.e. have a significant MI), and vice-versa.

By extending this analysis to all of the chromosomes, we found a significant overlap (correlation significance: $P < 0.01$) for all but three chromosomes (9, 20 and 21).

### 6.3 Prediction of gene function and protein localisation

We exploited the information embedded in the human and mouse networks to identify gene function, or protein subcellular localisation, via a guilty-by-association analysis. It consists in assigning a function to a gene (or a localisation to the encoded protein) by checking whether there is a shared function among the genes connected to it (or a shared localisation of their protein products). In what follows, we term “gene neighbours” the set of genes connected to a given gene of interest according to the predicted networks of connections.

We performed a Gene Ontology Enrichment Analysis (GOEA) on the set of gene neighbours for each gene, in both the human and mouse networks\(^1\) (see Sec. 6.3.1). We then selected, as a test set, the subset of 18,141 human and 22,774 mouse transcripts for which the function/localisation was known according to their Gene Ontology classification. In table 6.2 we report the percentage of correct predictions, for each of the three GO classes (Biological

---

\(^1\)http://netview.tigem.it
Table 6.2: GOEA: guilty-by-associations study. The CLASS column reports the percentage of correct predictions; probeset-GO term correct association by looking at the probeset neighborhoods. K: probesets associated to one or more GO term; N: probesets whose neighborhood is associated to one or more GO term; K\textsubscript{int}\_N\textsubscript{int}: intersection between K and N; K\textsubscript{int}\_N\textsubscript{correct}: probesets associated to one or more GO term that is enriched in its neighborhood.

<table>
<thead>
<tr>
<th>species</th>
<th>K</th>
<th>N</th>
<th>N\textsubscript{int}_K</th>
<th>CORRECT</th>
<th>PERC</th>
<th>CLASS</th>
</tr>
</thead>
<tbody>
<tr>
<td>human</td>
<td>17174</td>
<td>16750</td>
<td>13123</td>
<td>7626</td>
<td>58%</td>
<td>BP</td>
</tr>
<tr>
<td></td>
<td>18141</td>
<td>16646</td>
<td>13687</td>
<td>6601</td>
<td>48%</td>
<td>MF</td>
</tr>
<tr>
<td></td>
<td>18020</td>
<td>17561</td>
<td>14400</td>
<td>10247</td>
<td>71%</td>
<td>CC</td>
</tr>
<tr>
<td>mouse</td>
<td>19011</td>
<td>24848</td>
<td>11379</td>
<td>6682</td>
<td>59%</td>
<td>BP</td>
</tr>
<tr>
<td></td>
<td>22774</td>
<td>24796</td>
<td>13407</td>
<td>6789</td>
<td>51%</td>
<td>MF</td>
</tr>
<tr>
<td></td>
<td>21975</td>
<td>24942</td>
<td>13338</td>
<td>10375</td>
<td>78%</td>
<td>CC</td>
</tr>
</tbody>
</table>

Process, Molecular Function and Cellular Localisation), ranging from 48% to 78%.

### 6.3.1 Computational method

Gene Ontology Enrichment Analysis (GOEA) is a commonly used technique that allows the identification of statistically over-represented Gene Ontology terms in a set of genes. Suppose to have a set of $N$ genes, $m$ of which are annotated as associated to a Gene Ontology term of interest. Suppose to draw a subset of $n$ genes from the complete list of $N$ genes, then the probability of obtaining $k$ genes all sharing the same Gene Ontology term of interest follows an hypergeometric probability distribution, that is possible to compute applying the formula presented in Section 5.4. From this, we can compute the cumulative distribution and hence the significance, or p-value, of the draw; i.e. the probability of having at least $k$ genes sharing the same GO term of
interest. This is a classical statistical analysis that we also performed in [53] to associate significant ontology terms to communities of drug compounds. The drug compounds were first grouped according to the “distance” between the expression profiles they generate.

In order to predict gene function and/or localisation from the human and mouse networks, we proceeded as follows: for each probe (i.e. gene) of the human or mouse network, we selected the probes predicted to be connected with it in the network (i.e. the gene’s neighbours). For probes with more than 500 neighbors, only the top ranked 500 ones with the highest MI were retained; for nodes with less than 50 neighbors, we included also the gene’s second neighbours (i.e. the neighbours of the neighbours) up to a maximum of 500.

Multiple probes may refer to the same gene and thus be highly co-expressed. To avoid biases in the GOEA computation, we removed from the neighbours, the probes associated to the same gene. GOEA was then performed on this subset of neighbours for each of the probe in the human network. In the testing phase, a prediction was claimed to be correct if the GO term that was enriched within a neighbourhood was also the one associated to the probe itself according to the GO database [6].

6.3.2 Biological validation of gene expression localisation

To test experimentally a subset of new protein localisation predictions, we selected 6 genes, whose protein products were predicted to be localised in
6.3.2 Biological validation of gene expression localisation

Figure 6.5: Biological validation of predicted mitochondrion protein localisations. NSUN4 (A) and UQCC (B) have been modified in order to be expressed together with a red-fluorescent protein. Green fluorescence localises mitochondrion. Yellow spot are given by the combination of red and green showing the mitochondrion localisation of the expression of the protein we are testing.

We found that 2 genes, NSUN4 (NOP2/Sun domain family, member 4) and UQCC (ubiquinol-cytochrome c reductase complex chaperone, CBP3 homolog (yeast)) localised in mitochondria as shown in Figure 6.5); confirming that at least 33% of our predictions are correct. For comparison, had 6 genes been selected at random, the probability that 2 of them had protein products localised to mitochondria would have been less that 0.3%, hence two-order of magnitudes lower.
Experimental methods

**Cell culture and transfection for protein mitochondrial localisation assays.** HeLa cells were maintained at 37°C in a 5% CO2-humidified incubator, and cultured in DMEM (GIBCO BRL) supplemented with 10% heat-inactivated fetal bovine serum (FBS) (Invitrogen) and 1% antibiotic-antimycotic solution (GIBCO BRL). 50000 cells were transfected in a 24-well with 0.3µg of DNA using TransIT-LT1 Transfection Reagent following the standard protocol suggested by the manufacturer. After 16h cells were treated with a red MitoTracker (Invitrogen) for 30 minutes, in order to specifically bind the mitochondria, and then were fixed with 4% paraformaldehyde for 10 minutes. Cells were observed using the confocal microscope ZEISS LSM 710.

**Bacterial strain, Plasmids and Molecular Cloning.** Plasmid construction and molecular cloning were performed in the cloning host cell E.coli DH5a (Invitrogen) following standard protocols. For each gene of interest the coding sequence was amplified from a total preparation of cDNA obtained by RT-PCR of a murine RNA.

The amplification was carried out using the primers listed in Table 6.3. Each coding sequence was then cloned into the EcoRI/BglII sites (except for Fam73b that was cloned into EcoRI/BamHI sites) of pEGFP-N3 vector at the N terminal of the EGFP, thus resulting in a fusion protein. All clones were confirmed by sequencing analysis.
6.4 Elucidating the function of the granulin precursor ‘disease-gene’

We next asked whether the guilty-by-association approach could be helpful in elucidating the function of genes involved in genetic diseases. Since we were interested in lysosomal function and lysosomal disorders, we used our guilty-by-association analysis to identify human disease genes which may have a yet undiscovered role in lysosome function and organization.

To this end, we ranked in descending order of their p-values all of the genes predicted to be lysosomal, or associated with lysosome organization (http://netview.tigem.it). The top ranked genes included both lysosomal enzymes and other genes involved lysosomal

<table>
<thead>
<tr>
<th>Gene</th>
<th>Sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>9530058B02Rik F</td>
<td>(5'-CAGGAATTCACCATGACATACATACACCAAG-3')</td>
</tr>
<tr>
<td>9530058B02Rik R</td>
<td>(5'-CGAGATCTCGGAACAGTTGATTTTACG-3')</td>
</tr>
<tr>
<td>2010012O05Rik F</td>
<td>(5'-TTAGAATTCGACCTTCAGTGGGACTGGGACGC-3')</td>
</tr>
<tr>
<td>2010012O05Rik R</td>
<td>(5'-GGAGATCTCCTCCTCCACACACCTCTTTCA-3')</td>
</tr>
<tr>
<td>2810405K02Rik F</td>
<td>(5'-GTCGAATTCATTAGAATGTTGGACCTTTGCT-3')</td>
</tr>
<tr>
<td>2810405K02Rik R</td>
<td>(5'-GTAGAATTCGGTGGTTGGCGACTGGGCA-3')</td>
</tr>
<tr>
<td>Nsun4 F</td>
<td>(5'-CTAGAATTCACCATGGTGTTTATCACATCAGA-3')</td>
</tr>
<tr>
<td>Nsun4 R</td>
<td>(5'-CTAGAATTCACCATGGCTGCGCCCGTCTTTA-3')</td>
</tr>
<tr>
<td>Uqcc F</td>
<td>(5'-GCGGAATTCAGTTGGCGCTGGTGCGAG-3')</td>
</tr>
<tr>
<td>Uqcc R</td>
<td>(5'-GCGGAATTCAGTTGGCGCTGGTGCGAG-3')</td>
</tr>
<tr>
<td>Ccb1 F</td>
<td>(5'-GCGGAATTCACCATGGTTGCGACTGGGACGC-3')</td>
</tr>
<tr>
<td>Ccb1 R</td>
<td>(5'-GCGGAATTCACCATGGTTGCGACTGGGACGC-3')</td>
</tr>
</tbody>
</table>

Table 6.3: Genes tested for mitochondrial localisation. Genes predicted to be significantly associates to mitochondria by the guilty-by-association analysis. The sequence column reports the primers used to test the mitochondrial localisation. Names in italics indicate an alternative isoform of the protein. F and R state for Forward and Reverse, respectively.
6.4 Elucidating the function of the granulin precursor ‘disease-gene’

<table>
<thead>
<tr>
<th>P-value</th>
<th>Probeset ID</th>
<th>Gene Symbol</th>
<th>Gene Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>$7.89 \times 10^{-11}$</td>
<td>200701_at</td>
<td>NPC2</td>
<td>Niemann-Pick disease, type C2</td>
</tr>
<tr>
<td>$1.88 \times 10^{-36}$</td>
<td>211284_x_at</td>
<td>GRN</td>
<td>granulin</td>
</tr>
<tr>
<td>$2.41 \times 10^{-34}$</td>
<td>200743_x_at</td>
<td>TPP1</td>
<td>tripeptidyl peptidase 1</td>
</tr>
<tr>
<td>$7.74 \times 10^{-33}$</td>
<td>212663_at</td>
<td>FKBP15</td>
<td>FK506 binding protein 15, 133kDa</td>
</tr>
<tr>
<td>$3.76 \times 10^{-32}$</td>
<td>201212_at</td>
<td>LGMN</td>
<td>legumain</td>
</tr>
<tr>
<td>$7.10 \times 10^{-31}$</td>
<td>201494_at</td>
<td>PRCP</td>
<td>prolylcarboxypeptidase (angioteinase C)</td>
</tr>
<tr>
<td>$8.76 \times 10^{-31}$</td>
<td>218217_at</td>
<td>SCPEP1</td>
<td>serine carboxypeptidase 1</td>
</tr>
<tr>
<td>$5.99 \times 10^{-30}$</td>
<td>200744_at</td>
<td>NAGA</td>
<td>N-acetylgalactosaminidase, alpha</td>
</tr>
<tr>
<td>$8.99 \times 10^{-30}$</td>
<td>217118_x_at</td>
<td>C22orf9</td>
<td>chromosome 22 open reading frame 9</td>
</tr>
<tr>
<td>$9.02 \times 10^{-30}$</td>
<td>204294_at</td>
<td>SLCO1A2</td>
<td>solute carrier family 31, member 2</td>
</tr>
<tr>
<td>$2.31 \times 10^{-29}$</td>
<td>200871_x_at</td>
<td>PSAP</td>
<td>prosaposin</td>
</tr>
<tr>
<td>$9.49 \times 10^{-29}$</td>
<td>201944_at</td>
<td>HEXB</td>
<td>hexosaminidase B (beta polypeptide)</td>
</tr>
<tr>
<td>$2.73 \times 10^{-28}$</td>
<td>35828_at</td>
<td>GM2A</td>
<td>GM2 ganglioside activator</td>
</tr>
<tr>
<td>$3.67 \times 10^{-28}$</td>
<td>200839_x_at</td>
<td>CTSH</td>
<td>cathepsin H</td>
</tr>
<tr>
<td>$7.29 \times 10^{-28}$</td>
<td>202295_x_at</td>
<td>CTSH</td>
<td>cathepsin H</td>
</tr>
<tr>
<td>$7.29 \times 10^{-28}$</td>
<td>202838_at</td>
<td>FUCA1</td>
<td>fucosidase, alpha-L-1, tissue</td>
</tr>
<tr>
<td>$8.89 \times 10^{-28}$</td>
<td>200748_x_at</td>
<td>FTH1</td>
<td>ferritin, heavy polypeptide 1</td>
</tr>
<tr>
<td>$5.39 \times 10^{-27}$</td>
<td>202545_at</td>
<td>PRKCD</td>
<td>protein kinase C, delta</td>
</tr>
<tr>
<td>$7.00 \times 10^{-27}$</td>
<td>202087_x_at</td>
<td>CTSL1</td>
<td>cathepsin L1</td>
</tr>
<tr>
<td>$8.61 \times 10^{-27}$</td>
<td>208784_x_at</td>
<td>APLP2</td>
<td>amyloid beta (A4) precursor-like protein 2</td>
</tr>
</tbody>
</table>

Table 6.4: Top 20 genes significantly associated to Lysosome (cellular component gene ontology class) resulting from the application of the guilty-by-association analysis, sorted according to the associated p-values.

function and are listed in Table 6.4 and Table 6.5. Among these, at position one, *NPC2* (*Niemann-Pick disease, type C2*) disease-gene, known to be an intralysosomal gene [115]; at position two, we found another disease-gene *GRN*. Both genes are members of community 40, which is enriched both the presence of disease genes and for lysosomal genes, described in Section 5.4.

Despite recent extensive studies, the role of *GRN* is far from being understood and, to our knowledge, it has not been directly linked to lysosomal function in the literature. *GRN* is a highly conserved gene bearing multiple copies of the cysteine-rich granulin motifs. Proteolytic cleavage of the precursor protein by extracellular proteases, such as elastase, gives rise to smaller peptide fragments termed granulins, or epithelins, which have been linked to a range of biological functions including cell division, survival, and migration [34]. Mutations in *GRN* cause frontotemporal lobar degeneration with ubiquitin-immunoreactive neuronal inclusions (FTLD-U) [7, 24]. Of note, mu-
Table 6.5: Top 20 genes significantly associated to Lysosome organisation (biological process gene ontology class) resulting from the application of the guilty-by-association analysis, sorted according to the associated p-values.

<table>
<thead>
<tr>
<th>P-value</th>
<th>Probeset ID</th>
<th>Gene Symbol</th>
<th>Gene Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.58×10^{-09}</td>
<td>2006845_at</td>
<td>CTSA</td>
<td>cathepsin A</td>
</tr>
<tr>
<td>7.37×10^{-06}</td>
<td>2007422_at</td>
<td>TPP1</td>
<td>tripeptidyl peptidase I</td>
</tr>
<tr>
<td>3.20×10^{-05}</td>
<td>2078806_at</td>
<td>ATP6AP1</td>
<td>ATPase, H+ transporting, lysosomal accessory protein 1</td>
</tr>
<tr>
<td>3.48×10^{-05}</td>
<td>2112383_at</td>
<td>GRN</td>
<td>granulin</td>
</tr>
<tr>
<td>2.10×10^{-04}</td>
<td>2025455_at</td>
<td>PRKCD</td>
<td>protein kinase C, delta</td>
</tr>
<tr>
<td>2.25×10^{-04}</td>
<td>2010504_at</td>
<td>PLD3</td>
<td>phospholipase D family, member 3</td>
</tr>
<tr>
<td>2.90×10^{-04}</td>
<td>2007668_at</td>
<td>CTSD</td>
<td>cathepsin D</td>
</tr>
<tr>
<td>3.73×10^{-04}</td>
<td>2171186_at</td>
<td>C22ORF9</td>
<td>chromosome 22 open reading frame 9</td>
</tr>
<tr>
<td>3.74×10^{-04}</td>
<td>2199325_at</td>
<td>MCOLN1</td>
<td>maculopin 1</td>
</tr>
<tr>
<td>4.58×10^{-04}</td>
<td>2028120_at</td>
<td>GAA</td>
<td>glucosidase, alpha</td>
</tr>
<tr>
<td>2.54×10^{-03}</td>
<td>2089268_at</td>
<td>NEU1</td>
<td>sialidase 1 (lysosomal sialidase)</td>
</tr>
<tr>
<td>3.54×10^{-03}</td>
<td>2050904_at</td>
<td>NAGPA</td>
<td>N-acetylgalcosamine-1-phosphodiester alpha-N-acetylgalcosaminidase</td>
</tr>
<tr>
<td>3.81×10^{-03}</td>
<td>2006490_at</td>
<td>NUCB1</td>
<td>nucleobindin 1</td>
</tr>
<tr>
<td>4.56×10^{-03}</td>
<td>2182822_at</td>
<td>EDPM2</td>
<td>ER degradation enhancer, mannosidase alpha-like 2</td>
</tr>
<tr>
<td>4.56×10^{-03}</td>
<td>2190200_at</td>
<td>HS1BP3</td>
<td>HCLS1 binding protein 3</td>
</tr>
<tr>
<td>4.56×10^{-03}</td>
<td>2030452_at</td>
<td>NINJ1</td>
<td>ninjurin 1</td>
</tr>
<tr>
<td>4.56×10^{-03}</td>
<td>2014944_at</td>
<td>PRCP</td>
<td>prolylcarboxypeptidase (angiotensinase C)</td>
</tr>
<tr>
<td>4.56×10^{-03}</td>
<td>2126478_at</td>
<td>RRAS</td>
<td>related RAS viral (r-ras) oncogene homolog</td>
</tr>
<tr>
<td>4.56×10^{-03}</td>
<td>2182177_at</td>
<td>SCPEP1</td>
<td>serine carboxypeptidase 1</td>
</tr>
<tr>
<td>4.56×10^{-03}</td>
<td>2031677_at</td>
<td>TIMP2</td>
<td>TIMP metalloproteinase inhibitor 2</td>
</tr>
</tbody>
</table>

To investigate whether GRN was indeed related to lysosomal function, we first evaluated GRN expression levels following sucrose treatment, a known inducer of lysosomal biogenesis [58, 48]. A strong increase in the number of lysosomes was detected upon sucrose supplementation to the culture medium, as expected (Data not shown). Following sucrose treatment, in Figure 6.6A, we observed a 2-fold increase over baseline in GRN mRNA levels, along with a 3-fold increase in Cathepsin D (CTSD), a lysosomal enzyme used as positive control (Figure 6.6A).

The transcription factor EB (TFEB) has been recently identified as the transcription factor controlling most of the known lysosomal genes via direct binding to their proximal promoter [95], therefore we next asked whether GRN may indeed be regulated by this transcription factor. We first identified,
6.4 Elucidating the function of the granulin precursor ‘disease-gene’

Figure 6.6: *GRN* is involved in lysosomal biogenesis and function. (A) *GRN* and *CTSD* increase in expression level following sucrose treatment, as measured by realtime PCR. (B) Expression level of *GRN* and of *CTSD* increase following *TFEB* over-expression. (C) Immuno-fluorescence with antibody anti-*LAMP2*, used as a lysosomal marker, of transfected HEK293 cells over-expressing *GRN*, or EGFP, against a mock control. All values represent mean fluorescence intensity ±s.e.m; *GRN* overexpression (*P* = 0.003) and sucrose treatment (*P* = 0.01) significantly increase the fluorescence compared to mock control. (D) Immuno-fluorescence with antibody anti-*LAMP2* in wt HEK293 grown in medium collected from *GRN* over-expressing cells shows a significant increase in fluorescence (*P* = 0.008) compared to medium collected from EGFP over-expressing cells.
by bioinformatics analysis, two TFEB binding sites upstream of the GRN coding sequence (see section 6.4.2). We then over-expressed TFEB in human cell lines and detected a 3-fold increase in GRN mRNA levels, along with a 3-fold increase in CTSD, a known target of TFEB, used as a positive control (Figure 6.6B).

We next over-expressed GRN to observe if this had any effect on lysosomes; as shown in Figure 6.6C, the number of lysosomes significantly increased ($P = 0.01$), compared to a mock control, or over-expression of EGFP, as assessed by immuno-fluorescence against LAMP2. The effect of increased lysosomes was observed on the great majority of cells, despite transfection efficiency was not as extensive. Therefore, we hypothesised that GRN-transfected cells could secrete a factor inducing lysosome biogenesis in neighbouring cells. This hypothesis was supported by previous study indicating that GRN is uptaken by a mannose 6P-phosphate receptor mechanism [61]. To test this hypothesis, we collected the medium from transfected cells over-expressing GRN, and used it to grow untreated cells; this resulted in a consistent increase ($P = 0.003$) in lysosome number compared to control, as evident in Figure 6.6D.

### 6.4.1 Experimental material

HeLa or HEK293 cells were cultured in DMEM supplemented with 10% FBS and treated for 96 hours in the presence of sucrose to a final concentration of 100mM with daily changes of medium. For immunofluorescence cells grown on glass coverlips were fixed with methanol for 10 minutes, washed with PBS,
6.4.1 Experimental material

treated with 50 mM NH₄Cl for 15 minutes, and permeabilized with PBS 0.1% Triton, blocked in blocking buffer (0.5% BSA, 50mM NH₄Cl, 0.001% Triton in PBS pH 7.4), and incubated overnight at 4°C with anti-LAMP2 antibody Santa Cruz and for one hour with Alexa-594 (Invitrogen).

HEK293 cells were maintained at 37°C in a 5% CO2-humidified incubator, and cultured in DMEM (GIBCO BRL) supplemented with 10% heat-inactivated fetal bovine serum (FBS) (Invitrogen), 1% L-glutamine and 1% antibiotic/antimycotic solution (GIBCO BRL). 500,000 cells were trasfected with 4µg of DNA expressing the transcriptional factor EB (TFEB) using lipofectamine trasfection reagent (Invitrogen) following the standard protocol suggested by the manufacturer. After 48 hours from trasfection cells were collected, the mRNA extracted and the levels of Granulin and Catepsin D (CTSD) were evaluated by Real-Time PCR (Roche). The amplification was performed using the following primers for GAPDH, “Fw: GAAGGTGAAG-GTCGGAGTC” and “Rev: GAAGATGATGATGGGATTTC”, for GRN, “Fw: TCCAGAGTAAGTGCTCTCCA” and “Rev: TCACCTCCATGT-CACATTCCA”, and for CTSD, “Fw: AACTGCTGGACATCGCTTGCT” and “Rev: CATTCTTC-ACGTagGTGCTGGA”.

Medium was concentrated on filters (Vivaspin Sartorius Stedim) and loaded on 10% SDS-PAGE. Transfer membranes were incubated with anti human granulin antibody (Invitrogen) at 1:50 dilution.
6.4.2 Identification of Binding sites in Granulin promoter region

We used the Position Weight Matrix of the Transcription Factor EB and run MATCH [59] to find its binding sites across -1000 +1000 base pairs flanking the UTR region of GRN. We found two binding sites on chromosome 17, plus strain, with location 42,422,444 to 42,422,457 and chromosome 17, plus strain, location 42,422,460 42,422,473 (Genome Browser, Human assembly Feb. 2009).

6.5 Gene signature analysis

The human network can be used as a ‘consensus’ network to analyse gene signatures. A gene signature is a list of genes able to identify a specific condition (i.e. disease state, drug response, etc.). Genes part of a signature are the end point of a regulatory cascade. The identification of the primary genes, such as Transcription Factors (TFs), that control the genes in the signature via the regulatory cascade can help in identifying the “master regulators”, involved in the process of interest [21].

The Gene Signature Analysis is basically a Gene Set Enrichment Analysis and consists in computing the statistical significance of the intersection of two sets of objects. In this case, we check how significant the group of genes in the gene signature of interest are represented within the neighbours of each TF of the human network.
6.5.1 A case study

As an example of the use of the network, we analysed the genes part of the mesenchymal gene expression signature (MGES) found to be over-expressed in poor prognosis group of glioma patients [88]. In [21], a high-grade glioma (HGG) specific gene network was inferred from 176 expression profiles of HGG samples derived from patients. This condition-specific network was then used to infer the TFs responsible for the mesenchymal transformation of brain tumors. Such “master regulators” were the TFs that, according to the HGG-specific transcriptional network, regulated a significant number of genes in the MGES signature.

We identified 31 TFs listed in Table 6.6 that according to our network had a significant number of connections to the genes in the signature. Eighteen TFs, out of these 31, were identified also in [21] (significance of $p = 7.3 \times 10^{-16}$). The two key genes, C/EBPβ and STAT3, codifying for TFs necessary in human glioma cells for mesenchymal transformation [21] were correctly included among the 31 TFs.

We further tested whether the genes in the MGES signature were over-represented within some of the 393 human communities identified within the network. Community 11 and 46 were found to be significantly enriched ($p = 7.90 \times 10^{-16}$ and $p = 1.09 \times 10^{-12}$, respectively) for genes in the MGES signature. Community 11 is composed by genes involved in phosphate transport ($p = 5.35 \times 10^{-38}$), and whose expression is localised into the proteinaceous extracellular matrix ($p = 1.70 \times 10^{-99}$); whereas community 46 includes genes involved in actin filament-based process ($p = 1.50 \times 10^{-66}$),
### Table 6.6: Transcription Factors that are enriched of gene in the mesenchymal gene expression signature. MGES accounts 122 probes. Genes are ranked in ascending order according with their p-values.

<table>
<thead>
<tr>
<th>Common number of neighbours</th>
<th>p-value</th>
<th>Gene Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>$1.33 \times 10^{-11}$</td>
<td>MAPF</td>
</tr>
<tr>
<td>10</td>
<td>$9.17 \times 10^{-11}$</td>
<td>AEBP1</td>
</tr>
<tr>
<td>7</td>
<td>$3.39 \times 10^{-09}$</td>
<td>FOSL2</td>
</tr>
<tr>
<td>5</td>
<td>$3.48 \times 10^{-09}$</td>
<td>FOSL1</td>
</tr>
<tr>
<td>7</td>
<td>$1.55 \times 10^{-08}$</td>
<td>ELF4</td>
</tr>
<tr>
<td>7</td>
<td>$1.23 \times 10^{-07}$</td>
<td>TFE3</td>
</tr>
<tr>
<td>7</td>
<td>$4.66 \times 10^{-07}$</td>
<td>JUNB</td>
</tr>
<tr>
<td>10</td>
<td>$5.45 \times 10^{-07}$</td>
<td>IRF1</td>
</tr>
<tr>
<td>6</td>
<td>$5.54 \times 10^{-07}$</td>
<td>RELB</td>
</tr>
<tr>
<td>6</td>
<td>$1.51 \times 10^{-06}$</td>
<td>CEBPB</td>
</tr>
<tr>
<td>5</td>
<td>$5.05 \times 10^{-05}$</td>
<td>TNXA</td>
</tr>
<tr>
<td>4</td>
<td>$6.71 \times 10^{-05}$</td>
<td>BHLHB2</td>
</tr>
<tr>
<td>5</td>
<td>$1.86 \times 10^{-04}$</td>
<td>RELA</td>
</tr>
<tr>
<td>3</td>
<td>$3.80 \times 10^{-04}$</td>
<td>CEBPD</td>
</tr>
<tr>
<td>10</td>
<td>$1.62 \times 10^{-03}$</td>
<td>PRRX1</td>
</tr>
<tr>
<td>4</td>
<td>$1.93 \times 10^{-03}$</td>
<td>BTG2</td>
</tr>
<tr>
<td>3</td>
<td>$3.59 \times 10^{-03}$</td>
<td>NFKB1</td>
</tr>
<tr>
<td>2</td>
<td>$4.30 \times 10^{-03}$</td>
<td>LOC161527</td>
</tr>
<tr>
<td>3</td>
<td>$5.46 \times 10^{-03}$</td>
<td>SMAD3</td>
</tr>
<tr>
<td>4</td>
<td>$6.69 \times 10^{-03}$</td>
<td>IRF7</td>
</tr>
<tr>
<td>2</td>
<td>$6.98 \times 10^{-03}$</td>
<td>RARA</td>
</tr>
<tr>
<td>4</td>
<td>$7.77 \times 10^{-03}$</td>
<td>SPI1</td>
</tr>
<tr>
<td>4</td>
<td>$9.82 \times 10^{-03}$</td>
<td>STAT3</td>
</tr>
<tr>
<td>3</td>
<td>$1.14 \times 10^{-02}$</td>
<td>PYCARD</td>
</tr>
<tr>
<td>2</td>
<td>$1.25 \times 10^{-02}$</td>
<td>CREB3L1</td>
</tr>
<tr>
<td>2</td>
<td>$2.02 \times 10^{-02}$</td>
<td>FOSB</td>
</tr>
<tr>
<td>3</td>
<td>$2.43 \times 10^{-02}$</td>
<td>JUND</td>
</tr>
<tr>
<td>2</td>
<td>$2.63 \times 10^{-02}$</td>
<td>SNF1LK</td>
</tr>
<tr>
<td>2</td>
<td>$2.85 \times 10^{-02}$</td>
<td>HMOX1</td>
</tr>
<tr>
<td>3</td>
<td>$3.54 \times 10^{-02}$</td>
<td>MYD88</td>
</tr>
<tr>
<td>2</td>
<td>$4.41 \times 10^{-02}$</td>
<td>ATF3</td>
</tr>
</tbody>
</table>
and expressed into the cytoskeleton ($p = 6.23 \times 10^{-10}$).

### 6.6 Discussions and conclusions

We demonstrated that genes connected within functional modules in the human network tend to have a ‘physical counterpart’ in the three-dimensional conformation of the chromatin inside the nucleus. We observed a striking similarity between genes that appear to be connected, and, therefore, are co-expressed, and their physical proximity at the three-dimensional chromatin level. This suggests that regulation of coordinated gene expression is “hard-wired” in the physical arrangement of the chromatin within the nucleus.

We have shown, and experimentally validated, different examples of how the network can be queried to predict the the localisation of genes’protein product, to identify new protein-protein interactions and to explore the function of a disease-gene.

Upregulation of $GRN$ by known inducers of lysosomal biogenesis and function, together with the increase in the number of lysosomes following $GRN$ over-expression, or treatment with medium from over-expressing cells, clearly supports a role of $GRN$ in lysosome biology that has been unrecognized until now. This finding is also supported by previous evidence indicating that $GRN$ colocalises with lysosome-associated CD68 antigen in activated macrophages and microglia [80] and is overexpressed in the cerebral cortex of MPSIIIB and MPSI mice [84].

Moreover, $GRN$ has been shown to bind the mannose 6-phosphate recep-
tor [61]. Interestingly, lysosomal dysfunction has been proposed to play a role in other neurodegenerative diseases including Alzheimer’s disease [22] sharing clinical similarities with FTLD-U. In these disorders it has been proposed that lysosomal hydrolytic enzymes contribute to cell death through lysosomal destabilization and enzyme leakage into the cytoplasm as indicated by models of experimental brain ischemia injury resulting in cytosolic acidification and rupture/permeabilization of lysosomes [116]. Interestingly, one of the first recognized function of $GRN$ has been in wound healing [47] and release of lysosomal enzymes appears to play an important role in healing [57, 74].

The Gene Signature Analysis we proposed in Section 6.5, shows that it may be not always necessary to use a diseases-specific network to analyse a disease-specific gene signature, but using a “consensus” network, we could correctly identify the “master regulators” involved in the observed gene signature.
Chapter 7

Conclusions and future directions

Despite the common believe that the merge of massive and heterogeneous gene expression profiles would be to noisy to be used for inference, in this thesis work we shown the reliability of the human and mouse gene regulatory network we inferred. The results of the analysis were validated both on an in-silico scenario, where predictions were compared with known protein-protein and other types of interactions collected from literature, and on a in-vivo scenario, where biological experiments confirmed different types of predictions (protein-protein, gene function and gene localisation). Moreover, the validity of the inference process here proposed is enhanced when network characteristics are analysed; i. e. degree distribution, small protein disorder associated to hubs, modularity structure of the networks and the presence of co-expressed groups of genes that localise in spatially close chromosomal
loci.

The results collected are accessible, and can be queried online\(^1\). The system provides both the access to the connections of the human and mouse networks, and to a set of gene functions predictions obtained by analysing the topology of the network with the “guilty-by-association” approach presented in Chapter 6.

The computational core of our information theoretic approach is the mutual information (MI). The MI measures the statistical dependence between two variable, in general, and two gene in the particular case of gene network prediction. The algorithm reported in Chapter 4 allowed to compute the point estimate of each pair of genes in the mammals species we considered. The point estimate does not give a clue regarding the distribution of the MI. Such distribution could be used to directly estimate the significance of the estimation. Here, we propose a statistical model that theoretically could be used to obtain the distribution of the MI associated to a pair of genes.

\section{7.1 MI distribution via a hierarchical statistical model}

Suppose to have \( K \) experiments. The vectors of counts for experiments \( k = 1, ..., K \) are assumed to follow an independent multinomial distribution,

\[ n_k \sim \text{Mult}(M_k, \theta_k) \quad (7.1) \]

\(^1\)http://netview.tigem.it
with the number of hybridizations, $M_k$, known. The parameters $\theta_k$ are assumed to be independent samples from a Dirichlet distribution,

$$
\theta_k \sim \text{Dirichlet}(\alpha) \quad (7.2)
$$

and we shall assign a noninformative hyperprior distribution to reflect our ignorance about the unknown hyperparameters

$$
\alpha \sim \prod_{j=1}^{J} \text{Gamma}(\alpha_j \mid a, b). \quad (7.3)
$$

where $J$ is the dimension of the parameter vector $\alpha$ as well as $\theta_k$ and $n_k$, $k = 1, \ldots, K$.

### 7.1.1 Joint, conditional and marginal posterior distributions

We first perform the three steps for determining the analytic form of the posterior distribution. The joint posterior distribution, $p(\alpha, \theta_1, \ldots, \theta_K \mid n_1, \ldots, n_K)$, of all parameters is

$$
\propto p(\alpha)p(\theta_1, \ldots, \theta_K \mid \alpha)p(n_1, \ldots, n_K \mid \theta_1, \ldots, \theta_K, \alpha) \quad (7.4)
$$

$$
\propto p(\alpha)\prod_{k=1}^{K} [\text{Dirichlet}(\theta_k \mid \alpha)\text{Mult}(n_k \mid M_k, \theta_k)] \quad (7.5)
$$

$$
= p(\alpha)\prod_{k=1}^{K} \left[ \frac{\Gamma(\sum_{j=1}^{J} \alpha_j)}{\Gamma(\alpha_j)} \prod_{j=1}^{J} \theta_k^{\alpha_j-1} \frac{n_k!}{\prod_{j=1}^{J} n_{kj}!} \prod_{j=1}^{J} \theta_k^{n_{kj}} \right] \quad (7.6)
$$

$$
= p(\alpha)\prod_{k=1}^{K} \left[ \frac{\Gamma(\sum_{j=1}^{J} \alpha_j)}{\Gamma(\alpha_j)} \cdot c_k' \prod_{j=1}^{J} \theta_k^{n_{kj}+\alpha_j-1} \right], \quad (7.7)
$$
with \( c'_k = \frac{n_k!}{\prod_{j=1}^J n_{kj}} \). Notice that (7.5) follows (7.4) because \( n_k \) is independent from \( \alpha \) given \( \theta_k \), for \( k = 1, ..., K \).

Given \( \alpha \), all the \( \theta_k \)'s have independent posterior densities given by

\[
p(\theta_k \mid \alpha, n_k) \propto J \prod_{j=1}^J \theta_n^{n_k + \alpha_j - 1} \sim \text{Dirichlet}(\alpha + n_k),
\]

and the joint density is

\[
p(\theta_1, ..., \theta_K \mid \alpha, n_1, ..., n_K) \propto \prod_{k=1}^K \left[ \prod_{j=1}^J \theta_n^{n_k + \alpha_j - 1} \right].
\]

From (7.7), integrating over the \( \theta \)'s, we can determine the marginal posterior distribution of \( \alpha \)

\[
p(\alpha \mid n_1, ..., n_K) \propto \]

\[
p(\alpha) \prod_{k=1}^K \left[ c' \cdot \frac{\Gamma(\sum_{j=1}^J \alpha_j)}{\prod_{j=1}^J \Gamma(\alpha_j)} \int_{\theta_k} \prod_{j=1}^J \theta_n^{n_k + \alpha_j - 1} d\theta_k \right]
\]

\[
= p(\alpha)e'' \prod_{k=1}^K \frac{\Gamma(\sum_{j=1}^J \alpha_j)}{\prod_{j=1}^J \Gamma(\alpha_j)} \prod_{j=1}^J \frac{\Gamma(n_k + \alpha_j)}{\Gamma(n_{kj} + \alpha_j)} \int_{\theta_k} \prod_{j=1}^J \theta_n^{n_k + \alpha_j - 1} d\theta_k
\]

where the last factor of Equation 7.13 integrates to 1 for \( k = 1, ..., K \) given that \( \int_{\theta} p(\theta \mid \alpha) d\theta = 1 \); \( c'' = \prod_{k=1}^K c'_k \). The posterior distribution of the hyper-
parameters becomes

\[
\propto p(\alpha) \prod_{k=1}^{K} \left[ \frac{\prod_{j=1}^{J} [\alpha_j + n_kj]}{\left( \sum_{j=1}^{J} \alpha_j + M_k \right) \left( \sum_{j=1}^{J} \alpha_j + M_k - 1 \right) \cdots \left( \sum_{j=1}^{J} \alpha_j + 1 \right)} \right] \quad (7.14)
\]

\[
= p(\alpha) \prod_{k=1}^{K} \left[ \frac{\prod_{j=1}^{J} [\alpha_j]^{n_{k,j}}}{\sum_{j=1}^{J} \alpha_j} \right] \quad (7.15)
\]

where Equation 7.14 follows the Equation 7.15 by applying a well known property of the Gamma function that states that \( \Gamma(z + 1) = z \cdot \Gamma(z) \).

### 7.1.2 Obtaining a MI distribution

Given the posterior distribution over the hyper-parameter we are able to draw samples from it,

**Algorithm 1** \textsc{Simulation} \( (n) \)

\begin{algorithm}
\begin{algorithmic}
\REQUIRE \( p(\alpha \mid n_1, \ldots, n_K) \).
\STATE \( M = 100 \) \( I = 100 \)
\FOR \( m = 1 \) TO \( M \)
\STATE \( \alpha^{(m)} \sim p(\alpha \mid n_1, \ldots, n_K) \)
\FOR \( i = 1 \) TO \( I \)
\STATE \( \theta^{(i)} \sim \text{Dirichlet}(\alpha^{(m)}) \)
\STATE \( m_i^{(i)} = MI(\theta^{(i)}) \)
\ENDFOR
\STATE \( f^{(m)} = \frac{\sum_{h=1}^{M} m_i^{(h)}}{I} \)
\ENDFOR
\STATE \( MI(\theta \mid n_1, \ldots, n_K) \)
\end{algorithmic}
\end{algorithm}

The Algorithm 1 allows to obtain the distribution of the MI of two variables (genes in this context). However, the \( \theta \) parameters of the posterior distribution in Equation 7.10 have many possible use. These parameters embed the joint (and marginal) knowledge of the variables. Once the parameters
are estimated can be used as prior knowledge for future inferences, when more data come available. A set of useful information the Dirichlet distribution and the inference of the posterior parameters assuming a multinomial distribution can be found in [75, 78, 77, 76].
Appendix A

Parameters

<table>
<thead>
<tr>
<th>NIR</th>
<th>Connectivity</th>
</tr>
</thead>
<tbody>
<tr>
<td>For small gene networks of order 10:</td>
<td>5</td>
</tr>
<tr>
<td>For medium gene networks of order 100:</td>
<td></td>
</tr>
<tr>
<td>TopD</td>
<td>5</td>
</tr>
<tr>
<td>Reest-K</td>
<td>10</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Clustering (Hierarchal)</th>
<th>No of Clusters</th>
</tr>
</thead>
<tbody>
<tr>
<td>For small gene networks of order 10:</td>
<td>3</td>
</tr>
<tr>
<td>For medium gene networks of order 100:</td>
<td>10</td>
</tr>
<tr>
<td>For big gene networks of order 1000:</td>
<td>100</td>
</tr>
</tbody>
</table>
ARACNe

DPI: 0.15

BANJO

searcherChoice: SimAnneal
  initialTemperature 1000
  coolingFactor 0.9
  reannealingTemperature 500
  maxAcceptedNetworkBeforeCooling 1000
  maxProposedNetworkBeforeCooling 10000
  minAcceptedNetworkBeforeReannealing 200

proposerChoice: RandomLocalMove

evaluatorChoice: default

deciderChoice: default

discretizationPolicy: Q3

minMarkovLag (for dynamical data): 1
maxMarkovLag (for dynamical data): 1
maxMarkovLag (for dynamical data): 1
maxMarkovLag (for static data): 0

dbnMandatoryIdentityLags: 1

equivalentSampleSize: 1.0

maxParentCount: 5

maxTime

10 Genes 60 Seconds
100 Genes 600 Seconds

minNetworkBeforeChecking: 1000
Appendix B

Parallel implementation of the mutual information
{discretize local data expression profiles}
for $i = 1$ to $\frac{N}{p}$ do
    \(localE_{\text{dis}}(i,:) \leftarrow \text{discretize}(localE(i,:))\)
    for $j = 1$ to $i - 1$ do
        \{compute mutual information for each pair of local probes\}
        \(localMI(i,j) \leftarrow \text{computeMI}(localE_{\text{dis}}(i,:),localE_{\text{dis}}(j,:))\)
    end for
end for
for $i = 1$ to $\frac{p}{2}$ do
    \{\(\text{P}_{j}\text{mod}\ p\) sends its local discretized data expression profiles to \(\text{P}_{(j+i)}\text{mod}\ p\)\}
    Send\((localE_{\text{dis}},\text{my_rank} + i)\)
    \{\(\text{P}_{(j+i)}\text{mod}\ p\) receives discretized data expression profiles from \(\text{P}_{j}\text{mod}\ p\)\}
    \(E_{\text{recv}} \leftarrow \text{Recv}((\text{my_rank} - i)\)
    \{compute mutual information for all the pairs of probes where the first is the local probe while the second is the received probe\}
    for $j = 1$ to $\frac{N}{p}$ do
        for $k = 1$ to $j - 1$ do
            \(localMI(j,k) \leftarrow \text{computeMI}(localE_{\text{dis}}(j,:),E_{\text{recv}}(k,:))\)
        end for
    end for
end for

Figure B.1: Parallel mutual information algorithm
Appendix C

Matlab Codes

Listing C.1: Simulates network of genes

function simulNet(nGenes, sparsity, Ntgts, Nregs)

global A    \% A will contain the simulated net

\% nGenes: number of genes in the network
\% sparsity: average number of connections for each gene
\% Ntgts: number of non transcription factor genes
\% Nregs: number of transcription factor genes

NonzeroFrac = sparsity; \% Density of network

z = norminv(1−NonzeroFrac/2,0,1); \% threshold

UR = randn(Ntgts,Nregs);    \% generate a random matrix (will
\% be the up−right part of the final
\% matrix) from normal distribution

UR = UR.*(abs(UR)>z); \% keep only those values for which
\% the probability to be in the net
\% is greater than the threshold
\begin{lstlisting}[language=matlab]
21 LR = randn(Nregs);    % these lines of code allow to
22 UL = -eye(Ntgs);     % build a net in which some parts
23 LL = zeros(Nregs,Ntgs); % of it are not randomized (these
24  % represent, respectively, the low–
25  % right, the upper–left and the low
26  % left part of the end matrix)
27
28 A = [UL UR ; LL LR];    % the final net is assembled
29  % from the parts created above
30
31 A = LR.*(LR>z) - eye(Nregs); % keep only those values
32  % greater than the threshold
33  % and add negative values
34  % on the diagonal of it
35 end
\end{lstlisting}
Listing C.2: Simulates static expressions level

```matlab
function simulStaticExpr(nGenes, N, Ntgts, Nregs, policy)

global A Xss P

% these variables represent, respectively, the input net, the matrix of simulated experiments and the applied perturbation

// Ntgts: number of Non transcription factor genes
// Nregs: number of Transcription factor genes
// N: number of Experiments

if policy == 'global'
    % global perturbation policy
    % Create a random perturbation matrix from uniformly distributed values
    P = [zeros(Ntgts,N);rand(Nregs,N)];
end

if policy == 'local'
    % local perturbation policy
    % Create a diagonal perturbation matrix
    P = eye(nGenes,nGenes);
end

Xss = inv(A)*(-P); % Noise free data

return
```
Listing C.3: Simulates static expressions noisy

```matlab
function simulStaticExprNoisey( noiseLevel )

global Xss nXss  % these variables will contain, respectively,
                 % simulated expressions (input) and the noisy
                 % simulated expressions.

sX = noiseLevel;  % level of the external noise

nXss = sX*randn(size(Xss))+Xss;  % Noisy data points: the noise value is
                                  % extracted from a normal distribution
                                  % and than added to the input expressions

return
```

Listing C.4: Check PPV and Sensitivity

```matlab
function [connections_predicted, ppv, sensitivity] = ppv_sensitivity_mod(AA, original_A)

connections_predicted = nnz(AA);

true_positives = nnz((AA+(original_A==0))==2);
false_positives = nnz((original_A==0)-AA)==-1);
false_negatives = nnz((AA-(original_A==0))==1);

ppv = true_positives/(true_positives + false_positives);
sensitivity = true_positives/(true_positives + false_negatives);
```
Listing C.5: Compute Gene Set Enrichment

function [p_vals, idxs] = computeClusterEnrichment(genes, mat)
% genes: position (row) in mat of the genes in the cluster
% mat: matrix (genesXGoterms) of associations

m_vet = sum(mat); % # of genes in the net with a given GO term
k_vet = sum(mat(genes,:)); % # of genes in the cluster for GO term
idxs = find(k_vet); % indexes of non null elements

N = nnz(sum(mat,2)); % # of genes in the net

t_vet = sum(mat,2)~=0;
n = nnz(t_vet(genes)); % cluster size

k_vet = full(k_vet(idxs));
m_vet = full(m_vet(idxs));

%compute the cumulative hypergeometric distribution
p_vals = [];
for kk=1:numel(k_vet)
    p_vals(kk) = sum( hygepdf(k_vet(kk):min(m_vet(kk),n), N, m_vet(kk), n) )

end
Bibliography


[21] Maria S. Carro, Wei K. Lim, Mariano J. Alvarez, Robert J. Bollo, Xudong Zhao, Evan Y. Snyder, Erik P. Sulman, Sandrine L. Anne, Fiona Doetsch, Howard Colman, Anna Lasorella, Ken Aldape, Andrea Califano, and Antonio Iavarone. The transcriptional network for mes-


[27] Alberto de la Fuente, Nan Bing, Ina Hoeschele, and Pedro Mendes. Discovery of meaningful associations in genomic data using partial

Evaluation Studies.


[53] Francesco Iorio, Roberta Bosotti, Emanuela Scacheri, Vincenzo Belcastro, Pratibha Mithbaokar, Rosa Ferriero, Loredana Murino, Roberto Tagliaferri, Nicola Brunetti-Pierri, Antonella Isacchi, and Diego


[84] Kazuhiro Ohmi, David S. Greenberg, Kavitha S. Rajavel, Sergey Ryazantsev, Hong Hua Li, and Elizabeth F. Neufeld. Activated microglia in cortex of mouse models of mucopolysaccharidoses I and IIIB.


[97] Eran Segal, Michael Shapira, Aviv Regev, Dana Pe’er, David Botstein, Daphne Koller, and Nir Friedman. Module networks: identifying reg-


[116] Tetsumori Yamashima, Yukihiro Kohda, Katsuhiro Tsuchiya, Takashi Ueno, Junkoh Yamashita, and Tohru Yoshioka. Inhibition of ischaemic

