Ph.D. in Electronic and Computer Engineering
Dept. of Electrical and Electronic Engineering
University of Cagliari

Simulation and identification of gene regulatory networks

Andrea Pinna

Advisors: Alberto de la Fuente
Nicola Soranzo
Carla Seatzu

Curriculum: ING-INF/04 Automatica

XXVI Cycle
March 31st, 2014

Acknowledgments

Compiling a list of people to whom I am particularly grateful for their essential support during these years of struggles with genes, genotypes, phenotypes, markers, transcription factors, etraits and alike is, possibly, an even more daunting task!

I will start with my supervisors: Alberto, Nicola and Carla have been generous teachers and invaluable advisors, even and above all remotely through emails and Skype conversations – Alberto left CRS4 in 2012! Moreover, I guess that Nicola has been the person with which I shared the most time in the same room during these years... weird!

I also thank all the colleagues that, in one way or another, consciously or inadvertently showed me positive advices or avoidable bad examples, significant knowledge and forgettable boring-as-hell blathers: every little bit helps!

Other, and too many to be singularly mentioned, colleagues at CRS4 made our free lunches – otherwise awful and unhealthy – cheerful and noisy, with talks of dental examinations, fatty donkey challenges, seasonal niggard rankings, detailed soccer statistics and spoiled movie summaries. Thanks for these lighthearted but influential instants!

To all, friends and foes, thanks for having been a substantial part of the path towards another academic title and relevant life experience!

Thanks to my parents, for they uninterrupted support, confidence and love!
Thanks to Carlino, because he could not have got a doctorate degree earlier than me!

And thanks to Camilla, the one who shares her life with mine! ♡

Abstract

Gene regulatory networks are a well-established model to represent the functioning, at gene level, of utterly elaborated biological networks. Studying and understanding such models of gene communication might enable researchers to rightly address costly laboratory experiments, e.g. by selecting a small set of genes deemed to be responsible for a particular disease, or by indicating with confidence which molecule is supposed to be susceptible to certain drug treatments.

This thesis explores two main aspects regarding gene regulatory networks: (i) the simulation of realistic perturbative and systems genetics experiments in gene networks, and (ii) the inference of gene networks from simulated and real data measurements. In detail, the following themes will be discussed: (i) SysGenSIM, an open source software to produce gene networks with realistic topology and simulate systems genetics or targeted perturbative experiments; (ii) two state of the arts algorithms for the structural identification of gene networks from single-gene knockout measurements; (iii) an approach to reverse-engineering gene networks from heterogeneous compendia; (iv) a methodology to infer gene interactions from systems genetics dataset.

These works have been positively recognized by the scientific community. In particular, SysGenSIM has been used – in addition to providing valuable test benches for the development of the above inference algorithms – to generate benchmark datasets for international competitions as the DREAM5 Systems Genetics challenge and the StatSeq workshop. The identification methodologies earned their worth by accurately reverse-engineering gene networks at established contests, namely the DREAM Network Inference challenges. Results are explained and discussed thoroughly in the thesis.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

During the three-year term of the PhD studies I have conducted the related research activity mainly at the Bioinformatics Laboratory of the Center for Advanced Studies, Research and Development in Sardinia (CRS4) under the supervision and partnership of dr. Alberto de la Fuente and dr. Nicola Soranzo, and under the guidance of professor Carla Seatzu from the Department of Electric and Electronic Engineering (DIEE) at the University of Cagliari (UNICA).

The fundamental themes of the above-mentioned collaborative research have been the study of models for the simulation of systems genetics experiments, and the development of algorithms and methodologies for the identification of gene regulatory networks. Such topics, which constitute the first two parts of the thesis, are heavily interlinked since synthetic datasets of biological experiments are necessary – due to the uncertainty on real gene regulatory networks – to develop inference algorithms and compare their performances. A third section describes other side activities that have been nevertheless integral part of the PhD studies.

In particular, a definition of gene regulatory networks and a brief introduction to application tools for the simulation of gene expression data is given in Chapter 2, while our software SysGenSIM is presented in Chapter 3 and the datasets produced as benchmarks for the evaluation and comparison of inference algorithms are extensively described in Chapter 4. An overview of reverse-engineering techniques for gene regulatory networks is given in Chapter 5, while the inference algorithms we developed are presented in Chapter 6 (from datasets of single-gene knockout and knockdown), in Chapter 7 (from heterogeneous compendia), and in Chapter 8 (from systems genetics data). Finally, other activities to a less extent related to the previous, as the estimation of model parameters in gene networks and the processing and analysis of next generation sequencing data, are debated in Chapter 9. Some recapitulatory remarks are given in Chapter 10. The contents of all these topics are briefly introduced in the following paragraphs.

1.1 Simulation of systems genetics experiments

The open source MATLAB package SysGenSIM [16, 136] has been developed to simulate systems genetics experiments in model organisms with the aim of evaluating and comparing statistical and computational methods for the analysis of systems genetics data. In fact, the central goal of systems biology is the understanding of biological networks: this can be achieved by inferring causal networks from observations on a perturbed biological system. Several methodologies have been proposed to identify biological networks whose nodes represent e.g. phenotypes, DNA variants, etraits and other omics variables (e.g. Bayesan networks, differential equation models, structural equation modeling, co-expression networks); on the other hand, assessing strengths and weaknesses of such methods is a tough task unless their performances are fairly evaluated and compared. SysGenSIM therefore allows for the simulation of synthetic datasets meant to be used for training and evaluating algorithms and techniques for the inference of networks from systems genetics data. Benchmark datasets have been simulated and employed for the DREAM5 Systems Genetics challenge and for the StatSeq COST action. Our group is also preparing a review about gene expression simulators of large-scale gene regulatory networks [161].

SysGenSIM

Given a population of individuals, SysGenSIM simulates steady-state gene expression values based on a gene network topology and on the individuals' genotypes, using nonlinear ordinary differential equations. The mathematical model displays two main features of biochemical kinetics: saturation and cooperativity. Most of the equation parameters can be set according to pre-selected common distributions in order to facilitate the customization of the simulated experiments, e.g. the values of the biological variances can be regulated to obtain the desired heritability level of expression traits. An efficient implementation of the steady state solver allows for a quick data generation thanks to a decomposition of the underlying gene network in acyclic and cyclic components: this approach is efficient due to the sparse topology of the gene networks, which can be generated by SysGenSIM to reproduce observed characteristics of biochemical networks like e.g. clustering, degree distributions, motif occurrences. Genetic data produced by SysGenSIM are currently limited to inbred line cross employed in real systems genetics experiments in model organisms and plants, but the simulation of human genotype data is going to be implemented in the near future thanks to an ongoing collaboration with the Virginia Bioinformatics Institute. Last but not least, macroscopic phenotypes can be added to the network by also specifying the genes that affect and that are affected by the phenotypes themselves.

DREAM5 Systems Genetics challenge

The main objective of the Dialogue for Reverse-Engineering Assessments and Methods is to catalyze the interactions between experiment and theory in the area of cellular network inference and quantitative model building in systems biology. Our group co-organized the DREAM5 Systems Genetics challenge [2], which consisted of two sub-challenges, namely DREAM5 SysGenA based on in silico data and designed to elucidate causal network models among genes, and DREAM5 SysGenB based on experimental data on soybean and designed to predict complex phenotypes from a combination of genetics and expression data. We pro-

duced the datasets for the in silico challenge were the 16 participant teams were requested to infer the 15 networks by providing for each a list of edges sorted according to the confidence assigned by their reverse-engineering algorithm. The datasets associated to the networks consisted of genotype and gene expression values computed for 100, 300 or 999 individuals, to further evaluate the strength of the inference techniques with respect to the size of the population.

StatSeq benchmark dataset

During a workshop organized by the Genetical Genomics working group of the StatSeq COST action, the problem of evaluating and comparing the several techniques for the inference of gene networks from high-throughput next generation sequencing data has been thoroughly discussed. To this aim, we published the so-called StatSeq benchmark dataset [135] to investigate the performances of inference algorithms for systems genetics data simulated over various network and population sizes, marker distances, and heritability levels. The identification techniques proposed by the workshop participants are described in [44].

Pula-Magdeburg single-gene knockout benchmark dataset

A recent update of SysGenSIM introduced the possibility to simulate experimental perturbations (i.e. single-gene knockout, knockdown and over-expression experiments) besides the systems genetics experiments. A collection of single-gene knockout datasets has been produced as a genome-scale benchmark for the network inference algorithms described in detail in [132]. The compendium consists of 270 datasets simulated from 30 different 5000-gene networks according to nine different parameter settings. The networks show a similar modular structure but different connectivity, i.e. their average node degrees differ significantly (from about 7500 to 12500 edges) in order to evaluate the performances of the algorithms over the sparsity of the networks besides the different conditions of biological variance and measurement noise.

1.2 Identification of gene regulatory networks

As previously mentioned, the inference of intracellular networks is one of the key challenges of computational and systems biology. Several methods have been proposed and are currently presented, which can be categorized according to the formalism and principle used for inferring the network: sparse regression, correlation-based techniques, z-score, mutual information, Bayesian networks, Gaussian graphical models, random forest, differential equations and Petri networks are some of the approaches proposed in literature. We presented some inference techniques for gene regulatory networks applicable to single-gene perturbation datasets, to heterogeneous gene expression compendia, and to systems genetics experiments, whose effectiveness has been validated at international challenges and through their application on established benchmark datasets.

Inference from single-gene knockout datasets

One of the challenges proposed at the DREAM4 initiative demanded the participants to reverse-engineering a set of five 100-gene regulatory networks given the simulated gene expression value of each gene after separately performing the single-gene knockout and knockdown perturbations for the other genes of the network. We proposed a technique whose final purpose is the removal of the incorrectly predicted indirect edges: (i) an initial prediction of the network is obtained by assigning to each possible edge (i, j) a confidence score consisting in the deviation from the mean of the expression of gene j after the knockout of gene i; (ii) given a threshold value, the edges whose confidence exceeds such limit are selected to build a so-called perturbation graph $\mathcal{G}^P$; (iii) an acyclic graph $\mathcal{G}^A$ is obtained by condensing the perturbation graph $\mathcal{G}^P$, i.e. its nodes are the strongly connected components of the perturbation network $\mathcal{G}^P$; (iv) the edges of $\mathcal{G}^P$ are removed if they connect two different nodes of $\mathcal{G}^A$ and if there is a path of length at least 2 between the above nodes of $\mathcal{G}^A$; (v) the prediction list is created by sorting all the possible edges according to the confidence score cited in (i) with the exception of those still in $\mathcal{G}^P$ whose score is increased to ensure them a ranking higher than the edges deemed unessential. This algorithm [134], called down-ranking of feed-forward loops, was awarded the 1st place at the DREAM4 In Silico Network challenge by achieving a better overall score than other 18 participant teams. The technique has been significantly improved after a collaboration with the "Analysis and Redesign of Biological Networks" group of the Max Planck Institute, which ranked 3rd at the same DREAM4 challenge. The new methodology [132] improves the previous algorithms by (i) using novel statistical criteria for deriving a high quality perturbation graph from the experimental data and (ii) applying local transitive reduction to remove indirect edges. The technique has been successfully evaluated on the DREAM4 datasets, on the Pula-Magdeburg benchmark and on a yeast compendium of real knockout experiments, and its performances considerably exceeded those by previous state of the art inference algorithms.

Inference from heterogeneous datasets

The DREAM5 Network Inference challenge allowed the researchers to propose methodologies to be employed in the task of inferring gene regulatory networks from heterogeneous microarray datasets. One sub-challenge is based on a simulated dataset, while three other are dedicated to experiments performed on real microorganisms: each compendium consisted of steady state or time series of gene expression values simulated or measured under several known experimental conditions, like single- or multiple-gene deletions or overexpressions, drug applications, environmental perturbations, and their combinations. In order to exploit the different types of information included in such mixed datasets, we applied three different approaches to as many subsets of data: (i) on gene-perturbative experiments, potential targets are identified by computing the deviation due to the perturbation; (ii) on steady-state measurements, a confidence score to all potential edges is assigned through full order partial correlation analysis; (iii) on experiments featuring drug perturbations, another confidence score is assigned to edges after a co-deviation analysis. The results of the three approaches have been combined according to a weighted average, whose coefficients had been estimated by trials with comparable datasets simulated by an early stage version of SysGenSIM. The performance of our methodology was awarded with the 2nd position overall in the real networks sub-challenge, and in particular with the 1st place for the identification of

the *S. cerevisiae* network. A paper describing the technique has been submitted [160]. Moreover, our contribution has been included in the DREAM5 community paper [111] where it has been observed that integrating the predictions from multiple inference methods shows robust and high performance across different datasets.

Inference from systems genetics datasets

We developed a technique for the identification of gene regulatory networks from systems genetics experiments, namely genotype and gene expression measurements [133]. The inference methodology is based on three main steps: (i) provide an accurate initial confidence score for all the possible edges in the network; (ii) select the edges satisfying certain statistical conditions to build the perturbation graph; (iii) perform a transitive reduction of the perturbation graph to remove the edges whose effect is identified as indirect. The algorithm performs excellently on the DREAM5 Systems Genetics datasets, and yields scores comparable with the best methodologies presented in [44] when applied to the StatSeq benchmark.

1.3 Side projects

I have been involved in other research activities, in particular the estimation of parameters in models of gene regulatory networks, the wrapping of Galaxy tools for the analysis of Next Generation Sequencing data, and the development of a computational pipeline for the automated processing of Next Generation Sequencing data.

Estimation of model parameters

We took the opportunity to study the problem of estimating the kinetic parameters of given models of gene regulatory networks by participating to the DREAM6 Estimation of Model Parameters challenge, whose goal is the development of optimization methods for the estimate of parameters in the modeling of biological systems. The participants were provided with the full regulatory interaction topology for three small gene networks and were requested to estimate the value of the unknown parameters given a dataset of time-series measurements and the possibility to purchase, given a fixed budget, additional experimental data from a broad assortment. By using a customized version of SysGenSIM we simulated all types of challenges data with the known equations and topologies, and with randomly assigned values in place of the unknown parameters. This enabled us to evaluate optimization algorithms with different purchasing schemes to finally select a common strategy for all three models: unfortunately our approach performed badly compared to the best methodologies. Anyway, this work contributed to the DREAM6 community paper [121].

Next Generation Sequencing data analysis with Orione

Recently I have been involved in a new project: the development of Orione, a Galaxy-based framework designed to build complex and reproducible workflows for the analysis of Next Generation Sequencing microbiology data [39]. The platform allows the researchers to tackle with large-scale datasets, to conduct their own analysis using and interconnecting different software packages, and to easily assembling reproducible analysis workflows. In particular,

I have been responsible for the implementation of several tools in Orione through XML and Python wrappers: BLAT, Glimmer, Edena, SEQuel, Merlin, Mach, Pedstats, Beagle, PLINK, MetaGeneMark, MetaVelvet, and others. Most of these tools have already been released to the official Galaxy Tool Shed repository.

Automated processing of Next Generation Sequencing data

The genotyping platform at CRS4 has a theoretical maximal throughput capacity of nearly 1 TB per day of raw Next Generation Sequencing data. The goal is to fully automate the process of transforming the above-mentioned raw data into aligned and recalibrated data which can be outright analyzed with the software tools provided by Orione or promptly dispatched to the customer. Such automated processing involves several communicating tools, such as e.g. RabbitMQ to manage messages between applications, iRODS and OMERO to allow for the traceability of datasets and operations, Galaxy to run the single steps of the process through the cluster via Sun Grid Engine, Hadoop and Seal to handle the distributed computation of large files. In general, the processing of Next Generation Sequencing data is one of the key problems for bioinformatics: such issues have been discussed at a recent SeqAhead workshop meeting. My contribution to this activity is still negligible and it will not be included in the next chapters.

Chapter 2

Simulation of gene expression data

Living organisms are composed by basic biological units called cells, where all vital functions take place and the genetic information – to regulate such functions – is contained. Deoxyribonucleic acid, also known as DNA, is the molecule encoding the above-mentioned genetic instructions, i.e. during the process of *transcription* (see Figure 2.1) the DNA sequence of guanine, adenine, uracil and cytosine is copied into a complementary sequence of ribonucleic acid (RNA), a sub-category of which, the messenger RNA (mRNA), is then involved in the synthesis of proteins in a process called *translation*. Finally, proteins perform most of the functions needed in living organisms, e.g. catalysis of chemical reactions, transmission of signals between cells, protection, support and movement of tissues.

The functioning of each cell can be modeled through a so-called biological system, i.e. a collection of the interactions occurring between the molecules of the cell. The biological system might be thoroughly described as a whole, but such level of detail would lead to absurdly complex models, whose practical study would be unattainable. Therefore, a simplified model representing the cell functions, employed in most chapters of this thesis, is described in Section 2.1, and an overview[1] of the available software for the simulation and

[1]The review [161] is currently in preparation.

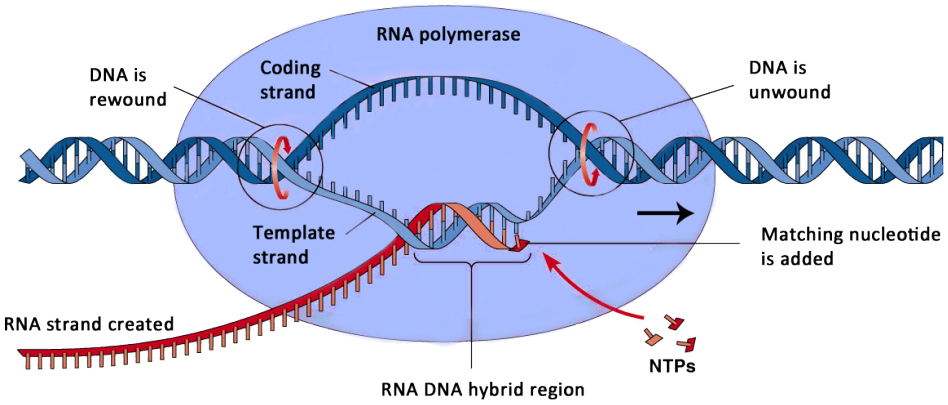

Figure 2.1: **Transcription of DNA into mRNA.**

the study of cell activities in given in Section 2.2.

2.1 Modeling of gene regulatory networks

In real biochemical networks, genes do not directly interact with each other: in fact, gene activation and inhibition processes are e.g. triggered by proteins, which are in turn products of genes; gene activity is also regulated by metabolites (Figure 2.2).

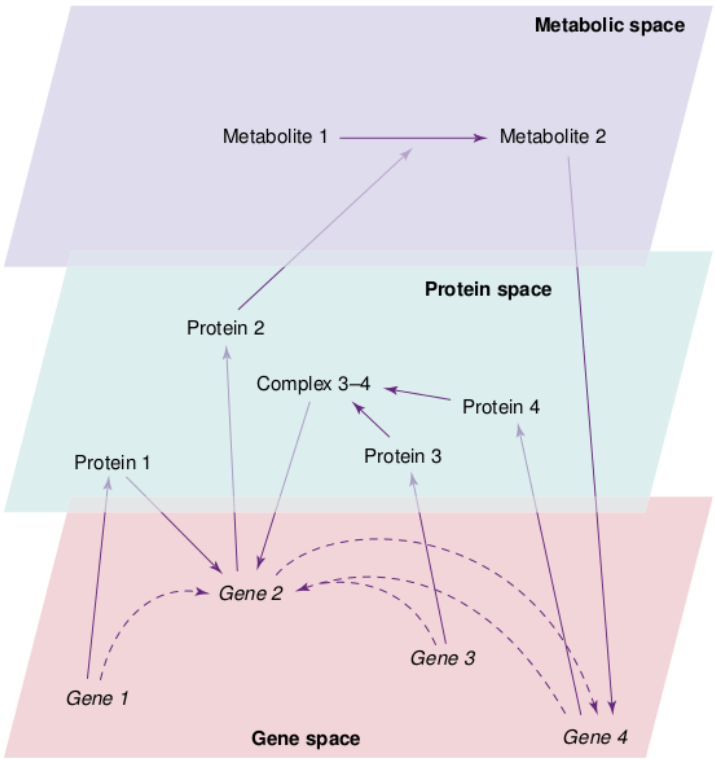

Figure 2.2: **An example of a biochemical network.** Molecular constituents (nodes of the network) are organized in three levels: mRNAs, proteins, and metabolites (from [32]).

On the other hand, a network consisting of only genes is obtained by removing proteins and metabolites from the model of biological network while turning the indirect interactions between genes (e.g. with proteins and/or metabolites as intermediate molecules) into direct influences (Figure 2.3). Gene networks are then representations of biological systems that do not explicitly include other molecules beside genes and their interactions, i.e. are abstract models of gene communication with nodes representing the gene activities, and directed edges representing causal influences. The causal influence of gene A on gene B could be, in fact, due to the transcription activation of gene B by the protein product of gene A upon binding to gene B's promoter sequence (as in a transcription factor–target relationship), but also be due to more complicated processes, such as gene A encoding a metabolic enzyme producing a metabolite which in turn regulates the transcription of gene B. These detailed biochemical events are hidden to the observed set of variables (gene expression levels) and their effects will merely result in an observable causal effect $A \rightarrow B$. Undirected edges in

gene networks are present due to unmeasured confounding variables. Gene networks are context specific: the regulatory structure among genes depends on the developmental stage, cell type, environment, genotype and disease state. For a comprehensive discussion on the nature of gene networks please refer to [43].

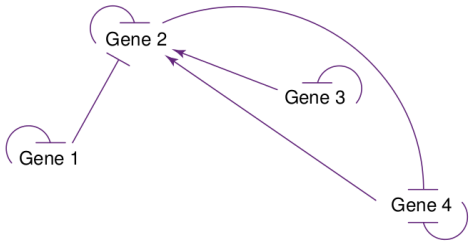

Figure 2.3: **Graph representation of the gene network.** The graph corresponds to the biochemical network in Figure 2.2 (from [32]).

An advantage of gene networks is given by the enormous availability of gene activity[2] measurements, that largely exceeds that of protein and metabolite profiles because of the relative ease and unified way to measure RNA levels; this disproportion will be further increased due to the appearance of gene expression measurements techniques based on novel sequencing technologies, e.g. [176]. The choice of the gene regulatory network model is also extensively supported and motivated [32]: (i) gene networks provide a large-scale but concise representation of the cell functions; (ii) gene expression levels can be quantified and analyzed to identify interactions and then reverse-engineering the network; (iii) newly identified interactions may help researchers to address their expensive studies and experiments; (iv) group of genes may be discovered to be linked to the expression of a particular phenotype, e.g. a disease; (v) understanding the topology of gene networks might explain the robustness of living organisms.

Therefore, the concept of gene network is of high importance for the purpose of describing the regulatory networks inside living cells. As a precise definition of gene networks is missing in current literature we here provide (one possible) formal definition.

Definition. A *gene network* is a mixed graph $\mathscr{G} := (\mathscr{V}, \mathscr{U}, \mathscr{D})$ over a set $\mathscr{V}$ of nodes, corresponding to gene activities, with unordered pairs $\mathscr{U}$, the undirected edges, and ordered pairs $\mathscr{D}$, the directed edges. A directed edge $d_{i,j}$ from v_i to v_j is present if and only if a causal effect runs from node v_i to v_j and there exist no nodes or subsets of nodes in $\mathscr{V}$ that are intermediating the causal influence (it may be mediated by hidden variables, i.e. variables not in $\mathscr{V}$). An undirected edge $u_{i,j}$ between nodes v_i and v_j is present if and only if gene activities v_i and v_j are associated by other means than a direct causal influence, and there exist no nodes or subsets of nodes in $\mathscr{V}$ that explain that association (i.e. it is caused by a variable hidden to $\mathscr{V}$).

In the following of this manuscript, we will mostly refer to a gene regulatory network as a directed graph $\mathscr{G}$ representing through edges (i, j) the causal interactions between the gene activities of genes i and j, i.e. the nodes. The graph is signed when the gene-gene influences

[2] Synonyms for gene activity are mRNA concentrations and gene expression levels.

can be modeled as positive (activation) or negative (inhibition) edges. Moreover, a weight $W_{i,j}$ is assigned to edges when the strength of the interactions is depicted. A convenient representation of gene regulatory network is through the so-called (weighted and signed) adjacency matrix **A** or the even more compact list of edges (particularly advantageous due to the huge sparsity of such networks). Therefore, in this thesis we concentrate uniquely on the simplified model of gene regulatory networks above defined, unless otherwise specified.

2.2 Simulating the transcriptome to evaluate algorithms for the inference of gene regulatory networks

Elucidating the structure of biomolecular networks continues to be a main challenge in modern biology. Many algorithms have been proposed for gene regulatory network inference from gene expression data, and new ones are being proposed at a high rate. Validation of these algorithms is of utmost importance, before they can be confidently applied to biological datasets. Simulated benchmarks provide a way to thoroughly evaluate and compare different approaches. However, these benchmarks are useful only if the simulated data realistically represents real biological data. Here, we review software that has been developed to simulate gene expression data using gene regulatory networks for the purpose of network inference algorithm verification. We highlight how each of these software programs incorporates biological realism.

2.2.1 Introduction

Living cells can be abstracted as biomolecular networks in which nodes represent biomolecules, such as genes, proteins and metabolites, and edges represent causal effects between them. Gene networks are a subset of such networks, specifically focusing on the causal regulatory effects between the expression levels of genes. Inferring such networks from gene expression measurements is one of the main activities in modern post-genomic biology, and many algorithms for this purpose have been proposed [32, 111]. Unfortunately, often algorithms are presented without a thorough evaluation or the results are highly biased [31, 127]. The need for unbiased evaluation of algorithms in computational biology is now internationally widely recognized and gave rise to projects such as DREAM [111, 112] and IMPROVER [120] (see also an international study of algorithms for gene regulatory network inference from systems genetics data [44]).

Ideally, real biological data is used in benchmarking studies. However, biological networks are largely unknown [167]. This severely limits the use of real biological benchmarks. For simple organisms some *bronze standard* networks have been defined [111], but it is not well known how incomplete and reliable these are (hence these are called *bronze standards* instead of *gold standards*). Only for simulated data the true complex system underlying the data is known. Using simulated data to evaluate algorithms is a common practice in many research areas and indeed has been widely used by the gene regulatory network inference community. Figure 2.4 demonstrates the mechanism of algorithm verification by the DREAM project. In early papers on gene regulatory network inference dating back to the

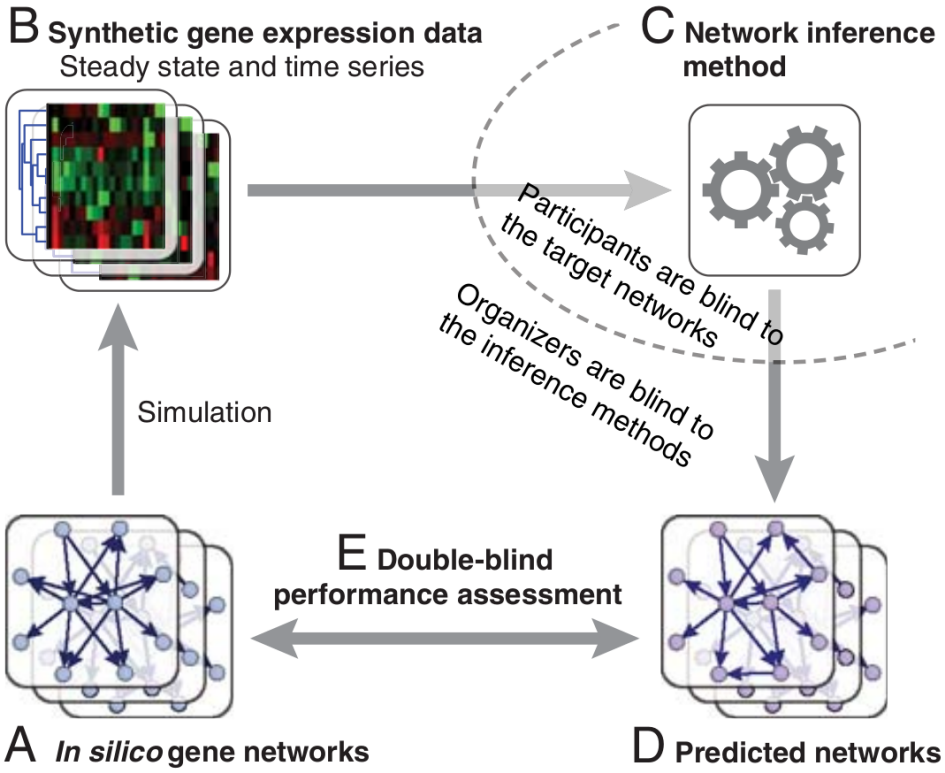

Figure 2.4: **Double-blind performance assessment of network inference methods. (A, B)** From a set of in silico benchmark networks (the so-called gold standards), steady-state and time-series gene expression data was generated and provided as a community-wide reverse engineering challenge. **(C, D)** Participating teams were asked to predict the structure of the benchmark networks from this data. They were blind to the true structure of these networks. **(E)** The submitted predictions were evaluated, blind to the inference methods that produced them. This allowed for a double-blind performance assessment (from PNAS [112]).

year 2000, simulated data were used to evaluate the algorithms [171, 158, 46]. However, data were simulated with the same equations as were used in the inference and/or the test networks were very small. While it is of course interesting by itself, to see whether the systems underlying the data can be identified using the same model as was used for data generation, it is more of mathematical interest than it is of biological interest. It was only in 2003 that Mendes and colleagues suggested simulating large networks with gene expression dynamics governed by biochemically realistic equations (e.g. displaying saturation and cooperativity) for evaluation purposes [119]. After that, several programs have been developed with the same goal in mind: generating biologically realistic in silico data.

Here we briefly review eight programs for gene expression simulation using gene regulatory networks for the purpose of inference algorithm evaluation.

2.2.2 Gene network simulators

GRENDEL

GRENDEL [76] is an open and extensible software toolkit for the generation of random gene regulatory networks according to user-defined constraints on the network topology and kinetics. It simulates the state of each regulatory network under various user-defined conditions (the experimental design) and produces simulated gene expression data, including experimental noise at a user defined level. This toolkit is distributed under a GPL license and is written in C++. It makes use of some external library, like Boost and Xerces, which have been included in the software sources. A concise text file with a description of how to use this tool is provided with the sources but a more detailed description of the parameters would be more useful.

The artificial networks generated by GRENDEL are continuous-time dynamical systems with three independent types of molecular species: mRNA, proteins and environmental stimuli. Generated genes can be over-expressed or knocked-out. The network generation involves two modular steps: topology generation and kinetic parametrization. The topology generation step defines the reagents, catalysts and products of each reaction providing a directed graph (with exponential in-degree and scale-free out-degree nodes distribution) where nodes represent signals and genes; in the kinetic parameterization are chosen the parameters for the linear differential equation that determine the concentration of each protein

$$\frac{\mathrm{d}p_i}{\mathrm{d}t} = \mathrm{TP}_i \cdot m_i - \mathrm{DP}_i \cdot p_i \tag{2.1}$$

where p_i is the protein, m_i is the mRNA, TP_i is the protein's translation rate constant and DP_i is the the degradation rate constant, and each mRNA

$$\frac{\mathrm{d}m_i}{\mathrm{d}t} = S_i(\mathbf{R}) - \mathrm{DM}_i \cdot m_i \tag{2.2}$$

where DM_i is the degradation rate constant of the mRNA, $\mathbf{R}$ is a vector of regulator concentration and S_i maps regulator concentrations to the transcript rate of gene i, reproducing the Hill kinetics. After generating a network, GRENDEL exports it in SBML in order to easily allow the further simulation with an external tool like COPASI [80, 118] or SOSlib [14]; the last one is present in the software sources of GRENDEL and allows to deterministically integrating the ODEs that define the dynamical system obtaining noiseless expression data. Simulated experimental noise can finally be added to the data according to a log-normal distribution, with user-defined variance.

The novelty of the GRENDEL kinetic model lies in its use of more realistic parameters: in fact, the parameter selection process begins by randomly pairing each gene in the synthetic network with a real gene from *S. cerevisiae*; the synthetic network's gene is assigned the translation rate, protein decay rate, mRNA decay rate and mRNA transcription rate of the real gene, which are available from high-throughput studies. In this way the generated networks should behave on the same timescale as a real biological system.

SynTReN

SynTReN [170] creates synthetic transcriptional regulatory networks and produces simulated gene expression data that approximates experimental data. Several user-definable parameters adjust the complexity of the resulting dataset with respect to the structure learning algorithm. SynTReN is distributed with an academic license and is written in Java. It provides a very simple GUI with a default initial value for each parameter and offers also the possibility of using the command line.

The data generation process is composed of three essential steps: in the first step a network topology is selected from a known biological source network (*E. coli* or *S. cerevisiae*), in the second step the transition functions and their parameters are assigned to the edges in the network and in the third step the mRNA expression levels for the genes in the network are simulated under different conditions. After optionally adding noise, a dataset representing normalized and scaled microarray measurements is obtained. Generated networks cannot be large: 400 genes is the limit. This method offers a valid alternative over generating networks using random graph models. However, using previously characterized transcriptional regulatory networks as a source of synthetic network topologies implies a dependency on available knowledge about these networks. Obviously not all the interactions are known and some described interactions might be false positives. Moreover *E. coli* and *S. cerevisiae* networks might be biased towards well studied pathways.

The choice of equations based on Michaelis-Menten and Hill kinetic equations to model regulatory interactions allows a variety of interaction types likely to occur in real biological systems, ranging from a nearly linear behavior to very steep interactions. Although in genuine networks all dynamic interactions are coupled, in the second step of the data generation process it has been assumed that the steady-state kinetics of the complete network of uncoupled equations are comparable to those of the coupled set of equations. Moreover all individual transcription rates are assumed to be in a steady-state regime. Several parameters controlling the gene network generation and sampling process are user-definable in order to generate datasets of increasing level of difficulty; this allows thorough benchmarking of inference algorithms, while low level parameters like kinetic parameters or enzyme kinetic equations are automatically chosen from predefined distributions.

GeneNetWeaver

GeneNetWeaver [153] is an open source toolbox for gene subnetwork extraction, expression data generation and performance profiling of reverse engineering methods. The software is written in Java and can be run either as a standalone program or through a web interface directly from the GeneNetWeaver homepage. Unlike most other gene network simulators, GeneNetWeaver does not generate topologies from scratch, but extract subnetworks from known biological networks, like the *E. coli* an *S. cerevisiae* transcriptional networks. Each subnetwork is grown from a source node by iteratively selecting one of the neighbors of the subnetwork which leads to the highest network modularity. In this way, the extracted networks preserve important properties of the original network such as degree distribution, network motifs and functional annotation. The subnetworks can be visualized with an embedded viewer.

Expression datasets for both mRNAs and proteins can then be generated using an extracted subnetwork as the topology for transcriptional regulations. Transcription and translation are modeled with nonlinear ODEs based on a thermodynamic approach which can express transcription factor binding cooperativity and synergistic interaction. The resulting system of ODEs can be integrated either deterministically or stochastically (i.e. using chemical Langevin equations [67] to simulate molecular noise) to produce steady states or time series. A wide range of experiments can be simulated, like gene knockouts, knockdowns, and multifactorial perturbations (which mimic individual differences as small random variations in the basal activation of all genes). Various models for the measurement noise are also implemented.

Finally, GeneNetWeaver offers some useful tools to evaluate network predictions from inference methods, like the standard precision-recall (PR) and receiver operating characteristic (ROC) curves, and a network motif analysis which profiles the performances of a method in the inference of network motifs. The software is updated with frequent releases, while the website features a complete user manual, a video tutorial and a bug tracker. In summary, GeneNetWeaver is a complete and powerful suite of tools for gene network simulations which has already proven its usefulness to network inference researchers. In fact, it has been used to generate the in silico datasets for the Network Inference challenges of DREAM3, DREAM4 and DREAM5 competitions.

RENCO

REgulatory Network generator with COmbinatorial control (RENCO) [145] is a small command line program for the generation of gene networks and ODEs describing the corresponding mRNA and protein expression dynamics. In RENCO each gene encodes for exactly one protein, proteins interact to form protein complexes, and the transcription of each gene can be combinatorially activated or repressed by various protein complexes. The program can create protein interaction networks with undirected scale-free topology [26], and transcriptional networks connecting proteins to genes with genes having exponential in-degree distribution. Alternatively, both networks can be imported from tab-separated text files.

The ODE for a mRNA or a protein is, as usual, the difference between its synthesis and degradation rates. Protein synthesis rate is directly proportional to the corresponding mRNA concentration. For mRNAs instead, the combinatorial regulation of transcription is expressed as a weighted sum of the contributions given by all possible combinations of protein complexes regulating a gene. The contribution of each set of protein complexes is instead the product of the activating or repressing effects due to the complexes belonging to this set. Finally, the effect of a protein complex is modeled with a typical Michaelis-Menten kinetics, where the substrate concentration is the product of the concentration of the proteins forming the complex. RENCO writes the computed equations to an SBML file, which can then be simulated with external software, e.g. COPASI [80, 118]. This tool has clearly some interesting features, like the explicit modeling of protein expression and protein complexes, combinatorial regulation of transcription, and SBML support. Unfortunately, it is quite limited in scope and has seen no further development after paper publication.

Netsim

Netsim [51] is a full-featured simulator of gene network topologies and gene expression data, implemented as an R package and completely configurable with a GUI. Networks can be generated according to the classic random, scale-free and geometric topologies, but Netsim offers also a novel directed hierarchical graph, called modular topology model (MTM), which exhibits three important characteristics found in real transcriptional networks: scale-free out-degree distribution, small-world property (i.e. low average path length), and high clustering coefficient (independent of the number of nodes). To build an MTM network, three flexible module structures of given average clustering coefficient are randomly replicated at all levels of a hierarchical structure. The user can customize many parameters of an MTM network like the total number of nodes, the power law exponent for the out-degree distribution, the maximum number of nodes of modules and their average clustering coefficient.

Another original feature of Netsim is the use of Boolean fuzzy logic to express the various types of biological interactions among the transcription regulators of a same gene. To this intent, first the expression level of each gene regulator is normalized between 0 and 1, then these values are randomly combined in a target function by using four rules formalizing cooperative, synergistic, negative and competitive regulations. For example, a cooperative effect is expressed as the minimum of the values of the gene regulators, while a negative regulation as 1 minus the value of the regulator. Gene expression profiles are generated by solving a system of ODEs in which the dynamics of each gene is calculated as the difference between its transcription and degradation rates. In particular, transcription depends on the previously described target function, optionally modulated by a sigmoidal function in order to implement saturation effect and activation threshold. Initial conditions and various equation parameters can be defined by the user. A few experimental setups can also be simulated, like gene knockouts and external stimuli to genes.

After running a simulation, the gene network structure can be visualized as a graph, and the expression time series can be easily plotted. Netsim is undoubtedly a complete and user-friendly package which, beyond the standard features, also pushes the state of the art with the new MTM topology and the use of fuzzy logic for regulatory interactions.

GreenSim

GreenSim [60] generates large, genome-size networks with biologically realistic structural characteristic and second order nonlinear regulatory functions. GreenSim, distributed under a GPL license, is a collection of MATLAB functions, so it works within all operating systems that support MATLAB. Unfortunately it lacks a GUI, and the employment of the scripts is not immediate for an inexperienced MATLAB user. On the other hand, a manual is provided. Its most important functions are `genNetwork.m` and `genSample.m`. The first function generates directed networks characterized by exponential in-degree and scale-free out-degree node distributions. This is done by a novel generative method based on half-edges. Moreover, nodes are modularly distributed in the network. The network structure includes cycles and the following motifs: auto-regulatory, cascade, convergence, feed-forward. Such motifs are over-represented in gene regulatory networks compared to random graphs with

the same degree distributions. Generated networks can be quite large (more than the stated 104 nodes). The user can set the size of the network and the only parameter in the exponential distribution. No other algorithms to generate networks are offered to the user.

Gene expression data are produced by `genSample.m` through a nonlinear second order set of functions. In fact, the gene expression is updated as:

$$Y(t+1) = Y(t) + f(Y(t)) = Y(t) + \mathbf{A} \cdot (Y(t) - T) + \epsilon \tag{2.3}$$

where $\mathbf{A}$ is a weighted adjacency matrix that specifies the functional relationships of genes as linear differential equations, T represents the base expression level for mRNA degradation, and ϵ is the biological noise. The user can only specify the maximum regulatory influence that genes can have on others (i.e. the weights in $\mathbf{A}$), and the standard deviation of the Gaussian noise. The outputs are the simulated time series for each gene of the network. Simulation is extremely slow when using nonlinear dynamics for large networks, probably because the code is not optimized. GreenSim, as already said, can simulate the biological (value) noise. Moreover, it can take into account the spot noise (when a time point in a time series is missing) and the span noise (a contiguous subset of time points is missing) issues. This is quite realistic since mRNA measurements are frequently somehow lost or missing. Moreover, GreenSim provides a set of functions to calculate the network statistics, to save and load GreenSim data, and to write data to text file. Finally, scripts are provided to assure that networks and samples are *good* enough (but they obviously slow down the performances).

GeNGe

The web application GeNGe [72] is a framework for the automatic generation of gene regulatory networks that can be used as valid benchmarks for reverse engineering methods. The user can select amongst several network topologies. Moreover, they can load their own network, create a new one, or modify an existing one. This is relevant for comparing the network behavior when the topology is slightly modified. Then, the user selects the kinetic for mRNA transcription, mRNA degradation and protein degradation. Finally, the user selects whether to perform global and/or local perturbations, as noise, single or multiple knockdowns with custom degree of knockdown (from 0% to 100%). Other parameters that can be set are the initial value of the variables and the kinetic parameters (singularly for each gene). The simulation is then executed and data (network, time series of mRNA and proteins, simulation and kinetic parameters, report file) can be downloaded.

Unfortunately the software is not downloadable, and computational performances cannot be compared to those of other simulation software. The web application is neat and user-friendly, but it *reacts* a bit slowly to users' actions. Moreover, after the simulation starts, it is not clear when the results are ready and how much time is needed to perform the calculations. It is also unclear whether the application is still maintained and/or supported.

SysGenSIM

SysGenSIM [136] is a software package to simulate systems genetics experiments in model organisms, for the purpose of evaluating and comparing statistical and computational methods and their implementations for analyses of systems genetics data (e.g. methods for expression quantitative trait loci (eQTL) mapping and network inference). SysGenSIM allows

the user to select a variety of network topologies, genetic and kinetic parameters to simulate systems genetics data (genotyping, gene expression and phenotyping) with large gene networks with thousands of nodes. The effects of the genotypes on the gene expression dynamics is modeled with special parameters as thoroughly explained later in Section 3.6.

In addition to systems genetics data, SysGenSIM can simulate experiments according to a systematic single-gene knockout screen, and has recently been extended to simulate data for *differential networking*. The software is encoded in MATLAB, and a user-friendly graphical interface is provided. SysGenSIM is available for download [16].

SysGenSIM can easily generate data for very large networks: in fact, due to a highly efficient implementation to solve for steady states, The software is able to generate data with networks of 10000 nodes with the nonlinear dynamical model (about two minutes per steady state using a single 2.5GHz core). This is because SysGenSIM decomposes the network into acyclic and cyclic parts. Then, it solves for steady-state values of genes in the acyclic parts analytically very quickly, and only deal with the genes in the cyclic components numerically by using the function `ode45` in MATLAB. The decomposition of the network in acyclic and cyclic components increases the computational efficiency substantially, because cyclic components usually make up a relatively small part of biological networks [107, 108].

2.2.3 Discussion

Here, we reviewed the available software programs to simulate gene expression data for gene regulatory network inference algorithm evaluation. We highlighted the pros and cons of each of these tools, and we put emphasis on the biological realism of the data. Obviously, it is impossible to take into account each and every detail of true data, but we have seen how these programs advanced upon earlier simulations. Much could be further improved.

Topological properties of the networks can be perfectioned by including as many as the properties observed in real biological networks. It has been observed that gene regulatory networks have scale-free out-degree and exponential in-degree distributions. Also it has been observed that biological networks are highly modular. Many other topological features of biological networks are known, which could be incorporated in the benchmark networks [131].

Distributional properties of simulated gene expression data can be improved as to match better the ones that are observed in real data. It has been observed that gene expression means and variances distribute roughly according to a power law. Benchmark datasets could be approved by being generated with similar distributions. Figure 2.5 shows some distributions for real data of *Arabidopsis thaliana* (left) and their counterparts in data simulated with SysGenSIM (right). As can be seen, the distributions match quite well. Still, parameter values could be adjusted to improve the match. An interesting recent paper proposes a kind of Turing test for simulated gene expression data [110]. The authors suggest certain measures and tested them for three of the above described simulators. Figure 2.6 shows the results of their comparison with *E. coli* and *S. cerevisiae* gene expression data. In addition of the properties listed in Figure 2.6 for the Turing test, there are lots of additional possibilities. Simulated benchmarks should always pass these tests as much as possible for them to be useful in ge-

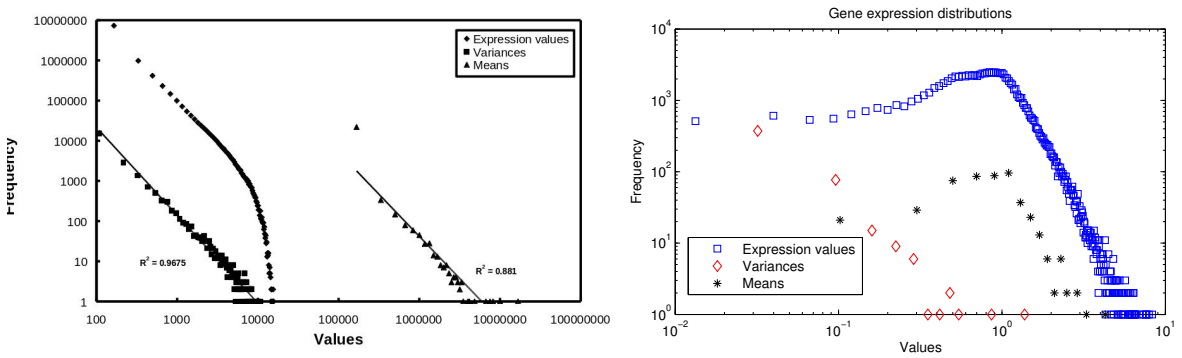

Figure 2.5: **Distributions of the gene expression values, means and variances.** Comparison between real *A. thaliana* measurements (left, from [180]) and data simulated with SysGenSIM (right).

Section	Type of analysis	Motivation and questions	Figure: Simulator
4.1	Replicate noise distribution	Reproducibility of measurements, likelihood of outliers	2a: GNW
4.1	Intensity distribution	Choose normalization method, evaluate effectiveness	2b: GNW
4.1	Range of gene expression	Count distinguishable gene levels	2cd: GNW
4.2	Gene silhouette coefficient	Assess gene cluster quality	3a: Grendel
4.2	Chip silhouette coefficient	Assess microarray cluster quality	3b: Grendel
4.3	TF-TG correlation	Dependencies between expression of TFs and TGs	4a: SynTReN
4.3	TG-TG correlation	Co-regulation of TGs induced by common TFs	4b: SynTReN
4.3	TF activity distribution	Distribution of TF activities across conditions	4c: SynTReN

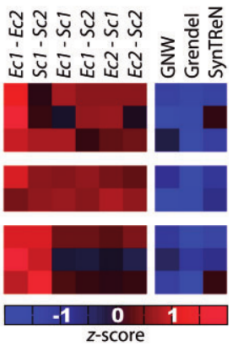

Figure 2.6: **Overview of tested properties and corresponding areas of expression data analysis.** Expression analyses were performed to compare four real and three in silico expression compendia. For each type of analysis, figures in [110] use in silico data of a different simulator. Additional analyses were performed across all simulators, and corresponding results are shown in the heat map on the right. In the heat map, different analyses (rows) compare various pairs of *E. coli* and *S. cerevisiae* datasets based on histogram overlaps. The heat map depicts the extent to which dataset properties are shared across compendia. Overlaps are shown in units of row-wise standard deviation. In case of the three simulators, the average overlap to the four real expression compendia is shown (from [110]).

nomic research.

With this review we hope to have given the reader a better idea of which simulator to choose for evaluation studies. We think that it is probably the best to use more than one and test algorithms on a variety of simulated data. Several internationally used benchmarks are already available, such as the ones of the DREAM, simulated with GeneNetWeaver [8], and a suite of benchmarks generated with SysGenSIM [15].

Chapter 3

Simulating systems genetics data with SysGenSIM

SysGenSIM is a software package, presented in [136], originally intended to simulate systems genetics experiments in model organisms, for the purpose of evaluating and comparing statistical and computational methods and their implementations for analyses of systems genetics data, as e.g. methods for expression quantitative trait loci (eQTL) mapping and network inference. SysGenSIM allows the user to select a variety of network topologies, genetic and kinetic parameters to simulate systems genetics data (genotyping, gene expression and phenotyping) with large gene networks with thousands of nodes. The software is encoded in MATLAB, and a friendly graphical user interface is provided.

3.1 Introduction

The central goal of systems biology is to gain a predictive, system-level understanding of biological networks. This entails inferring causal networks from observations on a perturbed biological system. An ideal experimental design for causal inference is randomized, multifactorial perturbation [58]. The recognition that the genetic variation in a segregating population represents randomized, multifactorial perturbation [84, 85] gave rise to *genetical genomics* and *systems genetics*, where a segregating or genetically randomized population is genotyped at (many) DNA variants, and is profiled for (disease) phenotypes of interest, genome-wide gene expression and potentially other omics variables (epigenomics, microRNA expression, proteomics, metabolomics, etc.). Systems genetics experiments and studies enable us to elucidate the genetic control of gene expression (and other omics variables) [35, 89, 150], to annotate DNA polymorphisms implicated in previous genome-wide association studies (GWAS) for particular diseases and to infer key control genes and pathways causally underlying a disease or biomedical trait of interest [143, 148].

Many statistical and computational methods are being developed for the analysis of systems genetics data. An important component of any systems genetics analysis is the quantitative trait locus (QTL) mapping of all expression traits (etraits) and other omics traits if available. It is well known that the etraits of groups of genes share common regulators (DNA

variants), which are more easily identified when associated with a group of etraits rather than with individual etraits. Several approaches to associating DNA variants with groups of etraits have recently been proposed, e.g. [38, 97, 130, 173, 185]. A major goal of systems genetics studies is to reconstruct a causal network whose nodes are the phenotypes, the etraits (and potentially other omics variables) and the DNA variants. Methods proposed to achieve this goal include Bayesian networks [186], differential equation models [23, 46], structural equation modeling [101, 103] and undirected dependency graph or co-expression network with edge orientation using DNA variants as causal anchors [21, 126]. While multiple methods for QTL mapping of etraits (omics variables) and for causal network inference are available, at the present time not much is known about the strengths and weaknesses of all of these proposed methods and whether or when some methods perform better than others. However, researchers increasingly realize that thorough verification of algorithms in bioinformatics and (genetical) systems biology is required. In fact, several international competitions are organized on an annual basis to compare computational methods for systems biology and genetic analysis. These include the Dialogue for Reverse Engineering Assessments and Methods (DREAM) project with its reverse-engineering challenges [17, 164, 165], for which SysGenSIM has been used to produce the systems genetics challenges in 2010, and the Genetic Analysis Workshops [9, 40], which compare analysis tools relevant for current analytical problems in genetic epidemiology, statistical genetics and genetical systems biology.

The availability of realistically simulated (artificial) datasets, which are generated under a set of assumptions most relevant to real systems genetics data, is of utmost importance for the verification of algorithms for systems genetics data analysis. Several systems genetics papers use simulations which are typically simplistic and not general, e.g. [103, 185, 186]. Other more general software packages have been developed for simulating gene expression data with network models for gene network inference algorithm evaluation (e.g. ABIOCHEM [119], GeneNetWeaver [114, 153] and Ingeneue [117]), but experimental designs are restricted to time-series and steady-state measurement after environmental or kinetic parameter perturbations, and single-gene perturbation experiments. These and other existing packages do not permit the simulation of systems genetics data, in particular the integration of DNA variation, transcriptomics, epigenomics, etc. This is the reason why we have developed and continue to develop SysGenSIM to simulate systems genetics data.

The above features are still maintained in the current release of the software, and are moreover supported by the newly implemented capability of reproducing experimental perturbations, i.e. single-gene knockout, knockdown, and over-expression experiments. Such data can be of great use for the evaluation of algorithms for the inference of gene networks, as demonstrated in [134, 132, 160, 44] where our in silico data turned out as valuable benchmarks for testing the developed inference algorithms. Moreover, thanks to an ongoing collaboration with the Virginia Bioinformatics Institute, the simulation of gene expression data given real or in silico human genotypes is being currently implemented in SysGenSIM. The produced datasets are used to develop analysis techniques for explaining the genetic basis of variation in complex traits.

The software is presented as a MATLAB toolbox complete with a graphical user interface, that has been recently updated from one large window into six compact panels, each

dedicated to a particular section of the simulation process:

- selection of the simulation type, and number of total runs;
- generation of the gene network;
- definition of the genetic experiments;
- definition of the gene perturbation experiments;
- selection of the model parameters;
- selection of the output files and figures.

The first panel allows the user to select the desired simulation type between the two possible choices `systems genetics` and `experimental perturbations`. Moreover, the number of replicates of the entire experiment can be augmented in order to replicate the simulation (from the network generation to the output selection) a desired number of times by keeping the same configuration of SysGenSIM settings. This feature allows the user to e.g. simulate similar datasets in order to verify and reproduce the result of a certain analysis performed on the data.

3.2 Network topology

The precise topological structure of genotype-gene-phenotype networks is largely unknown. Multiple studies (protein interaction, metabolomic, transcriptomic, etc.) provide evidence for topologies that are scale-free, hierarchical and modular. Many algorithms to generate (or *grow*) networks in silico have been proposed, each reproducing particular characteristics observed in biomolecular networks (such as clustering, degree distributions, motif occurrences, etc.), but none can generate networks displaying all observed topological properties simultaneously.

SysGenSIM provides the users with several choices to generate directed gene networks, including the following topology models:

- `random` [56] and `random acyclic`, with equiprobable edge directions;
- `scale-free` [25], with hub's edges generally directed outward; these networks are generated by sampling the out-degree sequence from a power law, and then randomly connecting the nodes;
- `small-world` [179], with user-specified `rewiring probability`;
- `EIPO`, which stands for exponential in-degree and power law out-degree distributions, which is of particular interest as these distributions have been observed in real gene networks [70]; these networks are generated by sampling the in- and out-degree sequences from the distributions of interest and subsequently the nodes are connected according to their degrees;
- `EIPO modular`, which is similar to the EIPO but consists of modules (genes clustered into densely connected components) which is also an often observed property of biological networks [26, 75]; these networks are created by generating a number of EIPO

modules and then by connecting them through edge rewiring (with a probability specified at parameter `rewiring probability`);

- `random modular`, which are generated in the same way, but starting with random modules instead of EIPO modules.

SysGenSIM also allows the user to input the network structure as inferred from an actual dataset in the form of a (signed) edge list, e.g. to reproduce the gene expression of a real network under particular experimental settings.

Moreover, the users can select from the graphical user interface (see Figure 3.1) the parameters to specify:

- the `size` of the network, i.e. the number of nodes (genes);
- the `average degree`, i.e. the average number of neighbors of a node in the networks. For the modular networks, a single, average degree common for all the modules can be specified, or an array of average degrees, one entry for each module of the network;
- the `signs` of the edges, determining which regulatory effects are activating or inhibiting;
- the `sign probability`, i.e. the probability for an edge to have a positive sign;
- the `module sizes`, applicable only when a modular topology has been selected;
- the `rewiring probability`, i.e. the probability of rewiring edges after the initial creation of a regular ring lattice [68].

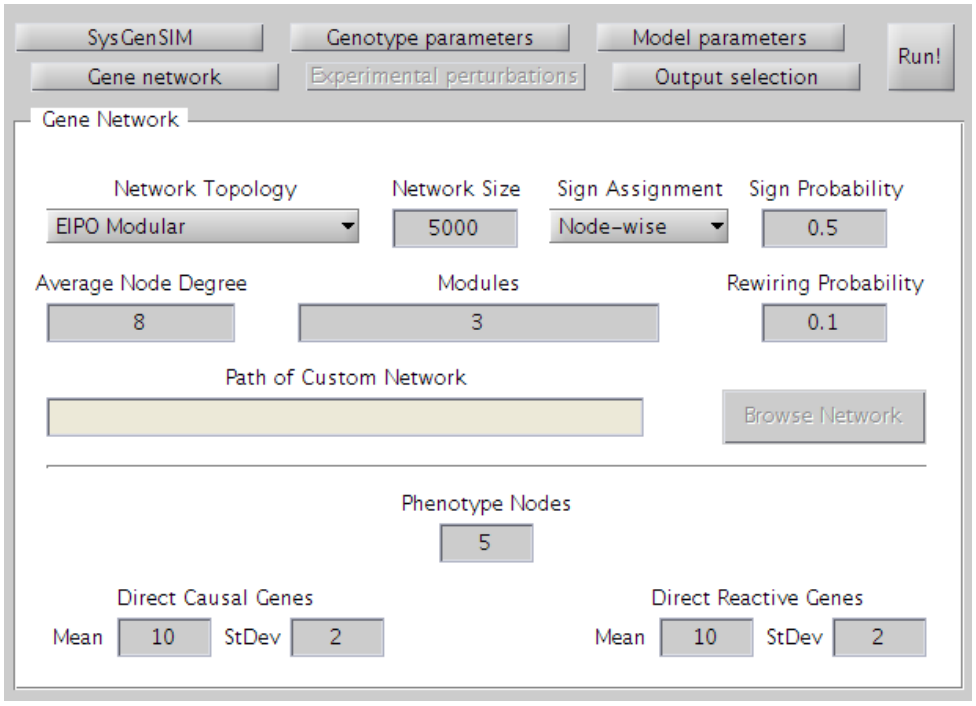

Figure 3.1: **Gene network panel from SysGenSIM's graphical user interface.**

3.3 Phenotype data

The user can select one or more continuous macroscopic phenotypes which will be added as nodes to the gene network (see the lower part of Figure 3.1). As genes can be causal or reactive to the phenotype(s) [149], the user can select the number of genes which directly affect a phenotype and the number of genes which are directly affected by a phenotype. Inputs and outputs of the phenotype node are randomly selected from the gene network. Currently, a phenotype is modeled with Equation 3.1 where its expression nonlinearly depends on its input genes and additional biological variability.

3.4 Genetic data

In terms of the type of the segregating population of individuals for which the systems genetics data are generated, SysGenSIM is currently limited (see Figure 3.2) to an inbred line cross commonly employed in real systems genetics experiments in model organisms (e.g. mouse) and plants: Recombinant Inbred Lines (RIL) created by `selfing` or `brother-sister matings` from two inbred parental lines.

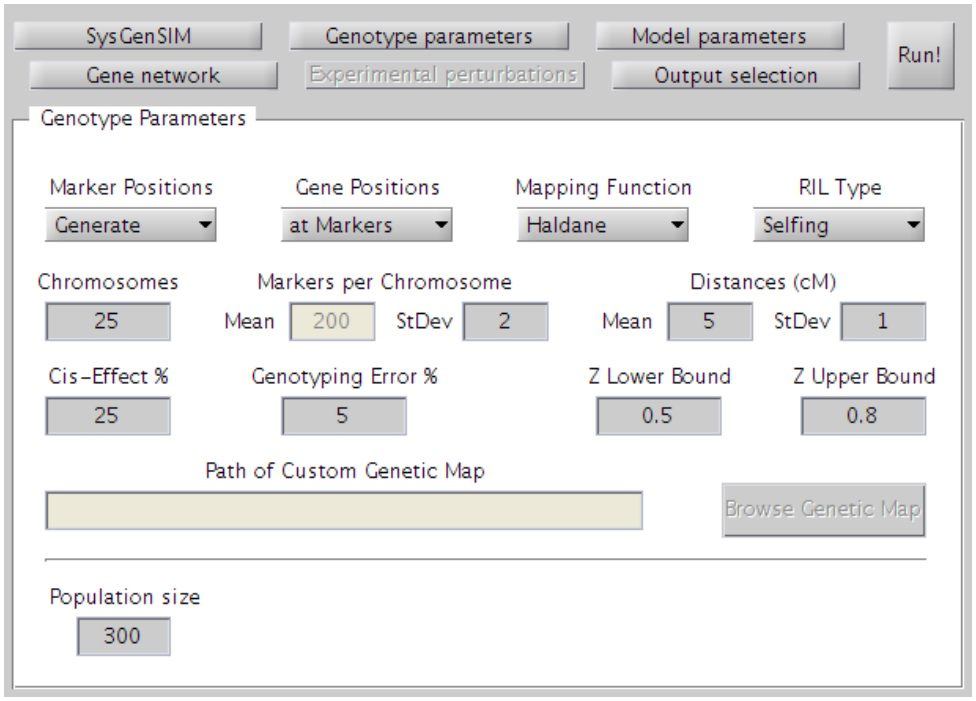

Figure 3.2: **Genotype panel from SysGenSIM's graphical user interface.**

In an RIL population, each DNA variant has two genotypes. SysGenSIM simulates genotype data at all functional (gene) and measured (marker) DNA variants according to a randomly generated `genetic map` based on user-specified parameter values (e.g. `chromosome number`, number of `genetic markers per chromosome` with constant or normally distributed `pair-wise distance` among DNA variant locations in centimorgan[1]) or based on a

[1]A centimorgan (cM) is a unit used to measure genetic linkage. One centimorgan equals a 1% chance that

Table 3.1: **Example of user-defined genetic map.**

Chromosome	Markers		cM
C1	M1	M2	5.1
C1	M2	M3	4.9
C1	M3	M4	3.5
...	...	...	...
C2	M21	M22	4.9
C2	M22	M23	7.2
...	...	...	...
C21	M2341	M2342	1.1
C21	M2342	M2343	3.7
...	...	...	...
C21	M2507	M2508	3.9

(real) map provided by the user (e.g. in Table 3.1). The user can choose between the `mapping functions` of Haldane [73] or Kosambi [93] to convert map distances to recombination rates in the generation of genotypes at linked loci. The user can choose between placing one marker in perfect linkage with each functional polymorphism (in this case the number of markers is equal to the number of genes, i.e. the network size) or generating a (sparser) marker map first and then placing functional variants randomly throughout the genome (at minimum distance of 100 kb; see Section 3.7).

With regards to human data, SysGenSIM can now simulate gene expression data when provided with an appropriate heterozygous genotype matrix, e.g. produced by [182].

3.5 Experimental perturbations

Experiments in which the expression of some genes is perturbed can be simulated by SysGenSIM. In particular, the available experimental perturbations are:

- `knockout`, where n_{ko} experiments (each involving a different gene) simulating single-gene deletions are performed;
- `knockdown`, where n_{kd} experiments (each involving a different gene) simulating single-gene knockdown are performed;
- `over-expression`, where n_{oe} experiments (each involving a different gene) simulating single-gene over-expressions are performed;
- `mixed perturbations`, where $n_{\text{tot}} = n_{\text{ko}} + n_{\text{kd}} + n_{\text{oe}}$ single-gene experiments of the three types above are performed.

The multiplicative coefficients (see Section 3.6) that affect the expression of the perturbed genes are sampled from a uniform distribution defined by the user-defined intensity range.

a marker on a chromosome will become separated from a second marker on the same chromosome due to crossing over in a single generation.

Genes undergoing systematic perturbations can be selected according to four choices:

- `all genes`, where all the n genes of the network are subjected to the selected perturbation;
- `only TFs`, where only the genes with at least one outgoing edge (transcription factors) undergo the selected perturbation;
- by `percentage`, where only a certain proportion (from 0 to 100%) of genes withstands the selected perturbation;
- by `indexes`, where only the specified genes (according to their indexes chosen from $\{1,2,\ldots,n\}$) will be subjected to the selected perturbation.

The above settings allowed us to simulate the datasets used to assess the performance of our inference algorithms for the DREAM challenges, and might then be useful to perform similar analyses for the development of techniques for the identification of gene networks.

3.6 Gene expression dynamics

Steady-state gene expression traits are simulated for a population of individuals, based on a gene network topology and the individuals' genotypes at a set of genome-wide DNA variants, using nonlinear ordinary differential equations (ODEs). The rate law used in SysGenSIM for transcription is not based on any explicit biochemical mechanism, but it displays two main features of biochemical kinetics: saturation and cooperativity [119]. We assume that mRNA decay is a first order process. The ODE for gene i is:

$$\frac{\mathrm{d}G_i}{\mathrm{d}t} = Z_i^c \cdot V_i \cdot \vartheta_i^{\mathrm{syn}} \cdot \prod_{j \in \mathscr{R}_i} \left(1 + A_{j,i} \frac{G_j^{h_{j,i}}}{G_j^{h_{j,i}} + (K_{j,i}/Z_j^t)^{h_{j,i}}} \right) - \lambda_i \cdot \vartheta_i^{\mathrm{deg}} \cdot G_i \tag{3.1}$$

where G_i is the mRNA concentration of gene i, V_i is its basal transcription rate and λ_i is the degradation rate constant. The G_j are the mRNA concentrations of genes $j \in \mathscr{R}_i$ which have directed edges into node i, i.e. $\mathscr{R}_i$ is the set of regulators of gene i. $K_{j,i}$ is a Michaelis constant (representing the concentrations of input gene j at which its effect on the transcription rate of gene i is half of its maximum effect), $h_{j,i}$ is a cooperativity coefficient and $A_{j,i}$ is an element of matrix $\mathbf{A}$ encoding the signed network structure ($A_{j,i} = -1$ for inhibitor, $A_{j,i} = 1$ for activator, $A_{j,i} = 0$ for no effect). The parameters $\vartheta_i^{\mathrm{syn}}$ and $\vartheta_i^{\mathrm{deg}}$ represent non-genetic internal biological noise in the transcription and degradation rates, respectively; their values are sampled from normal distributions with mean 1 and user-specified standard deviations prior to the calculation of each steady state. Z_i^c and Z_j^t are parameters which incorporate effects of DNA variants (see Section 3.7 for details). After generating a network topology (Section 3.2), the nonlinear equations are formulated according to this topology, encoded in matrix $\mathbf{A}$. Kinetic parameters V_i, $K_{j,i}$, $h_{j,i}$ and λ_i are initialized by sampling values from certain distributions[2] to generate a set of base parameter values, i.e. the *genetic background* of the organism. The gene expression variability among individuals in the population results

[2] Uniform, (truncated) Gaussian or Gamma with default or user-specified parameter values.

from different genotypes (values of the Z_i^c and Z_j^t parameters) and additional biological fluctuations (represented by the noise parameters ϑ_i^{syn} and ϑ_i^{deg}).

After setting the values of all parameters Z_i^c and Z_j^t according to the genotypes of an individual in the population, a value for the biological noise terms ϑ_i^{syn} and ϑ_i^{deg} is sampled, and the steady-state mRNA concentrations are calculated. This process is repeated for all individuals in the population. Finally, normally distributed multiplicative experimental noise is added to each mRNA concentration at a user-specified level, resulting in a set of expression values for all genes in the system and all individuals. The values for parameters ϑ_i^{syn} and ϑ_i^{deg} , and the experimental noise level can be chosen such that the distribution of estimated heritabilities of the etraits[3] is close to those found in real data. For example, in our previous work [103] the simulated expression traits had an average heritability of 56%, close to what was observed in a yeast systems genetics experiment [33]. Due to a highly efficient implementation to solve for steady states, SysGenSIM is able to efficiently generate data with networks of 10000 nodes with the nonlinear dynamical model (~ 2 minutes per steady state using a single core of an AMD Opteron X2380 QuadCore, 2.5 GHz). This approach will be described in detail elsewhere, but essentially we solve for steady-state values of genes that are not involved in any cycle very quickly and analytically, while we only deal with the cyclic components of the network numerically by using the function `ode45` in MATLAB. The decomposition of the network in acyclic and cyclic components increases the computational efficiency substantially, because cyclic components usually make up a relatively small part of biological networks [107, 108].

3.7 Genotype effects on expression dynamics

We currently assume that each gene in the network has a single functional DNA variant. The variant is located either in the gene's promoter region affecting its own transcription rate (cis-variant with, for example, $Z_i^c = 1$ for one genotype and $Z_i^c = 0.75$ for the other; reduced Z_i^c reflects a less efficient transcription process), or in the coding region of a regulatory gene altering the strength of its regulatory effect (trans-variant for which a reduced Z_j^t reflects a less potent inhibitor/activator). Promoter variants modify the kinetics of recruitment of the transcriptional machinery to the promoter sequence, which affects the efficiency of transcription, so a change in Z_i^c results in a change of the basal transcription rate of G_i. A trans-effect occurs through changes in the kinetic properties of the product of the gene containing the polymorphism in its coding region.

Because we do not explicitly include proteins in our networks, we model these kinetic changes by their effect on the transcription rates of the target genes, by altering their Michaelis constant. The protein products of allelic variants of G_j may have reduced or increased strength through adjustment of Z_j^t. The probabilities of a locus acting in cis or in trans can be set by the user, as well as the allelic values of Z_i^c and Z_j^t .

[3] Steady-state variances simulated without biological (ϑ_i^{syn} and ϑ_i^{deg}) and experimental noise divided by the steady-state variances simulated with these noise terms.

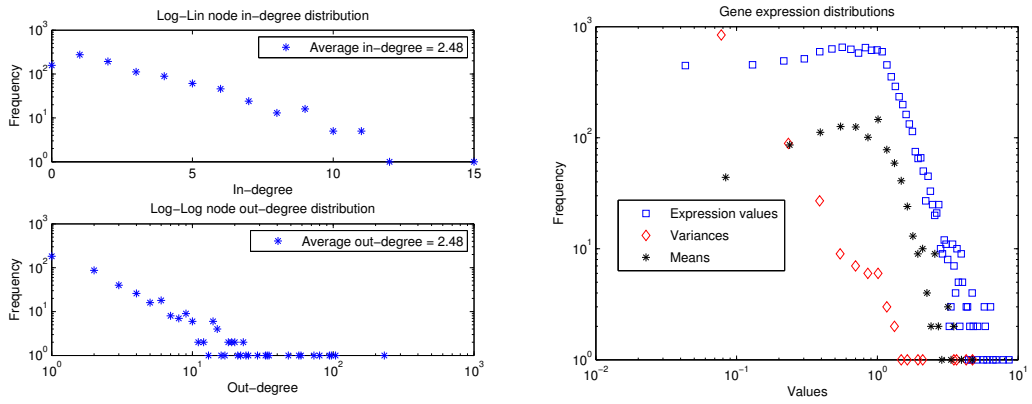

Figure 3.3: **Some of the output figures produced by SysGenSIM.** Network node in- and out-degree distributions (left) and simulated gene expression distributions (right).

3.8 Output files and figures

SysGenSIM's users can select which simulated data output to file or figure. As an example, `genotype`, `gene expression` and `phenotype` matrices can be produced. The network structure can be output by selecting `edge list`, and the `genetic map` can be selected as output as well. A `Pajek network file` [27] can be produced to visualize the network and perform some network analyses. The `module list` contains lists of genes pertaining to each module. The `topological properties` file provides some standard topological information, for example the in- and out-degree of each node, the network component each node belongs to (strongly connected, in- or out-component, tendrils or tubes) and other indexes.

In `genotype information` all the available information about the DNA variants is provided, including the chromosome number and location, and for functional variants the trans or cis status, the value of the corresponding Z parameter and which genotype (parental origin) was assigned the $Z = 1$ value. The `perturbation list` enumerates the experiments, described with their perturbation type and involved gene. Finally, the `simulation summary` file contains the chosen values of all simulation parameters, to eventually later reproduce a similar simulation.

The distributions of `in- and out-degree` and of `gene expression` (see Figure 3.3) are some of the output figures produced by SysGenSIM, together with the distributions of the `kinetic parameters`, of `heritability`, and the `gene-gene correlations`.

3.9 Future development

SysGenSIM is a work in progress with many possible future developments. Of highest priority are improvements to the simulation of the continuous phenotype nodes (e.g. realistic heritabilities, numbers and sizes of QTLs, numbers of causal and reactive modules), the inclusion of discrete (disease) phenotype data, and extensions of the simulation of genotype and steady-state data to other types of inbred line crosses and to human cohorts and case-

control designs.

To keep pace with recent and future real systems genetics experiments and studies, we plan to extend the simulation of genotype data from bi-allelic DNA variants (single nucleotide polymorphisms) to copy number variation and to incorporate epigenomics data (e.g. DNA methylation sites) and microRNAs into the gene networks. Given the general systems genetics simulation methodology described in this article and the existence of simulators for genome-wide association studies (HapSample [182] and genomeSIMLA [53]), these extensions are actually quite straightforward.

Furthermore, to ensure that the simulated data display known characteristics of real systems genetics data, such as distributions of means, variances and heritabilities of etraits and correlations among etraits, we will continue to estimate the values of such parameters from real systems genetics data and utilize the results from similar studies in the literature. Finally, we continue to implement additional topology models for the generation of gene networks (with emphasis on hierarchical modularity and scale-free out-degree and exponential in-degree distributions).

Chapter 4

Benchmark datasets

Datasets produced by SysGenSIM has been employed as benchmarks for the assessment of network inference algorithms in international challenges (DREAM and StatSeq initiatives in Sections 4.1 and 4.2) and for the testing and development of in-house methodologies presented in scientific journals (see Chapters 6, 7 and 8 for details about the techniques and their performances).

4.1 DREAM5 Systems Genetics challenge dataset

DREAM is the acronym for Dialogue for Reverse Engineering Assessments and Methods, an international initiative whose objective is "to catalyze the interaction between experiment and theory in the area of cellular network inference and quantitative model building in systems biology" [17]. We were invited to organize the DREAM5 Systems Genetics challenge, whose goal is the creation of models with biological interpretation, i.e. reverse-engineering gene networks from systems genetics data.

From previous DREAM challenges, especially both the DREAM3 [138, 112] and DREAM4 In Silico Network challenges, it has become unambiguously clear that systematic perturbations (e.g. experimental gene knockouts and knockdowns) and measurements of responses greatly contribute to establish the directed structure of gene networks. However, large scale systematic knockouts may be unrealistic or unfeasible for many cell types and even impossible for some organisms. Systems genetics experiments, as considered here, could provide an alternative. In systems genetics [85], in fact, a segregating or genetically randomized population is genotyped for many DNA variants, and profiled for phenotypes of interest, gene expression, and potentially other *omics* variables (e.g. protein expression, metabolomics, DNA methylation; see Figure 4.1).

Genetic polymorphisms, which are naturally present in populations, act as multifactorial genetic perturbations that could be used to elucidate causal links between genes. For example, if the mean expression levels of gene B are significantly different between two groups of individuals, one with one genetic variant of gene A and the other with another genetic variant of gene A, this observation is highly indicative for a causal regulatory effect $A \rightarrow B$.

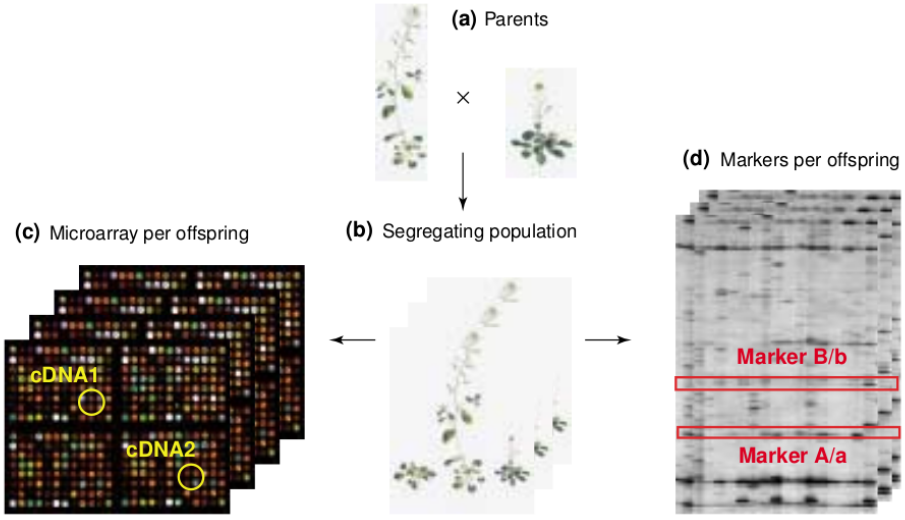

Figure 4.1: **Diagram for quantitative trait loci (QTL) analysis.** Expression profiling in combination with molecular marker analysis of a segregating population makes it possible to use QTL analysis for identification of influential genes and gene products (from [85]).

Recently, multiple approaches have been applied to systems genetics data in order to elucidate gene networks; see [143].

4.1.1 Simulation of genotype and gene expression datasets

Due to the lack of a reliable experimentally determined gold standard network, this challenge is based on synthetic systems genetics data [103] simulated with SysGenSIM [136] through the dynamical model described in Equation (3.1). We set the values of parameters Z^c and Z^t to either 1 or 0.75 depending on the binary genotype value. ϑ_i^{syn} and ϑ_i^{deg} represent fluctuations in the transcription and degradation rates, respectively, and are sampled from a normal distribution before the calculation of the steady state. All other parameters remain fixed throughout the generation of a dataset: for simplicity we have set $V_i = K_{ji} = \lambda_i = 1$. The value of cooperativity coefficients h_{ji} is set to 1, 2 or 4 with probabilities 0.6, 0.3 and 0.1, respectively.

In these simulations we consider data from Recombinant Inbred Lines (RILs), i.e. a set of homozygous lines derived from a cross between two genetically diverse inbred parent lines, through inbreeding for multiple generations. Each of these RILs is homozygous for the allele of one of the parents, and each RIL has inherited different combinations of parental alleles: the RILs constitute a genetically randomized population. In other words, the gene expression pattern of each RIL is the result of a different multifactorial genetic perturbation.

We proposed three sub-challenges, each with 5 different networks, and each network with different RIL populations of size: $p = 100$ (sub-challenge 1), $p = 300$ (sub-challenge 2) and $p = 999$ (sub-challenge 3). The genotyping and gene expression challenge data for the in silico RILs populations have been produced under these assumptions:

- a set of 15 gene networks with size $n = 1000$ were generated with *modular EIPO* topology, and the dynamical model was defined according to each network structure;
- for each of the networks we generated the genotypes of a population of p RILs, where each RIL is represented as a vector of binary genotype values (0/1), one for each of 1000 homozygous genes;
- for all networks, 20 chromosomes with 50 genes each were considered. The 0/1 values in the genotype vectors for each RIL were sampled with correlations between adjacent positions on the chromosomes (mimicking the *genetic linkage* phenomenon); no relationship between network positions of genes and their locations on chromosomes was assumed;
- each gene was assumed to have a single (functional) genetic variant, either in the gene's promoter region (leading to a *cis-effect* on its expression rate) with probability 0.25 or in the gene's coding region (leading to *trans-effects* on its targets) with probability 0.75;
- steady state gene-expression levels for all RILs were calculated after adjusting the Z parameters according to the corresponding genotype vector, and setting the values for ϑ_i^{syn} and ϑ_i^{deg};
- simulations were performed using the set of deterministic ordinary differential equations (3.1);
- finally, simulated experimental noise was added to the steady state values.

For each network we provided two files, i.e. the $n \times p$ matrices of gene expression and of genotype values. These data are still available for download at [15].

4.1.2 Predictions and scoring metrics

The DREAM5 Systems Genetics challenge is aimed at identifying the best approaches for gene network inference from systems genetics data for varying sample sizes, in particular considering the $p \ll n$ problem where the number of observations p (number of RILs) is less than the number of variables n (number of genes).

In order for this challenge to yield light on the performance of the algorithms under different data sizes, participants were strongly encouraged to submit predictions to the three sub-challenge. However, predictions to only one or two of the three sub-challenges were accepted. Participants were required to submit their predictions for each network, i.e. a ranked list of directed regulatory interactions $A \rightarrow B$ ordered according to the assigned confidence, from the most reliable (first row) to the least reliable (last row) prediction.

The results were scored using the area under the Precision versus Recall (PR) curve for the whole set of link predictions for a network. For the first k predictions (ranked by score, and for predictions with the same score, taken in the order they were submitted in the prediction files), precision is defined as the fraction of correct predictions to k, and recall is the proportion of correct predictions out of all the possible true connections. Also the area under the Receiver Operating Characteristic (ROC) curve was evaluated. The precise scoring system is described in [165]. Teams were ranked according to their overall performance over the five networks of each challenge.

4.2 StatSeq benchmark dataset

In this section, the in silico systems genetics dataset, used as a benchmark in [44], is described in detail, in particular regarding its simulation by SysGenSIM. Morever, the algorithms underlying the generation of the gene expression data and the genotype values are fully illustrated.

The presented benchmark dataset is meant to be used for training and evaluating algorithms and techniques for the inference of networks from systems genetics data. The goal is to find which methodologies exhibits the best overall network inference performance, and to analyze their performances under particular conditions (i.e. population size, large or small marker distances, high or low heritability, network size).

Section 4.2.1 describes how the dataset has been generated by means of SysGenSIM[1], a MATLAB toolbox for simulating systems genetics experiments in model organisms [136]. Detailed information is provided about the topology of the gene networks and the settings of the simulator used to produce the genotypes and the gene expression data. In Section 4.2.2, the algorithms employed to obtain the networks and genetic data are thoroughly explained.

4.2.1 Description of the systems genetics dataset

The Systems Genetics benchmark is a collection of 72 in silico datasets generated from nine artificial gene networks of different size. In the following, details on the in silico networks and on the configurations of SysGenSIM employed to produce the data are provided.

In silico networks

The systems genetics experiments have been simulated using nine different artificial gene networks. These networks have been generated using SysGenSIM with parameters chosen to produce the following topological properties:

- three networks for each of the sizes $n = \{100, 1000, 5000\}$, in the following referred to with the self-explanatory labels 100-{1,2,3}, 1000-{1,2,3}, and 5000-{1,2,3};
- exponential in-degree and power law out-degree (EIPO) distributions for the nodes;
- average node degree[2] $K \simeq 6$;
- size of the largest strongly connected component (subnetwork) equal to at least 20% of the network nodes ($n = 100$), 15% ($n = 1000$), and 10% ($n = 5000$).

Some other topological characteristics of the nine networks are summarized in Table 4.1, where for each of the networks the number of edges, the size of the largest strongly connected component (LSCC), the number of nodes in the in- and in the out-components[3],

[1] SysGenSIM 1.0.2, version released on May 8th, 2012. More information is available in the online manual at [16] or in Chapter 3.

[2] The average number of both ingoing and outgoing edges for a node: $K = K_{\text{in}} + K_{\text{out}}$.

[3] Respectively, all the nodes from which the LSCC is reachable and that are not reachable from the LSCC, and all the nodes reachable from the LSCC but from which the LSCC cannot be reached.

Table 4.1: **Topological characteristics of the in silico networks.**

Size	Network label	Edges	LSCC	In-	Out-	Tendrils	Tubes
100	100-1	285	21	25	18	30	6
100	100-2	304	26	7	63	4	0
100	100-3	296	34	9	57	0	0
1000	1000-1	3149	165	71	660	96	8
1000	1000-2	2881	162	58	648	108	23
1000	1000-3	3038	150	106	563	158	23
5000	5000-1	14678	528	224	3498	657	93
5000	5000-2	15672	526	233	3580	595	66
5000	5000-3	15270	598	231	3660	480	31

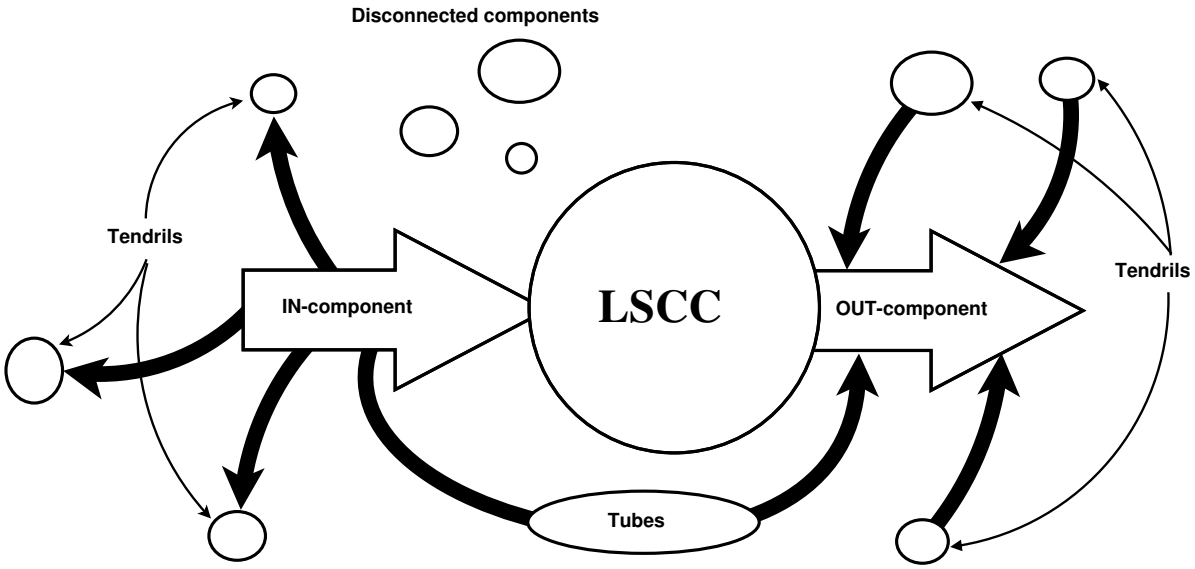

Figure 4.2: **Model representing the topology of the artificial networks.**

and the number of nodes amongst tendrils[4] and tubes[5] are shown. Network 1000-2 has one isolated node. Figure 4.2 shows a representation of such network topology.

Simulation of datasets

In order to provide a wide range of scenarios, datasets were simulated by combining the nine topologies with eight different parameter settings for a total of 72 datasets. The eight parameter settings resulted from a combination of two average marker distances $d = \{1\text{ cM}, 5\text{ cM}\}$, two median heritability values H (high $\simeq 0.8$ and low $\simeq 0.4$) and two population sizes $m = \{300, 900\}$. Simulations have been run with SysGenSIM's optional parameter settings set as described in Table 4.2. By keeping most of the parameters fixed, each dataset has been simulated according to the setting configurations summarized in Table 4.3, i.e. only the marker distance, the biological variance and the population size have been manipulated.

[4]Nodes from which the LSCC cannot be reached, and that cannot be reached from the LSCC itself.
[5]Nodes connecting the in- to the out-component, and not belonging to the LSCC.

Table 4.2: **Values of SysGenSIM's optional parameter settings used to generate the datasets.**

Settings	Selected value
Network topology	EIPO
Network size	{100, 1000, 5000}
Edge sign assignment	Node-wise
Edge sign probability	0.5
Average node degree	6
Marker positions	Generate
Gene positions	At markers
Mapping function	Haldane
RIL type	Selfing
Number of chromosomes	{5, 25, 25}
Markers per chromosome	N({20, 40, 200},2)
Marker distances	{N(1, 0.2), N(5, 1)}
Cis-effect %	25
Genotyping error %	5
Z lower	0.5
Z upper	0.8
Basal transcription rate	Constant
B.t.r. parameters	[1,-]
Interaction strength	Constant
I.s. parameters	[1,-]
Hill cooperativity coefficient	Gamma
H.c.c. parameters	[1,1.67]
Basal degradation rate	Constant
B.d.r. parameters	[1,-]
Transcription biological variance	Gaussian
T.b.v. parameters	[1, {0.1, 0.25}]
Degradation biological variance	Gaussian
D.b.v. parameters	[1, {0.1, 0.25}]
Expression measurement noise	Gaussian
E.m.n. parameters	[1, 0.1]
Number of phenotype nodes	0
Population size	{300, 900}
Number of experiments	1

Table 4.3: **SysGenSIM's settings applied to each network.**

Configuration	Marker distance	Biological variance	Heritability	Population size
1	N(5,1)	N(1,0.1)	$\simeq 0.8$	300
2	N(5,1)	N(1,0.1)	$\simeq 0.8$	900
3	N(5,1)	N(1,0.25)	$\simeq 0.4$	300
4	N(5,1)	N(1,0.25)	$\simeq 0.4$	900
5	N(1,0.1)	N(1,0.1)	$\simeq 0.8$	300
6	N(1,0.1)	N(1,0.1)	$\simeq 0.8$	900
7	N(1,0.1)	N(1,0.25)	$\simeq 0.4$	300
8	N(1,0.1)	N(1,0.25)	$\simeq 0.4$	900

For each of the 72 datasets, the following four components were made available, the first two for data analysis (network inference) and the other two for algorithm evaluation purposes:

Gene expression matrix A $n \times m$ matrix containing gene expression measurements. Entry (i, j) is the simulated steady state expression value of gene i in individual j.

Genotype matrix A $n \times m$ matrix of genotype values $\{0, 1\}$. Entry (i, j) is the genotype value of gene i in individual j.

Heritability The median value of the heritability.

Edge list A signed list of edges encoding the direct interactions between the nodes of the network.

4.2.2 Algorithms in SysGenSIM

This section is dedicated to the detailed description of the most relevant algorithms underlying the whole simulation process: the procedure to build the EIPO in silico networks, the simulation of the genotypes, and the equation modeling the evolution of the gene activity.

Generation of EIPO networks

The algorithm generates a network with exponential in-degree and power law out-degree node distributions[6], requiring the user to only specify the size n and the desired average degree K_d. Subsequently, the inverse scale parameter of the exponential distribution[7] is set to $\lambda = 1/K_d$, while the exponent γ in the power law distribution[8] is found after a quick iterative search. The algorithm works as follows:

1. Set $\lambda = 1/K_d$ defining the exponential distribution.

[6] In-degree and out-degree refer to the number of ingoing and outgoing edges of a node in a graph, respectively.

[7] The probability density function of an exponential distribution is $f(x;\lambda) = \lambda e^{-\lambda x}$ for $x \geq 0$.

[8] The power law distribution is described by the probability density function $f(x;\gamma) = x^{-\gamma}$, for $x \geq 0$.

2. Find γ and hence the power law distribution according to the desired average degree K_d.

3. Calculate the discrete probabilities $\mathbf{P}_{\text{in}}$ and $\mathbf{P}_{\text{out}}$ from the two distributions, respectively, for each possible degree K, i.e. from $K = 0$ to $K = n-1$ (a node can maximally be linked to all the remaining $n-1$ nodes).

4. Initialize the adjacency matrix $\mathbf{A}$ to zero.

5. Sample the in- and out-degree arrays $\mathbf{K}_{\text{in}}$ and $\mathbf{K}_{\text{out}}$ from the respective distributions, i.e. according to the probabilities $\mathbf{P}_{\text{in}}$ and $\mathbf{P}_{\text{out}}$. The i-th entries of the arrays $\mathbf{K}_{\text{in}}$ and $\mathbf{K}_{\text{out}}$ represent, respectively, the number of ingoing and outgoing edges for node i. The sampling is performed until the following conditions are met:

   - $K^i_{\text{in}} + K^i_{\text{out}} > 0, \forall i$, i.e. all nodes are connected to the rest of the network through at least one (ingoing or outgoing) edge.
   - $\sum_i K^i_{\text{in}} \geq \sum_i K^i_{\text{out}} \simeq n \cdot K_d$, to assure[9] that all the outgoing edges from $\mathbf{K}_{\text{out}}$ can reach a node in $\mathbf{K}_{\text{in}}$.

6. Sort the nodes with $K^i_{\text{out}} > 0$ (i.e. with at least one outgoing edge) by descending out-degree in list $\mathcal{N}_{\text{out}}$.

7. Then, for each node i in the ordered list $\mathcal{N}_{\text{out}}$:

   a) Place the nodes j with positive in-degree K^j_{in} in set $\mathcal{N}_{\text{in}}$.

   b) Remove, if included, node i from $\mathcal{N}_{\text{in}}$.

   c) If $|\mathcal{N}_{\text{in}}| < K^i_{\text{out}}$ then go back to step 5, else connect node i with K^i_{out} randomly selected nodes from $\mathcal{N}_{\text{in}}$.

   d) Decrease by 1 the in-degree of the nodes that have been just connected to node i.

   e) Update the adjacency matrix $\mathbf{A}$ with the new edges.

After the procedure has been performed for all nodes in $\mathcal{N}_{\text{out}}$, the network is complete and exhibits exponential in-degree and power law out-degree distributions with average degree $K \simeq K_d$.

Simulation of the genotypes

SysGenSIM simulates genotype data according to a user-defined or to a randomly generated genetic map based on the number of chromosomes in the genome and the number of genetic markers per chromosome with constant or normally distributed pair-wise distance among DNA variant locations. Map distances are converted to recombination rates for the generation of genotypes at ordered linked loci through the Haldane [73] or Kosambi [93] mapping functions. Markers can be either placed in perfect linkage with each functional polymorphism, or a marker map can be generated and then the functional variants randomly placed throughout the genome.

[9] The condition is requested in the continuation of the algorithm.

The genotype data for the benchmark datasets have been generated by the following algorithm:

1. Sample the number of markers for the n_{cr} chromosomes according to the size of the network n and the selected distribution of markers per chromosome.
2. Then, for each chromosome h in the genome:
   a) Generate the marker distances d by sampling the values according to the selected distribution ($N(1, 0.1)$ or $N(5, 1)$).
   b) Map the functional polymorphisms at markers.
   c) Convert[10] the distances d to recombination rates r and compute the probability[11] p_k of no recombination between any adjacent markers k and $k+1$.
   d) Generate the genotype vector for the entire chromosome $\mathbf{X}^h$ as:
      i. Randomly set X_1 to 0 or 1 with equal probability.
      ii. Sample u from a standard uniform distribution, set $X_k = X_{k-1}$ if $u < p_k$ and otherwise set $X_k = 1 - X_{k-1}$.
3. Combine all $\mathbf{X}^h$ into one single genotype vector $\mathbf{X}$.
4. The allelic effects, see Equation (4.1), of the *cis* (c) and *trans* (t) variants are generated as follows:
   a) For variant i (variant is synonymous with gene here as each gene is only allowed to have a single functional variant in *cis* or *trans*), set $Z_i = 1$. Sample u from a standard uniform distribution and if $u > 0.5$, sample Z_i from the uniform distribution $[Z^l, Z^u]$.
   b) Randomly select $n \cdot p_{\text{cis}}$ genes i to have a *cis* variant and set $Z_i^c = Z_i$.
   c) For the remaining genes j having a *trans* variant, set $Z_j^t = Z_j$.
5. A pre-specified number (proportion) of genotyping errors are added by changing randomly selected entries of the genotype vector $\mathbf{X}$ from 0 to 1 or vice versa.

The procedure is repeated for all the individuals m, while the conversion vector $\mathbf{B}$ is kept constant for the whole population.

Simulation of the gene expression data

For all the individuals, SysGenSIM computes the solution of a system composed of n differential equations, one for each gene i:

$$\frac{\mathrm{d}G_i}{\mathrm{d}t} = V_i Z_i^c \vartheta_i^{\text{syn}} \prod_{j \in \mathscr{R}_i} \left[1 + A_{ji} \frac{G_j^{h_{ji}}}{G_j^{h_{ji}} + \left(K_{ji} / Z_j^t \right)^{h_{ji}}} \right] - \lambda_i \vartheta_i^{\text{deg}} G_i \tag{4.1}$$

[10] According to Haldane: $r = 0.5(1 - e^{-0.02d})$.

[11] For recombinant inbred lines generated by selfing inbred line cross: $p = 1/(1+2r)$.

where $\mathscr{R}_i$ is a set containing the indexes of all regulators (both activators and inhibitors) j of gene i; G_i is the gene expression of gene i, V_i is its basal transcription rate and λ_i its degradation rate constant. K_{ji} is the interaction strength of G_j on G_i, h_{ji} is the Hill cooperativity coefficient, and A_{ji} is an element of the adjacency matrix **A** encoding the signed network structure. Finally, the parameters ϑ_i^{syn} and ϑ_i^{deg} represent the biological variances in the synthesis and degradation processes of gene i, while Z_i^c (cis-effect) and Z_i^t (trans-effect) incorporate the effects of DNA polymorphisms in the model.

Except for the case of a constant, the distributions from which the model parameters p are sampled are defined by two parameters a and b. The possible distributions are listed below:

Constant $p = a$, where a is a real number.

Uniform distribution $p = a + (b-a)\varrho_u$, where a and b are the lower and upper limits of the uniform distribution $[a,b]$, and ϱ_u is a random number sampled from the standard $[0,1]$ uniform distribution.

Normal distribution $p = a + b\varrho_n$, where a is the mean and b the standard deviation, and ϱ_n is a random value drawn from the standard normal $N(0,1)$ distribution. In the very unlikely (by choice of parameters a and b) case of $p < 0$, then $p = 0$ is forced to avoid negative parameters.

Gamma distribution To guarantee a positive value for the parameters, a gamma distribution is the best choice. Parameters are randomly sampled from a Gamma distribution with density function:

$$\text{Gamma}(a,b) = \frac{1}{b^a \Gamma(a)} x^{a-1} e^{-x/b} \tag{4.2}$$

where $\Gamma(\cdot)$ is the gamma function, and a and b are the shape and scale parameters, respectively. The exponential distribution is a special case of the gamma distribution with $a = 1$.

4.3 Pula-Magdeburg single-gene knockout benchmark dataset

We used SysGenSIM [136] to generate 30 realistic genome-scale networks and to simulate single-gene knockout experiments under different noise conditions. This benchmark has been used for the verification of network inference techniques [132].

The whole dataset consists of:

- 30 networks, composed each by 5000 genes, of which:
  - 10 networks have about 7500 edges (average degree $K = 1.5$);
  - 10 networks have about 10000 edges (average degree $K = 2$);
  - 10 networks have about 12500 edges (average degree $K = 2.5$).

- For each network, 5000 single-gene knockout experiments have been simulated under nine different noise conditions by the possible combinations of:
  - biological synthesis and degradation variances[12] sampled from distributions $N(1,0.025)$, $N(1,0.050)$, $N(1,0.100)$;
  - experimental noise[13] sampled from $N(1,0.025)$, $N(1,0.050)$, $N(1,0.100)$.

Therefore a grand total of 270 different networks (30 topologies with nine different noise configurations) with simulations of single-knockout experiments have been produced to be used as realistic and genome-scale benchmark datasets for testing inference methodologies under different conditions of edge density, biological variance, and multiplicative measurement noise. All datasets can be freely downloaded [15]; for each network the following files are available:

- a list of unsigned edges encoding the directed interactions in the 5000-gene network;
- the wild-type gene expression values $\mathbf{G}^{\text{wt}}$ (one file for each of the nine noise configurations);
- the matrix of expression values $\mathbf{G}^{\text{ko}}$ computed after the single-gene knockout of all genes in the network (one file for each of the nine noise configurations).

Some details about the production of the datasets are given in Sections 4.3.1 and 4.3.2.

4.3.1 Generation of networks

The gene networks have been generated in order to have a topology similar to those found in certain organisms, i.e. a modular structure with node degree distributions of exponential and power law behavior for, respectively, the number of ingoing and outgoing edges[14]. In particular, each network is constituted by:

- 10 modules of 100 genes,
- 8 modules of 250 genes,
- 2 modules of 500 genes,
- 1 module of 1000 genes.

The produced networks are represented as directed and signed graphs by a signed adjacency matrix $\mathbf{A}$. An edge (i,j) symbolizes an activating relationship between gene i and gene j when $A(i,j)=1$, or an inhibiting relationship when $A(i,j)=-1$. If $A(i,j)=0$, no direct influence exists from gene i to gene j. Each node of the networks has been randomly selected to have all its outgoing edges either positive or negative.

[12] Respectively parameters ϑ^{syn} and ϑ^{deg} in Equation (4.3).

[13] Parameter ν in Equation 4.3.

[14] SysGenSIM users can generate similar networks by running the software with the `EIPO modular` option selected.

4.3.2 Model dynamics

The following equation explains the synthesis and degradation processes that regulate the gene activity. The solution of the system composed by n first order nonlinear ordinary differential equations, one equation for each gene of the network, is a set of n steady state gene expressions:

$$\frac{\mathrm{d}G_i}{\mathrm{d}t} = V_i Z_i \vartheta_i^{\mathrm{syn}} \prod_{j \in \mathscr{R}_i} \left[1 + A_{ji} \frac{G_j^{h_{ji}}}{G_j^{h_{ji}} + K_{ji}^{h_{ji}}} \right] - \lambda_i \vartheta_i^{\mathrm{deg}} G_i \tag{4.3}$$

where $\mathscr{R}_i$ is a set containing the indexes of all regulators (both activators and inhibitors) j of gene i; G_i is the mRNA concentration (gene activity or gene expression) of gene i, V_i is its basal transcription rate, while λ_i is its degradation rate constant. K_{ji} is the interaction strength of G_j on G_i, h_{ji} is the Hill cooperativity coefficient, and A_{ji} is an element of the matrix $\mathbf{A}$ encoding the signed network structure. Finally, parameters $\vartheta_i^{\mathrm{syn}}$ and $\vartheta_i^{\mathrm{deg}}$ represent the biological variances in the synthesis and degradation processes of gene i, while Z_i sets to zero the transcription rate in case of knockout of gene i. Each steady-state gene expression value is then multiplied by experimental noise ν_i.

In particular:

- parameters V_i, λ_i and K_{ji} are constant for each gene or edge, and are set equal to 1;
- parameters h_{ji} are sampled from a gamma distribution with shape parameter $a = 1$ and scale parameter $b = 1.67$; values are increased by 1 in order to have them larger than 1;
- parameters Z_i are always equal to 1, except in case of knockout of gene i when $Z_i = 0$;
- biological variances $\vartheta_i^{\mathrm{syn}}$, $\vartheta_i^{\mathrm{deg}}$ and the experimental noise ν_i are sampled from Gaussian distributions with mean $\mu = 1$ and standard deviation $\sigma = \{0.025, 0.050, 0.100\}$; any very unlikely negative value is set to zero.

Chapter 5

Identification of gene regulatory networks

Gene regulatory networks have been defined in Chapter 2, and an overview of techniques and methodologies for the structural inference of such networks is presented in the following paragraphs in order to introduce and to position in their own appropriate *scope of application* the algorithms we developed to reconstruct the topology of gene regulatory networks, thoroughly explained in Chapters 6, 7 and 8, and respectively dedicated to the inference of networks from (i) single-gene knockout experiments; (ii) heterogeneous datasets; (iii) systems genetics observations.

Overview of network inference techniques

Reverse engineering is an interesting area of research currently receiving a lot of attentions from the Systems Biology community. In fact, reconstructed biomolecular networks may allow researchers to understand the molecular basis of complex traits and diseases [148], as well as the discovery of direct drug targets [50]. The data-driven inference of intracellular regulatory networks, in particular of those involved in gene regulation, remains to be one key challenge of computational and systems biology. Many methods for this daunting task have been proposed and new methods are appearing at a high rate [116, 65, 23, 155, 47, 111].

The different inference methodologies can be categorized based on the model formalism and the principle used for deriving interactions in a regulatory network: sparse regression [168], correlation-based approaches [45, 142, 152], z-score [138], ANOVA-based analysis [94], mutual information [36, 178, 115], Bayesian networks [62, 61], Gaussian graphical models [37], random forest [82], differential equations [125, 49, 46, 64], reaction networks [54] and Boolean networks [19, 146]. One final output of all these approaches is the reconstructed network topology, typically given as a (signed or unsigned, directed or undirected) graph. Recent efforts have shown that combining several of the aforementioned methods often outperform all single approaches [111].

Depending on the available measurements, different inference techniques can be em-

ployed. In case of experiments without targeted perturbations (*observational studies*, such as gene expression data collected over a group of similar individuals, typically done in the context of a disease) the expression profiles can be analyzed to build a undirected graph whose nodes are the genes, and whose edges represent the presence of significant associations. Without targeted perturbations it is not generally possible to infer directions of the edges. A wide variety of techniques for constructing such undirected co-expression networks has been proposed, typically based on marginal associations, conditional associations or information theory. Under some assumptions it is theoretically possible to decide the orientation of the edges using this type of data [129, 162], but unfortunately these assumptions (such as acyclicity of the network and absence of confounding factors) are very unlikely to be met in the present context. On the other hand, targeted perturbations (e.g., systematic single-gene gene knockouts, over-expressions) are needed to enable causal inference, and the reconstruction of the directed structure of gene networks. Many techniques for constructing gene networks have been proposed of which the most popular techniques are based on ordinary differential equations or Bayesian networks. A wide range of network inference methods have been developed to address this challenge, from those exclusive to gene-expression data [47, 112] to methods that integrate multiple classes of data [24, 141, 98, 113].

Verification of network inference algorithms

These approaches have been successfully used to address many biological problems [62, 115, 50, 57], yet when applied to the same data, they can generate disparate sets of predicted interactions [47, 112]. The rigorous evaluation and comparison of the large number of inference methods before one can put confidence in the results of their application is ascertained to be of utmost importance [164, 163], and the need for the verification of computational systems biology methods is now recognized worldwide. Understanding the advantages and limitations of different network inference methods is critical for their effective application in a given biological context. Notably, the Dialogue on Reverse Engineering Assessment and Methods (DREAM) project organizes international gene regulatory network inference challenges and evaluates the solutions submitted by participating research groups in a transparent manner [17, 112, 111]. This way a *collaborative-competition* is established in which complicated problems are addressed as a community rather than individual laboratories, as also shown by IMPROVER [120, 122]. Recently, it was demonstrated that such community efforts were fruitful for the inference of an improved gene regulatory network of *Escherichia coli* and the inference of a novel gene regulatory network for the bacterium *Staphylococcus aureus* [111].

The performance of these techniques can then be evaluated and compared by applying different inference methods to the data obtained from biomolecular networks of which the structure is assumed to be known *a priori*, i.e. *gold standard* networks [23, 159, 57]. However, real world biomolecular networks are mostly unknown. Even the most studied biomolecular networks are not only plagued by false positives, but suffer even worse from false negatives: they are largely incomplete [167]. Consequently such networks cannot be deemed as totally reliable benchmarks to compare inference algorithms. Therefore, it has been suggested to use data simulated with dynamical models of gene networks, i.e. in silico data. In

this case the underlying networks are precisely known and thus allow for thorough evaluation and comparison of reverse-engineering algorithms [119, 138]. Obviously, the relevance of evaluations on in silico data strongly depends on the realism of the simulation system, e.g. the network topology, the type of mathematical model, the type of kinetic functions, the noise model, etc. Verification of inference methods requires benchmark datasets [120]. Benchmarking on real biological data is challenging as true biological networks are largely unknown [167]. The availability of realistically simulated datasets is therefore of utmost importance for the verification of these methods. Only for simulated data can we be certain about the true complex system underlying the data. Simulated data has been used to validate methods, but typically the data was generated with small networks (containing 10-100 genes) [46, 90] and with the same models as used by the inference [171, 158]. The step to more realistic benchmark data was made in [119], generating simulated gene expression data using equations based on enzyme kinetics (for use of these data in method evaluations, see e.g. [45, 159]). As regulatory network inference methods are typically applied to genome-wide data, a necessary next step is to perform evaluations also on genome-scale, as we did in [132] (see Section 6.2).

Chapter 6

Inference from single-gene knockout datasets

Two of our most relevant works in the field of gene network inference from single-gene knockout datasets are presented in this chapter: the algorithm for the down-ranking of feed-forward loops [134] for which we have been awarded the 1st place in the DREAM4 In Silico Network challenge, and its evolution in a more accurate technique [132] developed through a profitable collaboration with the "Analysis and Redesign of Biological Networks" group of the Max Planck Institute in Magdeburg.

6.1 From knockouts to networks: establishing direct cause-effect relationships through graph analysis

Reverse-engineering gene networks from expression profiles is a difficult problem for which a multitude of techniques have been developed over the last decade. The yearly organized DREAM challenges allow for a fair evaluation and unbiased comparison of these methods. Here we propose an inference algorithm that combines confidence matrices, computed as the standard scores from single-gene knockout data, with the down-ranking of feed-forward edges. Substantial improvements on the predictions are obtained after the execution of this second step. In particular, our algorithm was awarded the best overall performance at the DREAM4 In Silico 100-gene Network sub-challenge, proving to be effective in inferring medium-size gene regulatory networks. This success demonstrates once again the decisive importance of gene expression data obtained after systematic gene perturbations and highlights the usefulness of graph analysis to increase the reliability of inference.

The outline of the paper is the following: we first describe the DREAM4 In Silico Network challenges, then explain the inference algorithm we developed and applied to the DREAM 4 data, followed by a description of the gene network simulator we developed to generate additional synthetic networks and data. Then, we show the results of evaluations of variants of our algorithm on both the DREAM3 in silico benchmarks and the additional simulated

datasets. Then, we show results of re-analysis of the DREAM4 in silico benchmarks, which we were able to perform after the gold standard networks were released. We conclude with a discussion of the method, data and future steps to be made.

6.1.1 DREAM4 In Silico Network challenge

The Dialogue for Reverse Engineering Assessments and Methods (DREAM) is an international initiative with the aim of evaluating methods for biomolecular network inference in an unbiased way [164, 163]. Evaluations proceed through organized competitions on a yearly basis in which teams from all over the world participate. For the 4th edition of DREAM in 2009, the organizers proposed three different challenges. Our team participated in the second one, the In Silico Network challenge, which asked to infer gene networks from simulated data. The challenge was, in turn, divided into three sub-challenges, respectively named `InSilico_Size10`, `InSilico_Size100`, and `InSilico_Size100_Multifactorial`.

These sub-challenges differ, as their names suggest, in the network size and the type of data provided. In the first sub-challenge the participants had to predict the topology of five 10-gene networks, and were provided with steady state gene expression levels from wild-type, knockouts, knockdowns, multifactorial perturbations, and time series data. The second sub-challenge concerns instead five 100-gene networks, with the same type of available data except the multifactorial perturbations. The third sub-challenge involves five other 100-gene networks provided with multifactorial perturbations data only. The contestants were challenged to predict the network structures underlying the above data, i.e. assigning a level of confidence for the presence of each possible edge.

We here provide a brief description of the available data provided to the DREAM4 participants. The number of genes in the network is denoted by n. The *wild-type* file contains the n steady-state levels of the unperturbed network. The *knockout* data (see an example in Table 6.1) consist of n rows with n steady-state values, each obtained after deleting one of the n genes. The *knockdown* data are similar to the above, but are obtained by halving the transcription rate constant of one gene at a time instead of setting it to zero. The *multifactorial perturbations* data consist of steady-state levels of small fluctuations of the values of all transcription rate constants simultaneously. The *time series* file contains trajectories of gene activity levels starting from the wild-type steady state to a perturbed state, and from the perturbed state back to the wild-type state upon removing the perturbations.

The network topologies to be inferred were generated by the organizers by extracting 10- or 100-node subnetworks from transcriptional regulatory networks of *E. coli* and *S. cerevisiae*, with preferential selection of parts containing cycles (but no self-interactions).

The challenge description mentioned also that the data was simulated through a dynamical model describing both independent and synergistic gene regulation, which included both gene and protein expression (but only the gene expression data was provided to the partecipants). Internal noise was modeled through stochastic (Langevin) differential equations, and measurement noise was added to the simulated gene expression levels. Networks and data were generated by the GeneNetWeaver 2.0 software [114], which was published only after the DREAM4 conclusion.

Table 6.1: **Sample of gene expression knockout data.** This is an example of the provided knockout data, related to an example 5-gene network. The first row contains the wild-type (unperturbed) gene activities, while the others contain the gene activities due to the knockout of the gene indicated on the left. A knocked-out gene has null expression. Data are affected by noise, but certain relationships are apparent: G_1 is likely to be regulated by (or at least downstream of) G_2, since the steady state value of G_1 responds strongly to perturbing G_2; in fact, $G_1^2 = 0.68$ noticeably differs from $G_1^{\text{wt}} = 0.14$.

	G_1	G_2	G_3	G_4	G_5
G^{wt}	0.14	0.89	0.01	0.87	0.14
G^1	0.00	0.96	0.00	0.86	0.06
G^2	0.68	0.00	0.04	0.90	0.05
G^3	0.17	0.86	0.00	0.88	0.02
G^4	0.13	0.86	0.08	0.00	0.09
G^5	0.12	0.78	0.09	0.91	0.00

6.1.2 Methods

Algorithm

The aim of these challenges is the prediction of the (directed and unsigned) network structures. How can we infer such gene regulatory networks? While the time-series data could be used for this purpose, the lack of protein measurements will make it difficult to infer relationships between gene activities from time dynamics: the protein dynamics will cause delays between the gene expression dynamics. Therefore, we resorted to the steady state levels, in particular to the knockout datasets, where the perturbations and the relative responses are stronger.

From this kind of data it is very easy to infer a so-called causal influence network: genes whose steady state values change as a result of a single-gene knockout are likely to be downstream of the perturbed gene [174, 154]. Most causal relationships (both activating and inhibiting) due to the knocked-out gene could be immediately recognized from the data table (e.g. Table 6.1), unless the influence is particularly weak and then overwhelmed by noise, or its effect is mitigated by other connections. This approach will not infer spurious relationships between co-regulated genes, which is instead a well-known problem of algorithms based on expression similarity (e.g. correlation) [45].

However, some of the edges of a causal influence network may be indirect, i.e. mediated by other (measured) gene activities [174]. The remaining task is thus to distinguish direct from indirect relationships. To accomplish this, we developed an algorithm consisting of two main steps: through statistical measures, a first estimate of the confidence of each possible edge is obtained directly from the available knockout data; then, by down-ranking the feed-forward edges, a refined prediction is given.

In the first step we quantify the importance of the responses of the gene activities toward single-gene perturbations and so how likely it is for each gene to be downstream of the perturbed genes. Let G^{wt} be the vector of wild-type gene expression, and let G^i be the vector of gene activity steady-states obtained by knocking out gene i. To obtain the initial predictions, we evaluated four possible different confidence matrices $\mathbf{W}$ in which elements $W(i,j)$

reflects the confidence in the existence of the edge $i \to j$:

Deviation matrix, $\mathbf{W^D}$ The confidence of edge (i, j) is simply estimated by the absolute value of the deviation from wild type of the expression of gene j after the knockout of gene i: $\mathbf{W^D}_{i,j} = |G^i_j - G^{\text{wt}}_j|$. The larger the deviation the higher the confidence we have that G_j is downstream of the perturbed G_i.

Normalized deviation matrix, $\mathbf{W^{ND}}$ As the absolute values of the steady state gene activities vary drastically (e.g. $G^{\text{wt}}_2 = 0.89$ and $G^{\text{wt}}_3 = 0.01$ in Table 6.1) it might be more appropriate to consider the relative deviations. Each column of the deviation matrix is normalized by the corresponding wild type: $\mathbf{W^{ND}}_{i,j} = \mathbf{W^D}_{i,j} / G^{\text{wt}}_j$.

Z-score on deviation matrix, $\mathbf{W^{ZD}}$ A more statistically motivated measure is the z-score; it indicates how many standard deviations σ an observation is far from the mean μ of a whole set of measurements. In this case, for each gene j we calculate μ_j and σ_j using the deviations from wild type after each knockout ($\mathbf{W^D}_{\cdot,j}$):

$$\mathbf{W^{ZD}}_{i,j} = \frac{\mathbf{W^D}_{i,j} - \mu_j}{\sigma_j} \tag{6.1}$$

Z-score on raw data matrix, $\mathbf{W^{ZR}}$ As both G^i_j and G^{wt}_j are noisy values, it may be better to consider raw expression values rather than deviations from the steady state values (subtracting a noisy value from another noisy value results in a even noisier value). Therefore, for each gene j we calculate μ_j and σ_j using the steady-state values after each knockout ($G^{\cdot}_j$):

$$\mathbf{W^{ZR}}_{i,j} = \frac{G^i_j - \mu_j}{\sigma_j} \tag{6.2}$$

Once a first prediction of the network has been calculated with one of the above methods, the second step of the inference algorithm comes into play. The logic behind this second step is also plain and simple. First, based on a threshold value on the derived confidence matrix, a network is obtained. This network contains edges which represent causal influences between the genes, which may be direct or indirect. The *true* network is thus embedded in this initial causal influence network and could be derived by removing edges (edges can not be added as they create causal influences not supported by the perturbation experiments). We recognize that certain edges can be removed without removing the causal influences: the edge from gene A to gene C could be removed if there is at least one additional path from gene A to C in the network [174]. The additional path(s) could explain the causal effect of gene A on C and therefore we have reduced confidence in the existence of the direct edge from A to C. Figure 6.1 provides an example of a feed-forward loop from which an edge could be removed. Our down-ranking algorithm systematically checks for paths through the initial networks and recognizes which edges can be removed (potentially indirect) and which edges can not be removed (these must be direct as removing them would result in a network missing one or more of the observed causal influences). Note that cyclic components in the networks are fully connected, as each gene in a cycle has a causal influence on all other genes in the cycle. Determining which edges in a cyclic component can be removed without removing causal paths depends on the order in which the edges are removed. Therefore, we

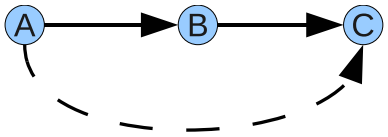

Figure 6.1: **Feed-forward loop in a 3-gene motif.** The edge between gene A and gene C might be erroneously predicted as the causal effect of gene A on gene C, which could in principle be explained by the indirect path through gene B.

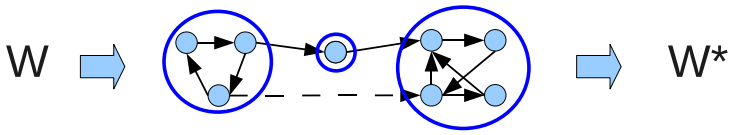

Figure 6.2: **Down-ranking of unnecessary feed-forward edges.** The thick rings highlight the strongly connected components of $\mathcal{N}$. The dashed edge is *removed* from the network.

decided not to touch any of the edges in cyclic components. We emphasize that we do not believe that the sparsest network possible is most biologically realistic. In fact, it is widely recognized that biomolecular networks are enriched in feed-forward loops [123]. However the absolute frequencies of their occurrence in the networks is much lower than that of the linear path motif ($A \to B \to C$). Therefore, it is reasonable to assume that down-ranking these edges improves the reliability of the network inference.

The second step of our algorithm proceeds in the following way (Figure 6.2):

1. Use a threshold value t for the edge confidence (selected after several test simulations, as explained in the Results section) to extract a directed network $\mathcal{N}$ from one of the above mentioned matrices **W**.

2. Calculate the condensation of network $\mathcal{N}$, i.e. the acyclic network formed by contracting each strongly connected component of $\mathcal{N}$ into a single vertex.

3. Obtain the subnetwork $\mathcal{N}'$ from $\mathcal{N}$ by deleting any edge such that:

   - its endpoints belong to two different strongly connected components C_i and C_j, and
   - there is a path of length at least 2 between C_i and C_j in the condensation of $\mathcal{N}$.

4. For all the remaining edges in network $\mathcal{N}'$, increase their corresponding weight by $\max_{(i,j)\in\mathcal{N}\setminus\mathcal{N}'} W_{i,j}$, in order to ensure them a ranking higher than all the unessential edges, i.e. the edges in $\mathcal{N} \setminus \mathcal{N}'$.

In silico data simulation

To be able to thoroughly evaluate and fine-tune the parameters of our algorithm we generated in silico data using our simulator developed in MATLAB.

In our model, the following nonlinear ordinary differential equation describes the evolution of the gene expression G_j:

$$\frac{\mathrm{d}G_j}{\mathrm{d}t} = Z_j V_j \prod_{i=1}^{n} \left(1 + A_{i,j} \frac{G_i^{h_{i,j}}}{G_i^{h_{i,j}} + K_{i,j}^{h_{i,j}}} \right) - \lambda_j \vartheta_j G_j \tag{6.3}$$

G_j is the gene activity (gene expression level, mRNA concentration) of gene j, V_j is its basal transcription rate, while λ_j is its degradation rate constant. $K_{i,j}$ is the interaction strength of G_i on G_j, $h_{i,j}$ is the Hill cooperativity coefficient, and $A_{i,j}$ is an element of the matrix $\mathbf{A}$ encoding the signed network structure (a positive sign corresponds to an activating regulation, while a negative one to an inhibition). Finally, ϑ_j represents the biological variance (sampled from a normal distribution with $\mu_\vartheta = 1$ and standard deviation $\nu_\vartheta = 0.1$), while Z_j is responsible for eventually knocking-out gene j. In our simulations, random networks were generated by the Erdős-Rényi algorithm [55], with various average degrees. Edge directions and signs were assigned randomly with uniform probability. Parameters Z_j, V_j, $K_{i,j}$, $h_{i,j}$, λ_j and $G_j(0)$ were all set equal to 1. We then calculated the wild-type steady state. To simulate the single-gene knockout experiments we initialize $G_j(0) = G_j^{\mathrm{wt}}$ and set $Z_j = 0$ in the j-th perturbed experiment in order to simulate the knockout of gene j; obviously $Z_{k\neq j} = 1$ since we only simulated single-gene knockout experiments. These simulations resulted in datasets similar to the ones provided in the DREAM4 challenges.

Evaluation

Method effectiveness are evaluated through the calculations of the Area Under the Receiver Operating Characteristic Curve (AUC(ROC)) and the Area Under the Precision versus Recall Curve (AUC(PvsR)) in the same way as is done by the DREAM organizers to evaluate the submitted networks [159, 138].

6.1.3 Results

In order to make informed decisions on the choice of the weight matrices to use and to fine-tune the threshold value for the second step of our algorithm, we practiced first on the DREAM3 benchmarks [114] and then on the additional datasets generated using our own network structures and dynamical model.
Then we show a re-analysis of the DREAM4 benchmarks, which were made available by the organizers after the competition.

Practice on the DREAM3 benchmarks

The DREAM3 In Silico Network challenge in 2008 was very similar to the DREAM4 one. Here too gene networks of different sizes (10, 50, and 100 genes) had to be inferred using steady states from wild-type, knockdown and knockout perturbations, and time series data. The kinetic equations were also similar, though in DREAM3 a deterministic model was used while in DREAM4 a stochastic one.

In order to choose which, amongst the confidence matrices $\mathbf{W^D}$, $\mathbf{W^{ND}}$, $\mathbf{W^{ZD}}$ and $\mathbf{W^{ZR}}$, gives the most reliable initial network prediction, tests were performed on the DREAM3

Table 6.2: **Performances of the four considered confidence matrices on the DREAM3 networks.** Average AUC(ROC) and AUC(PvsR) for the five networks of three different sizes from the DREAM3 In Silico benchmarks, calculated through the confidence matrices $\mathbf{W^D}$, $\mathbf{W^{ND}}$, $\mathbf{W^{ZD}}$ and $\mathbf{W^{ZR}}$. The best value of each row is highlighted.

	N. genes	$\mathbf{W^D}$	$\mathbf{W^{ND}}$	$\mathbf{W^{ZD}}$	$\mathbf{W^{ZR}}$
AUC(ROC)	10	**0.8194**	0.7741	0.7837	0.7901
	50	0.8444	0.8389	0.8769	**0.8875**
	100	0.8515	0.8454	0.8736	**0.8799**
AUC(PvsR)	10	**0.7028**	0.5619	0.5991	0.6732
	50	0.5396	0.4579	**0.6224**	0.6160
	100	0.5637	0.4616	**0.6200**	0.6143

Table 6.3: **Effect of the down-ranking algorithm on larger DREAM3 networks.** Average AUCs for the 50- and 100-gene networks from the DREAM3 In Silico challenge after the application of the down-ranking algorithm on matrices $\mathbf{W^{ZD}}$ and $\mathbf{W^{ZR}}$ with 8 different thresholds t. Setting $t = 0$ corresponds to not applying the down-ranking. The best value of each row is highlighted.

	n	$\mathbf{W}$	$t=0$	$t=1$	$t=1.5$	$t=2$	$t=2.5$	$t=3$	$t=3.5$	$t=4$
AUC(ROC)	50	$\mathbf{W^{ZD}}$	0.8769	0.8769	0.8766	0.8767	**0.8773**	0.8773	0.8772	0.8770
		$\mathbf{W^{ZR}}$	0.8875	0.8853	**0.8885**	0.8884	0.8881	0.8878	0.8877	0.8875
	100	$\mathbf{W^{ZD}}$	0.8736	0.8736	0.8735	0.8733	0.8735	**0.8739**	0.8738	0.8737
		$\mathbf{W^{ZR}}$	0.8799	0.8799	0.8793	**0.8804**	0.8804	0.8802	0.8801	0.8800
AUC(PvsR)	50	$\mathbf{W^{ZD}}$	0.6224	0.6224	0.6176	0.6175	0.6411	**0.6412**	0.6377	0.6303
		$\mathbf{W^{ZR}}$	0.6160	0.5835	**0.6669**	0.6666	0.6555	0.6461	0.6352	0.6259
	100	$\mathbf{W^{ZD}}$	0.6200	0.6200	0.6181	0.6111	0.6222	**0.6511**	0.6456	0.6387
		$\mathbf{W^{ZR}}$	0.6143	0.6143	0.6039	**0.6622**	0.6603	0.6502	0.6410	0.6326

benchmarks. We initially considered both the knockout and knockdown data, but since our algorithm consistently gave better results on the knockouts (data not shown), we will here further consider only the knockout steady states.

By applying the aforementioned inferring techniques on these data, the matrices $\mathbf{W^{ZD}}$ and $\mathbf{W^{ZR}}$ yielded the best results for the 50- and 100-gene networks, respectively for the AUC(PvsR) and for the AUC(ROC). On the other hand, a simple ordering of the edges based on the deviation from the wild type (i.e. matrix $\mathbf{W^D}$) gave the best results for the small 10-gene networks for both the evaluation measures AUC(ROC) and AUC(PvsR). The results are shown in Table 6.2. Then, given the confidence matrix $\mathbf{W}$, the down-ranking algorithm produces the modified matrix $\mathbf{W}^*$ as described in the Methods section. The result of this down-ranking step depends on the chosen value for the threshold t. Therefore, we performed test runs at different values of t to establish the value for which the best AUCs were obtained (Table 6.3). We here report only the results on the larger networks as the down-ranking step had almost no effect on the reliability of the small networks. This indicates that our down-ranking approach is beneficial only for larger networks. Negligible differences in the AUC(ROC), but more substantial improvements in the AUC(PvsR) measures were obtained for $\mathbf{W^{ZD}}$ and $\mathbf{W^{ZR}}$, with the latter slightly exceeding the former performances. In particular, the AUCs peak for $t = 2$ while down-ranking $\mathbf{W^{ZR}}$, and for $t = 3$ in $\mathbf{W^{ZD}}$ (100-gene networks).

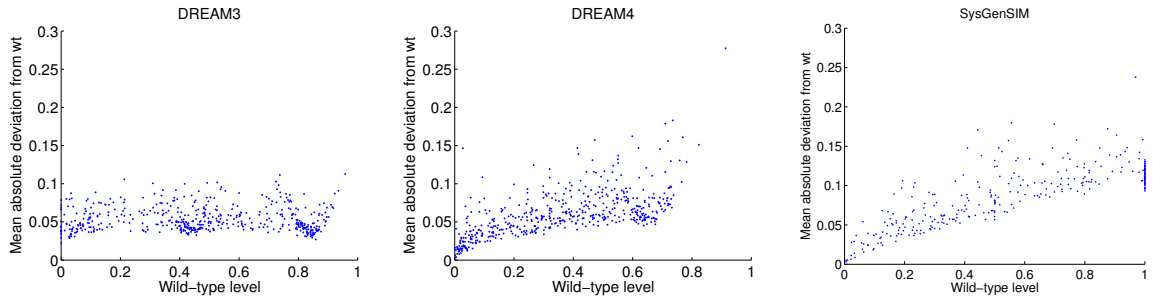

Figure 6.3: **Distribution of the mean absolute deviation for three knockout datasets.** Each point is the mean absolute deviation of the expression of a gene j with respect to its wild type G_j^{wt}, calculated as $\frac{1}{n-1}\sum_{i\neq j}|G_j^i - G_j^{\text{wt}}|$, obviously excluding the knockout of gene j from the averaged values. Our in silico knockout data (right) qualitatively resembles the distribution of the five DREAM4 `InSilico_Size100` knockout datasets (middle), in contrast to those from the five DREAM3 `InSilico_Size100` knockout datasets (left).

These tests suggested that using either matrix $\mathbf{W^{ZD}}$ or $\mathbf{W^{ZR}}$ in combination with $t = 2$ are the best choice. However, while the DREAM3 benchmarks are of great value, there were some notable differences between the DREAM3 and DREAM4 networks and data. All the networks in DREAM3 were acyclic, while the networks considered in DREAM4 do contain cycles.

Furthermore, the variance in the DREAM3 knockout data drastically differed from those in the DREAM4 knockout data. In the previous edition the mean deviation in each gene was uniform, while in the DREAM4 data it seemed proportional to the gene activity wild-type level (Figure 6.3). The same pattern can be observed in our self generated in silico data (Figure 6.3). So, by using our simulator, we can verify the previous choices for the confidence matrix and the threshold value on a much larger number of datasets than the DREAM3 benchmark (thus preventing overtraining), and on data which should be more similar to the DREAM4 ones.

Practice on additional in silico data

We performed further testing on our own simulated in silico knockout data. We generated 1000 100-gene networks with Erdős-Rényi topology with average degrees $\bar{k} \in \{2,3,5\}$[1]. The AUCs for the various confidence matrices are shown in Table 6.4, emphasizing that the z-score applied on the raw data ($\mathbf{W^{ZR}}$) clearly appears to be the most effective method to obtain a first prediction of the network from knockout data. This choice is also supported by the test on the DREAM3 benchmarks. In a similar fashion, we applied the down-ranking algorithm on matrix $\mathbf{W^{ZR}}$, showing that a small improvement on the AUCs (especially with the PvsR one) can be obtained with threshold $t = 2$ (Table 6.5), again in concordance with what we observed for the DREAM3 benchmarks.

DREAM4

After the extensive tests described above, we decided to base our predictions for the DREAM4 In Silico Network challenge on the z-score on raw data confidence matrix ($\mathbf{W^{ZR}}$), post-processed

[1] DREAM3 100-gene networks have the average degree $\bar{k}$ ranging from 1.2 to 5.5.

Table 6.4: **Performance of the four confidence matrices on additional in silico data.** Average AUCs for 1000 100-gene Erdős-Rényi networks with average degree $\bar{k} \in \{2,3,5\}$ calculated through the confidence matrices $\mathbf{W^D}$, $\mathbf{W^{ND}}$, $\mathbf{W^{ZD}}$ and $\mathbf{W^{ZR}}$. The best value of each row is highlighted.

	$\bar{k}$	$\mathbf{W^D}$	$\mathbf{W^{ND}}$	$\mathbf{W^{ZD}}$	$\mathbf{W^{ZR}}$
AUC(ROC)	2	0.8763	0.8829	0.9276	**0.9328**
	3	0.8223	0.8449	0.8910	**0.8972**
	5	0.7325	0.7751	0.8155	**0.8209**
AUC(PvsR)	2	0.3055	0.3839	0.5909	**0.6041**
	3	0.2602	0.3809	0.5383	**0.5519**
	5	0.2119	0.3513	0.4500	**0.4588**

Table 6.5: **Effect of the down-ranking algorithm on additional in silico data.** Average AUCs for 1000 100-node Erdős-Rényi networks, generated with average degree $\bar{k} \in \{2,3,5\}$, after the application of the down-ranking algorithm on matrix $\mathbf{W^{ZR}}$ with 6 different thresholds t. The best value of each row is highlighted.

	$\mathbf{W^{ZR}}$ ($t=0$)	$t=1.5$	$t=1.75$	$t=2$	$t=2.25$	$t=2.5$
AUC(ROC)	0.8317	0.8315	0.8317	**0.8317**	0.8317	0.8317
AUC(PvsR)	0.5913	0.5793	0.5892	**0.5962**	0.5954	0.5948

Table 6.6: **Performances of the four confidence matrices on the DREAM4 networks.** Average AUC(ROC) and AUC(PvsR) for the five 100-gene networks from the DREAM4 In Silico benchmarks, calculated through the confidence matrices $\mathbf{W^D}$, $\mathbf{W^{ND}}$, $\mathbf{W^{ZD}}$ and $\mathbf{W^{ZR}}$. The best value of each row is highlighted.

	$\mathbf{W^D}$	$\mathbf{W^{ND}}$	$\mathbf{W^{ZD}}$	$\mathbf{W^{ZR}}$
AUC(ROC)	0.7844	0.7927	0.8275	**0.8297**
AUC(PvsR)	0.2610	0.2786	**0.3710**	0.3602

with the down-ranking algorithm using threshold $t = 2$. Our submission as Team ALF was the best performer at the sub-challenge 2 (100-gene networks), ranking first among 19 participants. Interestingly, now that the gold standard networks have been published, we discovered that our choice for the confidence matrix was in fact good (see Table 6.6), but even better predictions would have been obtained by selecting $t = 2.5$ as the threshold for the down-ranking algorithm. Nevertheless, the improvement in the AUC(PvsR) obtained with the selected $t = 2$ has been considerable for networks 1 and 5, as shown in Figure 6.4 and in Table 6.7, compared to those from $\mathbf{W^{ZR}}$. It should also be noticed that the the average node degrees in the DREAM4 networks are smaller ($1.8 \leq \bar{k} \leq 2.5$) than those in DREAM3 and our simulated networks: a better estimation of the optimal threshold might have been obtained if our test networks had an average degree in the same range of the DREAM4 networks. Furthermore, we simulated data with networks generated with the Erdős-Rényi algorithm, which have significantly different topology than those used in DREAM4. Also, note that the performances on the DREAM4 benchmarks are much more sensitive to the value of t then we observed in the tests of our in silico data. Obviously this is due to the fact that we

Table 6.7: **Effect of the down-ranking algorithm on the DREAM4 100-gene networks.** Average AUC(PvsR) values for the 100-gene networks from DREAM4 In Silico challenge after the application of the down-ranking algorithm on matrix $\mathbf{W}^{\mathbf{ZR}}$ with 8 different thresholds t. The best value of each row is highlighted.

	$\mathbf{W}^{\mathbf{ZR}}$ ($t=0$)	$t=1$	$t=1.5$	$t=2$	$t=2.5$	$t=3$	$t=3.5$	$t=4$
Network 1	0.4928	0.4928	0.4847	0.5361	**0.6590**	0.6428	0.6225	0.5715
Network 2	0.3880	0.3880	0.3880	0.3771	**0.4144**	0.4125	0.4052	0.3886
Network 3	0.3816	0.3816	0.3834	0.3898	**0.4115**	0.4048	0.3939	0.3895
Network 4	0.3684	0.3684	0.3684	0.3494	**0.4433**	0.4338	0.4144	0.3841
Network 5	0.1703	0.1703	0.1697	**0.2133**	0.2008	0.1902	0.1782	0.1845

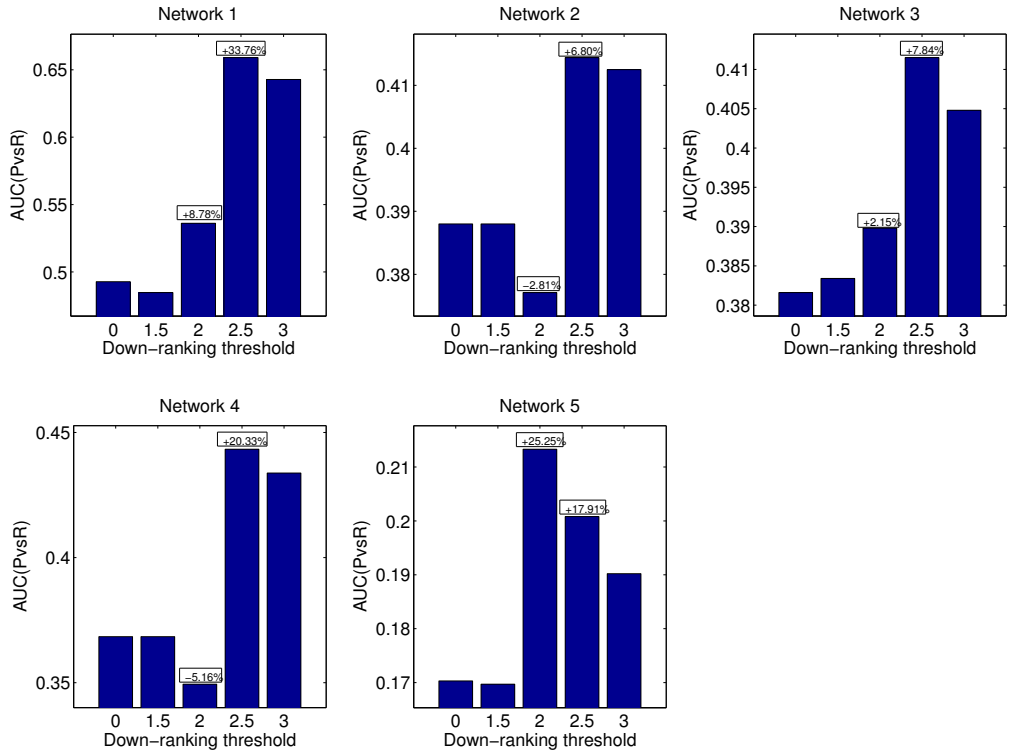

Figure 6.4: **Effect of the down-ranking algorithm on DREAM4 networks.** In each of the five plots, the bars show the values of the AUC(PvsR) for one of 100-gene networks from DREAM4 after the application of the down-ranking algorithm on matrix $\mathbf{W}^{\mathbf{ZR}}$ with 5 different threshold t. In the small boxes the most significant differences with respect to the threshold $t=0$ are shown.

used a large ensemble (1000 networks) over which the performances were averaged, but it also indicates that the DREAM4 benchmarks consist of a set of networks with widely varying topologies.

6.1.4 Discussion

We described an algorithm to infer gene regulatory networks from expression data, that proved to be effective by best performing at the DREAM4 In Silico Network challenge in the 100-gene networks sub-challenge. The proposed technique combines the advantages of the standard score in highlighting the deviation from the mean after a gene knockout, with the down-ranking algorithm that reduces the confidence initially predicted to unnecessary feed-forward edges.

Our algorithm is substantially different from the techniques used by the best performer teams of previous DREAM In Silico Network challenges. In particular, for DREAM2 the winning approach was fitting ordinary differential equation (ODE) models [71, 154]; for DREAM3, instead, the best method was based mainly on finding significant deviations from wild type in knockout data (so using the same primary source of information of our algorithm), but also applied ODE models on the time series for additional predictions [183].

To see how methods based on ODEs would perform on the DREAM4 data, we analyzed them with one of the best performer algorithm [154] for the DREAM2 In Silico Network challenges. The predictions of this algorithm on the DREAM4 100-gene networks was very poor (average AUC(ROC) = 0.5722, AUC(PvsR) = 0.0313). Note that in DREAM2 there was no noise added to the in silico data, while for DREAM4 both biological and experimental noise were present. Since the internal noise is propagated through gene relationships, its effect on large networks make sophisticated models (like ODEs) much less reliable than our method based on simple cause-effect logic and graph inspection.

Further improvements of the performance of our algorithm may be obtained by studying the possible relationships between the selected threshold t and other parameters, like the network average degree and size, the noise on the knockout data, and so on. Moreover, other subnetwork structures (e.g., fan-in, fan-out, and cascade motifs [112][2]) are poorly predicted even by the best algorithms, and so a lot of improvements can be done on these issues. Finally, also the rest of the available data from the DREAM challenges (knockdowns, time series, multifactorial) may be used to refine the network prediction, but the gain would probably be small, as already shown by the DREAM3 best performer [183].

It has become unambiguously clear that systematic perturbations (e.g. experimental gene knockouts) are needed to establish the directed structure of gene networks. However, systematic single-gene knockouts imply experimental requirements which are unrealistic and these experiments infeasible (and unethical) for many organisms. It is unlikely that data such as considered here will become available from real experiments. Fortunately, *systems genetics* experiments may provide an alternative. In systems genetics experiments a population under study is genotyped and gene expression profiled are simultaneously collected

[2] In [112], the motif we refer to as *feed-forward loop* is called *cascade*.

(possibly even including metabolomics and proteomics data [63]). It has been demonstrated that causal links in gene networks can be elucidated based on these data (see [103, 143] for reviews). Genetic polymorphisms, naturally present in the populations, act as genetic perturbations: if the gene activity of a gene B is affected by a polymorphism inside another gene A, this is highly indicative for a causal effect $A \rightarrow B$. In fact Liu *et al.* [103] proposed a very similar strategy as the one outlined in this paper: first creating a causal influence network (but based on systems genetics data instead of knockout data like is done here) and subsequent sparsification of this network to retain only the edges corresponding to direct causal influences. In that approach each edge in the initial network was statistically tested for being supported by the data, while we were here not able to do so based on the data considered here. Down-ranking edges based on our simple graphical inspection is very useful in the context of systems genetics data as it will provide the sparsest network supporting the causal influences. This then allows methods like the one of Liu *et al.* approach to statistically identify the networks best supported by the data by adding edges, rather than removing edges from the causal influence network. Heuristic model search algorithms are strongly dependent on a good initial guess in the network space: we argue that networks which result from the algorithm described in this paper will provide a better initial guess than the initial causal influence network, as gene networks are known to be sparse. In this sense, the resulting networks from our approach here should not be seen as the final prediction, but rather as inputs to more sophisticated methods involving thorough statistical testing. Nevertheless, as evidenced by its winning performance over 18 other participating teams in the DREAM4, this method can be considered state of the art on its own.

6.2 Reconstruction of large-scale regulatory networks based on perturbation graphs and transitive reduction: improved methods and their evaluation

The data-driven inference of intracellular networks is one key challenge of computational and systems biology. As suggested by recent works, a simple yet effective approach for reconstructing regulatory networks comprises the following two steps. First, the observed effects induced by directed perturbations are collected in a signed and directed perturbation graph (PG). In a second step, Transitive Reduction (TR) is used to identify and eliminate those edges in the PG that can be explained by paths and are therefore likely to reflect indirect effects.

In this work [132] we introduce novel variants for PG generation and TR leading to significantly improved performances. The key modifications concern: (i) use of novel statistical criteria for deriving a high-quality PG from experimental data; (ii) the application of local TR which allows only short paths to explain (and remove) a given edge; and (iii) a novel strategy to rank the edges with respect to their confidence. To compare the new methods with existing ones we not only apply them to a recent DREAM network inference challenge but also to a novel and unprecedented synthetic compendium consisting of 30 5000-gene networks simulated with varying biological and measurement error variances resulting in a

total of 270 datasets. The benchmarks clearly demonstrate the superior reconstruction performance of the novel PG and TR variants compared to existing approaches. Moreover, the benchmark enabled us to draw some general conclusions. For example, it turns out that local TR restricted to paths of only length 2 is often sufficient or even favorable. We also demonstrate that edge weights are highly beneficial for TR whereas edge signs are of minor importance. We explain these observations from a graph-theoretical perspective and discuss the consequences with respect to a greatly reduced computational demand to conduct TR. As a realistic application scenario we use our framework for inferring gene interactions in yeast based on a library of gene expression data measured in mutants with single knockouts of transcription factors. The reconstructed network shows a significant enrichment of known interactions, especially within the 100 most confident (and for experimental validation most relevant) edges.

This paper presents two major achievements. First, the novel methods introduced herein can be seen as state of the art for inference techniques relying on perturbation graphs and transitive reduction. The second main result of the study is the generation of a new and unprecedented large-scale in silico benchmark dataset accounting for different noise levels and providing a solid basis for unbiased testing of network inference methodologies.

6.2.1 Introduction

In this paper we revisit two related gene network inference methods: down-ranking of feed-forward loops (DR-FFL [134]) and TRANSitive reduction for WEighted Signed Digraphs (TRANSWESD [91]). Both approaches were successfully employed (ranked 1st and 3rd, respectively) in the DREAM4 In Silico 100-node Network challenge. In this challenge, the task was to reverse engineer gene networks from (simulated) steady-state and time-series data. DR-FFL and TRANSWESD share a common core as they both try to infer a minimal regulatory graph that can explain the gene expression changes observed in perturbation experiments. In particular, both methods apply the principle of *transitive reduction* to identify and eliminate edges reflecting indirect effects. Since both DR-FFL and TRANSWESD were ranked high, their underlying inference strategy could provide a generally promising approach for gene network inference.

Network reconstruction methods based on transitive reduction usually involve three steps of which the last can be seen as optional:

Step 1 (Generation of a perturbation graph): A *perturbation graph* $\mathcal{G}^P$ is generated from the perturbation data, i.e., a directed edge from a node i to a node j ($i \to j$) is included in $\mathcal{G}^P$ if a perturbation in i changed the level of j significantly (significance to be measured by a certain criterion). Sometimes, the edges are also labeled by a sign and might also get a weight indicating their confidence or likelihood.

Step 2 (Transitive reduction): As an edge in the perturbation graph may reflect a direct but also an indirect effect between two nodes, the goal of the second step – the transitive reduction – is to identify and eliminate indirect effects in $\mathcal{G}^P$ yielding the final reconstructed graph $\mathcal{G}^T$. As a general rule for transitive reduction, an edge introduced due to indirect effects is detected by searching for alternative paths in $\mathcal{G}^P$ which could induce the same net

effect as this edge. We say that such a path *explains* the edge and the latter is then removed.

Step 3 (Edge sorting): Normally one would consider all edges contained in $\mathcal{G}^T$ as the true edges. In an optional third step, all edges of the reconstructed graph $\mathcal{G}^T$ are ranked in a list according to a given confidence score for each edge. For certain applications it might be useful to augment this list also by edges (together with their confidence values) not contained in $\mathcal{G}^T$ (i.e., edges which were not contained in $\mathcal{G}^P$ or which were removed from the latter when computing the transitive reduction $\mathcal{G}^T$). In this way we get an ordered list of all potential pairwise interactions according to their confidence score.

These three steps are common to all approaches using transitive reduction (abbreviated by TR in the following) but different variants may arise (i) by using different approaches to derive the perturbation graph (abbreviated PG) in Step 1 or (ii) by considering different criteria a path must fulfill in order to explain a given edge in Step 2, or (iii) by different edge sorting schemes to be used in Step 3. For example, DR-FFL [134] uses a z-score-based strategy to generate the PG and does not consider edge signs in the TR step when searching for valid paths that can explain certain edges. In contrast, TRANSWESD [91] generates the PG by selecting edges that satisfy two distinct statistical conditions whereas the actual TR procedure accounts for edge signs and also edge weights when searching for suitable paths that can explain a given edge.

In the present study, we propose and test novel variants for each of the three steps mentioned above, i.e., for PG generation, for TR, and for edge sorting. As one major outcome, we present particular combinations of PG generation and TR strategies which yielded superior results in diverse benchmark tests outperforming by far the two original approaches. As benchmarks we used not only the DREAM4 In Silico Network challenge but also a novel and unprecedented synthetic compendium consisting of several realistic 5000-gene networks simulated with varying biological and measurement error variances resulting in a total of 270 datasets. In both benchmarks we focus on perturbations induced by single gene knockouts. Such experiments can be realistically carried out at genome-scale at least in some model organisms (see, for example, [181, 52, 81, 140]). As a realistic application scenario we use our framework for inferring gene interactions in yeast *Saccharomyces cerevisiae* based on a library of gene expression data measured in mutants with single knockouts of transcription factors [140]. The reconstructed network shows a significant enrichment of known interactions, especially within the (most relevant) edges identified with highest confidence.

The results of the benchmarks do not only demonstrate the relative performance of the different approaches but also enable us to draw some general conclusions. For example, it turns out that when pruning the PG by TR, it is often sufficient or sometimes even favorable to restrict the search on paths with a length of only 2. We also demonstrate that edge weights are highly beneficial for TR whereas edge signs are of minor importance (a finding which was also recently reported in [30]). We give an explanation for these observations from a graph-theoretical perspective.

6.2.2 Methods

We start with a brief description of the original TR methods DR-FFL and TRANSWESD which inspired the novel inference algorithms presented herein. Afterwards we introduce the new variants for PG generation, TR, and edge sorting. For the PG generation algorithms, we assume that we are given the following input variables (for a network of n genes):

- a $1 \times n$ row vector $\mathbf{G}^{\text{wt}}$ containing the (possibly preprocessed) wild-type gene expression data
- the $n \times n$ matrix $\mathbf{G}^{ko}$ containing the (possibly preprocessed) measured steady-state gene expression levels after perturbing/knocking-out each single gene. The element $G^{ko}(i,j)$ stores the gene expression level of gene j after perturbing gene i.

These input variables directly correspond to the datasets provided in the DREAM4 challenge and in our novel compendium of simulated large-scale networks (described below).

Down-ranking of feed-forward loops (DR-FFL)

The DR-FFL algorithm described in [134] used the following strategies for the three steps:

Step 1 (PG generation): In a preprocessing step, a confidence weight is assigned to each possible edge $i \to j$ of the network by computing the absolute value of the standard z-score z_{ij}. The latter quantifies the difference between the expression $G^{ko}(i,j)$ of gene j under knockout/perturbation of gene i and its mean μ_j, normalized by the standard deviation σ_j:

$$z_{ij} = \frac{G^{ko}(i,j) - \mu_j}{\sigma_j}. \tag{6.4}$$

Mean μ_j and standard deviation σ_j are computed on all available expression measurements of gene j, including the wild-type $G^{\text{wt}}(j)$. Then, the PG $\mathscr{G}^P$ is obtained by selecting all those edges whose $|z_{ij}|$ is larger than a given threshold β. We then denote the PG generated by the original DR-FFL method by PG^1.

Step 2 (TR): DR-FFL circumvents possible problems arising in TR of cyclic graphs by allowing only those edges to be removed that connect nodes from different strongly connected components (a strongly connected component in a directed graph is a maximal subgraph in which for each ordered pair of nodes a path exists connecting these nodes). DR-FFL uses unsigned and unweighted TR, i.e., an edge $i \to j$ is removed from $\mathscr{G}^P$ if i and j are from different components and if there is an alternative path connecting i and j without using edge $i \to j$.

Step 3 (Edge sorting): The confidence weights $|z_{ij}|$ of the remaining edges in the graph $\mathscr{G}^T$ obtained after TR are increased by a constant offset such that all edges in $\mathscr{G}^T$ are ranked higher than all other potential edges (not contained in $\mathscr{G}^T$). The latter are listed below the edges of $\mathscr{G}^T$ according to their confidence weight computed in Step 1.

TRANSWESD

TRANSWESD (TRANSitive reduction for WEighted Signed Digraphs) was introduced in [91] with the goal to generalize and improve previous TR approaches [169, 174, 86] to make it amenable for the reconstruction of large biological networks.

Step 1 (PG generation): TRANSWESD constructs the PG $\mathscr{G}^P$ via two thresholds: an edge $i \to j$ is introduced in $\mathscr{G}^P$ if (i) a measure similar to the z-score $|z_{ij}|$ used by DR-FFL exceeds a given threshold β and (ii) if the absolute change of the state of node j when perturbing i exceeds a certain minimal deviation γ, i.e. if $\mid G_j^{\text{wt}} - G_{ij}^{ko} \mid > \gamma$. Each edge $i \to j$ gets a sign $s_{ij} = \text{sign}\left(G_j^{\text{wt}} - G_{ij}^{ko}\right)$ indicating whether the changes in i and j have the same direction (positive sign) or not (negative sign). In addition, a weight w_{ij} is assigned to each edge $i \to j$ quantifying its *uncertainty* or behavioral distance (i.e., a large weight indicates a low confidence of this edge). Accordingly, TRANSWESD uses $w_{ij} = 1 - |c_{ij}|$ with c_{ij} being the conditional correlation coefficient between genes i and j which is computed from all experiments except those where gene i was directly perturbed. More specifically, herein the conditional correlation coefficient c_{ij} is defined as the Pearson correlation coefficient computed from all measurements of nodes i and j (columns in $\mathbf{G}^{\text{wt}}$ and $\mathbf{G}^{ko}$) except in the experiments where j was knocked-out. The PG generated by the original TRANSWESD procedure is denoted by PG^2.

Step 2 (TR): A particular feature of TRANSWESD is that it can deal with signed and weighted PGs and that cycles are allowed. The TR rule is as follows: An edge $i \to j$ with sign s_{ij} and weight w_{ij} is removed if there is an alternative path P_{ij} ($i \Longrightarrow j$) which connects i and j and fulfills the following requirements: (i) P_{ij} is simple, i.e., it does not contain a cycle; (ii) P_{ij} does not involve edge $i \to j$; (iii) the overall sign of P_{ij} (obtained by multiplying the signs of all its edges) is the same as s_{ij}; and (iv) the maximum weight of all edges on path P_{ij} (denoted by $w_{\max}(P_{ij})$) fulfills

$$w_{\max}(P_{ij}) < \alpha \cdot w_{ij}. \tag{6.5}$$

The confidence factor α is typically chosen close (but smaller) than unity; the default value used by Klamt et al. [91] is 0.95. With $\alpha < 1$ it is ensured that all edges in the path P_{ij} have a higher confidence than the edge $i \to j$. However, in some cases it can nevertheless be advantageous to use also $\alpha > 1$. If a path P_{ij} with the four required properties exists in the PG, then the observed effect of i upon j is considered to be explained (induced) by path P_{ij}. All edges $i \to j$ in the PG fulfilling these conditions are considered to be (potentially) removable and are collected in a set $\mathscr{R}$. If the graph is acyclic, TR is simple and unique and all potentially removable edges in $\mathscr{R}$ can be deleted immediately. The situation is more complicated in cyclic graphs: the result of TR can become non-unique, depending on the order of edge removals. TRANSWESD uses a reasonable rule to resolve non-uniqueness: it removes the edges of $\mathscr{R}$ iteratively starting with the highest weight (lowest association) first. As a second problem in cyclic graphs, it may then happen that a formerly removable edge in $\mathscr{R}$ becomes non-removable because certain paths may have been interrupted by preceding deletions of other removable edges. Even worse, an edge might still potentially be removable but its elimination would lead to the interruption of a path that was required to explain an edge already removed in a previous iteration (see the example below). It is therefore necessary to explicitly test, in each iteration, whether upon removal of the next edge of $\mathscr{R}$ all edges originally

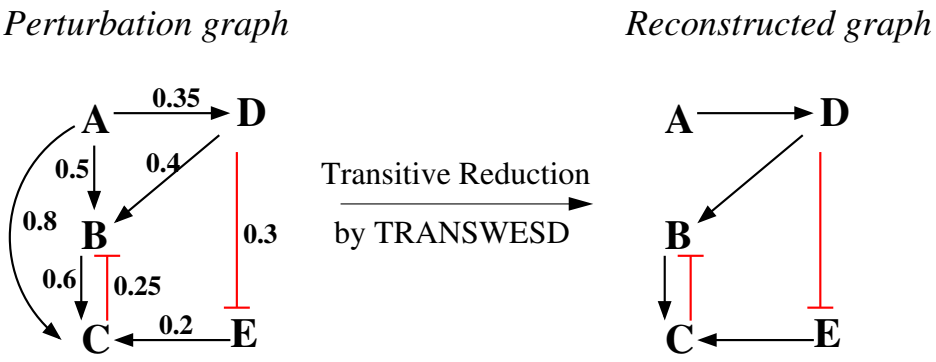

Figure 6.5: **Example of a perturbation graph and its transitive reduction computed with TRANSWESD.** A given signed and weighted perturbation graph (left) and its transitive reduction as computed by TRANSWESD (right).

contained in the PG $\mathscr{G}^P$ are still explainable by the remaining graph (otherwise this edge has to be reinserted). This may require extensive shortest path calculations.
Therefore, to reduce the computational effort in large-scale cyclic graphs, TRANSWESD provides two parameters (`path_exact` and `full_check`) to allow for the (optional) use of approximate solutions which may drastically reduce the required computation time. Since computing the shortest path of a given sign in cyclic signed digraphs is an NP-complete (and thus delicate) problem, starting TRANSWESD with `path_exact` = 0 enforces the use of approximate path calculation algorithms which have been shown to produce no or only few errors in large-scale biological networks [91, 92]. The `full_check` = 0 option can be used to suppress recomputation of shortest path lengths after deleting an edge (thus assuming that the relevant path lengths will not change). To our experience from numerous tests, there are usually only minor effects on the reconstruction quality when using this simplification. As we deal herein only with large-scale networks, we used `path_exact` = `full_check` = 0 in all calculations.
We illustrate the approach with the example shown in Figure 6.5. The graph on the left-hand side displays a hypothetical cyclic PG with its edge weights and signs. Using the standard confidence factor of $\alpha = 0.95$, in principle, three edges could be identified as indirect effects as for each of them a suitable explaining path would exist. This concerns the edge $A \to C$ (explained by path $A \to B \to C$ and alternatively also by path $A \to D \to B \to C$ both fulfilling the sign and weight conditions), the edge $A \to B$ (explained by path $A \to D \to B$) and the edge $D \to B$ (explainable by the path $D \dashv E \to C \dashv B$). These three edges form the set R of potentially removable edges. According to the rules, TRANSWESD removes first edge $A \to C$ as it has the largest weight (lowest confidence). In the second iteration, $A \to B$ can be safely removed. Now the algorithm has to stop even though the edge $D \to B$ is still explainable by the path given above. If we removed this edge, no positive path from A to B and from A to C would remain in the graph, i.e., the originally observed influence of A on B and C would not be captured anymore. This example shows that TRANSWESD may keep an edge in the graph, even if there is an explaining path for it. The resulting graph $\mathscr{G}^T$ for this example is shown on the right-hand side of Figure 6.5. (Note: with `full_check` = 0 the edge $D \to B$ would be (wrongly) removed additionally whereas `path_exact` = 0 had no effect).

Step 3 (Edge sorting): TRANSWESD ranks the edges according to their weights computed in Step 1: edges with highest confidence (lowest weights) are placed first. Edges retained in $\mathscr{G}^T$ are put first followed by edges that were contained in the PG $\mathscr{G}^P$ (but removed during TR). The last group comprises all other pairwise interactions; their order is also determined by the conditional correlation coefficient c_{ij}.

Novel variants

DR-FFL and TRANSWESD were successfully applied and highly ranked in the DREAM4 network reconstruction challenge. However, when we compared and mixed both methods (e.g., by replacing Step 1 of TRANSWESD with Step 1 of DR-FFL) we realized that even better approaches, in particular for Step 1 (PG generation) and Step 2 (TR), might exist. In the following we describe several new variants focusing on those which in the benchmarks performed significantly better than the original DR-FFL and TRANSWESD versions (see Results section).

Perturbation graph The novel PG generation procedure delivers:

- The signed and directed PG $\mathscr{G}^P$ itself.
- A matrix $\mathbf{W}^t$ containing the weights of the edges in $\mathscr{G}^P$ to be used by the transitive reduction algorithm. The element $W^t(i,j)$ contains the weight of the edge $i \to j$ in $\mathscr{G}^P$; it is set to ∞ if the edge was not included in the PG.
- A matrix $\mathbf{W}^r$ containing the confidence weights for all possible interactions (i,j) to be used in the edge ranking procedure in Step 3. In contrast to $\mathbf{W}^t$, this matrix contains a weight for all pairs (i,j) (except for $i=j$ as we exclude self-loops), even if $i \to j$ is not contained in $\mathscr{G}^P$.

A key difference of the novel PG algorithms compared to the strategies used by DR-FFL and TRANSWESD is that different edge weights are used for TR and for edge sorting. Moreover, the selection of candidate edges and the calculation of edge weights are based on (combinations of) correlation and z-score measures. In detail, the following calculations are performed:

1. Compute the $n \times n$ conditional correlation matrix $\mathbf{C}$ from the expression measurements $\mathbf{G}^{\text{wt}}$ and $\mathbf{G}^{ko}$.
2. Use $\mathbf{G}^{ko}$ to compute the $n \times n$ z-score matrix $\mathbf{Z}$ comprising the z-score values of all (potential) edges.
3. Compute the $n \times n$ matrix $\mathbf{Z}^c$ as the z-score calculated on the absolute value of the entries of the conditional correlation matrix $\mathbf{C}$, and add a (minimal) offset to obtain positive values: $\mathbf{Z}^c > 0$.
4. Build the PG by defining the following set of edges:
   - $\mathscr{S}^1$ comprises all node pairs (i,j) for which $|Z(i,j)| > \beta$.
   - $\mathscr{S}^2$ comprises all node pairs (i,j) for which $|C(i,j)| > \gamma$.
   - $\mathscr{S}^3$ is the set of all node pairs (i,j) whose z-score and correlation values have opposite sign: $C(i,j) \cdot Z(i,j) < 0$.

- $\mathscr{B} = \mathscr{S}^1 \cap \mathscr{S}^2 \cap \mathscr{S}^3$ is the set of node pairs (i, j) satisfying the three previous conditions.
- $\mathscr{Z}^p$ is the set of node pairs (i, j) with positive z-score value.
- $\mathscr{Z}^n$ is the set of node pairs (i, j) with negative z-score value.
- $\mathscr{E}^p = \mathscr{Z}^n \cap \mathscr{B}$ is the set of positive edges of the PG.
- $\mathscr{E}^n = \mathscr{Z}^p \cap \mathscr{B}$ is the set of negative edges of the PG.
- $\mathscr{G}^p = \mathscr{E}^p \cup \mathscr{E}^n = \mathscr{B}$ yields the PG.

5. Compute the ranking weight matrix by normalizing $\mathbf{W}^r = |\mathbf{Z}| + \mathbf{Z}^c$ between 0 and 1.
6. Compute the weight matrix $\mathbf{W}^t$ to be used for transitive reduction in TRANSWESD as $\mathbf{W}^t = \mathbf{1} - \mathbf{Z}^c$.

Using this scheme, the PG is built by selecting all edges where (i) the z-score exceeds a given threshold β, (ii) the conditional correlation exceeds another threshold γ, and (iii) the signs of z-score and conditional correlation are opposite. The latter condition is justified because a positive z-score for the edge (i, j) is computed when the deletion/decrease of i (due to knockout or knockdown) yields an increase in the activity of j which should corresponds to a negative correlation between i and j. The same correspondence exists between negative z-score and positive correlation. Obviously, measurement noise may invalidate the truth of these statements, thus we only keep edges that are consistent with respect to this sign rule. With the rule described above, the positive edges contained in $\mathscr{E}^p$ stem from a negative z-score and the negative edges contained in $\mathscr{E}^n$ from a positive z-score. In the following we denote the PG generated by the above procedure PG^{new}.

The weights $W^r(i, j)$ used for edge ranking take equally-weighted into account (i) the absolute value of the standard z-score (Equation (6.4)) of the deviations induced by the perturbation in i and (ii) the z-score of the deviations of the conditional correlation between i and j relative to the averaged conditional correlations related to gene j. As far as we know, a z-score of conditional correlations has not yet been used in the context of network inference, however, the ranking weights introduced above proved to be optimal in the benchmarks delivering an edge sorting of high quality. Below we show that the new PG generation approach in combination with the proposed ranking scheme may already deliver a valuable approximation of the network itself but can often be further improved by TR techniques. Regarding the weights to be used for TR ($\mathbf{W}^t$), benchmark tests showed us that it is beneficial to use only the z-score of the conditional correlation coefficients.

Transitive reduction Identifying and pruning edges representing the indirect interactions in $\mathscr{G}^P$ finally yielding $\mathscr{G}^T$ is the central goal of transitive reduction. We here present some novel and generalized variants of TR inspired from the original versions of DR-FFL and TRANSWESD.

We observed that the TR used by TRANSWESD (see Step 2 of TRANSWESD described above) can be generalized in multiple ways:

- One may consider unweighted TRANSWESD by setting $\alpha = \infty$ in the weight rule (6.5).

- One may consider unsigned TRANSWESD by setting all edge signs in $\mathcal{G}^P$ to "+". In this case, the algorithm becomes simpler (polynomial instead of NP-complete) as the calculation of shortest paths does not need to distinguish between positive and negative paths. It is then, however, still important to keep the weights to avoid non-unique results in cyclic networks.
- When searching for a suitable path P_{ij} that can explain a certain edge $i \to j$, one may restrict the search on paths involving not more edges than a predefined number L. In this way one would manifest the expectation that observed indirect effects can be traced back to short paths.

With these generalizations we introduce the notation TRANSWESD$^{\mathrm{S,W,L}}$ to specify the chosen TR variant: S $\in$ {u,s} indicates whether the signed (s) or unsigned (u) TR version is used; W $\in$ {u,w} specifies either the unweighted (u) or weighted (w) version; and L specifies the maximal path length allowed. Accordingly, the original TRANSWESD version corresponds to TRANSWESD$^{\mathrm{s,w,\infty}}$. We also observe that TRANSWESD$^{\mathrm{u,u,\infty}}$ mimics TR used by DR-FFL when removal of edges within one and the same component would be blocked. However, we soon realized that the unweighted variant of TRANSWESD does not perform very well, in particular when combined with the `full_check` = 0 option (see above). We therefore do not analyze the unweighted version in detail but keep the notation for consistency with respect to the following variant.

In addition to the modified version of TRANSWESD, we introduce a related but different strategy which we call *local transitive reduction* (LTR). There are two key differences: only paths of length 2 are considered as possible explanations for indirect effects and an alternative condition on the edge weight is introduced replacing rule (6.5). The LTR algorithm considers an edge $i \to j$ *potentially* removable if three criteria are fulfilled: (i) existence of a feed-forward loop, i.e. $\{i \to j, i \to k, k \to j\} \in \mathcal{G}^P$; (ii) sign consistency, i.e. $s_{ij} = s_{ik} \cdot s_{kj}$; and (iii) the weight condition:

$$\alpha \cdot Z^c_{ij} \leq Z^c_{ik} \cdot Z^c_{kj}, \qquad (\alpha > 0). \tag{6.6}$$

Recall that we introduced $\mathbf{Z}^c$ as the z-score of the correlation coefficients and that the relation to the edge weight $\mathbf{W}^t$ which we use for modified TRANSWESD is thus simply $\mathbf{Z}^c = \mathbf{1} - \mathbf{W}^t$. Therefore, the smaller Z^c_{ij} the higher the confidence that the path $i \to k \to j$ can explain the edge $i \to j$ (thus, a large weight is here associated with high confidence).
Analogously as described for TRANSWESD, to deal with non-uniqueness, the potentially removable edges are iteratively deleted according to the edge weights (lowest confidence first) and for each edge to be removed it is checked, whether all edges originally contained in $\mathcal{G}^P$ are still explainable by a 2-path in the remaining graph (otherwise this edge is kept).

Although LTR is also a weighted and signed TR variant, it is considerably simpler than TRANSWESD as it uses a simple triangle rule which is much easier to check than searching for suitable paths. For this reason, in contrast to TRANSWESD, we can easily use the exact variant with `path_exact` = `full_check` = 1 in large-scale networks. As will be shown in the Results section, despite its simplicity, LTR yielded excellent performance in the benchmarks. For LTR we also tested different variants, including the unweighted (condition (6.6) is dropped by setting $\alpha = 0$), the unsigned and the unsigned/unweighted version (in the latter, only the 2-path $i \to k \to j$ must exist to render edge $i \to j$ removable, irrespective of

edge signs and weights). We introduce a similar notation as for TRANSWESD: $\text{LTR}^{\text{S,W}}$ indicates whether edge signs (S ∈ {u,s}) and weights (W ∈ {u,w}) are considered or not; the length parameter L becomes obsolete as it is fixed to 2.

Edge sorting We use a simple edge ranking procedure which is similar to the strategy used by DR-FFL and (original) TRANSWESD. Note that all $n(n-1)$ potential edges (except self-loops) are included into this list, also those that were not contained in the PG $\mathcal{G}^P$ or that were removed during TR. The position of each edge is determined by the ranking weights stored in $\mathbf{W}^r$ (see above): edges with highest ranking weights are put first. To ensure that edges contained in the final graph $\mathcal{G}^T$ are really ranked higher than all other edges, an offset is added to the weight of all edges in $\mathcal{G}^T$.

6.2.3 Results and discussion

In the following we present performance results of the new PG generation algorithm in combination with the modified TRANSWESD and the new LTR technique for subsequent transitive reduction. We used two different case studies for benchmarking: (i) the datasets of the DREAM4 `InSilico_Size100` network inference challenge, and (ii) a novel large-scale synthetic compendium consisting of 30 5000-gene networks simulated by SysGenSIM [136] with different connectivities and noise levels. The DREAM4 benchmark also enables a comparison of the performances of the new approaches with its inspiring original techniques DR-FFL and (old) TRANSWESD. Generally, in the case of the two in silico datasets (where the gold standard is known) the goodness of the predictions are evaluated based on the established Area Under the Curve (AUC) measures of ROC (Receiver Operating Characteristic) and PR (Precision-Recall) curves. The AUPR is the most informative (and the only shown) performance measure for the case studies in this paper due to the sparsity of gene networks implying large AUROC values differing only insignificantly for the different methods.

Performance on DREAM4 networks

In the DREAM4 *InSilico_Size100* network reconstruction challenge [1, 112], simulated steady-state measurements of the expression of each gene in the wild-type as well as in the single-gene knockout and single-gene knockdown mutant were provided for 5 different in silico networks (100 nodes each) from which the networks had to be reconstructed. We only make use of wild-type and knockout data as they directly support the generation of the PG (knockdown data can, in principle, further improve the results; see below). For assessing the quality of reconstructed networks, an evaluation script is available at the DREAM website [4] which computes an overall score obtained from the geometric mean of p-values calculated for the AUPR and the AUROC measures from all 5 reconstructed networks.

We considered predictions by several combinations of the original as well as of the new PG generation and TR methods. The methods' parameters were chosen according to previously used values (e.g., α) or according to preliminary tests. Importantly, one and the same parameter set was used for all five networks, i.e., no optimization was conducted for every single network. The DREAM4 evaluation script was used to compute the respective overall scores which are summarized in Table 6.8. We recall that the combined use of the unsigned and z-score-based PG^1 with DR-FFL [134] originally obtained the best score (71.59) for the

Table 6.8: **Performance of the inference algorithms on the DREAM4 networks (100 nodes).** The table summarizes the performance (overall score, true positives (TPs), false positives (FPs) and false negatives (FNs)) of the different PG generation and TR algorithms when applied to the DREAM4 networks together with the displayed (optimal) parameters. The last column TP^{100} shows the average number of TPs within the first 100 top-ranked (reconstructed) edges. As a comparison, the scores of the 5 best-performing algorithms within the challenge are shown. PG^{2*} denotes PG^2 computed with a minor bug in the original implementation.

DREAM4 best performers				Score					
Team 395 (PG^1 + DR-FFL)				**71.5889**					
Team 296				71.2970					
Team 515 (PG^{2*} + $TRANSWESD^{s,w,\infty}$)				**64.7150**					
Team 466				63.4060					
Team 549				63.1050					
Inference algorithm	β	γ	α	Score	Edges	TPs	FPs	FNs	TP^{100}
PG^1	2.00	-	-	70.3495	349.4	103.4	246.0	101.4	58.0
PG^1 + DR-FFL	-	-	-	71.5889	267.4	83.6	183.8	121.2	62.4
PG^1 + $TRANSWESD^{u,w,\infty}$	**-**	**-**	**0.95**	**73.0444**	**305.2**	**97.4**	**207.8**	**107.4**	**60.8**
PG^1 + $TRANSWESD^{u,w,2}$	-	-	1.50	65.7845	103.0	9.8	93.2	195.0	53.6
PG^1 + $LTR^{u,u}$	**-**	**-**	**-**	**79.7428**	**261.2**	**92.4**	**168.8**	**112.4**	**72.2**
PG^1 + $LTR^{u,w}$	-	-	0.15	79.1609	262.0	92.8	169.2	112.0	72.2
PG^2	2.60	0.05	-	65.8012	398.2	98.0	300.2	106.8	58.2
PG^2 + DR-FFL	-	-	-	64.2614	372.8	94.4	278.4	110.4	58.0
PG^2 + $TRANSWESD^{u,w,\infty}$	-	-	0.95	65.6504	256.6	86.0	170.6	118.8	64.4
PG^2 + $TRANSWESD^{s,w,\infty}$	-	-	0.95	66.0970	260.8	86.8	174.0	118.0	66.6
PG^2 + $TRANSWESD^{u,w,2}$	-	-	1.50	66.5562	224.0	81.0	143.0	123.8	64.2
PG^2 + $TRANSWESD^{s,w,2}$	**-**	**-**	**1.50**	**68.1534**	**249.2**	**84.4**	**164.8**	**120.4**	**67.0**
PG^2 + $LTR^{u,u}$	-	-	-	65.4214	253.0	82.2	170.8	122.6	64.6
PG^2 + $LTR^{s,u}$	-	-	-	67.7407	274.0	86.4	187.6	118.4	66.6
PG^2 + $LTR^{u,w}$	-	-	0.15	67.2567	271.4	85.6	185.8	119.2	66.0
PG^2 + $LTR^{s,w}$	**-**	**-**	**0.15**	**68.5959**	**288.2**	**88.4**	**199.8**	**116.4**	**67.2**
PG^{new}	**2.00**	**0.00**	**-**	**81.7594**	**250.2**	**99.6**	**150.6**	**105.2**	**66.6**
PG^{new} + DR-FFL	-	-	-	80.3085	179.8	82.0	97.8	122.8	66.2
PG^{new} + $TRANSWESD^{u,w,\infty}$	-	-	0.95	85.3288	179.2	90.2	89.0	114.6	72.0
PG^{new} + $TRANSWESD^{s,w,\infty}$	-	-	0.95	85.7898	183.0	92.0	91.0	112.8	72.8
PG^{new} + $TRANSWESD^{u,w,2}$	-	-	1.50	88.0570	147.6	86.0	61.6	118.8	72.6
PG^{new} + $TRANSWESD^{s,w,2}$	**-**	**-**	**1.50**	**88.5728**	**150.4**	**87.8**	**62.6**	**117.0**	**72.8**
PG^{new} + $LTR^{u,u}$	-	-	-	88.2217	166.8	91.8	75.0	113.0	74.2
PG^{new} + $LTR^{s,u}$	-	-	-	88.6350	169.4	93.4	76.0	111.4	75.2
PG^{new} + $LTR^{u,w}$	-	-	0.15	88.5203	168.4	92.8	75.6	112.0	75.0
PG^{new} + $LTR^{s,w}$	**-**	**-**	**0.15**	**88.8005**	**169.8**	**93.8**	**76.0**	**111.0**	**75.8**

DREAM4 challenge, while the coupling of PG^2 and (original) TRANSWESD was ranked third with a score of 64.71 (see [91]). Although we used here only the knockout data (in [91] both knockout and knockdown data were used for computing the correlation coefficents) the results presented in Table 6.8 are slightly better (66.10) by fixing a small bug in the computation of PG^2.

The DREAM4 best overall score of 71.59 is already exceeded by just applying unsigned $\text{TRANSWESD}^{u,w,\infty}$ or unsigned and unweighted $\text{LTR}^{u,u}$ to the unsigned perturbation graph PG^1. This supports the statement in [153] about the weakness of the original DR-FFL algorithm, where transitive reduction is applied only to edges between but not within strongly connected components of the PG. In fact, both $\text{TRANSWESD}^{u,w,\infty}$ and especially $\text{LTR}^{u,u/w}$ improve the score of PG^1 significantly up to 79.74.
Regarding the results for PG^2 originally used by the TRANSWESD method in [91] we observe that the quality (score) of the PG is lower than for the simple z-score PG^1. Although all tested TR techniques (except DR-FFL) can improve the score, it remains below the performance results obtained for the z-score approach PG^1.

Next we tested the performance results of the TR techniques in PG^{new} where we also applied the new edge sorting scheme and sorting weights. As a first observation, a notable quality improvement is obtained by the novel PG^{new} alone achieving a score of 81.76 which is markedly higher than the scores obtained by PG^1 and PG^2, even after TR. A somewhat unexpected result was that the γ threshold for the conditional correlation coefficients was virtually not required as its optimal value turned out to be 0. However, in other tests described below, using a non-zero value for this threshold in combination with β (for the z-score) turns out to be beneficial. We also analyzed the robustness of the quality of PG^{new} and its edge ordering with respect to the chosen threshold parameters β and γ: Figure 6.6 displays the overall score of PG^{new} when varying the threshold parameters showing that it is (i) higher than the previous winning score (71.59), (ii) higher than PG^1, and (iii) higher than PG^2 – even for the complete space of meaningful parameter values scanned. Hence, a reasonable robustness of the quality of PG^{new} with respect to the two threshold parameters can be concluded. We then applied the TR techniques to PG^{new} which increase the scores up to 88.80, thus well above the best score recorded at the DREAM4 challenge. Regarding the different TRANSWESD variants, we see that the signed version (85.79) is only slightly better than the unsigned variant (85.33) whereas *local* TRANSWESD, which takes only paths of length 2 into account, results in a further significant improvement of the score (88.57). In line with these observations, unsigned and signed LTR differ only marginally whereas signed and weighted $\text{LTR}^{s,w}$ produces the best overall results, not far from the unweighted variant. Recall that $\text{TRANSWESD}^{s,w,2}$ and $\text{LTR}^{s,w}$ differ essentially only by condition (6.5) vs. (6.6). Although the results of both local variants are comparable, it seems that the rule used by LTR can better predict true indirect effects. Generally, Table 6.8 shows that all TR techniques work well by strongly decreasing the number of false positive edges (FPs) with only a slight decrease in number of true positives (TPs). Apparently, the best ratio is obtained by local TR variants (i.e., by LTR and $\text{TRANSWESD}^{s,w,2}$). We also noticed that the (average) enrichment of TPs under the first 100 reconstructed edges in the sorted edge list (column TP^{100} in Table 6.8) is especially large confirming the potential of our methods: it reaches 66.6 for PG^{new}, 72.8 for $\text{TRANSWESD}^{s,w,2}$ and even 75.4 for $\text{LTR}^{s,w}$. Hence, there is a high probability that top-ranked edges correspond to true interactions – a desirable property when validating the

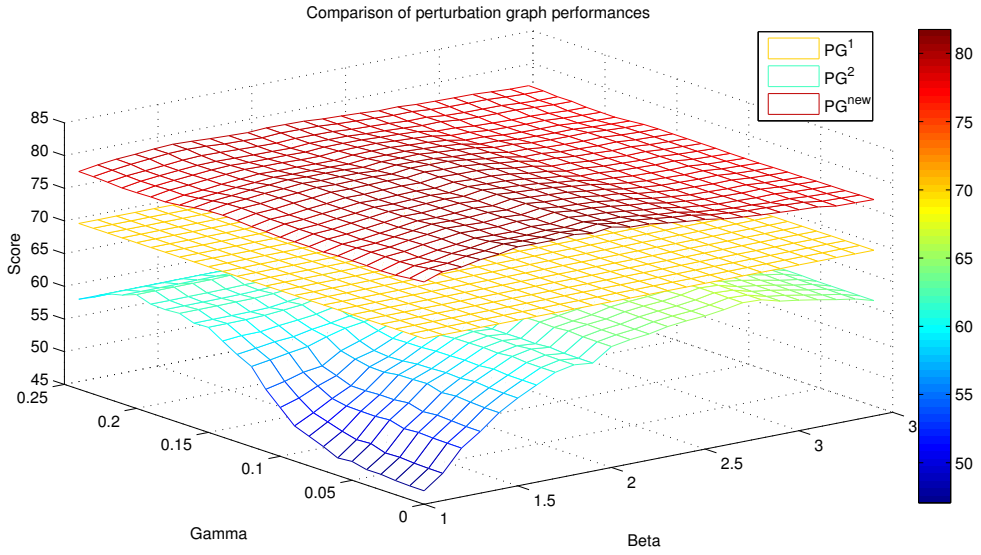

Figure 6.6: **Performance of the PG^{new} methodology in DREAM4 networks (100 nodes).** The overall score of PG^{new} is consistently higher than the scores achieved by the DREAM4 winning submission (71.59, not shown), by PG^1 (70.35; does not depend on any parameter), and by PG^2 for the complete space of meaningful parameter values (β, γ).

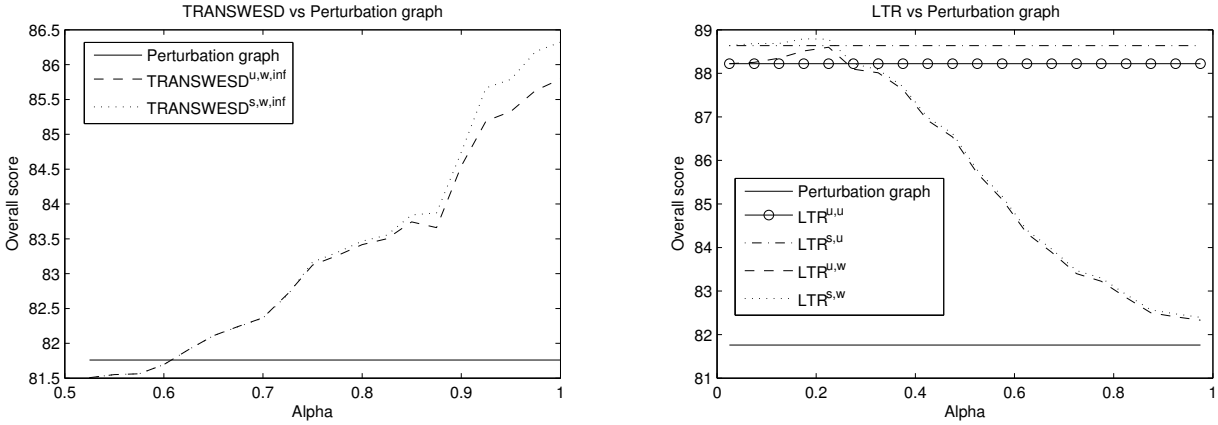

Figure 6.7: **Robustness of TRANSWESD (left) and LTR (right) variants in DREAM4 networks.** The overall scores achieved after TR with TRANSWESD$^{\cdot,w,\infty}$ or LTR$^{\cdot,\cdot}$ are consistently higher than the score obtained by PG^{new} for a large range of meaningful values of the confidence factor α. TRANSWESD$^{\cdot,\cdot,2}$ was not included because of its different operating range of values.

edges experimentally.

Figure 6.7 demonstrates that TRANSWESD (left) and LTR (right) are also fairly robust with respect to the confidence factor, as the score of the perturbation graph PG^{new} is highly improved by both methods for a broad range of meaningful values of α. In this context it is also of interest that unweighted LTR yielded very good predictions even without the need to specify any further parameter (as necessary for weighted LTR and TRANSWESD).

We summarize that the best-ranked algorithms of the DREAM4 competition are significantly outperformed by our new methods for PG generation, TR, and edge sorting. We noticed that also other recently published network inference techniques applied to the DREAM4 networks [175, 30] reported lower predictions. For example, the highest overall score in [175] is 81.10 obtained by using both knockout and knockdown datasets whereas a score of 73.33 was achieved in [30] by CUTTER-W, an approach similar to unsigned (and weighted) TRANSWESD. Moreover, if we also include knockdown data in our analysis (for calculating the conditional correlation coefficients), the scores in Table 6.8 grow up by approximately 3-4 points each, reaching a top of 92.03 with PG^{new} + $\text{LTR}^{\text{s,w}}$. As expected, this confirms that an increase of the number of measurements corresponds to an improvement of the prediction. However, most of the information is already provided by the knockout experiments.

Performance on SysGenSIM datasets

In order to provide an even more exhausting and more realistic test scenario for the developed inference algorithms, the software SysGenSIM [136] was used to create a new collection of synthetic gene networks and to simulate knockout experiments under different connectivities and noise conditions. SysGenSIM is able to generate large networks with a topology similar to those observed in real organisms, i.e., with a modular structure featuring exponential and power law behavior for the in- and out-degree distributions of nodes [70]. The generated 30 in silico networks have a considerable (close to genome-scale) size of 5000 nodes each. One third of them has a low average degree (about 7500 edges, i.e. $K \simeq 1.5$), 10 networks have a mean average degree (about 10000 edges, i.e. $K \simeq 2$), while the last third of the networks exhibits the largest average degree (about 12500 edges, i.e. $K \simeq 2.5$). Finally, using equations of biochemical kinetics where the degradation rate of gene expression is represented by a first order process and the transcription rate exhibits the essential features of cooperativity and saturation [119], single knockout experiments have been simulated for all the genes of each network with SysGenSIM's default kinetic parameters under 9 different combinations of noise conditions (for technical details see [136]). In fact, SysGenSIM allows for the selection of the standard deviation σ_ϑ of the Gaussian distribution from which the biological synthesis and degradation variances, are sampled (parameters ϑ^{syn} and ϑ^{deg} in Equation (3.1)) as well as the standard deviation σ_ν of the Gaussian distribution from which the experimental noise ν is sampled. As possible values for both standard deviations we considered $\{0.025, 0.05, 0.1\}$, yielding a total of 9 combinations summarized in Table 6.9. Therefore a grand total of 270 different networks (30 topologies with 9 different noise configurations) with simulations of single-gene knockout experiments have been produced, the goal being the testing of the inference methodologies under different conditions of edge density, biological variance, and multiplicative measurement noise.

Due to the superior performance of PG^{new} we present results only for this PG. Figure 6.8 exemplifying displays the performance of PG^{new} for the 9 different noise configurations of network 1 (connectivity $K \simeq 1.5$) in dependency of a wide range of (β, γ) parameters (examples for $K \simeq 2$ and $K \simeq 2.5$ are shown in Figures 6.9 and 6.10). The novel PG generation algorithm exhibits reasonable robustness with respect to both noise and threshold parameters. In fact it works decently with the same β and γ as used for the DREAM4 networks (for the optimal value, γ needs to be slightly raised from 0.00 to 0.05), while the procedures for

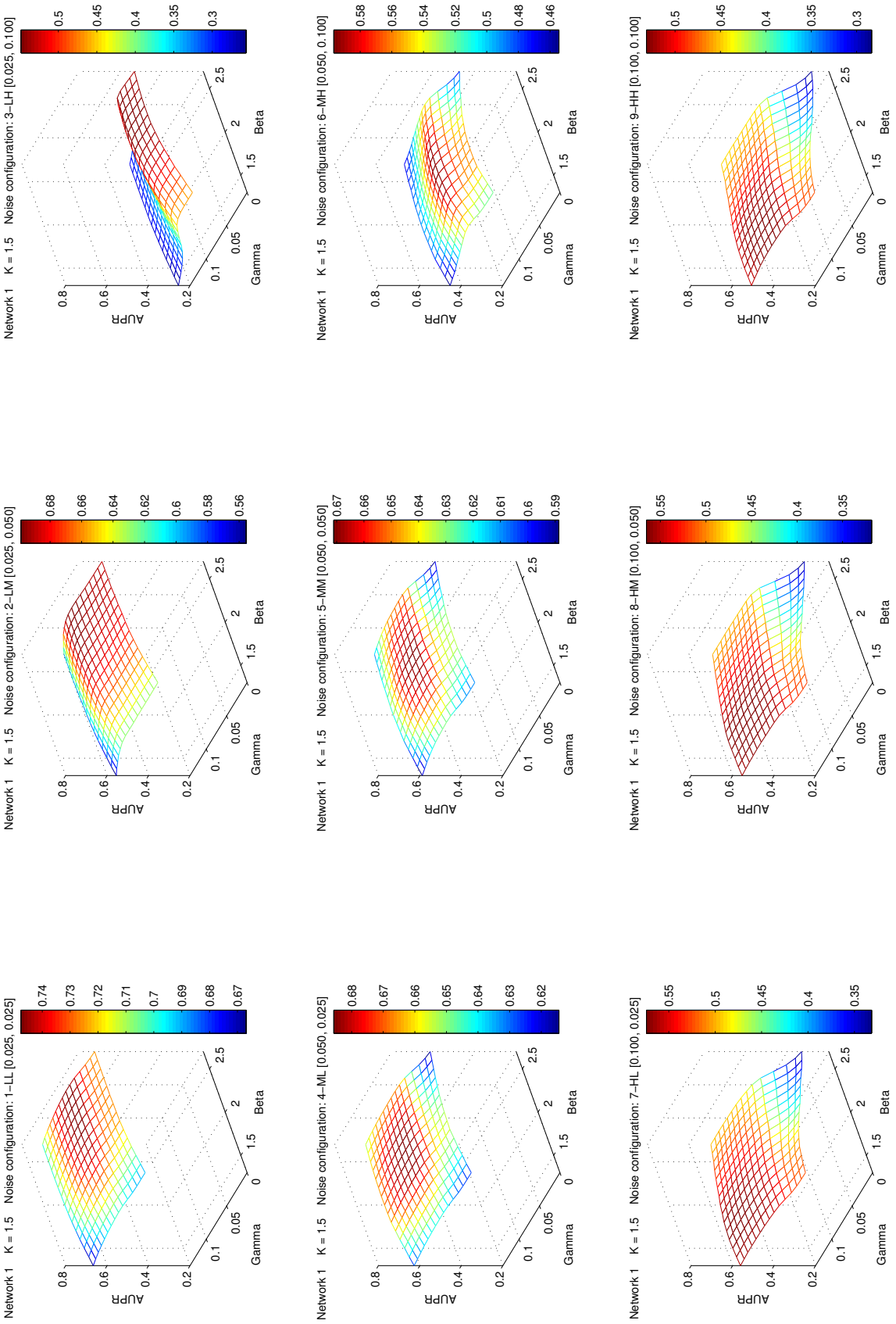

Figure 6.8: **Performance and robustness of the PG^{new} methodology applied to the 9 noise configurations of network 1 in the SysGenSIM dataset.** The AUPR scores of PG^{new} are fairly robust for a large range of meaningful parameter values β and γ. The picture shows the performance of the inference of network 1 (containing about 7500 edges) with respect to the 9 different noise conditions.

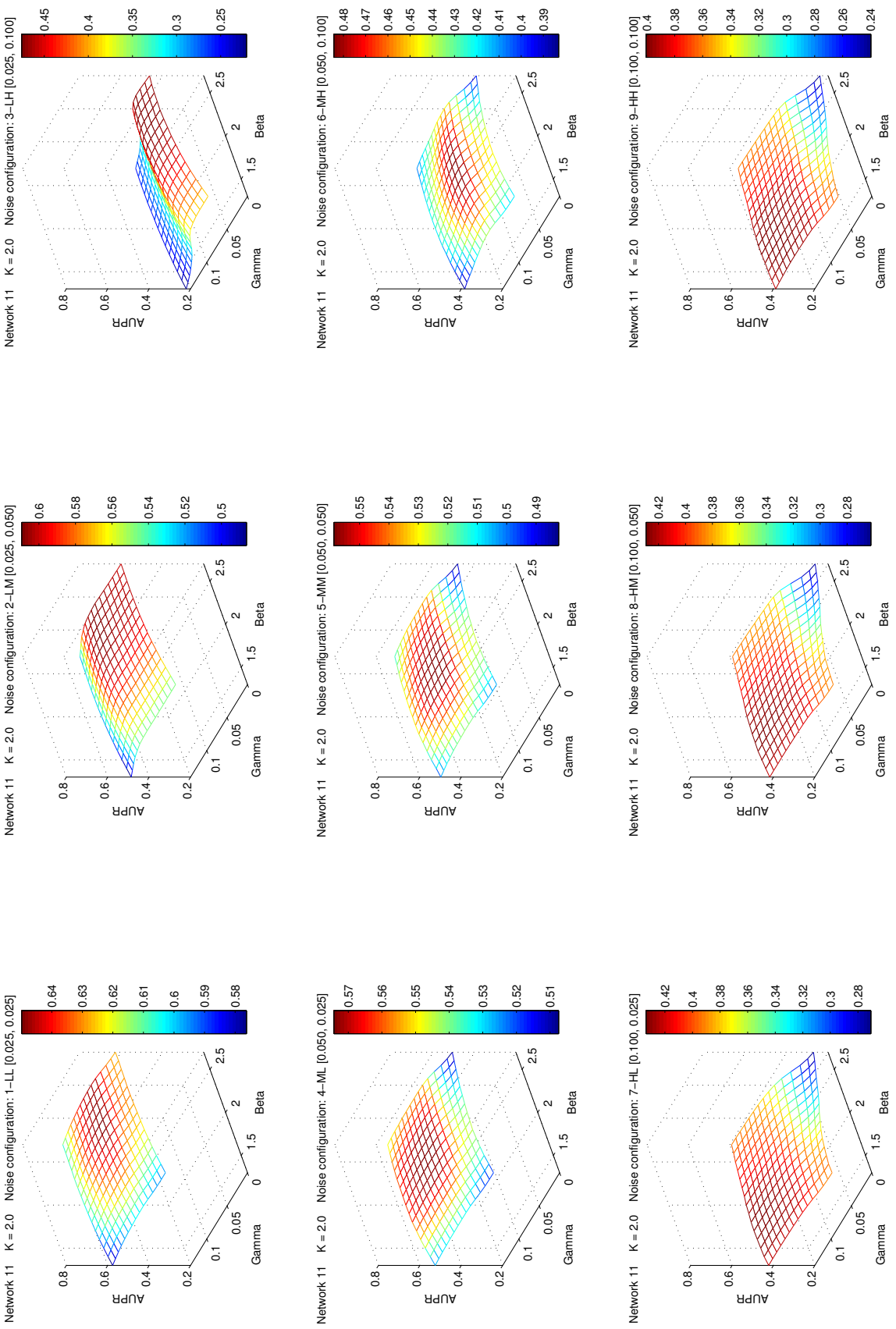

Figure 6.9: **Performance and robustness of the PG$^{\text{new}}$ methodology applied to the 9 noise configurations of network 11 in the SysGenSIM dataset.** The AUPR scores of PG$^{\text{new}}$ are fairly robust for a large range of meaningful parameter values β and γ. The picture shows the performance of the inference of network 11 (containing about 10000 edges, i.e. $K \simeq 2$) with respect to the 9 different noise conditions.

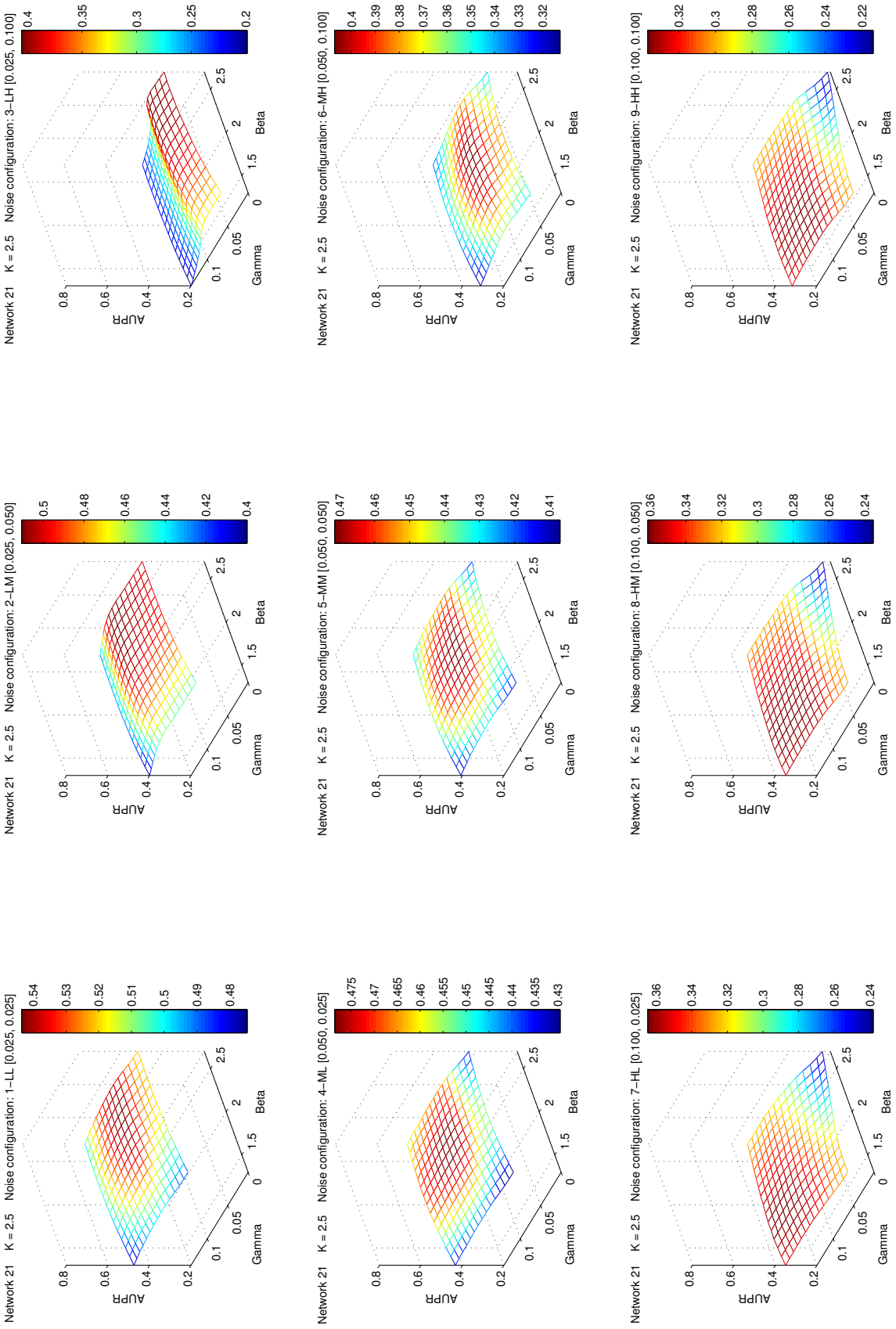

Figure 6.10: **Performance and robustness of the PG$^{\text{new}}$ methodology applied to the 9 noise configurations of network 21 in the SysGenSIM dataset.** The AUPR scores of PG$^{\text{new}}$ are fairly robust for a large range of meaningful parameter values β and γ. The picture shows the performance of the inference of network 21 (containing about 12500 edges, i.e. $K \simeq 2.5$) with respect to the 9 different noise conditions.

Table 6.9: **Noise configurations in simulated datasets.** Columns σ_ϑ and σ_ν show the values of the standard deviations of the Gaussian distributions with $\mu = 1$ from which the biological variances ϑ^{syn} and ϑ^{deg} and the measurement error ν were sampled in SysGenSIM. Each configuration is also represented by a 2-character string, indicating the intensity levels (low (L), medium (M), high (H)) of the biological variance (first letter) and of the measurement error (second letter), respectively.

Configuration	Label	σ_ϑ	σ_ν
1	LL	0.025	0.025
2	LM	0.025	0.050
3	LH	0.025	0.100
4	ML	0.050	0.025
5	MM	0.050	0.050
6	MH	0.050	0.100
7	HL	0.100	0.025
8	HM	0.100	0.050
9	HH	0.100	0.100

PG^1 and PG^2 would need a more extensive re-tuning of the parameters to obtain reasonable results (not shown).

The effect of the TR algorithms applied to PG^{new} (Tables 6.10 and 6.11, Figure 6.11) becomes more heterogeneous and differentiated compared to the DREAM4 networks. First of all, we observe that the unweighted versions of LTR decrease in all cases the quality of the perturbation graph PG^{new} whereas weighted LTR and (non-local versions of) TRANSWESD improve it – partially significantly – in all scenarios (with one minor exception). This demonstrates that weighted TR can be highly beneficial. However, local $\text{TRANSWESD}^{\text{s,w},2}$, which was comparable with LTR in the DREAM4 networks, achieves similar unfavorable results for these large and noisy networks as unweighted LTR. This confirms again that rule (6.6) seems to be better suited for *local* TR than rule (6.5). Furthermore, the quality of the PG as well as the relative improvement by the (weighted) TR techniques depends substantially on the magnitude of the noise level both with respect to AUPR and in the number of TPs and FPs. An interesting observation can be made regarding the effect of biological variance on the reconstruction quality: it appears that moderately increased (medium) biological noise is advantageous in case of high measurement noise for all K's (see Figure 6.11, Table 6.10 and Tables T1 and T2 in Additional File 1). Thus, higher biological noise may help to uncover true perturbation effects under high uncertainty of measurements. It can also be noticed that, in general, $\text{TRANSWESD}^{\text{s,w},\infty}$ and $\text{LTR}^{\text{s,w}}$ achieve similar superior AUPR performance, but by different means as manifested in Table 6.11 the LTR technique prunes the edges of the PG more generously than $\text{TRANSWESD}^{\text{s,w},\infty}$, resulting in a better reduction of false positive edges, but at the same time in an undesired higher decrease of true positives. Finally, we can confirm a result from the DREAM4 benchmark: signed (weighted) LTR and TRANSWESD achieved always better AUPR scores than their unsigned versions (except in one case) but only to a very small extent. This important observation are discussed in more detail in the Conclusion section. Finally, Figure 6.12 also indicates how the precision of PG^{new} and the effectiveness of TR decrease when the network connectivity (average node degree K) increases.

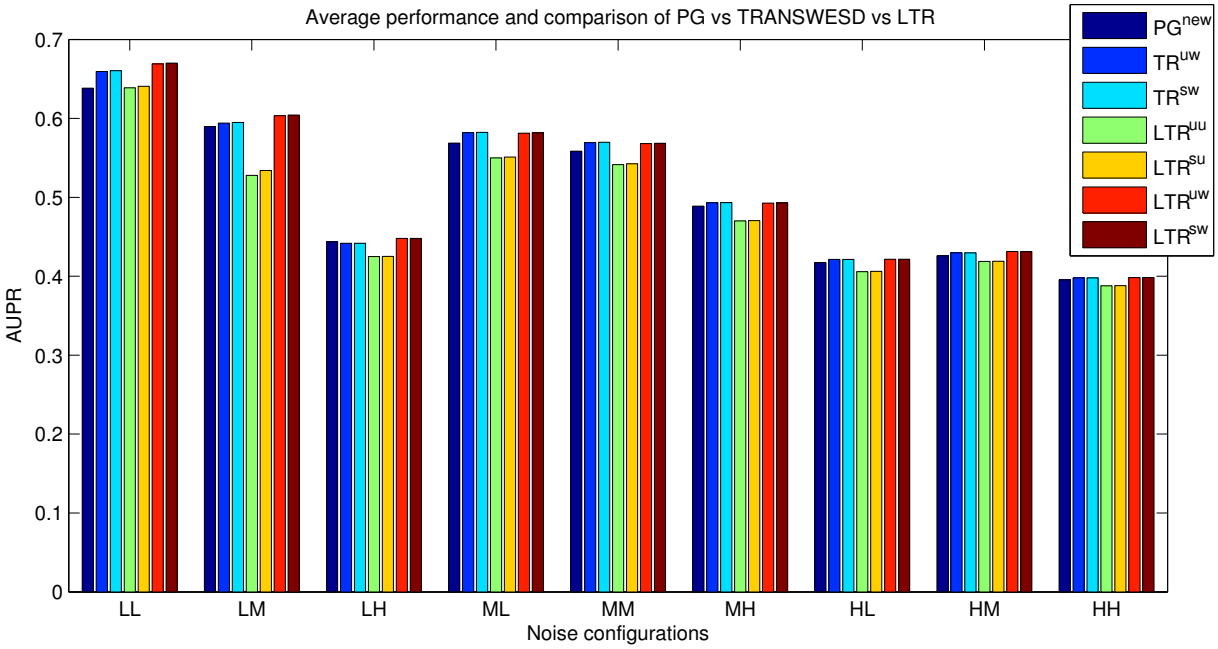

Figure 6.11: **Average performance of TRANSWESD and LTR variants on the SysGenSIM datasets.** Parameters used to obtain the perturbation graph were $\beta = 2.0$ and $\gamma = 0.05$, while $\alpha = 0.95$ and $\alpha = 0.15$ were selected for the TRANSWESD and LTR variants, respectively. AUPR scores are averaged across the 30 networks (10 networks for each of the three averaged node degrees considered) simulated with the same noise configuration. The efficacy of transitive reduction changes with the noise levels.

Table 6.10: **Performance of the inference algorithms on the SysGenSIM networks.** Each score is the mean of the AUPR computed for the 10 networks with $K \simeq 1.5$ simulated according to the same noise configuration. Thresholds used by the inference algorithms are $\beta = 2.0$ and $\gamma = 0.05$ for generating PG^{new}, $\alpha = 0.95$ for $\text{TRANSWESD}^{\cdot,\text{w},\infty}$, $\alpha = 1.50$ for $\text{TRANSWESD}^{\cdot,\text{w},2}$ and $\alpha = 0.15$ for $\text{LTR}^{\cdot,\text{w}}$. Analogous performances for $K \simeq 2$ and $K \simeq 2.5$ are shown in Tables T1 and T2 in Additional File 1.

	Noise configuration								
Inference algorithm	1 - LL	2 - LM	3 - LH	4 - ML	5 - MM	6 - MH	7 - HL	8 - HM	9 - HH
PG^{new}	0.7388	0.6835	0.5159	0.6735	0.6622	0.5850	0.5141	0.5218	0.4835
PG^{new} + $\text{TRANSWESD}^{\text{u,w},\infty}$	0.7695	0.6906	0.5141	0.6921	0.6778	0.5921	0.5192	0.5269	0.4868
PG^{new} + $\text{TRANSWESD}^{\text{s,w},\infty}$	0.7702	0.6910	0.5142	0.6923	0.6780	0.5922	0.5192	0.5269	0.4868
PG^{new} + $\text{TRANSWESD}^{\text{u,w},2}$	0.7335	0.5825	0.5041	0.6471	0.6410	0.5650	0.4963	0.5115	0.4737
PG^{new} + $\text{TRANSWESD}^{\text{s,w},2}$	0.7354	0.5929	0.5042	0.6478	0.6417	0.5653	0.4965	0.5116	0.4739
PG^{new} + $\text{LTR}^{\text{u,u}}$	0.7561	0.6320	0.5114	0.6701	0.6583	0.5783	0.5093	0.5209	0.4816
PG^{new} + $\text{LTR}^{\text{s,u}}$	0.7570	0.6390	0.5115	0.6705	0.6587	0.5784	0.5094	0.5210	0.4818
PG^{new} + $\text{LTR}^{\text{u,w}}$	0.7737	0.7051	0.5285	0.6898	0.6751	0.5924	0.5196	0.5291	0.4880
PG^{new} + $\text{LTR}^{\text{s,w}}$	0.7742	0.7057	0.5285	0.6900	0.6753	0.5925	0.5196	0.5291	0.4880

Table 6.11: **Statistics on edges from the inferred SysGenSIM networks.** The number of edges in the perturbation graph and the number of FPs and TPs are shown in the table. The relative reduction of these measures (edges, TPs and FPs) in the graphs obtained after applying the different TR techniques compared to the PG is also displayed. These averaged statistics are computed from the analysis of the graphs obtained after inferring the 30 SysGenSIM networks simulated according to configuration 1 (LL). Thresholds used are $\beta = 2.0$ and $\gamma = 0.05$ for generating PG^{new}, $\alpha = 0.95$ for $TRANSWESD^{\cdot,w,\infty}$, $\alpha = 1.50$ for $TRANSWESD^{\cdot,w,2}$ and $\alpha = 0.15$ for $LTR^{\cdot,w}$. Analogous tables for the other 8 configurations $(2, \ldots, 9)$ are shown in Tables T3–T10 in Additional File 1.

	$K \simeq 1.5$			$K \simeq 2$			$K \simeq 2.5$		
Inference algorithm	Edges	TPs	FPs	Edges	TPs	FPs	Edges	TPs	FPs
PG^{new}	14353	6239	8114	20477	7422	13055	27496	8258	19239
PG^{new} + $TRANSWESD^{u,w,\infty}$	-15.6%	-0.6%	-27.2%	-15.7%	-2.0%	-23.5%	-14.9%	-3.5%	-19.8%
PG^{new} + $TRANSWESD^{s,w,\infty}$	-15.4%	-0.4%	-26.8%	-15.2%	-1.6%	-22.9%	-14.4%	-3.0%	-19.3%
PG^{new} + $TRANSWESD^{u,w,2}$	-37.5%	-16.0%	-54.0%	-42.8%	-25.0%	-52.9%	-46.7%	-33.1%	-52.5%
PG^{new} + $TRANSWESD^{s,w,2}$	-37.2%	-15.7%	-53.9%	-42.4%	-24.2%	-52.7%	-46.1%	-31.8%	-52.2%
PG^{new} + $LTR^{u,u}$	-32.1%	-9.7%	-49.3%	-35.3%	-15.1%	-46.8%	-37.7%	-19.8%	-45.3%
PG^{new} + $LTR^{s,u}$	-31.9%	-9.5%	-49.2%	-35.0%	-14.6%	-46.6%	-37.4%	-19.2%	-45.2%
PG^{new} + $LTR^{u,w}$	-29.7%	-5.8%	-48.2%	-30.6%	-7.8%	-43.6%	-28.6%	-8.9%	-37.1%
PG^{new} + $LTR^{s,w}$	-29.6%	-5.7%	-48.1%	-30.5%	-7.5%	-43.5%	-28.5%	-8.5%	-37.0%

Moreover, the superiority of weighted vs. unweighted TR can again clearly be seen.

Application to a realistic yeast knockout dataset

The ultimate test for our reverse-engineering algorithm would be the application to a genome-scale real-world dataset of single-gene perturbation (e.g., knockout) experiments. Few such datasets are available. The most suitable for our purpose is the *S. cerevisiae* transcription factor knockout expression compendium of Hu *et al.* [81] where the expression of $n = 6253$ genes was measured after single knockouts (or knockdowns) of $m = 269$ transcription factors (TFs) being the most important regulators in yeast. Herein we refer to the revised dataset provided by Reimand *et al.* [140] where the original raw data of Hu *et al.* were reanalyzed with more sophisticated statistical techniques from the BioConductor package [66] leading to an increased informative content of the microarray measurements.

The processed data of Reimand *et al.* (available at [3]) consists of three matrices of size $m \times n$:

- **L** contains the log-fold change values for all genes across all knockout experiments;
- **P** includes the p-values for differential expression;
- **A** is the signed adjacency matrix (we can consider it as the reconstructed graph by Reimand *et al.*) whose entries $A(i, j)$ correspond to (inferred) edges with a p-value of $P(i, j) < 0.05$.

Our goal was to re-process the log-fold change values in order to apply our network inference algorithm and to produce a ranked list of edges. For comparing our reconstructed network with the predicted network of Reimand *et al.* we need a gold standard. However, as a reliable gold standard for gene regulation in *S. cerevisiae* is still not available (otherwise we would

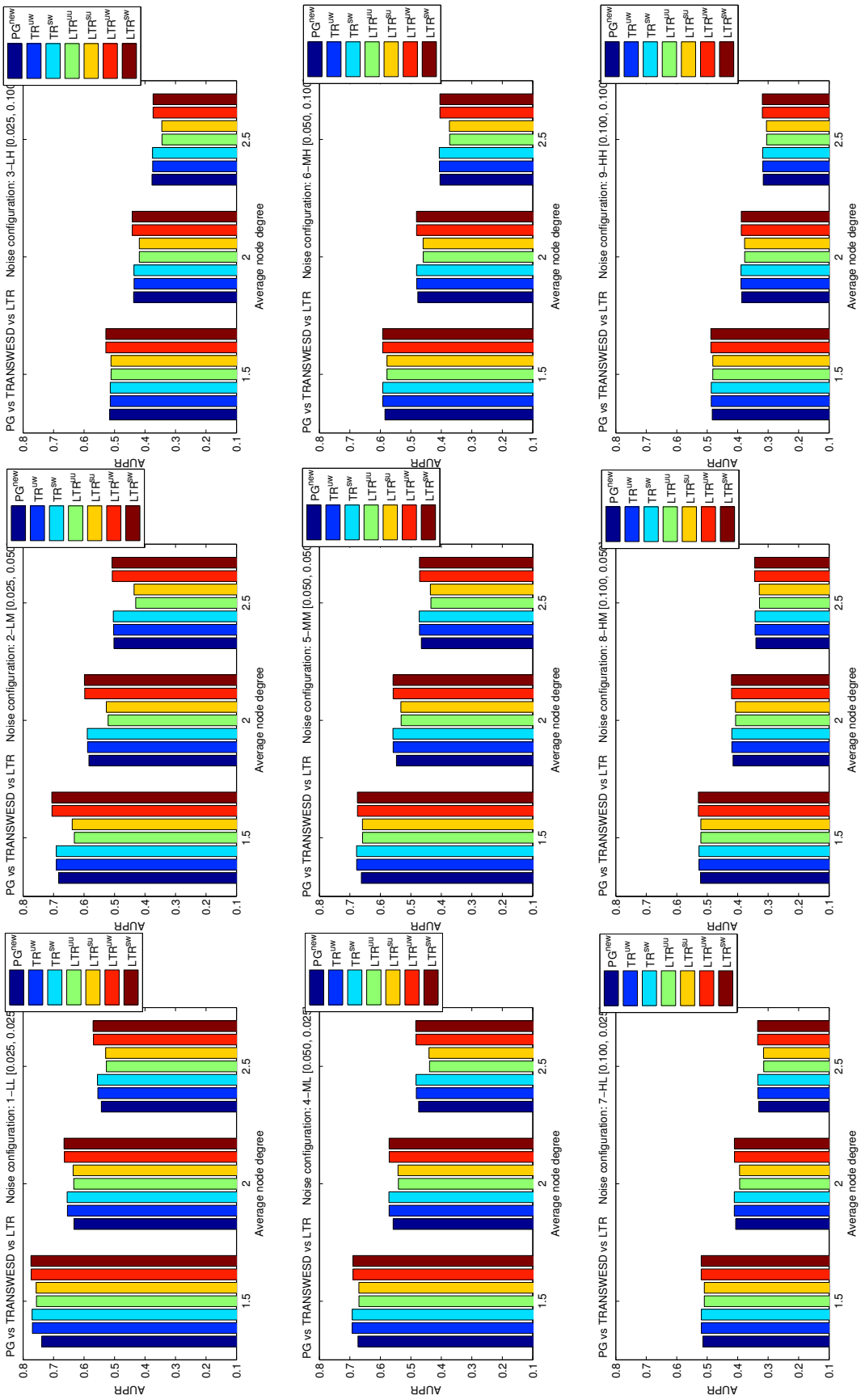

Figure 6.12: **Performance of TRANSWESD and LTR variants on the SysGenSIM datasets.** Parameters used to obtain the perturbation graph were $\beta = 2.0$ and $\gamma = 0.05$, while $\alpha = 0.95$ and $\alpha = 0.15$ were selected for the TRANSWESD and LTR variants, respectively. AUPR scores are averaged across the 10 networks generated with the same K and simulated with the same noise configuration. The efficacy of transitive reduction changes with the connectivity of the network and with the noise levels.

not need our inference methods), so we made use of four published *silver standard* networks comprising well-known interactions or edges (between TFs or from TFs to other genes) with high probability:

1. $\mathbf{SS}_1$: is a collection of found chip-chip results and motifs from 162 TFs [87] (size of silver standard network: 162 TFs × 6253 genes).

2. $\mathbf{SS}_2$: is a subset of the binding sites from $\mathbf{SS}_1$ which are also in nucleosome-depleted regions (a specific yeast environment of optimal growth conditions: targets are most likely to be active during optimal growth because they are located in open chromatin regions and therefore accessible for TF binding) [87] (size of silver standard network: 159 TFs × 6253 genes).

3. $\mathbf{SS}_3$: the silver standard network by Luscombe [106] contains known regulatory interactions between 142 TFs and 3459 targets compiled from the results of genetic, biochemical and ChIP (chromatin immunoprecipitation)-chip experiments (size of silver standard network: 142 TFs × 3459 genes).

4. $\mathbf{SS}_4$: it contains interactions between 114 TFs and 5667 targets and was used as a reference network for a sub-challenge of the DREAM5 competition [111] (size of silver standard network 114 TFs × 5667 genes).

In order to obtain a gene expression matrix $\mathbf{G}$ exploitable by our inference algorithm, we "inverted" the log-fold change values to obtain $G(i,j) = 2^{L(i,j)}$ for all the possible edges. In this way, expression values larger than 1 represent an increase of the gene expression of j after the knockout of i (i.e., i is a inhibitor of j), and vice versa a positive regulation for values smaller than 1.

As for the synthetic datasets, the gene expression matrix $\mathbf{G}$ served then as input to produce the perturbation graph $\mathrm{PG}^{\mathrm{new}}$ and the weight matrices $\mathbf{W}^r$ and $\mathbf{W}^t$, all of size $m \times n$. The transitive reduction techniques TRANSWESD and LTR were then applied to $\mathrm{PG}^{\mathrm{new}}$ and the resulting edges for each method were sorted according to our ranking scheme. This sorted edge list was delivered as output (prediction) of our procedure. We performed the whole inference process by employing the same parameters used for the simulated networks, i.e. $\beta = 2.00$, $\gamma = 0.05$, $\alpha = 0.95$ for TRANSWESD, $\alpha = 1.50$ for local TRANSWESD, and $\alpha = 0.15$ for LTR.

A (predicted) confidence-sorted edge list was also obtained for the original dataset of Reimand *et al.* by re-sorting the absolute values of the log-fold changes in $\mathbf{L}$ according to the adjacency matrix $\mathbf{A}$, which is then used as a reference to assess the performance of our inference algorithms.

Afterwards we evaluated all predictions against the 4 silver standards. To allow for a fair scoring, only common nodes from silver standard networks and prediction lists were taken into consideration, i.e. if edge (i,j) is in the prediction list but either node i or j is not in the silver standard, then the edge is not scored. On the other hand, if node k belongs to the silver standard but was not included in the microarray dataset, then all ingoing and outgoing edges of k were removed from the silver standard. Accordingly, the size of the silver standard

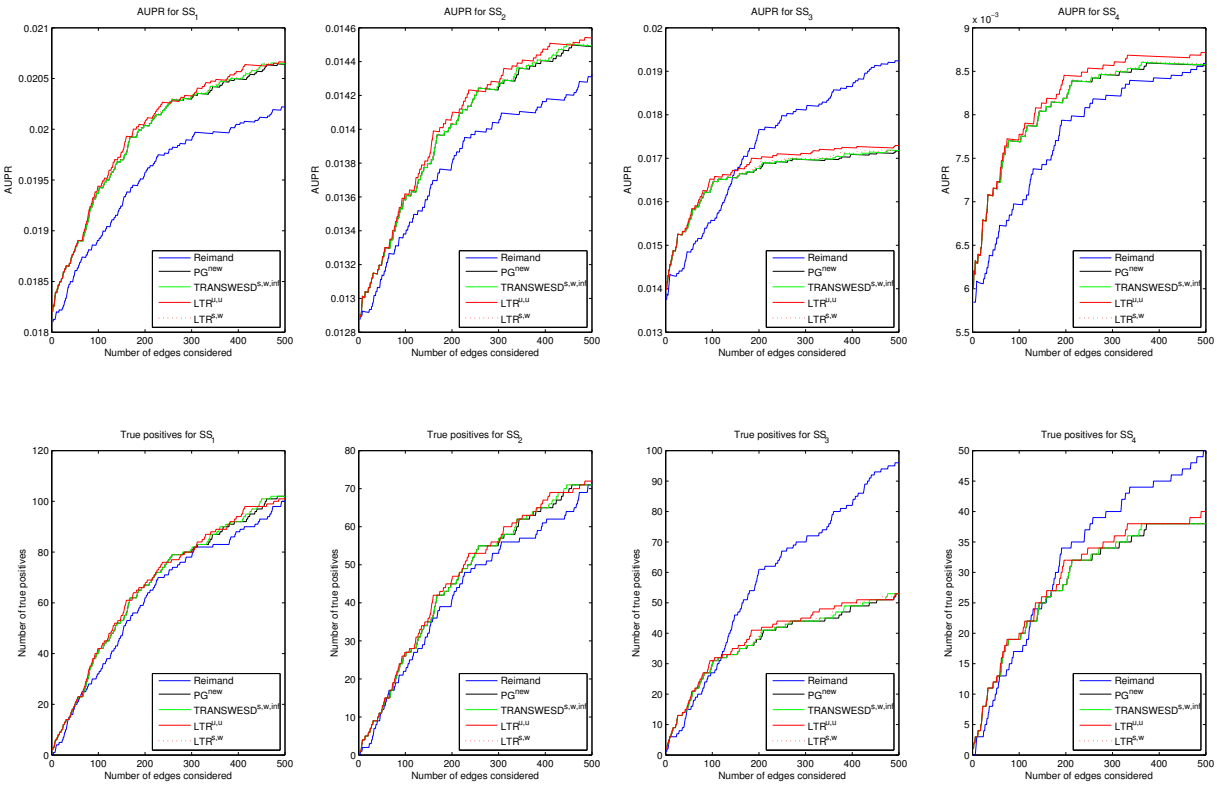

Figure 6.13: **Performance of the novel inference techniques on the *S. cerevisiae* dataset validated against four silver standards.** The plots show the AUPR and the number of true positive edges computed for the 500 best-ranked edges against four *silver standard* networks (see text for explanations). Parameters used to infer the networks are $\beta = 2.0$ and $\gamma = 0.05$ for the PG, $\alpha = 0.95$ for TRANSWESD and $\alpha = 0.15$ for LTR.

SS_3 reduced from 142×3459 to 122×3444 and of SS_4 from 114×5667 to 108×5469.

For each of the four silver standards, Figure 6.13 shows the AUPR and the number of true positive edges (TP) computed for an increasing number of edges selected from top of the ordered edge lists as given by (1) Reimand's predictions, (2) the perturbation graph PG^{new}, (3) $TRANSWESD^{s,w,\infty}$, (4) $LTR^{u,u}$, and (5) $LTR^{s,w}$. Generally, the results of our methods (2)-(5) with respect to the four different silver standard networks appear to be satisfactory, though with different measure for the four silver standard networks. It is apparent that our methods work especially well within the 100-200 top-ranked edges where all of the inferred networks (2)-(5) show better agreement with the silver standards than the interactions found by Reimand. Even the PG itself performs quite well and better than Reimand's in this region. All variants of TR show positive effects but not among the top-ranked edges because these are immune against pruning (accordingly, for these edges, the results of PG^{new} is identical with the TR methods). We observe that unweighted and unsigned $LTR^{u,u}$ performed best for this dataset. However, one should keep in mind that no tuning or adaptation of the parameters has been performed which could prevent a better result for the weighted versions.

When increasing the number of considered edges to 500, it appears that Reimand's net-

work becomes better and better eventually, in some cases, getting higher overall agreement with the silver standards than our methods. However, we argue that for practical applications (e.g. validation of edge candidates) the first 100 edges are the most important ones. In this region, given the silver standards, our approach seems to work most efficient yielding high statistical significance: for the four silver standard networks we obtained [42, 27, 31, 20] TPs in the network reconstructed with $\text{LTR}^{\text{u,u}}$ yielding corresponding p-values of $[6.31 \cdot 10^{-46}, 6.68 \cdot 10^{-28}, 3.93 \cdot 10^{-33}, 6.79 \cdot 10^{-25}]$, based on the hypergeometric distribution. These values are very similar for the other four PG/TR-based methods. It is most likely that the number of TPs is even larger given the high probability that not all interactions might be contained in the silver standards. Our prediction list might thus provide useful targets for validations. A list of the 300 first identified edges (with highest confidence) and a comparison with the four silver standards can be found in the Additional File 2.

6.2.4 Conclusion

We presented novel algorithms for the inference of gene regulatory networks from systematic perturbation experiments. These algorithms support the reconstruction of regulatory networks via three steps: (i) PG generation, (ii) TR to remove edges representing indirect effects in the PG, and (iii) sorting of edge candidates. We presented new variants for all of these three steps whose combined use yielded superior results over previous methods when tested with standardized benchmark scenarios.

Regarding the PG, it proved advantageous to identify, weight and sort candidate edges by a mixture of two measures, one being the standard z-score of deviations, the other one the z-score of conditional correlation coefficients. In particular, the latter was highly informative for edge pruning by TR whereas a combined weight of both z-scores proved beneficial for edge sorting. With the new candidate edge selection and edge sorting schemes, we observed that the PG alone (without TR) achieved a reconstruction quality that is far above the results of previous methods *after* TR. Importantly, the quality of the PG appeared to be robust against larger variations in the two required threshold parameters. In this regard, one aspect for future work is to develop algorithms for automatic thresholding, that is to estimate the threshold parameters from the data.

We proposed new variants of TR and, based on unbiased in silico benchmarks, compared them with the original versions of the algorithms. Several key observations could be made:

1. The DR-FFL method [134] was inferior to all other TR methods tested which led us to the conclusion that TR should be employed not only between but also within cyclic structures. The winning performance of the original DR-FFL in the DREAM challenge can mainly be attributed to its PG which is in parts similar to the one used by PG^{new}.

2. We found that explicitly accounting for edge signs almost always improves the results in terms of AUPR but only to a very minor extent. While this is in agreement with the observations made in [30], we give here an extended explanation for this unexpected result. Generally, neglecting the edge sign can only be *harmful* during TR, if the true network contains a negative feed-forward loop (FFL). As an example, Figure 6.14 (left) shows a hypothetical interaction graph containing one such negative FFL between

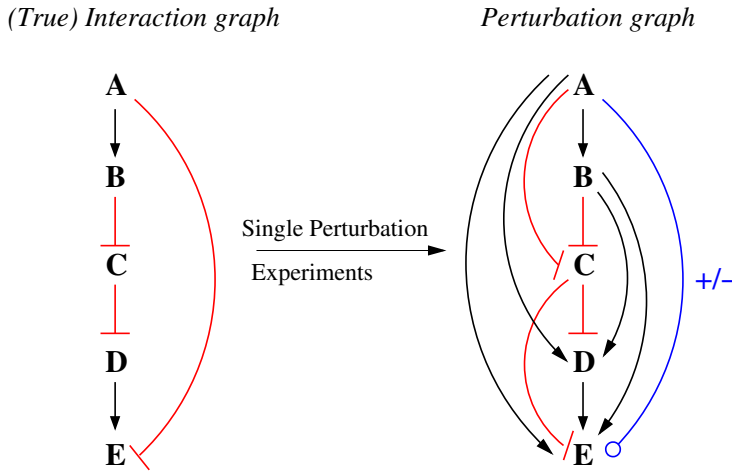

Figure 6.14: **Example of a (true) graph and its (perfect) perturbation graph representing the transitive closure.** An interaction graph (left) and its (expected) perturbation graph which forms the transitive closure of the original graph (right).

node A and E (consisting of a positive path and a negative edge from A to E). If we now assume that all nodes are perturbed in single perturbation experiments we would get, in the ideal case, a PG as shown in Figure 6.14 (right; weights not considered here). This PG corresponds to the *transitive closure* of the original graph, in which each node i induces a significant effect on another node j if there is an edge or path from i to j in the true graph. What we can now see is that each edge contained in the PG but not in the true graph (reflecting thus an indirect effect) is part of a positive FFL consisting of this edge and a path of the same sign. This happens because all these edges will have the same sign as the path they were induced from. Hence, if we compute the TR within the unsigned version of this PG (e.g., by neglecting the signs in the TR step or by setting all signs to "+") all edges that stem from indirect effects and span one branch of the induced FFLs would still correctly be removed.

Regarding the original negative FFL included in the true graph (Figure 6.14, left), we cannot be sure which sign it will get in the PG as there is a positive path as well as a negative edge from A to E and, hence, the direction of change in E when perturbing A cannot be predicted uniquely. Only if the overall effect of A on E measured during perturbation of A is negative (i.e., the true edge of A to E is dominating over the positive path from A to E), it may happen that it will be falsely removed during TR if edge signs are neglected. However, when using edge weights for (weighted) TR, it is rather unlikely that the path from A to E fulfills the rule (6.5) or, for a 2-path used by LTR, rule (6.6) since the measured overall effect from A to E turned out to be negative, hence, the path seems to have a low potential to transduce an effect from A to E. Thus, it is unlikely that the true edge $A \dashv E$ would be falsely removed.

To summarize this aspect, there is a low probability that (weighted) TR removes a true edge within a negative feed-forward loop and neglecting edge signs in TR will therefore have only minor impact on the reconstruction quality. This has important consequences since then the computationally expensive search for the shortest sign-consistent paths (an NP-complete problem) can be safely turned into a simple search

for a shortest (unsigned) path connecting a given pair of nodes (a polynomial problem). Thus, when applying TRANSWESD to the 5000-nodes networks, we may then use an exact (`path_exact` = 1) instead of an approximate (`path_exact` = 0) sub-algorithm for computing shortest signed path. In contrast, `full_check` = 0 is still required for TRANSWESD in large networks.

3. With LTR and TRANSWESD^{s,w,2} we considered local variants of TR removing an edge only if there is an explaining path of length 2. Whereas TRANSWESD^{s,w,2} performed well for the DREAM challenge but unfavorable for the SysGenSIM data, weighted LTR yielded superior performance in almost all benchmark tests and only (signed or unsigned) TRANSWESD applied to *all* paths could deliver comparable results. We can thus first conclude that using the multiplicative rule (6.6) is better suited than the max rule (6.5) when focusing on short paths. However, it remains still paradoxical why TR restricted to paths of length 2 should be sufficient. This can once more be illustrated by Figure 6.14. If we again assume that the true graph induces a complete PG (i.e., the transitive closure of the true graph as shown on the right-hand side of Figure 6.14) then we can indeed recognize that there is always a 2-path that can, in principle, explain an edge from an indirect effect (e.g., edge $A \to E$ is explained by the 2-path $A \to B \to E$). Hence, in principle, all false positive edges could be identified and removed explaining why LTR exhibits good behavior. However, one has to keep in mind that 2-paths may contain edges that are themselves indirect effects (as $B \to E$ in the example above), hence, the order of edge removal might then become crucial. Here, the strategy to cut lowest-confidence-edges first worked apparently well in the benchmarks.

   Again, showing that local TR based on 2-paths does not lead to lower performance has important consequences, as we can then restrict the search on simple triangles whose detection is computationally easier than detecting paths of arbitrary length. In fact, unsigned (signed) LTR required in the average only 8 (9) seconds in networks with 5000 nodes whereas TRANSWESD (in approximation mode!) needed 150 (260) seconds.

4. The SysGenSIM benchmark showed that edge weights really matter to obtain good results with LTR. Since (signed or unsigned) local LTR and unsigned TRANSWESD are computationally feasible in 5000-nodes networks and as they achieved superior results in all benchmarks (outperforming the winning methods of the DREAM4 challenge by far) these techniques appear to be well-suited for the reconstruction of large-scale regulatory networks in the particular case of a fully perturbed experimental framework, i.e. when single-gene knockout experiments are performed for all the genes.

5. Applied to a realistic application scenario with gene expression data from yeast mutants with single knockouts of transcription factors we could demonstrate that our approach delivers a high enrichment of known interactions especially within the top-ranked edge candidates. With this property, our method holds great potential to identify true unknown gene interactions that can subsequently be validated in experiments.

We noticed that LTR shares some similarities with ARACNE presented in [115]. ARACNE also eliminates an edge in a feed-forward loop consisting of three edges (so-called triplets) if a certain weight condition is fulfilled. However, there are several key differences since ARACNE only operates on undirected and unsigned graphs and uses different weights based

on mutual information.

A potential weakness of our PG- and TR-based methods is the requirement to perturb each node in the network at least once. At a genome-scale level, such datasets are (still) available only for a small number of organisms. On the other hand, one might focus on smaller subnetworks where all nodes can be perturbed. Furthermore, if m nodes out of n nodes can be perturbed in a network, we can use the information of the corresponding m perturbation experiments to (i) infer the complete subnetwork containing only the m perturbed nodes and (ii) to infer edges leading from the m perturbed nodes to the $n-m$ unperturbed nodes. In the latter, TR cannot work effectively (no edge will be removed since only single edges and no paths between these nodes exist) meaning that some of the (false positive) edges in the PG reflecting indirect edges cannot be identified as such. However, the provided output might still have its own value and indicate direct or indirect functional relationships. In fact, we employed this approach for the yeast knockout dataset where only TFs were knocked-out.

We also emphasize that perturbation graphs (as a requirement for applying TR) could also be constructed by other approaches than systematic knockouts of all genes. One example are genetical genomics data containing gene expressions measurements from naturally occurring multifactorial perturbations (polymorphisms). As an example for using PG- and TR-based methods based on genetical genomics data see [59].

As an important side results of our study, we have generated new and unprecedented large-scale benchmark datasets that, in contrast to comparable simulations, account for different noise levels. We think that these datasets, which can be downloaded from [15], are generally useful for unbiased testing of network inference methodologies complementing other available in silico benchmarks.

List of abbreviations used

PG - Perturbation Graph
TR - Transitive Reduction
LTR - Local Transitive Reduction
DR-FFL - Down-Ranking of Feed-Forward Loop
FFL - Feed-Forward Loop
TRANSWESD - TRANSitive reduction for WEighted Signed Digraphs
TP - True Positive
FP - False Positive
FN - False Negative

Chapter 7

Inference from heterogeneous datasets

The inference of whole gene networks from a dataset of single-gene knockout experiments is not a viable option yet, due to practical difficulties: real datasets are, in fact, composed by microarray measurements originated from the most diverse experiments and laboratories. In this chapter, we present in Section 7.1 the technique to infer transcriptional regulatory networks from such heterogeneous datasets that we developed for the DREAM5 Network Inference challenge [160], and in Section 7.2 a summary of the ensuing community paper by the DREAM5 organizers is given, where the inference algorithms and the network predictions by all 29 participants to the challenge are deeply analyzed and combined to obtain a robust community prediction [111]. We appear, being part of the so-called DREAM5 consortium, as authors of this community paper.

7.1 Elucidating transcriptional regulatory networks from heterogeneous gene expression compendia

Elucidating the wiring pattern of biomolecular systems continues to be a main challenge in Systems Biology. Many algorithms have been proposed for e.g. transcriptional regulatory network inference, but little is known about their strengths and weaknesses. Validation is of utmost importance and is the main goal of the DREAM project. The DREAM5 Network Inference challenge allows researchers to propose reverse-engineering algorithms to be employed in the stimulating task of inferring transcriptional regulatory networks from microarray datasets.

The DREAM5 Network Inference challenge provided compendia of gene expression profiles from a variety of experiments on several organisms. Here, a method using a combination of different inference approaches applied to different subsets of data is presented. The technique allows for a wise selection of the proper algorithm for the analysis of a particular type of data and underlying experiment.

The proposed approach proved to be reliable in reverse-engineering transcriptional regulatory networks from real expression compendia, ranking in 2nd position when considering the prediction of real networks (sub-challenges 3 and 4) only, and obtaining the best performance in the yeast sub-challenge. This emphasizes the need to account for the experimental design in network inference approaches, instead of blindly applying methods on the entire dataset.

7.1.1 Introduction

This paper describes a thorough analysis of a transcriptional regulatory network inference approach on an internationally recognized benchmark: the DREAM5 Network Inference challenge. The paper first illustrates in detail the above challenge, and then the proposed reverse-engineering algorithm is described. Each step in the approach is thoroughly evaluated. To this aim, in silico data have been produced using a modified version of SysGenSIM [136]. Finally, the results obtained at the challenge are examined.

The DREAM5 Network Inference challenge consists of four sub-challenges, one dedicated to data simulated using GeneNetWeaver [114, 153], and three concerning data of real microorganisms (*Staphylococcus aureus*, *Escherichia coli*, *Saccharomyces cerevisiae*, unveiled after the submission deadline only) and related experimental microarray compendia. Each dataset consists of three files containing the gene expression data, the description of the experimental setup, and the genes acting as transcription factors.

Expression data: a $m \times n$ matrix of gene expression values, with m the number of chips in the dataset, and n the number of genes in the network.

Chip features: the description of the experimental conditions under which the gene expression values of each row have been measured, such as the list of applied drugs or the environmental perturbations, the list of transcription factors that have been deleted or over-expressed, and the measurement time instants in case of time series.

Transcription factors: a list of genes acting as transcription factors.

Through analysis of these datasets, the participants are challenged to provide a list of *transcription factor* → *target* directed interactions sorted according to the assigned confidence, for each network. Predictions will be then scored according to the values of AUC(ROC) and AUC(PvR) [138, 112]. Interactions in *S. aureus* are not evaluated due to the lack of a reliable gold standard of the microorganism in literature. This challenge thus resulted in the first community effort to infer the unknown transcriptional regulatory network of *S. aureus* [111].

7.1.2 Methods

In order to exploit the different types of information included in these mixed datasets, three distinct approaches are applied to subsets of data, and their partial results are then combined into a final prediction.

The first approach is about studying the deviation from the wild-type of the gene expressions after gene-specific perturbations, i.e. knockouts and over-expressions. The second approach considers all the steady state data and calculates partial correlations, while the third

approach investigates the experiments concerning drug perturbations and other perturbations not considered before. Different versions of each of these approaches have been tested, in order to identify those providing the best predictions on a suite of training data, simulated by mimicking the chip features of the challenge compendia. The selection of the individual approaches is followed by the search for an optimal combination of the three approaches to obtain the final prediction of the network.

Approach 1: perturbation-response analysis

We identified experiments in which wild-type and knockout and/or over-expression of transcription factors are measured. We used steady state data, but also the time series data from which we considered the first (unperturbed) and last (perturbed) time-point. To identify the potential targets T_j of perturbed TF_i, we calculated for all target genes (including other TFs) the normalized deviation (superscript indicates the perturbation):

$$R_{\mathrm{TF}_i \to \mathrm{T}_j} = \frac{\mathrm{T}_j^{\mathrm{TF}_i} - \mathrm{T}_j^{\mathrm{WT}}}{\mathrm{T}_j^{\mathrm{WT}}} \tag{7.1}$$

In addition we employed double knockouts and/or over-expressions: to gain more confidence in $\mathrm{TF}_i \to \mathrm{T}_j$ we calculated (when possible):

$$R_{\mathrm{TF}_i \to \mathrm{T}_j} = \frac{\mathrm{T}_j^{\mathrm{TF}_i,\mathrm{TF}_k} - \mathrm{T}_j^{\mathrm{WT}}}{\mathrm{T}_j^{\mathrm{WT}}} - \frac{\mathrm{T}_j^{\mathrm{TF}_k} - \mathrm{T}_j^{\mathrm{WT}}}{\mathrm{T}_j^{\mathrm{WT}}} \tag{7.2}$$

The double knockout and/or over-expression can have extra information for $\mathrm{TF}_i \to \mathrm{T}_j$ however the response to the double perturbation might be explained by the effect of TF_k, so this must be subtracted. In some cases TF_k was not perturbed and then we simply used:

$$R_{\mathrm{TF}_i \to \mathrm{T}_j} = \frac{\mathrm{T}_j^{\mathrm{TF}_i,\mathrm{TF}_k} - \mathrm{T}_j^{\mathrm{WT}}}{\mathrm{T}_j^{\mathrm{WT}}} \tag{7.3}$$

Large values might be due to the TF_k perturbation, but we accept making some mistakes as a trade-off for also identifying real targets of TF_i. In case of triple knockouts, we proceeded in a similar way. Finally, we define the score S^1_{ij} as the confidence in $\mathrm{TF}_i \to \mathrm{T}_j$, obtained by averaging the $R_{\mathrm{TF}_i \to \mathrm{T}_j}$ values from single, double and triple knockout and/or over-expression experiments.

Approach 2: full order partial correlation analysis of steady state data

The second approach processes the information provided by the steady state experiments. All the time series chips are removed by the studied dataset to obtain a matrix $\mathbf{G}^{\mathrm{ss}}$ of steady state gene expression measurements only. Then, full order partial correlation [152], computed through the GeneNet R package [151] (available for download at [7]), is applied to matrix $\mathbf{G}^{\mathrm{ss}}$.

The score S^2_{ij} (confidence in the interaction between transcription factors i and target j) is then the absolute value of the full order partial correlation $\omega_{\mathrm{TF}_i,\mathrm{T}_j}$. The full order partial correlation approach was selected after evaluating a variety of approaches, such as plain Pearson correlation, first order partial correlation [45], the CLR algorithm [57] and several scaled versions (see Table 7.2), because of its superior performance.

Approach 3: co-deviation analysis of aspecific perturbation data

The third approach is applied to the matrix containing the gene expression of chips featuring drug perturbation experiments or *non-TF single gene perturbations*. The goal is to check whether a target T_j is consistently subject to large deviations in expression when transcription factor TF_i makes large deviations too. The algorithm underlying the approach follows.

1. Gene expression values are converted into a z-scores matrix $\mathbf{Z}$.
2. Then, for each transcription factor i:
   a) The data are split into two subsets according to a threshold d: one group of observations $\mathscr{D}_i^H$ with $Z_i > d$ (TF_i is *high*) and another group of observations $\mathscr{D}_i^L$ with $Z_i < -d$ (TF_i is *low*).
   b) The potential targets T_j of TF_i are identified by performing for each T_j a two-sided t-test to check whether its mean in $\mathscr{D}_i^H$ is significantly different from its mean in $\mathscr{D}_i^L$.
3. The confidence in each $TF_i \rightarrow T_j$ interaction S^3_{ij} is then the absolute value of the t-statistic t_{ij} (test performed for T_j when datasets are formed based on deviation of TF_i).

Poorer performances were obtained when analyzing the same dataset with different techniques, such as ranking based on p-value of t-test, Pearson correlation, and different values of the threshold d (see Table 7.3).

Combining the approaches

Each of the three approaches independently scores the interactions between transcription factors and targets. Individual predictions are combined into a final sorted list of interactions by means of a simple weighted sum of the scores. In particular, when approach 1 is applicable (i.e. if at least one experiment where the only perturbation is the knockout and/or over-expression of TF_i is available), then the overall score of the interaction $TF_i \rightarrow T_j$ is:

$$S_{ij} = a \cdot S^1_{ij} + b \cdot S^2_{ij} + (1 - a - b) \cdot S^3_{ij} \tag{7.4}$$

Otherwise:

$$S_{ij} = c \cdot S^2_{ij} + (1 - c) \cdot S^3_{ij} \tag{7.5}$$

The values of the weights a, b, and c are obtained through an optimization process aimed to maximize the AUC(ROC) and AUC(PvR) values computed by inferring the networks from the simulated datasets.

Simulation of synthetic datasets

To *train* our approach, we evaluated it on a large set of simulated data generated with a modified version of the SysGenSIM software [136]. Gene expression data are simulated with the aim of reproducing the experiments described in the *chip features* files provided by the DREAM5 organizers. Artificial gene networks were of the same sizes as used in the challenge. An accurate simulation of the experiments might allow for an effective testing and selection of the inference algorithms. Several typologies of experiments are proposed in the four compendia, and are described below.

Gene knockout: one or more genes (transcription factors and/or targets) are deleted.

Gene over-expression: one or more genes are up-regulated.

Perturbation: a drug or an environmental disturbance is applied to the experiment.

Steady states: observations are measured when the gene activity is deemed constant.

Time series: observations are taken when the gene expression is still varying from the initial wild-type due to any perturbation.

Most of the experiments consists of combinations of more than one of the previous possibilities, i.e. a microarray experiment could comprise the knockout of two genes and the over-expression of a third one, and at the same time undergo a drug perturbation.

Before explaining the simulation process, a clear definition of the terms *chip*, *experiment*, *sub-experiment* and *repeat* is given.

Chip: contains the microarray measurements from a single experiment, i.e. a row of n gene expression values.

Experiment: comprises all the chips from the same experiment, i.e. the chips undergoing the same experimental settings as defined by the authors of the challenge.

Sub-experiment: is defined for time series only. It indicates all the chips sharing established characteristics and belonging to the same experiment (i.e., an experiment may contain two time series sub-experiments: one is a wild-type time series, the other is a perturbed time series).

Repeat: is defined for time series only. It indicates the measurements belonging to the same repeat in a sub-experiment (a sub-experiment, in fact, may contain two or more repeat time series, i.e. time series with the same exact characteristics; the only distinct parameter among repeats is δ, as it will be afterwards explained). A repeat corresponds to a time series, and each of them needs to be simulated independently.

For each chip, gene expression values are simulated by solving the system of n ordinary differential equations, with n the number of genes in the network:

$$\frac{\mathrm{d}G_i}{\mathrm{d}t} = \zeta_i \tau_i \delta_i^{\mathrm{syn}} \vartheta_i^{\mathrm{syn}} V_i \prod_{j \in \mathscr{R}_i} \left(1 + A_{ji} \frac{G_j^{h_{ji}}}{G_j^{h_{ji}} + (K_{ji}/\pi_j)^{h_{ji}}} \right) - \delta_i^{\mathrm{deg}} \vartheta_i^{\mathrm{deg}} d_i G_i \tag{7.6}$$

Some parameters are common to all the experiments featured in a single dataset, other are shared by chips belonging to the same experiment, and other characterize the single steady state chip or the chips related to the same time series. The following paragraphs explain how the simulation of the different microarrays has been set.

Dataset parameters: the indexes of all regulators (both activators and inhibitors) j of gene i are contained in the set $\mathscr{R}_i$; G_i is the mRNA concentration (gene activity) of gene i, V_i is its basal transcription rate, while d_i is its degradation rate constant. K_{ji} is the interaction strength of G_j on G_i, h_{ji} is the Hill cooperativity coefficient, and A_{ji} is an

element of the matrix **A** encoding the signed network structure. For simplicity, parameters V_i, K_{ji}, and d_i were kept constant, and set equal to 1. Cooperativity coefficients h_{ji} are set to 1, 2, or 4 with probabilities 60%, 30%, and 10%, respectively. The elements of the adjacency matrix are $A_{ji} = 1$ when gene j is an activator for gene i, or are $A_{ji} = -1$ when gene j is an inhibitor for gene i; $A_{ji} = 0$ otherwise, i.e. when a direct regulation from gene j to gene i does not exist. The above parameters are set once, and are used to simulate all the chips of the current dataset.

Experiment parameters: biological variances ϑ^{syn} and ϑ^{deg} are both sampled from the Gaussian distribution $N(1, 0.1)$, and are constant for all the chips belonging to the same experiment.

Chip parameters: each single chip differs from the others by parameters ζ, π, δ^{syn}, δ^{deg} and τ. Parameter ζ is different from 1 in case of knockout and/or over-expression (i.e., $\zeta_i = 0$ if gene i is knocked-out, and $\zeta_j = 5$ if gene j is over-expressed), while π enables for drug perturbations of about 10% of targets, and is sampled from $N(1, 0.4)$; δ^{syn} and δ^{deg}, sampled from $N(1, 0.025)$, represent stochasticity in both transcription and degradation rates; τ, sampled from $N(1, 0.1)$, gives variability in time series simulations. Then, measurement noise ϵ is added to the gene expression values.

7.1.3 Results

The section displays the performance of the proposed inference technique in the DREAM5 Network Inference challenge, showing in particular how the goodness of the predictions increases with the combination of the approaches. Moreover, details are provided about the selection of the inference techniques with the assistance of the simulated datasets. The predictive performance of the inference algorithms is evaluated by computing the Area Under the Receiver Operating Characteristic Curve and the Area Under the Precision versus Recall Curve of the networks, respectively shortened to AUC(ROC) and AUC(PvR), as already done in previous DREAM competitions.

Performance in DREAM5 Network Inference challenge

The here described methodology showed to be performing well in the inference of transcriptional regulatory networks from real expression compendia, ranking 2nd overall in the *real networks* sub-challenge (networks 3 and 4), and 1st overall with respect to the prediction of the *S. cerevisiae* network. Particularly relevant is the improvement of the AUC scores due to the sequential combination of the approaches, as shown in Figure 7.1, where the score by approach 2 alone is compared with the scores by the combination of approaches 2 and 3, and of approaches 1, 2 and 3. Such results confirm that the combination of different inference techniques, especially in the analysis of heterogeneous compendia, is indeed effective, provided a correct subdivision of the data.

Practice on Synthetic Data

The simulated data are a collection of 40 datasets corresponding to the simulation of as many networks. For $i \in \{1, 2, 3, 4\}$, 10 networks have been generated with the size of chal-

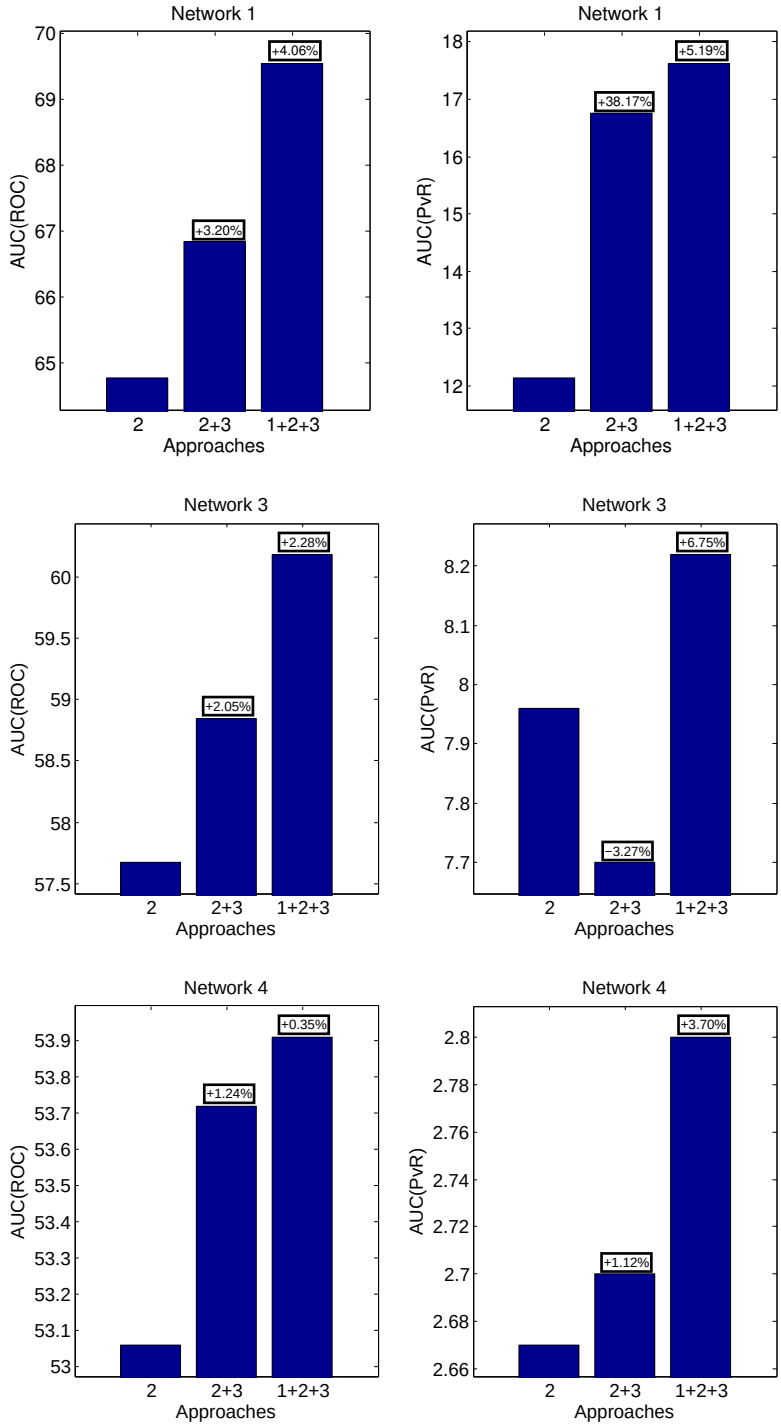

Figure 7.1: **Increase of the AUC scores in DREAM5 networks.** The AUC(ROC) and AUC(PvR) scored by approach 2 are compared with those obtained by combining approaches 2 and 3, and approaches 1, 2 and 3, respectively for the in silico (top), *E. coli* (center) and *S. cerevisiae* (bottom) networks.

Table 7.1: **Comparison of the inference techniques on DREAM5 networks for approach 1.** Normalized deviations (1.A) strongly outperforms non-normalized deviations (1.B). Network 2 does not have any suitable chip to be included in the subset.

	Network 1		Network 3		Network 4	
Approach	AUC(ROC)	AUC(PvR)	AUC(ROC)	AUC(PvR)	AUC(ROC)	AUC(PvR)
1.A	89.05	8.52	91.62	6.99	80.15	13.58
1.B	70.74	2.56	70.65	1.69	61.35	2.95

Table 7.2: **Comparison of the inference techniques on DREAM5 networks for approach 2.** Technique 2.CLRC yields a random prediction. Other algorithms have similar decent performances, while full order partial correlation (2.FPC) allows for a more accurate prediction of the interactions.

	Network 1		Network 2		Network 3		Network 4	
Approach	AUC(ROC)	AUC(PvR)	AUC(ROC)	AUC(PvR)	AUC(ROC)	AUC(PvR)	AUC(ROC)	AUC(PvR)
2.FPC	78.97	11.02	74.04	2.50	80.72	5.62	79.01	3.58
2.C	76.03	7.75	65.64	1.17	74.23	2.81	71.70	1.53
2.ZC	76.09	8.87	65.69	1.16	74.36	3.38	71.74	1.62
2.CLRC	50.01	1.07	47.80	0.39	49.74	0.36	49.03	0.23
2.CLRZ	75.16	8.02	64.57	1.11	73.33	3.19	70.82	1.52
2.PC	72.51	7.63	64.33	1.15	71.43	3.74	69.20	1.81
2.ZPC	65.85	7.54	54.83	1.05	63.64	3.32	59.9	1.56

lenge network i, and experiments have been simulated according to the *chip features* file of compendium i and to the dynamical model (7.6).

Regarding the first approach, the performances of the deviations normalized by the wild-type (1.A) proved to be arguably better than the deviations with no normalization (1.B), as established in Table 7.1. The inference methodologies compared for the selection of the second approach are applied to a subset of chips corresponding to steady state experiments. The full order partial correlation (2.FPC) performed extremely good compared to Pearson correlation (2.C), Pearson first order partial correlation (2.PC), z-score computed on Pearson correlation (2.ZC) and first order partial correlation (2.ZPC), and Context Likelihood of Relatedness (CLR) applied to Pearson correlation (2.CLRC) and to z-score computed on Pearson correlation (2.CLRZ). Scores are shown in Table 7.2. The third approach (Table 7.3) considers chips with perturbations triggered by drugs or due to the knockout or over-expression of non-TF genes. T-test statistic (3.TT) provides more precise predictions than p-value (3.PV) and Pearson correlation (3.C). After the approaches have been individually selected, the synthetic datasets allowed for an estimate of the weights for combining the approaches by means of an optimization process. By maximizing the AUC(ROC) and AUC(PvR), the values of the weights and have been set to $a = 0.4$ and $b = 0.5$ in (7.4), and $c = 0.8$ in (7.5).

Here, we presented a method for transcriptional regulatory network inference from gene expression compendia, based on dividing the compendia in similar datasets and analyzing these datasets individually with distinct approaches, and combining their results into a final prediction. The idea behind this method is that one has to account for the experimental setup when analyzing data, to extract as much useful information as possible. Different sta-

Table 7.3: **Comparison of the inference techniques on DREAM5 networks for approach 3.** The presented methodologies are compared according to the employed threshold d. For large thresholds ($d > 1$) predictions become random guesses. T-test gives more precise predictions than p-value and Pearson correlation.

		Network 1		Network 2		Network 3		Network 4	
Approach	d	AUC(ROC)	AUC(PvR)	AUC(ROC)	AUC(PvR)	AUC(ROC)	AUC(PvR)	AUC(ROC)	AUC(PvR)
3.TT		66.93	2.85	59.78	0.56	67.29	1.36	61.26	0.35
3.PV	0.5	66.89	1.46	59.43	0.50	66.81	0.52	60.56	0.25
3.C		67.79	1.83	59.42	0.49	66.40	0.61	61.09	0.29
3.TT		63.20	2.26	56.08	0.46	63.27	0.98	58.32	0.27
3.PV	1.0	46.82	0.78	55.97	0.37	62.08	0.29	57.30	0.19
3.C		64.87	1.23	56.52	0.40	63.27	0.41	58.35	0.21
3.TT		53.65	1.32	50.24	0.45	54.32	0.57	51.99	0.24
3.PV	1.5	51.19	0.53	49.66	0.31	50.34	0.20	50.90	0.15
3.C		58.07	0.81	53.70	0.37	56.73	0.28	53.86	0.18

tistical techniques require different assumptions. When applying a particular technique, one must try to consider data for which these assumptions are, at least approximately, valid. Indeed, we have shown that our approach based on this idea was successful, as evidence by its performance in the DREAM5 Network Inference challenge.

7.2 Wisdom of crowds for robust gene network inference

The DREAM5 community paper [111] is a high quality work resulted from the effort by the organizers of the Network Inference challenge, which analyzed and compared the performances of several network identification techniques, developed a community methodology to infer real networks, and validated some of these interactions. Here, we merely report the abstract of the paper and an interesting composition showing the performance of the submitted techniques compared with the community approach (see Figure 7.2).

Abstract

Reconstructing gene regulatory networks from high-throughput data is a long-standing challenge. Through the Dialogue on Reverse Engineering Assessment and Methods (DREAM) project, we performed a comprehensive blind assessment of over 30 network inference methods on *Escherichia coli*, *Staphylococcus aureus*, *Saccharomyces cerevisiae* and in silico microarray data. We characterize the performance, data requirements and inherent biases of different inference approaches, and we provide guidelines for algorithm application and development. We observed that no single inference method performs optimally across all datasets. In contrast, integration of predictions from multiple inference methods shows robust and high performance across diverse data sets. We thereby constructed high-confidence networks for *E. coli* and *S. aureus*, each comprising $\simeq$ 1700 transcriptional interactions at a precision of $\simeq$ 50%. We experimentally tested 53 previously unobserved regulatory interac-

tions in *E. coli*, of which 23 (43%) were supported. Our results establish community-based methods as a powerful and robust tool for the inference of transcriptional gene regulatory networks.

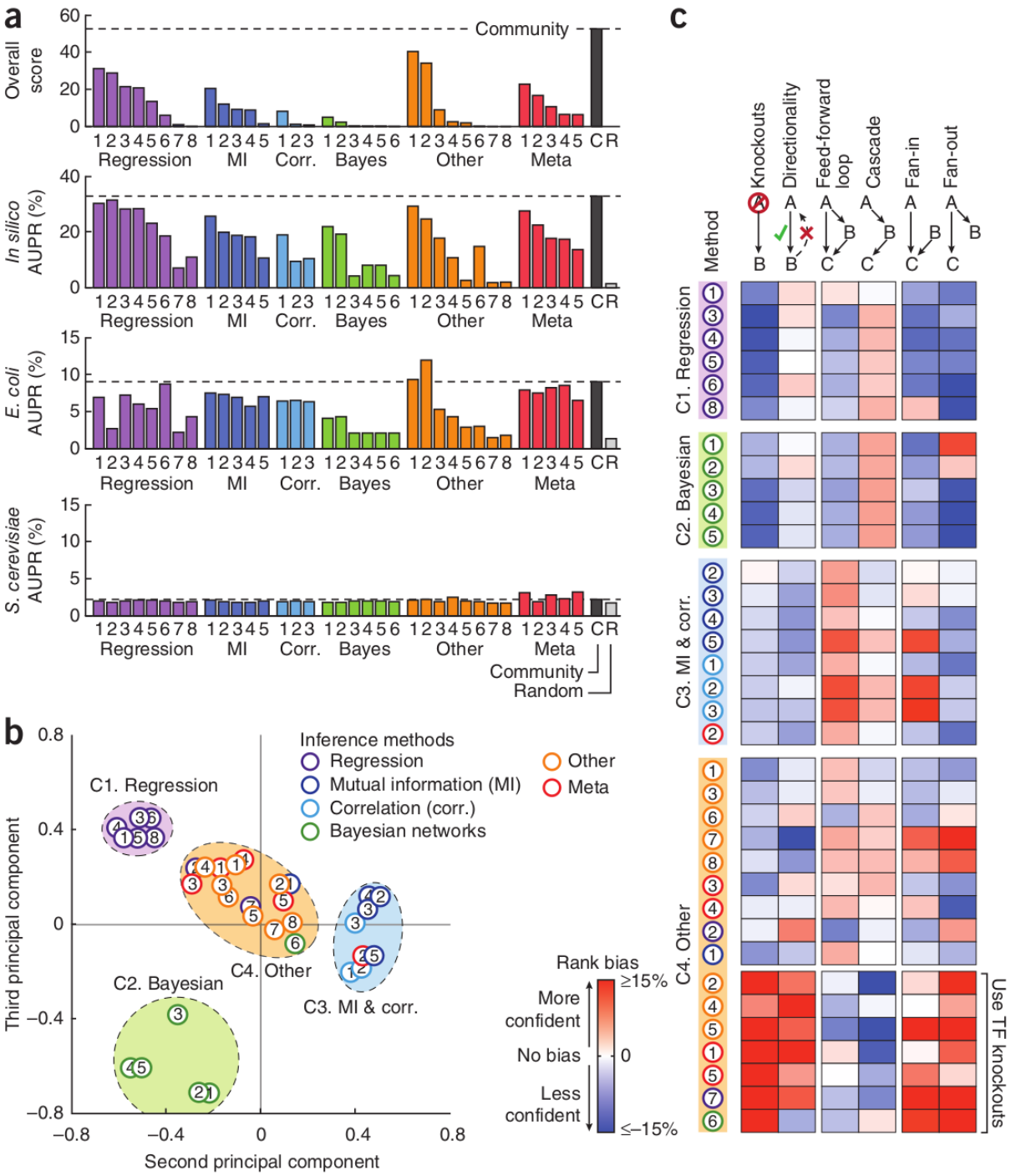

Figure 7.2: **Evaluation of the DREAM5 Network Inference methods.** Inference methods are indexed according to Table 1 in [111]. The technique we presented in Section 7.1 is method 3 amongst the Meta approaches. **(a)** The plots depict the performance for the individual networks (area under precision-recall curve, AUPR) and the overall score summarizing the performance across networks. R, random predictions; C, integrated community predictions. **(b)** Methods are grouped according to the similarity of their predictions via principal-component analysis. The second versus third principal components are shown; the first principal component accounts mainly for the overall performance (see Supplementary Note 4 in [111]). **(c)** The heat map depicts method-specific biases in predicting network motifs. Rows represent individual methods and columns represent different types of regulatory motifs. Red and blue show interactions that are easier and harder to detect, respectively.

Chapter 8

Inference from systems genetics datasets

This chapter describes RAGNO [133], i.e. a Reconstruction Algorithm for Gene Networks from multi-Omics data; in this particular case, the -omics in the acronym[1] actually corresponds to genotype and gene expression data measurements. After statistically analyzing the data, the methodology yields a sorted list of edges, i.e. it ranks all the possible edges of the gene regulatory networks according to the computed confidence. Scoring the possible edges of a gene network allows for identifying with a higher degree of certainty which are the most probable existing regulatory interactions between genes. The identification technique performs quite well on established systems genetics synthetic datasets [15], already employed as benchmarks in the DREAM5 Systems Genetics challenge [2] and in the book [44] to evaluate and compare several inference algorithms. Its application on a real genotyping and expression yeast dataset yields some valuable insights and might suggest connections between genes not yet investigated experimentally.

The hereafter introduced inference methodology starts by providing an accurate initial confidence score for all the possible edges in the gene network. Then, according to the value of basic statistical measures, a subset of edges is selected in order to build the correlation graph, which is finally reduced by pruning the edges whose effect between genes is identified as indirect. As shown in the following, the output network contains a large fraction of correctly predicted gene connections amongst the first hundreds of edges, thus proving the goodness of the algorithm.

8.1 Methods

Identifying the directed structure of a gene regulatory network from a set of observational data is a challenging task, which could be however assisted by the introduction of genetic markers into the inference process. In the considered datasets, in fact, the expression of a gene Y_i is directly linked to the value of genotype X_i, while e.g. the expression Y_j is highly correlated with X_j. If a connection exists between genes i and j, then the correlation be-

[1] *Ragno* is the Italian word for *spider*.

tween the respective mRNA activities is expected to be large[2]. Unfortunately, this does not allow to assign a direction to the edge: is gene i regulating gene j, or vice versa? An answer to a such dilemma might be obtained by computing the correlation between the genotype values and the expression measurements: if the association between the genetic marker of gene i and the expression of gene j is large, then it is highly probable that gene i regulates gene j; vice versa, if the correlation between the genotype of gene j and the activity of gene i is large, then the edge $j \rightarrow i$ might actually exist.

The presented technique avails itself of the above reasoning as its starting point. Indeed, the complete algorithm consists of four steps, namely (i) the initial sorting of all possible edges by computing correlation matrices subject to z-score scaling, (ii) the construction of the correlation graph $\mathscr{G}^C$ containing the most relevant edges, (iii) the transitive reduction of $\mathscr{G}^C$ into $\mathscr{G}^T$ to remove unessential connections, and (iv) the production of a sorted list of edges.

The datasets required to run the algorithm are simply the genotype matrix $\mathbf{X}$ and the associated gene expression matrix $\mathbf{Y}$, whose sizes are both $m \times n$, being m the number of individuals in the population and n the number of genes in the network. Therefore, the generic row i of the matrices contains the genotype values or expression measurements for the n genes of individual i, and the column j stores the genotype or the expression values of gene j for all the m individuals of the population. The particular datasets employed in this analysis consists of simulated genotype and gene activity for a population of haploid individuals, and therefore the genotype matrix $\mathbf{X}$ contains only the genotype markers $X_{i,j} = \{0,1\}$. The four steps of the algorithm are thoroughly described below.

Step 1: initial sorting of all possible edges. Pearson correlation is computed between the columns of $\mathbf{Y}$ to obtain the $n \times n$ matrix $\mathbf{C}_{yy}$, whose entries $C_{yy}(i,j)$ represent the correlation amongst the expression values of genes i and j. By computing the same Pearson correlation between matrices $\mathbf{X}$ and $\mathbf{Y}$ we obtained the cross correlation between genotype and gene expression values $\mathbf{C}_{xy}$. The entry (i,j) of this matrix represents the correlation between the genotype values of gene i and the gene expression values of gene j.

We computed the z-score on the columns of these matrices (after removing their main diagonal), i.e. gene-wise, to obtain $\mathbf{Z}_{yy}$ as the z-score matrix of $\mathbf{C}_{yy}$, and $\mathbf{Z}_{xy}$ as the z-score matrix of $\mathbf{C}_{xy}$. In particular, an entry $Z(i,j)$ is the z-score value of the i-th entry of the j-th column of $\mathbf{C}$.

By adding together the absolute values of the above two matrices, we obtain a new matrix $\mathbf{R}$ containing the confidence score for each possible edge of the network: $\mathbf{R} = |\mathbf{Z}_{yy}| + |\mathbf{Z}_{xy}|$. This allows for a better initial sorting of the edges, and a fair assignment of confidence values. The choice of $\mathbf{Z}_{yy}$ with respect to $\mathbf{C}_{yy}$ is due to the better scoring of edges, that might be explained by the EIPO topology – which in turn attempts to replicate topologies observed in real gene networks – used to generate the gene networks in SysGenSIM. Coefficients of $\mathbf{Z}_{yy}$ are computed column-wise on $\mathbf{C}_{yy}$ because correlations on columns are more significant

[2]The correlation between the expression measurements of two genes might be large even if the genes are not directly linked, e.g. when the genes have a common regulator.

than correlations on rows, i.e. in general the number of genes with significant correlation coefficients $C_{yy}(i,j)$ amongst their input edges is larger than the number of genes with significant correlation coefficients $C_{yy}(i,j)$ in their output edges. On average, columns of $\mathbf{C}_{yy}$ have correlation coefficients more significant than rows, i.e. the few input edges stand out more than the (larger number of) output edges amongst the n correlation coefficient entries.

Step 2: construction of correlation graph. The correlation graph $\mathscr{G}^C$ is built by selecting the edges (i,j) – amongst all possible edges of the network – that satisfy two conditions: (i) the absolute value of gene-gene correlation is larger than a certain value, i.e. $|\mathbf{C}_{yy}(i,j)| > \vartheta$; (ii) the value of genotype-gene expression correlation is larger than the correlation of the symmetric edge plus a threshold, i.e. $\mathbf{C}_{xy}(i,j) > \mathbf{C}_{xy}(j,i) + \sigma$ (for positive edges) or $\mathbf{C}_{xy}(i,j) < \mathbf{C}_{xy}(j,i) - \sigma$ (for negative edges). This allows for differentiating between positive and negative edges of the correlation graph. Both threshold parameters σ and ϑ are chosen between 0 and 1.

Step 3: transitive reduction. The correlation graph undergoes a transitive reduction process that removes unnecessary edges, i.e. the edges that are not necessary to explain regulation or inhibition between nodes, given the existence of other paths that explain such dependencies. By applying the local transitive reduction (LTR) approach recently presented in [132], a further refinement of the gene network might be obtained by down-ranking the edges in $\mathscr{G}^C$ but not in $\mathscr{G}^T$. In particular, the signed and weighted configuration of LTR can definitely improve the AUPR scores computed after Steps 1 and 2, as shown in Section 8.2.

Step 4: edge sorting. This step simply provides an edge list by ranking the edges according (i) to their *group* (first the edges in $\mathscr{G}^T$, then the remaining connections not included in $\mathscr{G}^T$) and, (ii) internally to each group, to their confidence score $\mathbf{R}$ computed in Step 1. This last step is optional, but it is needed to compute e.g. the AUROC and AUPR scores. Else, $\mathscr{G}^T$ itself might be considered as a reasonable estimate of the gene network.

The RAGNO algorithm to infer a gene network from genotype and gene expression data is detailed as follows:

1. Define the input datasets:
   - Define $\mathbf{X}$ as the $m \times n$ matrix containing the genotype values of the dataset. Entry $X(i,j)$ contains the genotype value for gene j in individual i.
   - Define $\mathbf{Y}$ as the $m \times n$ matrix containing the gene expression values of the dataset. Entry $Y(i,j)$ contains the gene expression value measured for gene j in individual i.
2. Compute the Pearson correlation matrices:
   - Compute the $n \times n$ matrix $\mathbf{C}_{yy}$ by calculating Pearson correlation between columns of $\mathbf{Y}$. Entry $C_{yy}(i,j) = C_{yy}(j,i)$ is the Pearson correlation coefficient computed between columns i and j of $\mathbf{Y}$. The main diagonal is set to zero.
   - Compute the $n \times n$ matrix $\mathbf{C}_{xy}$ by calculating Pearson correlation between columns of $\mathbf{X}$ and $\mathbf{Y}$. Entry $C_{xy}(i,j)$ is the Pearson correlation coefficient computed between column i of $\mathbf{X}$ and column j of $\mathbf{Y}$. The main diagonal is set to zero.

3. Compute the z-score matrices:
   - Compute the $n \times n$ matrix $\mathbf{Z}_{yy}$ by calculating the z-score on matrix $\mathbf{C}_{yy}$. Entry $Z_{yy}(i,j)$ is then the deviation of $C_{yy}(i,j)$ from the mean of column j of $\mathbf{C}_{yy}$.
   - Compute the $n \times n$ matrix $\mathbf{Z}_{xy}$ by calculating the z-score on matrix $\mathbf{C}_{xy}$. Entry $Z_{xy}(i,j)$ is then the deviation of $C_{xy}(i,j)$ from the mean of column j of $\mathbf{C}_{xy}$.
4. Compute the $n \times n$ confidence matrix $\mathbf{R} = |\mathbf{Z}_{yy}| + |\mathbf{Z}_{xy}|$.
5. Build the correlation graph:
   - Select the edges (i,j) whose gene-gene correlation is larger than a certain threshold ϑ, and store them in set $\mathscr{E}_t$: $|C_{yy}(i,j)| > \vartheta$.
   - Select the edges (i,j) for which their cross-correlation is significantly larger than that of the opposite edge (j,i) by a threshold σ, and store them in set $\mathscr{E}_s$: $|C_{xy}(i,j)| > |C_{xy}(j,i)| + \sigma$.
   - Define the correlation graph as $\mathscr{G}^C = \mathscr{E}_t \cap \mathscr{E}_t$.
   - Define the set of positive edges of the correlation graph $\mathscr{G}^C$ as those associated with a positive correlation: $C_{yy}(i,j) > 0$.
   - Define the set of negative edges of the correlation graph $\mathscr{G}^C$ as those associated with a negative correlation: $C_{yy}(i,j) < 0$.
6. Perform transitive reduction on the correlation graph $\mathscr{G}^C$ according to the LTR algorithm, using the normalized matrix $\mathbf{R}$ as weight matrix and α as threshold in case of inference with a weighted variant of the technique. The obtained reduced graph is $\mathscr{G}^T$.
7. Generate the output list by inserting first the edges in $\mathscr{G}^T$ sorted according to their weight in matrix $\mathbf{R}$, and then the other edges still according to their weight in $\mathbf{R}$.

8.2 Results

The inference technique has been successfully applied to the simulated datasets of the DREAM5 Systems Genetics A challenge [2] and to the recently published StatSeq benchmarks [135]. Both datasets have been produced with SysGenSIM [136], and are available for download at [15]. Moreover, the identification of transcriptional interactions in yeast from a real dataset is verified against a silver standard network.

8.2.1 Performance at the DREAM5 Systems Genetics challenge

One of the challenges proposed at the fifth edition of the well-known DREAM project involved the inference of gene networks from (real and synthetic) systems genetics data. The in silico sub-challenge requested to reverse-engineer 15 networks, all of size $n = 1000$ but with a different number of available individuals ($p = \{100, 300, 999\}$) and with increasing node average degree.

The scores obtained by inferring with RAGNO the networks from the DREAM5 Systems Genetics A datasets are quite large, as shown in Tables 8.1, 8.2 and 8.3, and the performance

Table 8.1: **Performance of the RAGNO techniques in DREAM5 sub-challenge A1 (100 RILs).**

Network	Method	AUROC	AUPR	TPs	FPs	ToP1000
100-1	$\mathbf{R}$	0.838	0.188	-	-	396
	$\mathbf{R}+\mathcal{G}^C$	0.838	0.216	530	1741	466
	$\mathbf{R}+\mathcal{G}^T$	0.838	0.207	494	1666	438
	Vignes (DREAM5)	0.754	0.085	-	-	-
	Vignes *et al.*	0.750	0.074	-	-	-
	Flassig *et al.*	0.843	0.247	-	-	-
100-2	$\mathbf{R}$	0.816	0.154	-	-	418
	$\mathbf{R}+\mathcal{G}^C$	0.816	0.181	666	2078	491
	$\mathbf{R}+\mathcal{G}^T$	0.816	0.178	630	1976	484
	Vignes (DREAM5)	0.718	0.060	-	-	-
	Vignes *et al.*	0.713	0.054	-	-	-
	Flassig *et al.*	0.821	0.203	-	-	-
100-3	$\mathbf{R}$	0.794	0.148	-	-	456
	$\mathbf{R}+\mathcal{G}^C$	0.794	0.171	762	2402	534
	$\mathbf{R}+\mathcal{G}^T$	0.794	0.167	716	2274	524
	Vignes (DREAM5)	0.696	0.053	-	-	-
	Vignes *et al.*	0.694	0.045	-	-	-
	Flassig *et al.*	0.802	0.186	-	-	-
100-4	$\mathbf{R}$	0.781	0.135	-	-	473
	$\mathbf{R}+\mathcal{G}^C$	0.781	0.146	764	2477	524
	$\mathbf{R}+\mathcal{G}^T$	0.781	0.143	707	2320	507
	Vignes (DREAM5)	0.676	0.054	-	-	-
	Vignes *et al.*	0.671	0.046	-	-	-
	Flassig *et al.*	0.788	0.158	-	-	-
100-5	$\mathbf{R}$	0.774	0.143	-	-	527
	$\mathbf{R}+\mathcal{G}^C$	0.775	0.151	880	2610	583
	$\mathbf{R}+\mathcal{G}^T$	0.774	0.146	808	2458	559
	Vignes (DREAM5)	0.670	0.054	-	-	-
	Vignes *et al.*	0.666	0.044	-	-	-
	Flassig *et al.*	0.781	0.158	-	-	-

would have been awarded with the 1st place due to the *infinite* overall score obtained for the three sub-challenges (explained by the *infinite* values computed for the AUROC scores; anyway, the AUPR scores are substantially larger than those by the challenge participants: 161.8, 160.7 and 192.4 instead of 81.9, 89.4 and 140.6 by the best performer [172] for A1, A2 and A3, respectively). The tables show, for each of the 15 networks, the AUROC and AUPR scores for the three steps of the algorithm, the number of true positive (TP) and false positive (FP) edges in the correlation graph $\mathcal{G}^C$ and in the transitive reduced graph $\mathcal{G}^T$, and the number of true positive edges amongst the best 1000 predicted edges (column ToP1000, sorry for the wordplay). The threshold parameters are set to $\sigma = 0.15$, $\vartheta = 0.25$, $\alpha = 0.6$ to infer all networks, i.e. the selection of parameters has not been optimized to maximize the score for the single networks.

Besides the rows for methods $\mathbf{R}$, $\mathbf{R}+\mathcal{G}^C$ and $\mathbf{R}+\mathcal{G}^T$, two more contain the AUROC and AUPR values scored by Vignes and colleagues: rows labeled with Vignes (DREAM5) show the performance officially registered at the DREAM5 challenge, while those labeled with Vignes *et al.* show the scores published in [172] after minor corrections in the implementation of the inference algorithm. Another row exalts the performance of the technique developed by Flassig *et al.* [59], that is by far the most accurate in reverse-engineering the networks, excelling in particular with the availability of RILs. The several combinations of inference techniques presented in [18] perform better than the DREAM5 winner, but it is not clear whether these algorithms would top the performance of RAGNO (the values of AUROC and AUPR are not shown numerically, only in form of comparative bar plots).

Table 8.2: **Performance of the RAGNO techniques in DREAM5 sub-challenge A2 (300 RILs).**

Network	Method	AUROC	AUPR	TPs	FPs	ToP1000
300-1	$\mathbf{R}$	0.920	0.305	-	-	526
	$\mathbf{R}+\mathscr{G}^C$	0.921	0.365	670	417	654
	$\mathbf{R}+\mathscr{G}^T$	0.921	0.368	660	358	657
	Vignes (DREAM5)	0.855	0.211	-	-	-
	Vignes *et al.*	0.845	0.248	-	-	-
	Flassig *et al.*	0.927	0.475	-	-	-
300-2	$\mathbf{R}$	0.882	0.220	-	-	519
	$\mathbf{R}+\mathscr{G}^C$	0.882	0.270	699	425	663
	$\mathbf{R}+\mathscr{G}^T$	0.882	0.271	680	402	661
	Vignes (DREAM5)	0.793	0.144	-	-	-
	Vignes *et al.*	0.779	0.175	-	-	-
	Flassig *et al.*	0.892	0.356	-	-	-
300-3	$\mathbf{R}$	0.875	0.220	-	-	553
	$\mathbf{R}+\mathscr{G}^C$	0.876	0.268	816	581	701
	$\mathbf{R}+\mathscr{G}^T$	0.876	0.269	780	479	703
	Vignes (DREAM5)	0.786	0.141	-	-	-
	Vignes *et al.*	0.774	0.159	-	-	-
	Flassig *et al.*	0.890	0.316	-	-	-
300-4	$\mathbf{R}$	0.859	0.193	-	-	553
	$\mathbf{R}+\mathscr{G}^C$	0.859	0.233	832	456	721
	$\mathbf{R}+\mathscr{G}^T$	0.859	0.236	796	359	739
	Vignes (DREAM5)	0.759	0.132	-	-	-
	Vignes *et al.*	0.739	0.141	-	-	-
	Flassig *et al.*	0.873	0.292	-	-	-
300-5	$\mathbf{R}$	0.841	0.190	-	-	573
	$\mathbf{R}+\mathscr{G}^C$	0.841	0.229	908	484	745
	$\mathbf{R}+\mathscr{G}^T$	0.841	0.231	877	416	756
	Vignes (DREAM5)	0.737	0.113	-	-	-
	Vignes *et al.*	0.719	0.131	-	-	-
	Flassig *et al.*	0.854	0.291	-	-	-

Table 8.3: **Performance of the RAGNO techniques in DREAM5 sub-challenge A3 (999 RILs).**

Network	Method	AUROC	AUPR	TPs	FPs	ToP1000
999-1	$\mathbf{R}$	0.952	0.324	-	-	543
	$\mathbf{R}+\mathscr{G}^C$	0.953	0.407	660	213	672
	$\mathbf{R}+\mathscr{G}^T$	0.953	0.410	642	156	668
	Vignes (DREAM5)	0.933	0.358	-	-	-
	Vignes *et al.*	0.902	0.482	-	-	-
	Flassig *et al.*	0.969	0.630	-	-	-
999-2	$\mathbf{R}$	0.923	0.259	-	-	563
	$\mathbf{R}+\mathscr{G}^C$	0.923	0.313	800	530	683
	$\mathbf{R}+\mathscr{G}^T$	0.923	0.315	779	461	688
	Vignes (DREAM5)	0.885	0.258	-	-	-
	Vignes *et al.*	0.845	0.364	-	-	-
	Flassig *et al.*	0.942	0.468	-	-	-
999-3	$\mathbf{R}$	0.906	0.241	-	-	580
	$\mathbf{R}+\mathscr{G}^C$	0.906	0.297	844	356	749
	$\mathbf{R}+\mathscr{G}^T$	0.906	0.296	817	315	755
	Vignes (DREAM5)	0.844	0.195	-	-	-
	Vignes *et al.*	0.808	0.292	-	-	-
	Flassig *et al.*	0.926	0.442	-	-	-
999-4	$\mathbf{R}$	0.893	0.223	-	-	598
	$\mathbf{R}+\mathscr{G}^C$	0.893	0.267	912	385	774
	$\mathbf{R}+\mathscr{G}^T$	0.893	0.268	889	307	788
	Vignes (DREAM5)	0.821	0.183	-	-	-
	Vignes *et al.*	0.784	0.260	-	-	-
	Flassig *et al.*	0.915	0.378	-	-	-
999-5	$\mathbf{R}$	0.876	0.216	-	-	627
	$\mathbf{R}+\mathscr{G}^C$	0.876	0.261	916	259	827
	$\mathbf{R}+\mathscr{G}^T$	0.876	0.261	889	209	837
	Vignes (DREAM5)	0.813	0.178	-	-	-
	Vignes *et al.*	0.768	0.244	-	-	-
	Flassig *et al.*	0.898	0.373	-	-	-

In sub-challenge A1, the 1000-gene networks are inferred with the worst accuracy due to the limited number of RILs (a population of only $p = 100$ individuals). The scores by RAGNO are improved by ranking with highest priority the edges in the correlation graph, while the transitive reduction appears detrimental for the accuracy of the prediction. This can be explained by a sub-optimal choice of the threshold parameters (σ, ϑ) for the selection of the edges to be included in the correlation graph, where the number of FPs is about three times larger than the number of TPs: larger values of σ and θ would have prevented the inclusion of several FPs into $\mathscr{G}^C$, and this would have thus helped the process of transitive reduction.

The improvement on the predictive performance after applying transitive reduction is finally significant on sub-challenges A2 and A3 ($p = 300$ and $p = 999$, respectively), in contrast with respect to sub-challenge A1. The edges assembling the correlation graph are chosen with better accuracy (the number of TPs is even larger than the number of FPs), and therefore the process of transitive reduction is less heavily hampered by the presence of too many distracting (false) edges. According to Tables 8.2 and 8.3, in general the decrease in TPs from $\mathscr{G}^C$ to $\mathscr{G}^T$ is moderate with respect to the decrease of FPs, and simultaneously the percentage of TPs amongst the top ranked edges increases.

For all the networks the change in the AUROC score from RAGNO to $\mathscr{G}^C$ and to $\mathscr{G}^T$ is negligible because the correlation graph and the transitive reduction relocate only few edges (about 1 thousand) compared to the number of possible edges in the networks (nearly 1 million), while the improvement in the AUPR is significant (about 20%) for the upraising of true edges and the downgrading of false edges in the top positions on the prediction list.

The beneficent effect of selecting particular edges into a correlation graph is undeniable when looking at the surfaces plotted in Figure 8.1, where the AUPR scores obtained by ranking the edges in the correlation graph before the other are larger than those scored by sorting the edges according uniquely to the confidence expressed in $\mathbf{R}$. In fact, the precision increases for a wide combination of parameters σ and ϑ, i.e. a reasonable choosing of the thresholds (e.g. $\sigma = 0.1$ and $\vartheta = 0.2$) will always lead to an improvement of the network reconstruction with respect to the output yielded by the confidence matrix $\mathbf{R}$, which might be significant when the threshold selection is optimal.

8.2.2 Performance with the StatSeq benchmark datasets

The StatSeq datasets [15] have been simulated to provide a varied benchmark to test how the inference of gene networks is conditioned by relevant features as the number of observations, the genetic linkage, the heritability, and the network size. For detailed information on the 72 syntethic datasets, see Section 4.2.

The application of the RAGNO algorithm on these benchmarks produces, in general, satisfactory results that are comparable with the best performances of the competition – in particular for sizes 1000 and 5000, and especially in configurations 1 (large marker distance, high heritability, small population), 3 and 7 (low heritability and small population), as shown as an example in the comparative plots of Figures 8.2 and 8.3, where the RAGNO technique is compared with the three best overall performing algorithms by Huynh-Thu *et al.* [83], Heise

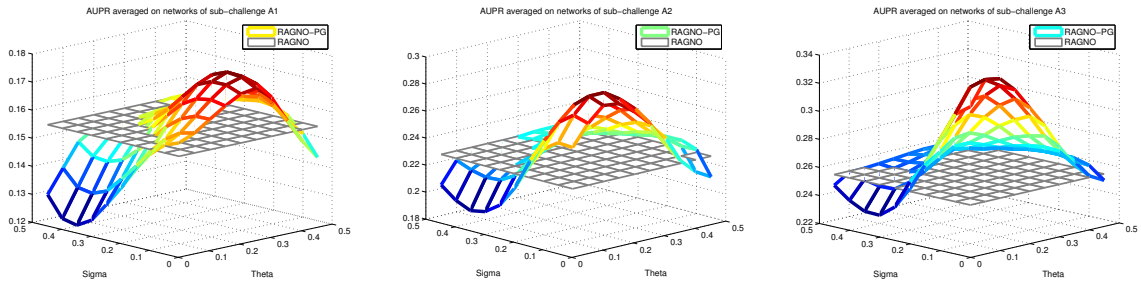

Figure 8.1: **AUPR score averaged by the RAGNO techniques on the networks of the DREAM5 Systems Genetics sub-challenges A1, A2 and A3.** The figures show the reconstruction performance of methods **R** (AUPR = $[0.1536, 0.2252, 0.2525]$) and $\mathbf{R}+\mathcal{G}^C$ for threshold parameters $(\sigma, \vartheta) \in [0.05, 0.5]$ in the three DREAM5 Systems Genetics sub-challenges. The best scores (AUPR = $[0.1752, 0.2855, 0.3396]$) are obtained for $[(\sigma = 0.15, \vartheta = 0.30), (\sigma = 0.10, \vartheta = 0.15), (\sigma = 0.05, \vartheta = 0.10)]$.

et al. [77] and Sambo *et al.* [147]. Analogously to the DREAM5 benchmarks, the surface plots in Figure 8.4 show how easily the correlation graph allows for an improvement of the AUPR scores (left), while the effect of the transitive reduction might be detrimental when too many edges are removed (e.g. for $\alpha < 0.5$) or slightly positive when the threshold is tightened (right).

8.2.3 Performance with a yeast dataset

A greatly challenging application of the algorithm involves the identification of the gene network representing a real-world organism from experimental data. We tested our approach by analyzing genotype and gene expression data measured in a cross between two strains of *Saccharomyces cerevisiae* [33, 34, 166]. The dataset consists of 5736 gene expression levels[3] measured for $p = 112$ segregants, i.e. individuals. Each gene has been related to its closest marker (amongst a set of 2956 markers), in order to complete the associated genotype matrix. Some genotype values are missing, and thus substituted with a `NaN` value in **X**. Both genotype and gene expression matrices **X** and **Y** have size 112×5670, i.e. $p \ll n$. Evaluation of the inference methodology is not forthright because the gene network of yeast is still unknown, yet the organism has been studied enough to compile reliable but incomplete silver standard networks. We then quantified the performance of the reverse-engineering technique by comparing the results with the 114×5667 yeast network used as a reference at the DREAM5 Network Inference challenge. The inference algorithm yields a score for each possible edge of a 5670×5670 matrix, but only the edges within its intersection with the silver standard, a 112×5418 network, can be evaluated.

The complete RAGNO algorithm has been applied to the input datasets. The confidence score matrix **R** already ranks 16 true edges amongst its most confident 100 edges; after rearranging the edges following the correlation graph procedure, the number of true edges amongst the top-100 increase to 20 with ($\sigma = 0.15$, $\vartheta = 0.25$) and even to 23 with ($\sigma = 0.30$,

[3] The expression of 33 genes is measured twice, therefore the number of genes represented with a single expression level is $n = 5670$.

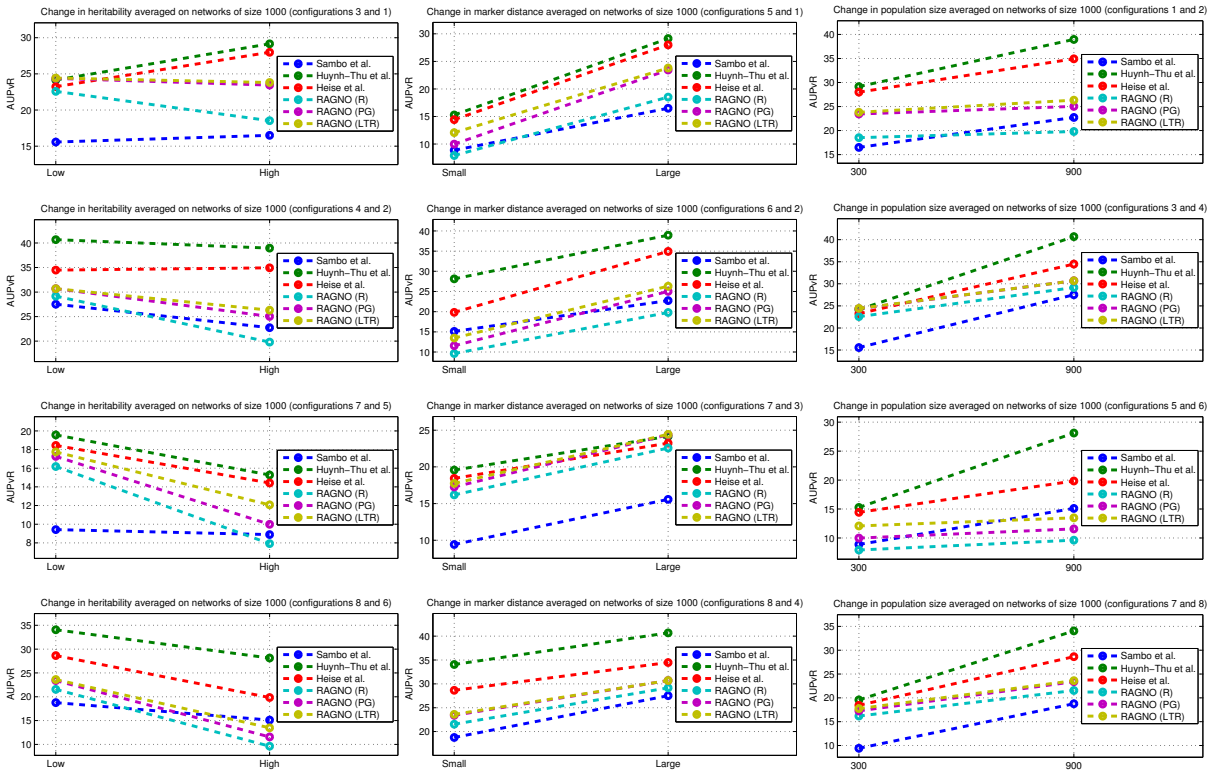

Figure 8.2: **Comparison of performances by RAGNO techniques on the StatSeq benchmark datasets ($n = 1000$).** The plots show the change in AUPR scores when two *variables* are fixed and the other one – heritability (left), marker distance (center), population size (right) – changes.

$\vartheta = 0.30$). Transitive reduction is not particularly effective, and the related improvements (if any!) are negligible. With regards to the AUPR computed by considering all the possible edges, it nearly doubles the score by the random prediction (from $\simeq 0.006$ to $\simeq 0.011$).

8.3 Discussion

We have presented a straightforward yet effective algorithm for the reconstruction of gene regulatory networks from genotype and gene expression measurements. The approach ranks itself amongst the top performing techniques within this particular area in the larger field of network inference methodologies, by proving its strength in the identification of gene networks after analyzing established benchmark datasets, as those released to the community at international projects such DREAM and StatSeq.

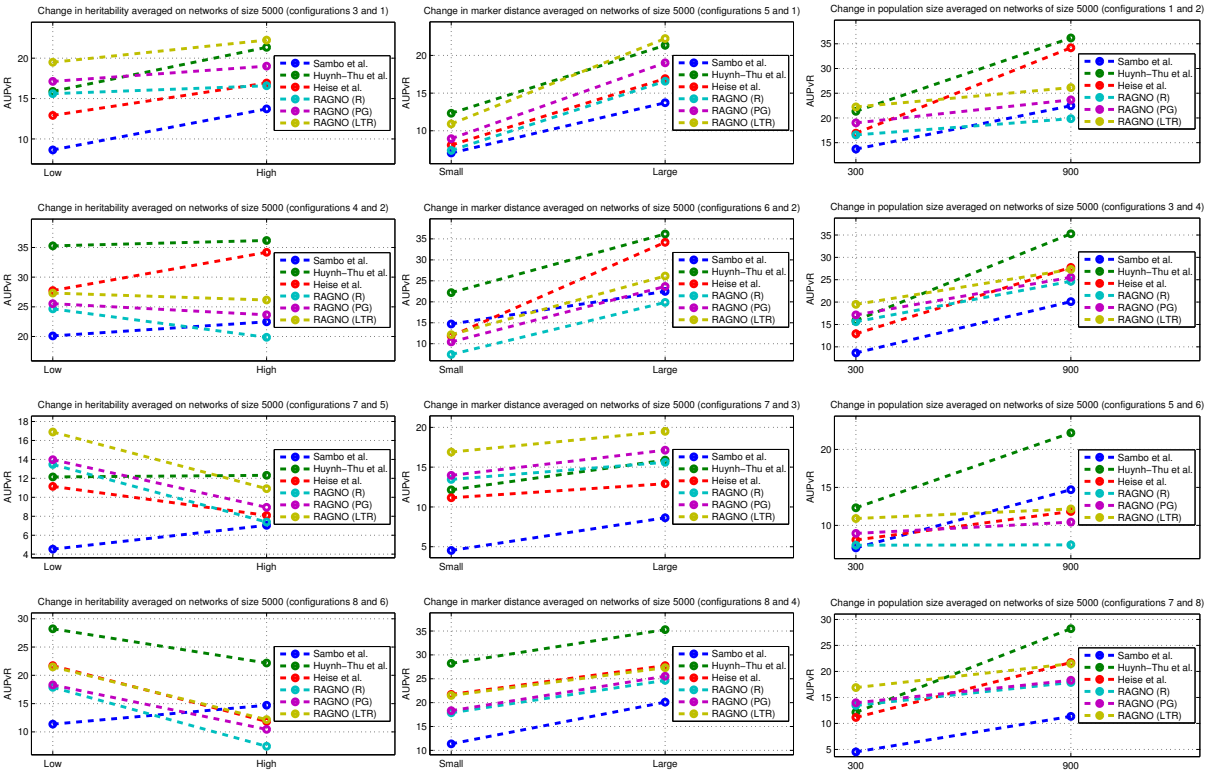

Figure 8.3: **Comparison of performances by RAGNO techniques on the StatSeq benchmark datasets (n = 5000).** The plots show the change in AUPR scores when two *variables* are fixed and the other one – heritability (left), marker distance (center), population size (right) – changes.

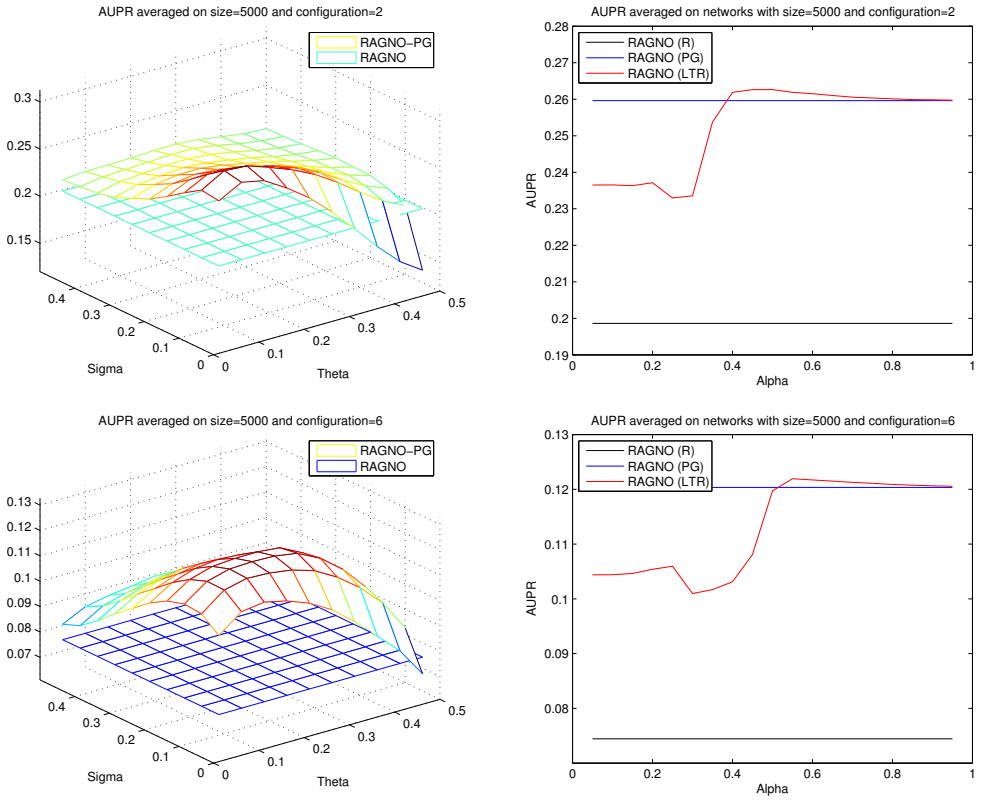

Figure 8.4: **Average AUPR scores by RAGNO techniques on StatSeq networks for configurations 2 and 6.** AUPR scores averaged on StatSeq networks of size $n = 5000$ and simulated according to configuration 2 and 6 for $\mathbf{R}$ and $\mathbf{R} + \mathscr{G}^C$ (left). AUPR score averaged on the same StatSeq networks with correlation graph computed for parameters $\sigma = 0.15$ and $\vartheta = 0.25$, and transitive reduction applied with thresholds $\alpha \in [0.05, 0.95]$ (right).

Chapter 9

Side projects

This chapter collects two works partially associated with the topics previously presented, where we mostly dealt with the simulation and the structural identification of gene regulatory networks. In fact, in Section 9.1.2 a community-based research [121], where the results and methods of the DREAM6 Estimation of Model Parameters and the DREAM7 Network Topology and Parameter Inference challenges are analyzed by a panel of organizers, is briefly summarized. Our team only participated to the DREAM6 challenge by submitting a below-the-average estimate of the small biological network parameters (see Section 9.1.1), and we are therefore credited as authors amongst the DREAM 6&7 Parameter Estimation consortium. Section 9.2 contains the paper [39], by our Bioinformatics group at CRS4, describing the Galaxy platform Orione [12] and the several microbiology tools there implemented and freely available to researchers.

9.1 DREAM6 parameter estimation challenge

In Section 9.1.1 we describe the technique we developed to estimate the parameters and to predict the outcomes of applied perturbations in systems biology models at the DREAM6 Estimation of Model Parameters challenge. Moreover, the organizers of the competition thoroughly examined the methodologies and their performances to compile a community approach for the estimation of gene regulatory network kinetics: this achievement is concisely recapitulated in Section 9.1.2.

9.1.1 Submitted technique at the DREAM6 Estimation of Model Parameters challenge

We studied the problem of estimating the kinetic parameters of given models of gene regulatory networks by participating to the DREAM6 Estimation of Model Parameters challenge, whose goal was therefore the development of optimization methods for the estimate of parameters in the modeling of biological systems. The participants were provided with the full regulatory interaction topology for three small gene networks and were requested to estimate the value of the unknown parameters given a dataset of time-series measurements and the possibility to purchase, given a fixed budget, additional experimental data from a

broad assortment. By using a customized version of SysGenSIM [136] we simulated all types of challenge data with the known equations and topologies, and with randomly assigned values in place of the unknown parameters. This enabled us to evaluate optimization algorithms with different purchasing schemes to finally select a common strategy for all three models: unfortunately our approach performed badly compared to the best methodologies, but still yielded an adequate prediction.

We applied our strategy as follows. First, we started out by simulating data with the given equations for all 3 models, using randomly assigned parameters values. In particular we generated all types of data that can be purchased and the time courses with the same perturbation as we are requested to predict. This enabled us to evaluate alternative optimization algorithms (Nelder-Mead, trust-region-reflective, CoByLa, simulated annealing, genetic algorithms) with different data purchasing schemes. After thorough simulation studies we decided to use a common strategy for all three models, as detailed in the following ordered list.

1. **Estimation of protein degradation rate constants:** we purchased one or more Gene Deletion Experiments in combination with protein time series of the protein encoded by the deleted gene. Since initial mRNAs are set to zero, the protein of the deleted gene will display simple first order decay, so that the protein degradation rate constants can be relatively easily estimated. We used `nlinfit` in MATLAB to find the best value for the protein degradation rate constants by fitting the following analytical function to the protein time series: $p(t) = p(0) \cdot e^{p\_degradation\_rate \cdot t}$ with $p(0)$ the initial concentration of the protein. Once a good estimate of the protein degradation constant has been obtained we can use this value for the protein degradation rate constants of all proteins in the model (as they were set equal as mentioned in the challenge description). For model 1 and 2 we purchased two Gene Deletion Experiments to obtain better estimates, while for model 3 only gene 1 can be used since it is the only one with $p(0) > 0$.

2. **Estimation of ribosome binding strengths:** we purchased high-quality wild type mRNA time series. We also purchased wild type time series for all proteins. For some proteins we purchased two wild type time series to get better estimates in the light of the noise. Also we selected the second protein in the pair purchased after Gene Deletions (Step 1) to be unaffected by the deletion, so we could use it as a wild type series replicate too. From these time series we estimated the ribosome binding strengths by fitting the following analytical function to the wild type mRNA and protein time series (using `nlinfit`):

$$p(t) = rbs\_strength \cdot \frac{pp\_mRNA(t)}{p\_degradation\_rate} + \frac{p(0) - rbs\_strength \cdot pp\_mRNA(t)}{p\_degradation\_rate} e^{p\_degradation\_rate \cdot t} \tag{9.1}$$

   We fixed the protein degradation rate constants to the values obtained in Step 1, so that only the ribosome binding strengths had to be optimized.

3. **Estimation of promoter strength of gene 1:** in each of the three models, gene 1 receives no inputs from other genes, therefore its mRNA time course depends only on its promoter strength and degradation rate constant. Since the value of the degradation rate constant was provided in challenge description, we used `nlinfit` to estimate only

the promoter strength by fitting the following analytical function to the high-quality wild type mRNA time series:

$$pp1\_mRNA(t) = pro1\_strength \cdot (1 - e^t) \tag{9.2}$$

given that the initial concentration of the mRNA $pp1\_mRNA(0) = 0$ and the degradation rate constant is equal to 1.

4. **Fixing some Binding affinity (K_d) and Hill coefficient (h) parameters:** through our simulation studies we clearly saw large improvements when a set of K_d and h were known, so we purchased 3 Gel Shift Experiments (giving 3 pairs of K_d and h) for each model. For model 1 and 2 we purchased parameters appearing in the equations of the genes encoding the proteins for which we needed to make time series predictions. For model 3 we performed several fits before deciding which parameters to purchase.

5. **Estimation of promotor strengths using forced inputs:** for the genes for we have purchased the K_d and h we need to estimate only promotor strength. In this case we decouple their equation from the global model by fitting its inputs by a polynomial and replace these state variables by the polynomial function (similar to the approach we have applied to the 5-gene network inference challenge in DREAM2 [154] and then optimize the promotor strength independent of all other parameters in the model.

6. **Global optimization:** we used all parameters estimated (and purchased) in Steps 1-5 to create an initial estimation vector $\mathbf{x}_0$, with the remaining promoter strengths and K_d initialized to 1 and h initialized to 2. Then we performed a series of global parameter optimizations to fit the wild type mRNA and protein time series. In particular, we alternated `fmincon` and genetic algorithms in MATLAB, always using as starting point the best fit of the previous optimization. `fmincon` was used to reach rapidly a local optimum, while ga was added for its ability in escaping the local optimum by mutations and recombinations. For all optimizations we fixed the purchased parameters, and also forced the other parameters between lower and upper bounds: between 1 and 4 for the h parameters, between 10^{-6} and 10^3 for the other ones. We used as fitness function the sum of the absolute values of the differences between the measured data points and the corresponding values calculated by solving the model ODEs for the estimated parameters. We continue the series of optimizations until the decrease in the fitness function is smaller than 0.5 in the last 10 optimizations.

7. **Final predictions:** the last best fit in Step 6 for the model parameters were submitted and used to calculate the submitted protein time series predictions, again by solving the model ODEs.

9.1.2 Network topology and parameter estimation: from experimental design methods to gene regulatory network kinetics using a community based approach

The summary of the community paper [121], to which we contributed as members of the DREAM 6&7 Parameter Estimation consortium, is reported in the following.

Background

Accurate estimation of parameters of biochemical models is required to characterize the dynamics of molecular processes. This problem is intimately linked to identifying the most informative experiments for accomplishing such tasks. While significant progress has been made, effective experimental strategies for parameter identification and for distinguishing among alternative network topologies remain unclear. We approached these questions in an unbiased manner using a unique community-based approach in the context of the DREAM initiative (Dialogue for Reverse Engineering Assessment of Methods). We created an in silico test framework under which participants could probe a network with hidden parameters by requesting a range of experimental assays; results of these experiments were simulated according to a model of network dynamics only partially revealed to participants.

Results

We proposed two challenges; in the first, participants were given the topology and underlying biochemical structure of a 9-gene regulatory network and were asked to determine its parameter values. In the second challenge, participants were given an incomplete topology with 11 genes and asked to find three missing links in the model. In both challenges, a budget was provided to buy experimental data generated in silico with the model and mimicking the features of different common experimental techniques, such as microarrays and fluorescence microscopy. Data could be bought at any stage, allowing participants to implement an iterative loop of experiments and computation.

Conclusion

A total of 19 teams participated in this competition. The results suggest that the combination of state of the art parameter estimation and a varied set of experimental methods using a few datasets, mostly fluorescence imaging data, can accurately determine parameters of biochemical models of gene regulation. However, the task is considerably more difficult if the gene network topology is not completely defined, as in challenge 2. Importantly, we found that aggregating independent parameter predictions and network topology across submissions creates a solution that can be better than the one from the best-performing submission.

9.2 Orione, a web-based framework for NGS analysis in microbiology

End-to-end NGS microbiology data analysis requires a diversity of tools covering bacterial resequencing, *de novo* assembly, scaffolding, bacterial RNA-Seq, gene annotation and metagenomics. However, the construction of computational pipelines that use different software packages is difficult due to a lack of interoperability, reproducibility, and transparency. To overcome these limitations we present Orione, a Galaxy-based framework consisting of publicly available research software and specifically designed pipelines to build complex, reproducible workflows for NGS microbiology data analysis. Enabling microbiology researchers to conduct their own custom analysis and data manipulation without software installation or programming, Orione provides new opportunities for data-intensive

computational analyses in microbiology and metagenomics.

Application of Next Generation Sequencing (NGS) in microbiology is becoming a common practice with a profound impact on research, diagnostic and clinical microbiology [104]. Recent applications include genomic sequencing, differential transcription analysis, variant investigation, as well as metagenomics studies. Major challenges include draft assemblies finishing followed by reliable genome annotation or robust dissection of microbial communities including those associated with human health and disease. Furthermore, there is an increasing need to process and present data in a fashion that is transparent and reproducible and to provide analysis frameworks that are usable and cost-effective for biomedical researchers.

To address these challenges, we developed Orione, an online framework for integrative analysis of NGS microbiology data. Orione is based on Galaxy [69], an open platform for reproducible data-intensive computational analysis utilized in many diverse biomedical research environments. Orione is the first freely available platform that supports the whole life cycle of microbiology research data from production and annotation to publication and sharing. Other commercial alternative exists (e.g. CLC Genomics Workbench by CLC Bio), but Orione is unique in transparently combining the most used open source bioinformatics tools for microbiology. Orione is currently applied to a variety of microbiological projects including bacteria resequencing, *de novo* assembling and microbiome investigations, see [10] for a list. Furthermore, Orione is part of an ongoing project to integrate Galaxy with: Hadoop-based tools to provide scalable computing [99]; a specialized version of OMERO [20] to model biomedical data and the chain of actions that connect them; and iRODS [139] to efficiently support inter-institutional data sharing. This infrastructure is already used in production at CRS4 for the automated processing of sequencing data [137] and for quality control in gene therapy applications [28].

9.2.1 Features and methods

Orione consists of *best-of-breed* NGS bioinformatics tools covering end-to-end data analysis for bacterial resequencing, *de novo* assembly, scaffolding, bacterial RNA-Seq, gene annotation, metagenomics and metatranscriptomics. Publicly available research tools were integrated under the open source Galaxy framework with pipelines and workflows newly developed by our group for ready-to-go microbiological analysis. While several of the tools for NGS microbiology data analysis were already available in Galaxy, a significant effort was required to expand the Galaxy functionalities with new features such as SSPACE [29], SSAKE [177], SOPRA [42], SEQuel [144], EDGE-pro [109], Glimmer [48], and Prokka [13]. We refer to the Supplementary information for a description of the complete set of Orione tools and workflows.

9.2.2 Functionalities

Orione complements the flexible Galaxy workflow environment, allowing microbiologists without any specific hardware or informatics skill to consistently access a set of NGS data analysis tools and conduct reproducible data-intensive computational analyses from quality

control to microbial gene annotation. Figure 9.1 illustrates an overall schema of the main Orione functionalities that are described in detail in the following paragraphs.

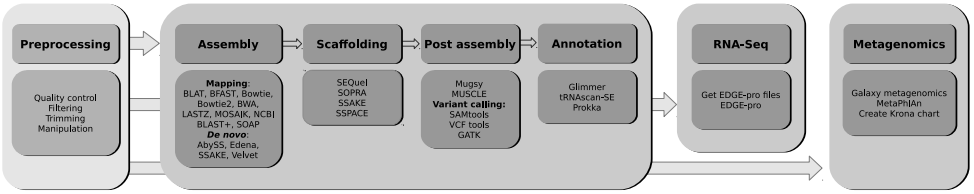

Figure 9.1: **Overall schema of the main Orione functionalities.** Boxes represent collections of tools performing specific tasks.

Preprocessing, quality control and trimming. The fundamental step before any NGS analysis is the quality control of reads and their trimming. To cope with with long reads and paired-end technology, FastX [6] and FASTQC [5] were complemented with specifically developed tools (see also workflow #1 in the Supplementary information).

Reads mapping. Mapping is a key step in many NGS applications from bacteria resequencing to variant calling. The most widely used aligners are integrated in Orione, including BWA [100], Bowtie1 [96], Bowtie2 [95], SOAP [102], MOSAIK [11]. We further added BLAT [88], SHRiMP [41], LASTZ [74] and BFAST [79] for use with long reads from 454 Roche.

***De novo* assembly.** *De novo* assembly produces contigs without the aid of a reference genome. Different methods, either based on a de Bruijn graph (Velvet [184], ABySS [157], SPAdes [22]) or on a greedy approach (SSAKE, Edena [78]) are available in Orione.

Scaffolding. After mapping, contigs are ordered and oriented to produce even longer sequences called scaffolds, exploiting the mate pair/paired-end information. Orione includes the most established scaffolders such as SSAKE, SSPACE, SEQuel, and SOPRA.

Post assembly, contigs statistics, (multi) aligning, and variant calling. This section of Orione includes tools we have developed covering task such as genome-scale alignment, high quality contigs extraction, statistics over contigs or draft genomes (N50/NG50 values, contigs length distribution, high/low quality regions/gaps in draft genomes).

Annotation. Annotation is the process of identifying meaningful biological information from sequences. Glimmer and tRNAscan-SE [105] were wrapped into Orione together with the Prokka pipeline, enabling easy Genbank/DDJB/ENA submission.

RNA-Seq. We integrated EDGE-pro tool for bacterial RNA-Seq analysis. Since EDGE-pro requires genome annotation files, we developed an accessory tool (Get EDGE-pro files) which retrieves them directly from the NCBI RefSeq repository.

Metagenomics and other tools. We added to the standard Galaxy metagenomics pipeline MetaPhlAn [156], and MetaVelvet [124]. The MetaGeneMark [187] annotation tool has been added for gene prediction in metagenomic sequences and a workflow has been developed

for (bacterial) metatranscriptome analysis. We complete this section with instruments for data filtering, conversion and taxonomy abundance displaying into the Krona visualizer [128].

Chapter 10

Concluding remarks

The projects described in this thesis allowed me to investigate several issues in bioinformatics topics of current interest, while still employing methodologies and techniques borrowed from information engineering, as e.g. system modeling and simulation, network identification, graph theory, scientific programming.

Thanks to a mixture of hard work, obstinacy, excellent advising and above all good luck, I have been able to substantially contribute to certain aspects of bioinformatics, below summarized.

- The development of SysGenSIM, a software to generate gene networks and efficiently simulate systems genetics and systematic perturbative experiments. The toolbox produces datasets for the purpose of evaluating and comparing e.g. methodologies to perform eQTL mapping and algorithms to infer gene networks. SysGenSIM is available as an open source software and it is still maintained and updated.

- The release of benchmark datasets to the scientific community for the evaluation and verification of gene network inference techniques. International competitions (as the DREAM5 challenge and the StatSeq workshop) have been organized to this aim – using SysGenSIM's datasets as benchmark. Works presenting network identification techniques, developed and verified using these benchmarks, have been and are currently being published by peer-reviewed journals. SysGenSIM can be then considered as a valuable instrument for the realistic simulation of such particular types of biological systems and genetic experiments.

- The implementation of algorithms for the accurate identification of directed gene interactions from data generated by different types of experiments. One of the developed inference techniques proved itself particularly effective in the analysis of simulated and even incomplete real datasets of single-gene knockout experiments. Another technique, based on multiple approaches singularly applied on subsets of data, proved to be reliable in reverse-engineering transcriptional regulatory networks from real and heterogeneous expression compendia. A third algorithm effectively identifies interactions in gene networks from genotyping and expression data. These methodologies,

applied on realistic datasets, might help scientists in addressing their decisions when performing in vivo research.

Besides the mere scientific contributions, I truly benefited from the participation to international conferences and meetings, and thrived in collaborating with researchers from established centers as Max Planck Institute and Virginia Bioinformatics Institute.

Future developments, besides the further improvement of the presented methods, might include the application of the inference algorithms on novel experimental datasets from real organisms and the actual finding of true but yet undiscovered gene interactions. Another possibility is the implementation of the simulation software and the identification techniques into a Galaxy platform, e.g. one specifically dedicated to systems biology. This would allow the immediate utilize of such powerful tools – but still demanding for inexperienced computer users – to biologists and physicians.

Finally, an official organism for the verification of gene network inference techniques should be established to develop shared strategies, as e.g. the type of biological datasets to be used as benchmarks, or the measures to fairly evaluate the proposed methodologies.

Bibliography

[1] DREAM4 In Silico Network challenge. http://wiki.c2b2.columbia.edu/dream/index.php/D4c2. [cited at p. 67]

[2] DREAM5 Systems Genetics challenge. http://wiki.c2b2.columbia.edu/dream/index.php/D5c3. [cited at p. 2, 97, 100]

[3] E-MTAB-109: transcription profiling of yeast transcription factor knockout compendium. http://www.ebi.ac.uk/arrayexpress/experiments/E-MTAB-109. [cited at p. 77]

[4] Evaluation script for the DREAM4 In Silico Network challenge. http://wiki.c2b2.columbia.edu/dream/results/DREAM4. [cited at p. 67]

[5] FASTQC. http://www.bioinformatics.babraham.ac.uk/projects/fastqc. [cited at p. 114]

[6] FASTX-Toolkit. http://hannonlab.cshl.edu/fastx_toolkit. [cited at p. 114]

[7] GeneNet. http://strimmerlab.org/software/genenet. [cited at p. 87]

[8] GeneNetWeaver benchmarks. http://gnw.sourceforge.net/dreamchallenge.html. [cited at p. 19]

[9] Genetic Analysis Workshop. http://gaworkshop.org. [cited at p. 22]

[10] Microbiology projects using Orione. http://ncbi.nlm.nih.gov/bioproject/?term=CRS4. [cited at p. 113]

[11] MOSAIK. http://github.com/wanpinglee/MOSAIK. [cited at p. 114]

[12] Orione. http://orione.crs4.it. [cited at p. 109]

[13] Prokka. http://vicbioinformatics.com/software.prokka.shtml. [cited at p. 113]

[14] SOSlib. http://www.tbi.univie.ac.at/~raim/odeSolver. [cited at p. 12]

[15] SysGenSIM benchmarks. http://sysgensim.sourceforge.net/datasets.html. [cited at p. 19, 33, 41, 84, 97, 100, 103]

[16] SysGenSIM website. http://sysgensim.sourceforge.net. [cited at p. 2, 17, 34]

[17] The DREAM project. http://the-dream-project.org. [cited at p. 22, 31, 44]

[18] Marit Ackermann, Mathieu Clément-Ziza, Jacob J Michaelson, and Andreas Beyer. Teamwork: improved eQTL mapping using combinations of machine learning methods. *PLoS ONE*, 7(7):e40916, 2012. [cited at p. 101]

[19] Tatsuya Akutsu, Satoru Kuhara, Osamu Maruyama, and Satoru Miyano. Identification of genetic networks by strategic gene disruptions and gene overexpressions under a boolean model. *Theoretical Computer Science*, 298(1):235–251, 2003. [cited at p. 43]

[20] Chris Allan, Jean-Marie Burel, Josh Moore, Colin Blackburn, Melissa Linkert, Luca Lianas, Simone Leo, Gerard J Kleywegt, Gianluigi Zanetti, Jason R Swedlow, et al. OMERO: flexible, model-driven data management for experimental biology. *Nature Methods*, 9(3):245–253, 2012. [cited at p. 113]

[21] Jason E Aten, Tova F Fuller, Aldons J Lusis, and Steve Horvath. Using genetic markers to orient the edges in quantitative trait networks: the NEO software. *BMC Systems Biology*, 2(1):34, 2008. [cited at p. 22]

[22] Anton Bankevich, Sergey Nurk, Dmitry Antipov, Alexey A Gurevich, Mikhail Dvorkin, Nikolay Vyahhi, Glenn Tesler, Max A Alekseyev, Pavel A Pevzner, et al. SPAdes: a new genome assembly algorithm and its applications to single-cell sequencing. *Journal of Computational Biology*, 19(5):455–477, May 2012. [cited at p. 114]

[23] Mukesh Bansal, Vincenzo Belcastro, Alberto Ambesi-Impiombato, and Diego di Bernardo. How to infer gene networks from expression profiles. *Molecular Systems Biology*, 3(1), 2007. [cited at p. 22, 43, 44]

[24] Ziv Bar-Joseph, Georg K Gerber, Tong Ihn Lee, Nicola J Rinaldi, Jane Y Yoo, François Robert, D Benjamin Gordon, Ernest Fraenkel, Tommi S Jaakkola, Richard A Young, et al. Computational discovery of gene modules and regulatory networks. *Nature Biotechnology*, 21(11):1337–1342, 2003. [cited at p. 44]

[25] Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, 1999. [cited at p. 23]

[26] Albert-László Barabási and Zoltan N Oltvai. Network biology: understanding the cell's functional organization. *Nature Reviews Genetics*, 5(2):101–113, 2004. [cited at p. 14, 23]

[27] Vladimir Batagelj and Andrej Mrvar. Pajek – program for large network analysis. *Connections*, 21(2):47–57, 1998. [cited at p. 29]

[28] Alessandra Biffi, Eugenio Montini, Laura Lorioli, Simone Leo, Gianluigi Zanetti, Elia Stupka, Alessandro Aiuti, Maria Sessa, Luigi Naldini, et al. Lentiviral hematopoietic stem cell gene therapy benefits metachromatic leukodystrophy. *Science*, 341(6148):1233158, 2013. [cited at p. 113]

[29] Marten Boetzer, Christiaan V Henkel, Hans J Jansen, Derek Butler, and Walter Pirovano. Scaffolding pre-assembled contigs using SSPACE. *Bioinformatics*, 27(4):578–579, 2011. [cited at p. 113]

[30] Dragan Bošnački, Maximilian R Odenbrett, Anton Wijs, Willem Ligtenberg, and Peter Hilbers. Efficient reconstruction of biological networks via transitive reduction on general purpose graphics processors. *BMC Bioinformatics*, 13(1):281, 2012. [cited at p. 60, 71, 81]

[31] Anne-Laure Boulesteix. Over-optimism in bioinformatics research. *Bioinformatics*, 26(3):437–439, 2010. [cited at p. 10]

[32] Paul Brazhnik, Alberto de la Fuente, and Pedro Mendes. Gene networks: how to put the function in genomics. *TRENDS in Biotechnology*, 20(11):467–472, 2002. [cited at p. 8, 9, 10]

[33] Rachel B Brem and Leonid Kruglyak. The landscape of genetic complexity across 5,700 gene expression traits in yeast. *Proceedings of the National Academy of Sciences of the United States of America*, 102(5):1572–1577, 2005. [cited at p. 28, 104]

[34] Rachel B Brem, John D Storey, Jacqueline Whittle, and Leonid Kruglyak. Genetic interactions between polymorphisms that affect gene expression in yeast. *Nature*, 436(7051):701–703, 2005. [cited at p. 104]

[35] Rachel B Brem, Gaël Yvert, Rebecca Clinton, and Leonid Kruglyak. Genetic dissection of transcriptional regulation in budding yeast. *Science*, 296(5568):752–755, 2002. [cited at p. 21]

[36] Atul J Butte and Isaac S Kohane. Mutual information relevance networks: functional genomic clustering using pairwise entropy measurements. In *Pacific Symposium on Biocomputing*, volume 5, pages 418–429, 2000. [cited at p. 43]

[37] Julien Chiquet, Yves Grandvalet, and Christophe Ambroise. Inferring multiple graphical structures. *Statistics and Computing*, 21(4):537–553, 2011. [cited at p. 43]

[38] Hyonho Chun and Sündüz Keleş. Expression quantitative trait loci mapping with multivariate sparse partial least squares regression. *Genetics*, 182(1):79–90, 2009. [cited at p. 22]

[39] Gianmauro Cuccuru, Massimiliano Orsini, Andrea Pinna, Andrea Sbardellati, Nicola Soranzo, Antonella Travaglione, Paolo Uva, Gianluigi Zanetti, and Giorgio Fotia. Orione, a web-based framework for NGS analysis in microbiology. *Bioinformatics*, 2014. [cited at p. 5, 109]

[40] L Adrienne Cupples, Joseph Beyene, Heike Bickeböller, E Warwick Daw, M Daniele Fallin, W James Gauderman, Saurabh Ghosh, Ellen Goode, Elizabeth Hauser, Anthony Hinrichs, et al. Genetic Analysis Workshop 16: strategies for genome-wide association study analyses. In *BMC Proceedings*, volume 3, page S1. BioMed Central Ltd, 2009. [cited at p. 22]

[41] Matei David, Misko Dzamba, Dan Lister, Lucian Ilie, and Michael Brudno. SHRiMP2: Sensitive yet practical short read mapping. *Bioinformatics*, 27(7):1011–1012, 2011. [cited at p. 114]

[42] Adel Dayarian, Todd P Michael, and Anirvan M Sengupta. SOPRA: scaffolding algorithm for paired reads via statistical optimization. *BMC Bioinformatics*, 11:345, 2010. [cited at p. 113]

[43] Alberto de la Fuente. What are gene regulatory networks? In *Handbook of research on computational methodologies in gene regulatory networks*, chapter 1, pages 1–27. IGI Global, Hershey, PA, USA, 2010. [cited at p. 9]

[44] Alberto de la Fuente. *Gene network inference – verification of methods for systems genetics data*. Springer, 2014. [cited at p. 3, 5, 10, 22, 34, 97]

[45] Alberto de la Fuente, Nan Bing, Ina Hoeschele, and Pedro Mendes. Discovery of meaningful associations in genomic data using partial correlation coefficients. *Bioinformatics*, 20(18):3565–3574, 2004. [cited at p. 43, 45, 49, 87]

[46] Alberto de la Fuente, Paul Brazhnik, and Pedro Mendes. Linking the genes: inferring quantitative gene networks from microarray data. *TRENDS in Genetics*, 18(8):395–398, 2002. [cited at p. 11, 22, 43, 45]

[47] Riet De Smet and Kathleen Marchal. Advantages and limitations of current network inference methods. *Nature Reviews Microbiology*, 8(10):717–729, 2010. [cited at p. 43, 44]

[48] Arthur L Delcher, Kirsten A Bratke, Edwin C Powers, and Steven L Salzberg. Identifying bacterial genes and endosymbiont DNA with Glimmer. *Bioinformatics*, 23(6):673–679, 2007. [cited at p. 113]

[49] Patrik D'haeseleer, Xiling Wen, Stefanie Fuhrman, and Roland Somogyi. Linear modeling of mRNA expression levels during CNS development and injury. *Pacific Symposium on Biocomputing*, 4(1):41–52, 1999. [cited at p. 43]

[50] Diego di Bernardo, Michael J Thompson, Timothy S Gardner, Sarah E Chobot, Erin L Eastwood, Andrew P Wojtovich, Sean J Elliott, Scott E Schaus, and James J Collins. Chemogenomic profiling on a genome-wide scale using reverse-engineered gene networks. *Nature Biotechnology*, 23(3):377–383, 2005. [cited at p. 43, 44]

[51] Barbara Di Camillo, Gianna Toffolo, and Claudio Cobelli. A gene network simulator to assess reverse engineering algorithms. *Annals of the New York Academy of Sciences*, 1158(1):125–142, 2009. [cited at p. 15]

[52] Elie Dolgin. Mouse library set to be knockout. *Nature*, 474(7351):262–263, 2011. [cited at p. 60]

[53] Scott M Dudek, Alison A Motsinger, Digna R Velez, Scott M Williams, and Marylyn D Ritchie. Data simulation software for whole-genome association and other studies in human genetics. In *Pacific Symposium on Biocomputing*, pages 499–510, 2005. [cited at p. 30]

[54] Markus Durzinsky, Annegret Wagler, Robert Weismantel, and Wolfgang Marwan. Automatic reconstruction of molecular and genetic networks from discrete time series data. *BioSystems*, 93(3):181–190, 2008. [cited at p. 43]

[55] Paul Erdős and Alfréd Rényi. On random graphs. *Publicationes Mathematicae Debrecen*, 6:290–297, 1959. [cited at p. 52]

[56] Paul Erdős and Alfréd Rényi. On the evolution of random graphs. *Publications of the Mathematical Institute of the Hungarian Academy of Sciences*, 5:17–61, 1960. [cited at p. 23]

[57] Jeremiah J Faith, Boris Hayete, Joshua T Thaden, Ilaria Mogno, Jamey Wierzbowski, Guillaume Cottarel, Simon Kasif, James J Collins, and Timothy S Gardner. Large-scale mapping and validation of *Escherichia coli* transcriptional regulation from a compendium of expression profiles. *PLoS Biology*, 5(1):e8, 2007. [cited at p. 44, 87]

[58] Ronald A Fisher. The arrangement of field experiments. In *Breakthroughs in Statistics*, pages 82–91. Springer, 1992. [cited at p. 21]

[59] Robert J Flassig, Sandra Heise, Kai Sundmacher, and Steffen Klamt. An effective framework for reconstructing gene regulatory networks from genetical genomics data. *Bioinformatics*, 29(2):246–254, 2013. [cited at p. 84, 101]

[60] Christopher Fogelberg and Vasile Palade. GreenSim: a network simulator for comprehensively validating and evaluating new machine learning techniques for network structural inference. In *Tools with Artificial Intelligence (ICTAI), 2010 22nd IEEE International Conference on*, volume 2, pages 225–230. IEEE, 2010. [cited at p. 15]

[61] Nir Friedman. Inferring cellular networks using probabilistic graphical models. *Science*, 303(5659):799–805, 2004. [cited at p. 43]

[62] Nir Friedman, Michal Linial, Iftach Nachman, and Dana Pe'er. Using Bayesian networks to analyze expression data. *Journal of Computational Biology*, 7(3-4):601–620, 2000. [cited at p. 43, 44]

[63] Jingyuan Fu, Joost JB Keurentjes, Harro Bouwmeester, Twan America, Francel WA Verstappen, Jane L Ward, Michael H Beale, Ric CH De Vos, Martijn Dijkstra, Richard A Scheltema, et al. System-wide molecular evidence for phenotypic buffering in *Arabidopsis*. *Nature Genetics*, 41(2):166–167, 2009. [cited at p. 58]

[64] Timothy S Gardner, Diego di Bernardo, David Lorenz, and James J Collins. Inferring genetic networks and identifying compound mode of action via expression profiling. *Science*, 301(5629):102–105, 2003. [cited at p. 43]

[65] Timothy S Gardner and Jeremiah J Faith. Reverse-engineering transcription control networks. *Physics of Life Reviews*, 2(1):65–88, 2005. [cited at p. 43]

[66] Robert C Gentleman, Vincent J Carey, Douglas M Bates, Ben Bolstad, Marcel Dettling, Sandrine Dudoit, Byron Ellis, Laurent Gautier, Yongchao Ge, Jeff Gentry, et al. Bioconductor: open software development for computational biology and bioinformatics. *Genome Biology*, 5(10):R80, 2004. [cited at p. 77]

[67] Daniel T Gillespie. The chemical Langevin equation. *The Journal of Chemical Physics*, 113(1):297–306, 2000. [cited at p. 14]

[68] Michelle Girvan and Mark EJ Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, 99(12):7821–7826, 2002. [cited at p. 24]

[69] Jeremy Goecks, Anton Nekrutenko, James Taylor, and The Galaxy Team. Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences. *Genome Biology*, 11(8):R86, 2010. [cited at p. 113]

[70] Nabil Guelzim, Samuele Bottani, Paul Bourgine, and François Képès. Topological and causal structure of the yeast transcriptional regulatory network. *Nature Genetics*, 31(1):60–63, 2002. [cited at p. 23, 71]

[71] Mika Gustafsson, Michael Hörnquist, Jesper Lundström, Johan Björkegren, and Jesper Tegnér. Reverse engineering of gene networks with LASSO and nonlinear basis functions. *Annals of the New York Academy of Sciences*, 1158(1):265–275, 2009. [cited at p. 57]

[72] Hendrik Hache, Christoph Wierling, Hans Lehrach, and Ralf Herwig. GeNGe: systematic generation of gene regulatory networks. *Bioinformatics*, 25(9):1205–1207, 2009. [cited at p. 16]

[73] John BS Haldane. The combination of linkage values and the calculation of distances between the loci of linked factors. *Journal of Genetics*, 8(29):299–309, 1919. [cited at p. 26, 38]

[74] Robert S Harris. *Improved pairwise alignment of genomic DNA*. PhD thesis, Pennsylvania State University, 2007. [cited at p. 114]

[75] Leland H Hartwell, John J Hopfield, Stanislas Leibler, and Andrew W Murray. From molecular to modular cell biology. *Nature*, 402:C47–C52, 1999. [cited at p. 23]

[76] Brian C Haynes and Michael R Brent. Benchmarking regulatory network reconstruction with GRENDEL. *Bioinformatics*, 25(6):801–807, 2009. [cited at p. 12]

[77] Sandra Heise, Robert J Flassig, and Steffen Klamt. Benchmarking a simple yet effective approach for inferring gene regulatory networks from systems genetics data. In *Gene Network Inference*, pages 33–47. Springer, 2013. [cited at p. 104]

[78] David Hernandez, Patrice François, Laurent Farinelli, Magne Østerås, and Jacques Schrenzel. De novo bacterial genome sequencing: millions of very short reads assembled on a desktop computer. *Genome Research*, 18(5):802–809, 2008. [cited at p. 114]

[79] Nils Homer, Barry Merriman, and Stanley F Nelson. BFAST: an alignment tool for large scale genome resequencing. *PLoS ONE*, 4(11):e7767, 2009. [cited at p. 114]

[80] Stefan Hoops, Sven Sahle, Ralph Gauges, Christine Lee, Jürgen Pahle, Natalia Simus, Mudita Singhal, Liang Xu, Pedro Mendes, and Ursula Kummer. COPASI – a complex pathway simulator. *Bioinformatics*, 22(24):3067–3074, 2006. [cited at p. 12, 14]

[81] Zhanzhi Hu, Patrick J Killion, and Vishwanath R Iyer. Genetic reconstruction of a functional transcriptional regulatory network. *Nature Genetics*, 39(5):683–687, 2007. [cited at p. 60, 77]

[82] Vân Anh Huynh-Thu, Alexandre Irrthum, Louis Wehenkel, and Pierre Geurts. Inferring regulatory networks from expression data using tree-based methods. *PLoS ONE*, 5(9):e12776, 2010. [cited at p. 43]

[83] Vân Anh Huynh-Thu, Louis Wehenkel, and Pierre Geurts. Gene regulatory network inference from systems genetics data using tree-based methods. In *Gene Network Inference*, pages 63–85. Springer, 2013. [cited at p. 103]

[84] Ritsert C Jansen. Studying complex biological systems using multifactorial perturbation. *Nature Reviews Genetics*, 4(2):145–151, 2003. [cited at p. 21]

[85] Ritsert C Jansen and Jan-Peter Nap. Genetical genomics: the added value from segregation. *TRENDS in Genetics*, 17(7):388–391, 2001. [cited at p. 21, 31, 32]

[86] Sema Kachalo, Ranran Zhang, Eduardo Sontag, Réka Albert, and Bhaskar DasGupta. NET-SYNTHESIS: a software for synthesis, inference and simplification of signal transduction networks. *Bioinformatics*, 24(2):293–295, 2008. [cited at p. 62]

[87] Noam Kaplan, Irene K Moore, Yvonne Fondufe-Mittendorf, Andrea J Gossett, Desiree Tillo, Yair Field, Emily M LeProust, Timothy R Hughes, Jason D Lieb, Jonathan Widom, et al. The DNA-encoded nucleosome organization of a eukaryotic genome. *Nature*, 458(7236):362–366, 2008. [cited at p. 79]

[88] W James Kent. BLAT – the BLAST-like alignment tool. *Genome Research*, 12(4):656–664, April 2002. [cited at p. 114]

[89] Joost JB Keurentjes, Jingyuan Fu, CH Ric De Vos, Arjen Lommen, Robert D Hall, Raoul J Bino, Linus HW van der Plas, Ritsert C Jansen, Dick Vreugdenhil, and Maarten Koornneef. The genetics of plant metabolism. *Nature Genetics*, 38(7):842–849, 2006. [cited at p. 21]

[90] Boris N Kholodenko, Anatoly Kiyatkin, Frank J Bruggeman, Eduardo Sontag, Hans V Westerhoff, and Jan B Hoek. Untangling the wires: a strategy to trace functional interactions in signaling and gene networks. *Proceedings of the National Academy of Sciences*, 99(20):12841–12846, 2002. [cited at p. 45]

[91] Steffen Klamt, Robert J Flassig, and Kai Sundmacher. TRANSWESD: inferring cellular networks with transitive reduction. *Bioinformatics*, 26(17):2160–2168, 2010. [cited at p. 59, 60, 62, 63, 69]

[92] Steffen Klamt and Axel von Kamp. Computing paths and cycles in biological interaction graphs. *BMC Bioinformatics*, 10(1):181, 2009. [cited at p. 63]

[93] Damodar D Kosambi. The estimation of map distances from recombination values. *Annals of Eugenics*, 12(1):172–175, 1943. [cited at p. 26, 38]

[94] Robert Küffner, Tobias Petri, Pegah Tavakkolkhah, Lukas Windhager, and Ralf Zimmer. Inferring gene regulatory networks by ANOVA. *Bioinformatics*, 28(10):1376–1382, 2012. [cited at p. 43]

[95] Ben Langmead and Steven L Salzberg. Fast gapped-read alignment with Bowtie 2. *Nature Methods*, 9(4):357–359, 2012. [cited at p. 114]

[96] Ben Langmead, Cole Trapnell, Mihai Pop, and Steven L Salzberg. Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. *Genome Biology*, 10(3):R25, 2009. [cited at p. 114]

[97] Su-In Lee, Dana Pe'er, Aimée M Dudley, George M Church, and Daphne Koller. Identifying regulatory mechanisms using individual variation reveals key role for chromatin modification. *Proceedings of the National Academy of Sciences*, 103(38):14062–14067, 2006. [cited at p. 22]

[98] Karen Lemmens, Tijl De Bie, Thomas Dhollander, Sigrid C De Keersmaecker, Inge M Thijs, Geert Schoofs, Ami De Weerdt, Bart De Moor, Jos Vanderleyden, Julio Collado-Vides, et al. DISTILLER: a data integration framework to reveal condition dependency of complex regulons in *Escherichia coli*. *Genome Biology*, 10(3):R27, 2009. [cited at p. 44]

[99] Simone Leo, Luca Pireddu, and Gianluigi Zanetti. SNP genotype calling with MapReduce. In *Proceedings of The Third International Workshop on MapReduce and its Applications*, MapReduce '12, pages 49–56, New York, NY, USA, 2012. ACM. [cited at p. 113]

[100] Heng Li and Richard Durbin. Fast and accurate short read alignment with Burrows–Wheeler transform. *Bioinformatics*, 25(14):1754–1760, 2009. [cited at p. 114]

[101] Renhua Li, Shirng-Wern Tsaih, Keith Shockley, Ioannis M Stylianou, Jon Wergedal, Beverly Paigen, and Gary A Churchill. Structural model analysis of multiple quantitative traits. *PLoS Genetics*, 2(7):e114, 2006. [cited at p. 22]

[102] Ruiqiang Li, Yingrui Li, Karsten Kristiansen, and Jun Wang. SOAP: short oligonucleotide alignment program. *Bioinformatics*, 24(5):713–714, 2008. [cited at p. 114]

[103] Bing Liu, Alberto de la Fuente, and Ina Hoeschele. Gene network inference via structural equation modeling in genetical genomics experiments. *Genetics*, 178(3):1763–1776, 2008. [cited at p. 22, 28, 32, 58]

[104] Nicholas J Loman, Chrystala Constantinidou, Jacqueline ZM Chan, Mihail Halachev, Martin Sergeant, Charles W Penn, Esther R Robinson, and Mark J Pallen. High-throughput bacterial genome sequencing: an embarrassment of choice, a world of opportunity. *Nature Reviews Microbiology*, 10(9):599–606, September 2012. [cited at p. 113]

[105] Todd M Lowe and Sean R Eddy. tRNAscan-SE: A program for improved detection of transfer RNA genes in genomic sequence. *Nucleic Acids Research*, 25(5):955–964, 1997. [cited at p. 114]

[106] Nicholas M Luscombe, M Madan Babu, Haiyuan Yu, Michael Snyder, Sarah A Teichmann, and Mark Gerstein. Genomic analysis of regulatory network dynamics reveals large topological changes. *Nature*, 431(7006):308–312, 2004. [cited at p. 79]

[107] Hong-Wu Ma and An-Ping Zeng. The connectivity structure, giant strong component and centrality of metabolic networks. *Bioinformatics*, 19(11):1423–1430, 2003. [cited at p. 17, 28]

[108] Avi Ma'Ayan, Guillermo A Cecchi, John Wagner, A Ravi Rao, Ravi Iyengar, and Gustavo Stolovitzky. Ordered cyclic motifs contribute to dynamic stability in biological and engineered networks. *Proceedings of the National Academy of Sciences*, 105(49):19235–19240, 2008. [cited at p. 17, 28]

[109] Tanja Magoc, Derrick Wood, and Steven L Salzberg. EDGE-pro: Estimated degree of gene expression in prokaryotic genomes. *Evolutionary Bioinformatics Online*, 9:127–136, 2013. [cited at p. 113]

[110] Robert Maier, Ralf Zimmer, and Robert Küffner. A Turing test for artificial expression data. *Bioinformatics*, 29(20):2603–2609, 2013. [cited at p. 17, 18]

[111] Daniel Marbach, James C Costello, Robert Küffner, Nicole M Vega, Robert J Prill, Diogo M Camacho, Kyle R Allison, Andrea Pinna, Nicola Soranzo, Vincenzo De Leo, Alberto de la Fuente, Manolis Kellis, James J Collins, Gustavo Stolovitzky, et al. Wisdom of crowds for robust gene network inference. *Nature Methods*, 2012. [cited at p. 5, 10, 43, 44, 79, 85, 86, 93, 95]

[112] Daniel Marbach, Robert J Prill, Thomas Schaffter, Claudio Mattiussi, Dario Floreano, and Gustavo Stolovitzky. Revealing strengths and weaknesses of methods for gene network inference. *Proceedings of the National Academy of Sciences*, 107(14):6286–6291, 2010. [cited at p. 10, 11, 31, 44, 57, 67, 86]

[113] Daniel Marbach, Sushmita Roy, Ferhat Ay, Patrick E Meyer, Rogerio Candeias, Tamer Kahveci, Christopher A Bristow, and Manolis Kellis. Predictive regulatory models in *Drosophila melanogaster* by integrative inference of transcriptional networks. *Genome Research*, 22(7):1334–1349, 2012. [cited at p. 44]

[114] Daniel Marbach, Thomas Schaffter, Claudio Mattiussi, and Dario Floreano. Generating realistic in silico gene networks for performance assessment of reverse engineering methods. *Journal of Computational Biology*, 16(2):229–239, 2009. [cited at p. 22, 48, 52, 86]

[115] Adam Margolin, Ilya Nemenman, Katia Basso, Chris Wiggins, Gustavo Stolovitzky, Riccardo Dalla Favera, and Andrea Califano. ARACNE: an algorithm for the reconstruction of gene regulatory networks in a mammalian cellular context. *BMC Bioinformatics*, 7(Suppl 1):S7, 2006. [cited at p. 43, 44, 83]

[116] Florian Markowetz and Rainer Spang. Inferring cellular networks – a review. *BMC Bioinformatics*, 8(Suppl 6):S5, 2007. [cited at p. 43]

[117] Eli Meir, Edwin M Munro, Garrett M Odell, and George Von Dassow. Ingeneue: a versatile tool for reconstituting genetic networks, with examples from the segment polarity network. *Journal of Experimental Zoology*, 294(3):216–251, 2002. [cited at p. 22]

[118] Pedro Mendes, Stefan Hoops, Sven Sahle, Ralph Gauges, Joseph Dada, and Ursula Kummer. Computational modeling of biochemical networks using COPASI. In *Systems Biology*, pages 17–59. Springer, 2009. [cited at p. 12, 14]

[119] Pedro Mendes, Wei Sha, and Keying Ye. Artificial gene networks for objective comparison of analysis algorithms. *Bioinformatics*, 19(suppl 2):ii122–ii129, 2003. [cited at p. 11, 22, 27, 45, 71]

[120] Pablo Meyer, Leonidas G Alexopoulos, Thomas Bonk, Andrea Califano, Carolyn R Cho, Alberto de la Fuente, David de Graaf, Alexander J Hartemink, Julia Hoeng, Nikolai V Ivanov, et al. Verification of systems biology research in the age of collaborative competition. *Nature Biotechnology*, 29(9):811, 2011. [cited at p. 10, 44, 45]

[121] Pablo Meyer, Thomas Cokelaer, Andrea Pinna, Nicola Soranzo, Alberto de la Fuente, et al. Network topology and parameter estimation: from experimental design methods to gene regulatory network kinetics using a community based approach. *BMC Systems Biology*, 8(1):13, 2014. [cited at p. 5, 109, 111]

[122] Pablo Meyer, Julia Hoeng, J Jeremy Rice, Raquel Norel, Jörg Sprengel, Katrin Stolle, Thomas Bonk, Stephanie Corthesy, Ajay Royyuru, Manuel C Peitsch, et al. Industrial methodology for process verification in research (IMPROVER): toward systems biology verification. *Bioinformatics*, 28(9):1193–1201, 2012. [cited at p. 44]

[123] Ron Milo, Shalev Itzkovitz, Nadav Kashtan, Reuven Levitt, Shai Shen-Orr, Inbal Ayzenshtat, Michal Sheffer, and Uri Alon. Superfamilies of evolved and designed networks. *Science*, 303(5663):1538–1542, 2004. [cited at p. 51]

[124] Toshiaki Namiki, Tsuyoshi Hachiya, Hideaki Tanaka, and Yasubumi Sakakibara. MetaVelvet: an extension of Velvet assembler to *de novo* metagenome assembly from short sequence reads. *Nucleic Acids Research*, 40(20):e155, 2012. [cited at p. 114]

[125] Sven Nelander, Weiqing Wang, Björn Nilsson, Qing-Bai She, Christine Pratilas, Neal Rosen, Peter Gennemark, and Chris Sander. Models from experiments: combinatorial drug perturbations of cancer cells. *Molecular Systems Biology*, 4(1), 2008. [cited at p. 43]

[126] Elias Chaibub Neto, Christine T Ferrara, Alan D Attie, and Brian S Yandell. Inferring causal phenotype networks from segregating populations. *Genetics*, 179(2):1089–1100, 2008. [cited at p. 22]

[127] Raquel Norel, John Jeremy Rice, and Gustavo Stolovitzky. The self-assessment trap: can we all be better than average? *Molecular Systems Biology*, 7(1), 2011. [cited at p. 10]

[128] Brian D Ondov, Nicholas H Bergman, and Adam M Phillippy. Interactive metagenomic visualization in a web browser. *BMC Bioinformatics*, 12:385, 2011. [cited at p. 115]

[129] Rainer Opgen-Rhein and Korbinian Strimmer. From correlation to causation networks: a simple approximate learning algorithm and its application to high-dimensional plant gene expression data. *BMC Systems Biology*, 1(1):37, 2007. [cited at p. 44]

[130] Elena Parkhomenko, David Tritchler, and Joseph Beyene. Genome-wide sparse canonical correlation of gene expression with genotypes. In *BMC Proceedings*, volume 1, page S119. BioMed Central Ltd, 2007. [cited at p. 22]

[131] Enrico Pieroni, Sergio de la Fuente van Bentem, Gianmaria Mancosu, Enrico Capobianco, Heribert Hirt, and Alberto de la Fuente. Protein networking: insights into global functional organization of proteomes. *Proteomics*, 8(4):799–816, 2008. [cited at p. 17]

[132] Andrea Pinna, Sandra Heise, Robert J Flassig, Alberto de la Fuente, and Steffen Klamt. Reconstruction of large-scale regulatory networks based on perturbation graphs and transitive reduction: improved methods and their evaluation. *BMC Systems Biology*, 7(1):73, 2013. [cited at p. 3, 4, 22, 40, 45, 47, 58, 99]

[133] Andrea Pinna, Carla Seatzu, and Alberto de la Fuente. RAGNO: reconstruction algorithm for gene networks from multi-omics data. In *Methods in Molecular Biology*. Springer, 2014. [cited at p. 5, 97]

[134] Andrea Pinna, Nicola Soranzo, and Alberto de la Fuente. From knockouts to networks: establishing direct cause-effect relationships through graph analysis. *PLoS ONE*, 5(10):e12912, 2010. [cited at p. 4, 22, 47, 59, 60, 61, 67, 81]

[135] Andrea Pinna, Nicola Soranzo, Alberto de la Fuente, and Ina Hoeschele. Simulation of the benchmark datasets. In *Gene Network Inference*, pages 1–8. Springer, 2013. [cited at p. 3, 100]

[136] Andrea Pinna, Nicola Soranzo, Ina Hoeschele, and Alberto de la Fuente. Simulating systems genetics data with SysGenSIM. *Bioinformatics*, 27(17):2459–2462, 2011. [cited at p. 2, 16, 21, 32, 34, 40, 67, 71, 86, 88, 100, 110]

[137] Luca Pireddu, Gianmauro Cuccuru, Luca Lianas, Matteo Vocale, Giorgio Fotia, and Gianluigi Zanetti. Automated and traceable processing for large-scale high-throughput sequencing facilities. *EMBnet.journal*, 19(A):23–24, 2013. [cited at p. 113]

[138] Robert J Prill, Daniel Marbach, Julio Saez-Rodriguez, Peter K Sorger, Leonidas G Alexopoulos, Xiaowei Xue, Neil D Clarke, Gregoire Altan-Bonnet, and Gustavo Stolovitzky. Towards a rigorous assessment of systems biology models: the DREAM3 challenges. *PLoS ONE*, 5(2):e9202, 2010. [cited at p. 31, 43, 45, 52, 86]

[139] Arcot Rajasekar, Reagan Moore, Chien-Yi Hou, Christopher A Lee, Richard Marciano, Antoine de Torcy, Michael Wan, Wayne Schroeder, Sheau-Yen Chen, Lucas Gilbert, Paul Tooby, and Bing Zhu. iRODS primer: integrated rule-oriented data system. *Synthesis Lectures on Information Concepts, Retrieval, and Services*, 2(1):1–143, 2010. [cited at p. 113]

[140] Jüri Reimand, Juan M Vaquerizas, Annabel E Todd, Jaak Vilo, and Nicholas M Luscombe. Comprehensive reanalysis of transcription factor knockout expression data in *Saccharomyces cerevisiae* reveals many new targets. *Nucleic Acids Research*, 38(14):4768–4777, 2010. [cited at p. 60, 77]

[141] David J Reiss, Nitin S Baliga, and Richard Bonneau. Integrated biclustering of heterogeneous genome-wide datasets for the inference of global regulatory networks. *BMC Bioinformatics*, 7(1):280, 2006. [cited at p. 44]

[142] John Jeremy Rice, Yuhai Tu, and Gustavo Stolovitzky. Reconstructing biological networks using conditional correlation analysis. *Bioinformatics*, 21(6):765–773, 2005. [cited at p. 43]

[143] Matthew V Rockman. Reverse engineering the genotype–phenotype map with natural genetic variation. *Nature*, 456(7223):738–744, 2008. [cited at p. 21, 32, 58]

[144] Roy Ronen, Christina Boucher, Hamidreza Chitsaz, and Pavel Pevzner. SEQuel: improving the accuracy of genome assemblies. *Bioinformatics*, 28(12):i188–i196, 2012. [cited at p. 113]

[145] Sushmita Roy, Margaret Werner-Washburne, and Terran Lane. A system for generating transcription regulatory networks with combinatorial control of transcription. *Bioinformatics*, 24(10):1318–1320, 2008. [cited at p. 14]

[146] Julio Saez-Rodriguez, Leonidas G Alexopoulos, Jonathan Epperlein, Regina Samaga, Douglas A Lauffenburger, Steffen Klamt, and Peter K Sorger. Discrete logic modelling as a means to link protein signalling networks with functional analysis of mammalian signal transduction. *Molecular Systems Biology*, 5(1), 2009. [cited at p. 43]

[147] Francesco Sambo, Tiziana Sanavia, and Barbara Di Camillo. Integration of genetic variation as external perturbation to reverse engineer regulatory networks from gene expression data. In *Gene Network Inference*, pages 107–118. Springer, 2013. [cited at p. 104]

[148] Eric E Schadt. Molecular networks as sensors and drivers of common human diseases. *Nature*, 461(7261):218–223, 2009. [cited at p. 21, 43]

[149] Eric E Schadt, John Lamb, Xia Yang, Jun Zhu, Steve Edwards, Debraj GuhaThakurta, Solveig K Sieberts, Stephanie Monks, Marc Reitman, Chunsheng Zhang, et al. An integrative genomics approach to infer causal associations between gene expression and disease. *Nature Genetics*, 37(7):710–717, 2005. [cited at p. 25]

[150] Eric E Schadt, Stephanie A Monks, Thomas A Drake, Aldons J Lusis, Nam Che, Veronica Colinayo, Thomas G Ruff, Stephen B Milligan, John R Lamb, Guy Cavet, et al. Genetics of gene expression surveyed in maize, mouse and man. *Nature*, 422(6929):297–302, 2003. [cited at p. 21]

[151] Juliane Schäfer, Rainer Opgen-Rhein, and Korbinian Strimmer. Reverse engineering genetic networks using the GeneNet package. *Journal of the American Statistical Association*, 96:1151–1160, 2001. [cited at p. 87]

[152] Juliane Schäfer and Korbinian Strimmer. An empirical Bayes approach to inferring large-scale gene association networks. *Bioinformatics*, 21(6):754–764, 2005. [cited at p. 43, 87]

[153] Thomas Schaffter, Daniel Marbach, and Dario Floreano. GeneNetWeaver: in silico benchmark generation and performance profiling of network inference methods. *Bioinformatics*, 27(16):2263–2270, 2011. [cited at p. 13, 22, 69, 86]

[154] Alan Scheinine, Wieslawa I Mentzen, Giorgio Fotia, Enrico Pieroni, Fabio Maggio, Gianmaria Mancosu, and Alberto De La Fuente. Inferring gene networks: dream or nightmare? *Annals of the New York Academy of Sciences*, 1158(1):287–301, 2009. [cited at p. 49, 57, 111]

[155] Thomas Schlitt and Alvis Brazma. Current approaches to gene regulatory network modelling. *BMC Bioinformatics*, 8(Suppl 6):S9, 2007. [cited at p. 43]

[156] Nicola Segata, Levi Waldron, Annalisa Ballarini, Vagheesh Narasimhan, Olivier Jousson, and Curtis Huttenhower. Metagenomic microbial community profiling using unique clade-specific marker genes. *Nature Methods*, 9(8):811–814, 2012. [cited at p. 114]

[157] Jared T Simpson, Kim Wong, Shaun D Jackman, Jacqueline E Schein, Steven JM Jones, and İnanç Birol. ABySS: a parallel assembler for short read sequence data. *Genome Research*, 19(6):1117–1123, 2009. [cited at p. 114]

[158] V Anne Smith, Erich D Jarvis, and Alexander J Hartemink. Evaluating functional network inference using simulations of complex biological systems. *Bioinformatics*, 18(suppl 1):S216–S224, 2002. [cited at p. 11, 45]

[159] Nicola Soranzo, Ginestra Bianconi, and Claudio Altafini. Comparing association network algorithms for reverse engineering of large-scale gene regulatory networks: synthetic versus real data. *Bioinformatics*, 23(13):1640–1647, 2007. [cited at p. 44, 45, 52]

[160] Nicola Soranzo, Andrea Pinna, Vincenzo De Leo, and Alberto de la Fuente. Elucidating transcriptional regulatory networks from heterogeneous gene expression compendia. *Submitted to PLoS ONE*, 2014. [cited at p. 5, 22, 85]

[161] Nicola Soranzo, Andrea Pinna, Vincenzo De Leo, and Alberto de la Fuente. Simulating the transcriptome for the evaluation of gene regulatory network inference algorithms. *In preparation*, 2014. [cited at p. 2, 7]

[162] Peter Spirtes, Clark Glymour, and Richard Scheines. *Causation, prediction, and search*, volume 81. The MIT Press, 2000. [cited at p. 44]

[163] Gustavo Stolovitzky, Pascal Kahlem, and Andrea Califano. The challenges of systems biology community efforts to harness biological complexity. *Annals of the New York Academy of Sciences*, 1158(1):ix–xii, 2009. [cited at p. 44, 48]

[164] Gustavo Stolovitzky, Don Monroe, and Andrea Califano. Dialogue on reverse-engineering assessment and methods: the dream of high-throughput pathway inference. *Annals of the New York Academy of Sciences*, 1115(1):1–22, 2007. [cited at p. 22, 44, 48]

[165] Gustavo Stolovitzky, Robert J Prill, and Andrea Califano. Lessons from the DREAM2 challenges. *Annals of the New York Academy of Sciences*, 1158(1):159–195, 2009. [cited at p. 22, 33]

[166] John D Storey, Joshua M Akey, and Leonid Kruglyak. Multiple locus linkage analysis of genomewide expression in yeast. *PLoS Biology*, 3(8):e267, 2005. [cited at p. 104]

[167] Michael PH Stumpf, Thomas Thorne, Eric de Silva, Ronald Stewart, Hyeong Jun An, Michael Lappe, and Carsten Wiuf. Estimating the size of the human interactome. *Proceedings of the National Academy of Sciences*, 105(19):6959–6964, 2008. [cited at p. 10, 44, 45]

[168] Robert Tibshirani. Regression shrinkage and selection via the LASSO. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996. [cited at p. 43]

[169] Achim Tresch, Tim Beissbarth, Holger Sültmann, Ruprecht Kuner, Annemarie Poustka, and Andreas Buness. Discrimination of direct and indirect interactions in a network of regulatory effects. *Journal of Computational Biology*, 14(9):1217–1228, 2007. [cited at p. 62]

[170] Tim Van den Bulcke, Koenraad Van Leemput, Bart Naudts, Piet van Remortel, Hongwu Ma, Alain Verschoren, Bart De Moor, and Kathleen Marchal. SynTReN: a generator of synthetic gene expression data for design and analysis of structure learning algorithms. *BMC Bioinformatics*, 7(1):43, 2006. [cited at p. 13]

[171] Eugene P van Someren, Lodewyk FA Wessels, and Marcel JT Reinders. Linear modeling of genetic networks from experimental data. *Proceedings of the International Conference on Intelligent Systems for Molecular Biology*, 8:355–66, 2000. [cited at p. 11, 45]

[172] Matthieu Vignes, Jimmy Vandel, David Allouche, Nidal Ramadan-Alban, Christine Cierco-Ayrolles, Thomas Schiex, Brigitte Mangin, and Simon de Givry. Gene regulatory network reconstruction using Bayesian networks, the Dantzig selector, the LASSO and their meta-analysis. *PLoS ONE*, 6(12):e29165, 2011. [cited at p. 101]

[173] Sandra Waaijenborg, Philip C Verselewel de Witt Hamer, and Aeilko H Zwinderman. Quantifying the association between gene expressions and DNA-markers by penalized canonical correlation analysis. *Statistical Applications in Genetics and Molecular Biology*, 7(1), 2008. [cited at p. 22]

[174] Andreas Wagner. How to reconstruct a large genetic network from n gene perturbations in fewer than n^2 easy steps. *Bioinformatics*, 17(12):1183–1197, 2001. [cited at p. 49, 50, 62]

[175] Yali Wang and Tong Zhou. A relative variation-based method to unraveling gene regulatory networks. *PLoS ONE*, 7(2):e31194, 2012. [cited at p. 71]

[176] Zhong Wang, Mark Gerstein, and Michael Snyder. RNA-Seq: a revolutionary tool for transcriptomics. *Nature Reviews Genetics*, 10(1):57–63, 2009. [cited at p. 9]

[177] René L Warren, Granger G Sutton, Steven JM Jones, and Robert A Holt. Assembling millions of short DNA sequences using SSAKE. *Bioinformatics*, 23(4):500–501, 2007. [cited at p. 113]

[178] John Watkinson, Kuo-ching Liang, Xiadong Wang, Tian Zheng, and Dimitris Anastassiou. Inference of regulatory gene interactions from expression data using three-way mutual information. *Annals of the New York Academy of Sciences*, 1158(1):302–313, 2009. [cited at p. 43]

[179] Duncan J Watts and Steven H Strogatz. Collective dynamics of *small-world* networks. *Nature*, 393(6684):440–442, 1998. [cited at p. 23]

[180] Marilyn AL West, Kyunga Kim, Daniel J Kliebenstein, Hans van Leeuwen, Richard W Michelmore, RW Doerge, and Dina A St Clair. Global eQTL mapping reveals the complex genetic architecture of transcript-level variation in *Arabidopsis*. *Genetics*, 175(3):1441–1450, 2007. [cited at p. 18]

[181] Elizabeth A Winzeler, Daniel D Shoemaker, Anna Astromoff, Hong Liang, Keith Anderson, Bruno Andre, Rhonda Bangham, Rocio Benito, Jef D Boeke, Howard Bussey, et al. Functional characterization of the *S. cerevisiae* genome by gene deletion and parallel analysis. *Science*, 285(5429):901–906, 1999. [cited at p. 60]

[182] Fred A Wright, Hanwen Huang, Xiaojun Guan, Kevin Gamiel, Clark Jeffries, William T Barry, Fernando Pardo-Manuel de Villena, Patrick F Sullivan, Kirk C Wilhelmsen, and Fei Zou. Simulating association studies: a data-based resampling method for candidate regions or whole genome scans. *Bioinformatics*, 23(19):2581–2588, 2007. [cited at p. 26, 30]

[183] Kevin Y Yip, Roger P Alexander, Koon-Kiu Yan, and Mark Gerstein. Improved reconstruction of in silico gene regulatory networks by integrating knockout and perturbation data. *PLoS ONE*, 5(1):e8121, 2010. [cited at p. 57]

[184] Daniel R Zerbino and Ewan Birney. Velvet: algorithms for de novo short read assembly using de Bruijn graphs. *Genome Research*, 18(5):821–829, 2008. [cited at p. 114]

[185] Wei Zhang, Jun Zhu, Eric E Schadt, and Jun S Liu. A Bayesian partition method for detecting pleiotropic and epistatic eQTL modules. *PLoS Computational Biology*, 6(1):e1000642, 2010. [cited at p. 22]

[186] Jun Zhu, Pek Yee Lum, John Lamb, Debraj GuhaThakurta, Steve W Edwards, Rolf Thieringer, Joel P Berger, Min Shung Wu, John R Thompson, Alan B Sachs, and Eric E Schadt. An integrative genomics approach to the reconstruction of gene networks in segregating populations. *Cytogenetic and Genome Research*, 105(2-4):363–374, 2004. [cited at p. 22]

[187] Wenhan Zhu, Alexandre Lomsadze, and Mark Borodovsky. *Ab initio* gene identification in metagenomic sequences. *Nucleic Acids Research*, 38(12):e132, 2010. [cited at p. 114]

List of thesis-related publications

Published works

- Andrea Pinna, Nicola Soranzo, Alberto de la Fuente (2010). *From knockouts to networks: establishing direct cause-effect relationships through graph analysis*. PLoS ONE, 5(10), e12912.
- Andrea Pinna, Nicola Soranzo, Ina Hoeschele, Alberto de la Fuente (2011). *Simulating systems genetics data with SysGenSIM*. Bioinformatics, 27(17), 2459-2462.
- Daniel Marbach, James C Costello, Andrea Pinna, et al. (2012). *Wisdom of crowds for robust gene network inference*. Nature Methods, 9, 796-804.
- Andrea Pinna, Sandra Heise, Robert J Flassig, Alberto de la Fuente, Steffen Klamt (2013). *Reconstruction of large-scale regulatory networks based on perturbation graphs and transitive reduction: improved methods and their performance*. BMC Systems Biology, 7:73.
- Andrea Pinna, Nicola Soranzo, Alberto de la Fuente, Ina Hoeschele (2013). *Simulation of the benchmark datasets*. In *Gene network inference – verification of methods for systems genetics data*. Springer.
- Pablo Meyer, Thomas Cokelaer, Andrea Pinna, et al. (2014). *Network topology and parameter estimation: from experimental design methods to gene regulatory network kinetics using a community based approach*. BMC Systems Biology, 8:13.
- Gianmauro Cuccuru, Massimiliano Orsini, Andrea Pinna, et al. (2014). *Orione, a web-based framework for NGS analysis in microbiology*. Bioinformatics.

Submitted works

- Nicola Soranzo, Andrea Pinna, Vincenzo De Leo, Alberto de la Fuente (submitted). Elucidating transcriptional regulatory networks from heterogeneous gene expression compendia.
- Nicola Soranzo, Andrea Pinna, Vincenzo De Leo, Alberto de la Fuente (in preparation). Simulating the transcriptome for the evaluation of gene regulatory network inference algorithms.
- Andrea Pinna, Carla Seatzu, Alberto de la Fuente (in preparation). RAGNO: reconstruction algorithm for gene networks from multi-omics data.

Conference posters

- Groningen (2009). SysGenSIM: simulating large systems genetics datasets for the evaluation of analysis methods.
- DISC PhD School (2011). Structural identification of gene regulatory networks from gene expression datasets.
- FEBS-SystemsX-SysBio (2011). Elucidating transcriptional regulatory networks from heterogeneous gene expression compendia.
- StatSeq meeting (2013). Simulating systems genetics with SysGenSIM.
- FEBS-SystemsX-SysBio (2011). Differential networking studies of microRNA regulation.
- RECOMB Systems Biology Conference (2011). Simulating large systems genetics datasets for the evaluation of analysis methods.
- Galaxy Community Conference (2013). Engaging Galaxy in microbiology.
- Galaxy Community Conference (2013). Microbiome profiling on a Galaxy-based framework for microbiology.